Les CHARTREUX

Nom : .....

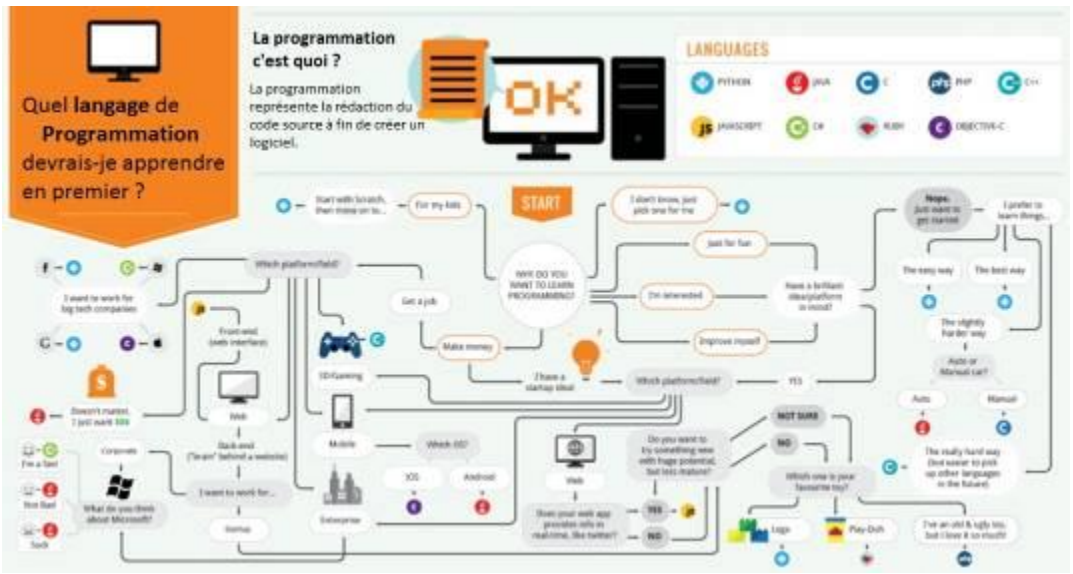
Prénom : .....

Classe : .....

NUMERIQUE ET SCIENCES INFORMATIQUES (NSI)

CLASSE DE PREMIERE

6ème édition



Réforme du lycée
NOUVEAU PROGRAMME 2019

MONSIEUR FONTAINE
MONSIEUR VINSU
MONSIEUR PELLERIN

INSTITUTION DES CHARTREUX

SOMMAIRE

D/ REPRESENTATION DES DONNEES (TYPES CONSTRUITS) :

E/ TRAITEMENT DE DONNEES EN TABLES :

F/ ARCHITECTURES MATERIELLES ET SYSTEMES D'EXPLOITATION :

G/ ALGORITHMIQUE :

ANNEXES :

La spécialité NSI en classe de première

I – Matériel nécessaire :

- ✓ Un porte-vue de 60 pages / 120 vues (ou un classeur souple) de couleur verte (si possible) pour le cahier d'activité.
- ✓ Ajouter au cahier d'activité le règlement informatique des Chartreux.
- ✓ Une adresse mail personnelle accessible.
- ✓ Une clé USB.
- ✓ À la maison : un ordinateur (fixe ou portable), de préférence sous Windows (la liste des logiciels sera indiquée au fur et à mesure). Le plus important reste la distribution Python « Anaconda » avec l'IDLE Spyder.

II – Travail et méthode :

- ✓ Cette spécialité demande beaucoup d'autonomie et de curiosité intellectuelle.
- ✓ Il y aura peu de « cours magistraux » : ce sera à vous de reprendre les exercices pour maitriser les notions au programme de la spécialité NSI.

III – Evaluations :

- ✓ En cours, des mini-projets en fin de chaque chapitre (généralement à deux sur machine, parfois seul en déconnecté, de temps en temps sous la forme d'une interrogation). Rendre les projets sous le format **nom1_nom2.py** (noms par ordre alphabétique).
- ✓ Des DS moitié sous forme de QCM à points négatifs (+3 / 0 / -1), moitié sous forme d'exercices (coefficient 1 pour 2). Les DS représenteront 50% (si un DS) ou 60% (si deux DS) de la moyenne.
- ✓ Des interrogations orales en début de chaque cours.
- ✓ Spécialité abandonnée : 8 % (contrôle continu).
- ✓ Spécialité non abandonnée : 16 % (épreuves de terminale) et 5% (grand oral).

IV - Programme :

- ✓ Voir le sommaire.

Signature de l'élève : Signature du responsable légal :

Langages de programmation

Spécialisée dans les missions informatiques et le travail indépendant, la plate-forme IT Profil a dressé les tendances du moment du secteur, où la gestion des données gagne en intérêt.

En se basant sur l'analyse de 20 000 utilisateurs de sa plate-forme, IT Profil - spécialiste du freelance et des missions - a dessiné les tendances de l'année en matière de recrutement dans le numérique, un secteur où les projets d'embauches sont supérieurs à celles des autres secteurs : 88 % contre 51 %, d'après l'Apec (Association Pour l'Emploi des Cadres).

Les deux secteurs qui embauchent le plus sont l'informatique de gestion (business intelligence, progiciels de gestion intégrée, systèmes de gestions de bases de données) avec 30 % des recrutements, suivis de près par le marketing et la communication (29 %), devenus centraux dans le numérique. Suivent ensuite les services d'infrastructure (17 %), le service commercial (13 %) et encore le consulting et la gestion de projets (6 %).

En se penchant sur le métier de développeur, l'un des plus recherchés, IT Profil a dressé le top 10 des langages les plus demandés. Voici le classement :

1. **JavaScript** : scripts utilisés dans les pages web interactives créé par Mozilla en 1995.
2. **Python** : langage de programmation objet créé en 1990 sous licence libre.
3. Java : langage permettant de développer des applications client-serveur créé par Sun (Oracle en 2009) en 1995.
4. PHP : langage utilisé pour produire des pages web dynamiques créé en 1994 (licence libre). Il est généralement associé au **SQL** (exploitation des bases de données – IBM 1974).
5. C# (C sharp) : langage dérivé du C++ commercialisé par Microsoft depuis 2002.
6. C++ : langage de programmation compilé créé en 1990 (libre de droits).
7. **CSS** : langage permettant la mise en page des pages Web (1996) écrit en langage **HTML** (créé par le CERN en 1993).
8. TypeScript : langage développé par Microsoft, améliorant le JavaScript (2012).
9. Ruby : langage de programmation fortement orienté objet (1995).
10. **C** : langage de programmation impératif créé par les laboratoires Bell en 1972.

Autre tendance : l'appétence des entreprises pour les experts en gestion de bases de données, reflet, d'après l'auteur de l'étude, de l'enjeu grandissant de la data - pour le marketing, l'aide à la décision, etc.

Sur ce terrain, les compétences les plus demandées sont : MySQL, Access, SQL Server, Oracle Database et PostgreSQL. Sur l'administration des systèmes et réseaux, les experts Windows sont les plus courus, devant Linux, TCP/IP, Unix et Cisco. Le profil type du « freelance IT » a un bac +4, a cumulé 5 ans d'expériences, maîtrise 4 langages, réalise 2 missions de 6 mois par an et facture un taux journalier moyen de 480 euros.

D'après Clubic Pro, « Recrutement informatique : les tendances 2016 », 13 janvier 2016
[Classement mis à jour en 2021]
En gras / soulignés les langages enseignés en SNT/NSI (seconde à terminale) / BTS info

Présentation de Python

Guido van Rossum est le créateur du langage informatique Python – nommé en hommage aux Monty Python –, l'un des plus populaires au monde. Après près de trente ans comme responsable de l'évolution du langage, cet informaticien néerlandais, passé par Google et qui travaille aujourd'hui pour Dropbox, a annoncé au début de juillet son intention de prendre sa retraite et de laisser le développement du langage à la communauté des développeurs. Il revient pour Pixels sur l'étonnant parcours de Python, qui a connu un succès exponentiel depuis le début des années 2010.

Une étude récente du site d'aide aux développeurs Stack Overflow montre que Python, qui a 30 ans, est devenu le langage le plus recherché sur la plate-forme. Les tendances Google soulignent que, depuis 2013, Python a quasiment éclipsé tous les autres langages. Comment expliquez-vous ce succès ?

Guido van Rossum : Quand j'ai commencé à travailler sur Python, je n'imaginais certainement pas qu'il connaîtrait un tel succès ! Dans les années 1990, Python était un « petit » acteur, même au sein de la catégorie des langages dynamiques. Le grand acteur, c'était Perl. Je connaissais Larry [Wall, le créateur de Perl], et j'étais en désaccord avec certaines de ses idées. Le temps a montré que d'autres personnes partageaient ma vision, et j'ai été très content quand, au milieu des années 2000, Python a commencé à prendre le dessus sur Perl. La croissance récente a plusieurs explications, mais l'un des éléments-clés, c'est que le secteur du big data a adopté Python comme l'un de ses principaux outils. Le big data a très fortement progressé ces derniers temps, et c'est clairement l'une des raisons qui ont poussé le succès de Python.

Guido van Rossum en 2014, au siège de Dropbox.

Python est aussi très utilisé par les chercheurs en intelligence artificielle, un autre secteur qui a le vent en poupe. Pourtant, ces deux disciplines n'existaient pas, en tout cas sous cette forme, à la création de Python…

C'est vrai. Mais à la fin des années 1990, il y avait un certain nombre de personnes qui voulaient utiliser des bases de code écrites dans les langages de programmation les plus rapides, comme C++, mais qui avaient aussi besoin d'un langage leur permettant de lier ces briques logicielles ensemble. C'est alors qu'est né Numeric Python, et plus tard NumPy. Cela a lancé tout un mouvement : des gens ont commencé à ajouter des outils de visualisation de données, [la bibliothèque graphique] matplotlib…
Une fois qu'il y a un certain nombre d'outils disponibles, dans une discipline donnée, les chercheurs ont tendance à en ajouter d'autres ! Par exemple IPython, qui leur a permis de jouer avec leurs données, de manière dynamique et expérimentale. Que se passe-t-il si je projette telle ou telle donnée sur tel ou tel axe ? Cette interactivité était incroyablement importante. IPython a grossi, puis est arrivé IPython Notebook, un autre outil très important pour les communautés de chercheurs, et plus généralement pour les gens qui manipulent de grandes quantités de données.
Toute la communauté Python est incroyablement impliquée dans l'open source [le partage libre et ouvert des logiciels créés]. Les développeurs copient ce qu'ils connaissent et ce qu'ils aiment, l'améliorent ou l'étendent, et rendent le résultat open source. C'est une philosophie commune.

D'autres langages, comme Perl, étaient aussi open source, mais ont eu moins de succès à long terme. Qu'est-ce qui faisait la différence pour Python ?

Il y a deux raisons, je pense. D'abord, c'est très facile de brancher du code C++ à Python, et pour quelqu'un qui n'a pas une expérience d'administrateur système Linux, Python est beaucoup plus facile à apprendre que Perl.
En administration système, on utilise beaucoup de scripts – Linux a tous ces outils que l'on peut combiner pour faire des choses utiles. Perl utilise de manière très astucieuse beaucoup de choses issues de ce système – le dollar pour les variables, les expressions régulières… – et est donc devenu une meilleure version des scripts Unix. Mais quand vous enseignez à des gens qui n'ont aucune expérience en programmation, Perl n'est pas un bon langage. Pour l'apprendre, il faut maîtriser beaucoup de choses, qui ne sont pas du niveau d'un lycéen ou d'une lycéenne par exemple.
La philosophie de Python est que vous avez uniquement besoin de connaître un peu les mathématiques, niveau lycée, et quelques clefs du langage pour commencer à faire des choses. C'est un langage très facile à apprendre pour des chercheurs, qui connaissent les mathématiques et ont quelques bases en informatique. C'est notamment pour cela que la communauté scientifique s'en est rapidement emparée.

Python est effectivement très utilisé dans l'enseignement de l'informatique, y compris en lycée en France. Etait-ce l'un de vos buts quand vous l'avez créé ?

Ce n'était pas sur ma liste ! Mais Python a emprunté beaucoup de choses à ABC [un langage jamais finalisé sur lequel M. van Rossum avait précédemment travaillé], qui était conçu pour être facile à enseigner. Dans les années 1990, j'ai rencontré un développeur qui m'a raconté avoir donné Python à son fils, 12 ans à l'époque, et qu'il avait adoré. Son fils a même découvert un bug qui faisait planter l'interpréteur [le logiciel qui exécute le programme] ! Python est assez intuitif, et les enfants n'aiment pas devoir lire des pages et des pages de documentation. Ils apprennent en tapant sur le clavier ! Mais à un certain point, il faut au minimum savoir comment chercher dans la documentation – arrivé à un certain niveau, chercher la réponse sur Google ne sera plus suffisant.

Quels conseils donneriez-vous à quelqu'un qui débute la programmation avec Python ?

Je leur dirais de s'amuser ! Et de lire le code d'autres personnes, à partir du moment où ils ont le niveau suffisant pour comprendre comment il fonctionne. Un autre conseil important : ne pas s'y attaquer seul. Trouvez un ami, résolvez les problèmes ensemble ; si personne dans votre entourage n'est intéressé, trouvez un groupe amical sur Internet. Poser des questions est une manière incroyablement efficace d'apprendre. Il faut juste trouver le bon moment pour appeler à l'aide, une fois que vous avez vraiment essayé de trouver par vous-même. En général, c'est le moment où vous commencez à avoir envie de vous cogner la tête contre le mur !

Dans un discours il y a deux ans, vous disiez que « les langages de programmation sont la manière dont les programmeurs expriment et communiquent leurs idées ». Quelle idée vouliez-vous communiquer en créant Python ?

Je voulais que Python soit un peu différent des autres, et montrer qu'il n'y a rien de mal à être différent. A l'époque, la clarté n'était pas au cœur de la philosophie des langages. Ils se concentraient sur l'efficacité du code, c'était un héritage des débuts de l'informatique, dans les années 1940 : quand il faut une journée entière pour faire tourner un programme, vous avez intérêt à ce que votre code soit aussi efficace que possible. Mais cette efficacité se faisait au détriment de la lisibilité, et les langages étaient très complexes pour les utilisateurs.
Dans la philosophie de Python, le temps d'exécution n'est pas aussi important que le temps nécessaire à l'écriture du code. Prenons un exemple : pour accomplir une certaine tâche, il vous faut une journée pour écrire un programme, qui s'exécute en une seconde. Dans bien des cas, certains utilisateurs préféreront, pour accomplir la même tâche, avoir un programme qui s'exécute en cinq minutes, mais qui s'écrit en une demi-heure. C'est une meilleure solution, sauf si vous devez faire tourner ce programme des centaines de fois par jour. L'idée était vraiment de dire : « tant pis pour toutes ces astuces très malines qui rendent la vie des utilisateurs impossible ». D'accord, le code s'exécutait un peu moins vite, mais quand la performance est devenue un problème, la communauté a amélioré le langage. Comme le disait le célèbre informaticien Donald Knuth [reprenant une formule de Charles Antony Hoare], « l'optimisation prématurée est la source de tous les maux ».

Pourquoi abandonnez-vous aujourd'hui la direction du projet ? Vous aviez pourtant le titre honorifique de « dictateur bienveillant à vie »…

Bien sûr, cette idée de « dictature à vie » est une blague – originellement, le titre complet était même « dictateur bienveillant à vie par intérim ». C'était juste une manière amusante de désigner le leader d'un projet open source. Je l'ai fait pendant vingt-huit ans. J'en ai aujourd'hui 62, et cela fait plusieurs années que j'envisage de prendre ma retraite. C'est un travail stressant.
La goutte d'eau qui a fait déborder le vase a probablement été la manière dont les discussions en ligne se sont compliquées ces dernières années. Pendant tout ce temps, j'ai connu beaucoup de désaccords avec mes décisions, mais plus récemment, les discussions ont pris un ton désagréable, et parfois insultant. Cela a fini par me ronger. Un matin, je me suis levé – la veille, j'avais validé une nouvelle décision controversée –, et j'ai senti que c'était trop pour moi. Il est temps de laisser la communauté des développeurs trouver une autre manière de prendre les décisions, une manière qui ne me place pas en permanence en première ligne.

Qu'est-ce qui a changé dans la manière dont les discussions se déroulent ?

J'ai vécu avec les désaccords pendant quarante ans, et ce n'était pas un problème. Ce qui est nouveau, c'est la manière dont les gens expriment leur désaccord. Cela m'a donné l'impression qu'on ne me faisait plus confiance. Un message agressif publié sur un réseau social par un développeur important, ce n'est pas une bonne manière de faire un retour. Auparavant, quand quelqu'un me disait « je ne suis pas d'accord », je pouvais dire « O.K., je prends la responsabilité de cette décision ». Sur les réseaux sociaux, je ne veux pas me noyer dans une discussion infinie avec 100 000 followers. On ne peut pas gagner un débat sur Twitter.
Est-ce plus généralement le climat sur les réseaux sociaux qui a changé ? Après l'élection de Donald Trump, beaucoup d'Américains se sont plaints d'une hostilité grandissante…
Je ne suis pas un expert, mais sur les réseaux sociaux, les débats politiques ont tendance à virer à la foire d'empoigne. Si c'est ce que le président des Etats-Unis ou l'Union européenne, veulent faire, c'est malheureux. Je ne suis pas fait pour ça – je n'aime pas dire des choses méchantes sur d'autres personnes.

D'après Damien Leloup, Le Monde, 25 juillet 2018

Histoire de l'informatique

I – Les dates les plus importantes :

1642 (invention) et 1652 (réalisation) :

Blaise …… (1623-1662), savant …… produit une machine à calculer, (la « Pascaline ») capable d'additionner et de soustraire des nombres de huit chiffres !

1673 (invention) et 1694 (réalisation) :

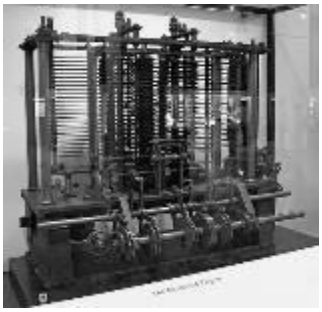
Gottfried Leibniz (1646-1716), savant …… améliore la Pascaline en construisant une machine capable de multiplier et …… sans utiliser les additions successives !

1725 (Bouchon) et 1801 (Jacquard) :

Invention du métier à tisser semi-automatique par le français Basile …… et amélioration par le …… Joseph …… : début de la programmation !

1823 (première idée) et 1834 (avancée majeure) :

L'…… Charles Babbage (1791-1871) imagine une machine pour résoudre le calcul polynomial. Elle ne sera construite par le Musée des Sciences de Londres qu'en 1991 !



1854 :

L'anglais George …… (1815-1864) publie un article sur la logique binaire. Son algèbre sera étudiée cette année. Les systèmes informatiques l'utilisent !

1936 (premier ouvrage) et 1942 (décryptage d'Enigma) :

Le mathématicien …… Alan …… (1912-1954) est le fondateur des sciences informatiques. Le prix …… récompense l'informaticien de l'année.

Il se suicida en 1954, mangeant une pomme imbibée de cyanure, suite à des accusations d'homosexualité. Une légende (fausse) rapporte que cet épisode est à l'origine du logo d'Apple !

1944 :

L'architecture de John von …… (1903-1957) est utilisée dans la quasi-totalité des ordinateurs modernes. Elle sera étudiée cette année.

1945 (bug) et 1959 (COBOL) :

L'informaticienne américaine Grace Hopper (1906-1992) a inventé le mot « …… » après avoir décoincé un papillon de nuit dans les circuits du calculateur Mark II.

1954 :

Création du premier langage info, le Fortran (FORmula TRANslator), par l'informaticien américain John Backus (1924-2007), nom de l'ancienne salle info des ……

1955 (ordinateur) et 1962 (informatique) :

Invention du mot « …… » par Jacques Perret, sur demande d'*IBM*.

Invention du mot « …… » par Philippe Dreyfus (information-automatique).

1963 :

Ingénieur informaticien chez *Stanford Research Institute*, Douglas Engelbart (1925-2013) a inventé la première ……

1967 (8 pouces), 1976 (5 pouces ¼) et 1980 (3 pouces ½) :

Création de la …… par l'ingénieur américain Alan Shugart (1930-2006). La première version pouvait contenir 80 ko !

1969 :

Création du réseau Arpanet, ancêtre d'……, qui a relié 4 universités américaines. Franck Heart (1929-2018), ……, en est un des pionniers.

1975 (Microsoft) et 1976 (Apple) :

Création de …… par Bill …… (1955) et Paul Allen (1953-2018).
Création de …… par Steve …… (1955-2011), Wozniak et Wayne.

1982 (CD) et 1997 (DVD) :

Invention du CD par Sony & Philips (dont Immink) dont la version standard fut de …… Mo.
Invention du DVD par Philips, Sony, Toshiba et Panasonic, version standard de …… Go.

1985 :

Entreprise française crée en 1972. M. …… découvre l'informatique avec le modèle 520 STE, acheté 3 490 francs à l'époque (950 euros en 2018).

1989 :

Invention du World Wide Web (appelé également ……) par Tim Berners-Lee, anglais (né en 1955) afin que les chercheurs du CERN puissent partager ……

1994 (Amazon) et 1998 (Google) :

Création d'…… par Jeff Bezos (né en 1964), employant 500 000 personnes en 2017.
Création de …… par Larry Page et Sergey Brin (nés en 1973), le G des GAFA.

1997 :

Deep Blue, un superordinateur (32 cores) développé par IBM perd en 1996 mais gagne un an plus tard contre Garry ……, champion du monde d'échecs (3,5 – 2,5).

2001 (Wikipedia) et 2010 (Pinterest) :

Création de l'encyclopédie …… par Jimmy Wales (né en 1966), top 10 des sites.

Lancement de …… (partage d'albums) par Ben Silbermann (né en 1982).

2004 (Facebook) et 2006 (Twitter) :

Création de FB par Mark …… (né en 1984), 2,2 milliards d'abonnés en 2018.

Lancement de ……, avec environ 500 millions de tweets par jour.

2009 (Whatsapp) et 2010 (Instagram) :

Création de ……, appli utilisée par plus d'un milliard d'utilisateurs en 2017.

Lancement d'…… par Kevin Systrom (né en 1983), un milliard d'utilisateurs.

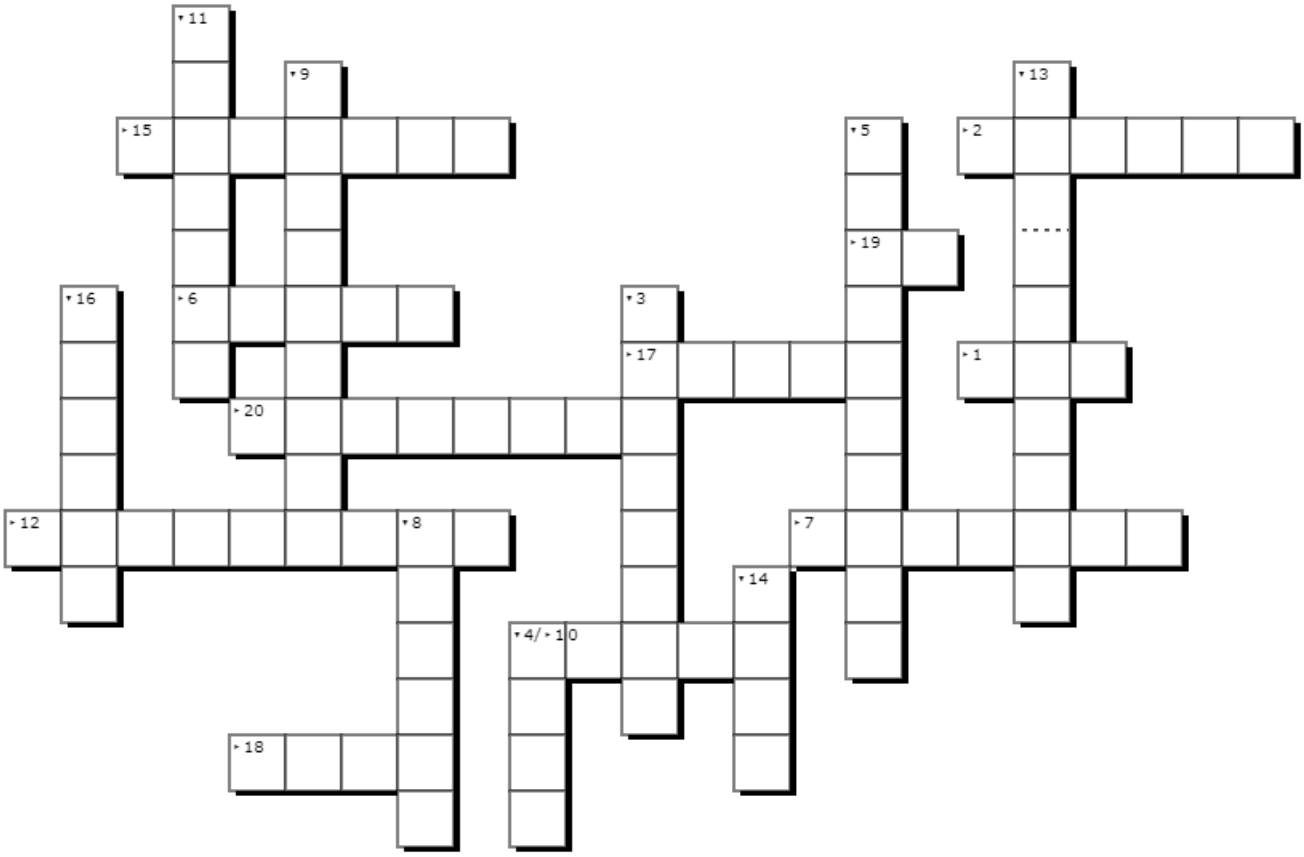
2017 :

Libratus bat 4 joueurs professionnels de …… (en heads-up), dont Dong Kim.

AlphaGo a battu le champion du monde de …… Ke Jie (né en 1997).

2022 : OpenAI développe l'agent conversationnel ChatGPT utilisant l'intelligence artificielle générative.

II – Mots-croisés :



Horizontalement :	**Verticalement :**
1. Insecte en anglais **2.** Firme fondée par Larry Page et Sergey Brin **6.** Marque du premier ordinateur de M. Fontaine **7.** Inventeur de la première calculatrice capable de multiplier **10.** Inventeur de la logique binaire **12.** Ancêtre de la clé USB **15.** Ancêtre d'Internet **17.** Société la plus ancienne des GAFA **18**. Ville où a été inventé le métier à tisser semi-automatique. **19**. Sa capacité standard était de 700 Mo. **20**. IA vainqueur en heads-up des 4 meilleurs joueurs professionnels.	**3.** Champion d'échec battu par Deep Blue en 1997. **4.** Prénom du créateur de Microsoft **5.** Créateur de Facebook **8.** Récompense du 'Nobel' en informatique **9.** Nom de la première machine à calculer **11.** Nom du premier langage informatique **13.** Architecture utilisée dans la quasi-totalité des ordinateurs modernes **14.** Lieu où a été inventé le WEB **16.** Appareil possédant généralement deux clics et une molette.

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 01
Le langage HTML

Les Interactions Homme-Machine (IHM) sur le web définissent les moyens et les outils mis en œuvre afin qu'une personne physique puisse agir (contrôler, communiquer) à distance par l'intermédiaire d'Internet par exemple.

Parmi ces moyens, on trouve les langages HTML, CSS et JavaScript.

I – Généralités :

1) Introduction :

Tous les sites web sont basés sur le langage HTML, il est incontournable et universel aujourd'hui. Il est à la base même du Web. Le langage HTML a été inventé par un certain Tim Berners-Lee en 1991…

Exercice 01 : ouvrir un site Web et visualiser le code HTML de la page.

Le langage HTML signifie ***HyperText Markup Language*** : son rôle est de gérer et organiser le contenu. C'est donc en HTML que vous écrirez ce qui doit être affiché sur la page : du texte, des liens, des images… Vous direz par exemple : « Ceci est mon titre, ceci est mon menu, voici le texte principal de la page, voici une image à afficher, etc. ».

Nous utiliserons comme éditeur de texte NotePad ++ (NPP) : c'est une version améliorée d'un simple bloc note, mais il ajoute la couleur syntaxique (mise en évidence des balises).

Il faudra également utiliser un navigateur, permettant de voir vos créations.

2) Première page en HTML :

Exercice 02 : ouvrir NPP, écrire un court texte comme « Hello World ! » puis sur la seconde ligne « Bonjour le monde ! », enregistrer le fichier sous le nom exercice_02.html dans un dossier « NSI > A – IHM WEB » de vos documents. Ouvrir le fichier avec un navigateur.

« Hello World ! » fait référence à une tradition dans le monde informatique : c'est un des premiers messages mis en place pour tester un programme et cela appartient désormais à l'histoire de l'informatique (1974).

1. Que remarquez-vous quant à la mise en page ? ……

   ……

   ……

En fait, pour créer une page web, il ne suffit pas de taper simplement du texte comme on vient de le faire. En plus de ce texte, il faut aussi écrire ce qu'on appelle des **balises**, qui vont donner des instructions à l'ordinateur comme « aller à la ligne », « afficher une image », etc.

3) Les balises :

Les balises sont invisibles à l'écran pour vos visiteurs, mais elles permettent à l'ordinateur de comprendre ce qu'il doit afficher. Les balises se repèrent facilement. Elles sont entourées de « chevrons », c'est-à-dire des symboles < et >, comme ceci : <balise>

À quoi est-ce qu'elles servent ? Elles indiquent la nature du texte qu'elles encadrent. Elles veulent dire par exemple : « Ceci est le titre de la page », « Ceci est une image », « Ceci est un paragraphe de texte », etc.

- **Balises en paires :**

Elles s'ouvrent, contiennent du texte, et se ferment plus loin. Voici à quoi elles ressemblent :

```
<titre>Ceci est un titre</titre>
```

On distingue une balise ouvrante (<titre>) et une balise fermante (</titre>) qui indique que le titre se termine. Cela signifie pour l'ordinateur que tout ce qui n'est pas entre ces deux balises… n'est pas un titre.

```
Ceci n'est pas un titre <titre>Ceci est un titre</titre> Ceci n'est pas un titre
```

- **Balises orphelines :**

Ce sont des balises qui servent le plus souvent à insérer un élément à un endroit précis (par exemple une image). Il n'est pas nécessaire de délimiter le début et la fin de l'image, on veut juste dire à l'ordinateur « Insère une image ici ».

Une balise orpheline s'écrit comme ceci :

```
<image />
```

Notez que le / de fin n'est pas obligatoire (en HTML 5). On pourrait écrire seulement <image>. Néanmoins, pour ne pas les confondre avec le premier type de balise, les webmasters recommandent de rajouter ce / (slash) à la fin des balises orphelines (obligatoire en XHTML).

- **Les attributs :**

Les attributs sont un peu les options des balises. Ils viennent les compléter pour donner des informations supplémentaires. L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

```
<balise attribut="valeur">
```

À quoi cela sert-il ? Prenons la balise <image /> que nous venons de voir. Seule, elle ne sert pas à grand-chose. On pourrait rajouter un attribut qui indique le nom de l'image à afficher :

```
<image nom="photo.jpg" />
```

L'ordinateur comprend alors qu'il doit afficher l'image contenue dans le fichier photo.jpg.

Dans le cas d'une balise fonctionnant « par paire », on ne met les attributs que dans la balise ouvrante et pas dans la balise fermante. Par exemple, ce code indique que la citation est de Neil Armstrong et qu'elle date du 21 Juillet 1969 :

```
<citation auteur="Neil Armstrong" date="21/07/1969">
C'est un petit pas pour l'homme, mais un bond de géant pour l'humanité.
</citation>
```

Toutes les balises que nous venons de voir sont fictives. Les vraies balises ont des noms en anglais…

II – Les bases d'une page HTML :

1) Structure minimale d'une page HTML :

Exercice 03 : recopier le code source ci-dessous dans NPP, ce code correspond à la base d'une page web en HTML et l'enregistrer au format exercice_03.html.

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>Mon titre à moi</title>
   </head>

   <body>

   </body>
</html>
```

2. Ouvrir la page et chercher « Mon titre à moi ». Où apparaît-il ? Le modifier :

……

Il y a des tabulations au début de certaines lignes pour « décaler » les balises. Ce n'est pas obligatoire et cela n'a aucun impact sur l'affichage de la page, mais cela rend le code source plus lisible. On appelle cela l'indentation.

Vous noterez que les balises s'ouvrent et se ferment dans un ordre précis. Par exemple, la balise <html>est la première que l'on ouvre et c'est aussi la dernière que l'on ferme (tout à la fin du code, avec</html>). Les balises doivent être fermées dans le sens inverse de leur ouverture.

Un exemple :
<html><body></body></html> : correct. Une balise qui est ouverte à l'intérieur d'une autre doit aussi être fermée à l'intérieur.
<html><body></html></body> : incorrect, les balises s'entremêlent.

2) Le doctype :

```
<!DOCTYPE html>
```

La toute première ligne s'appelle le doctype. Elle est indispensable car c'est elle qui indique qu'il s'agit bien d'une page web HTML.

Ce n'est pas vraiment une balise comme les autres (elle commence par un point d'exclamation). Vous pouvez considérer que c'est un peu l'exception qui confirme la règle : elle signifie que la page est écrite en HTML.

3) La balise <html> :

```
<html>
</html>
```

C'est la balise principale du code. Elle englobe tout le contenu de votre page. Comme vous pouvez le voir, la balise fermante </html> se trouve tout à la fin du code !

4) L'en-tête <head> et <body> :

Une page web est constituée de deux parties :

L'en-tête <head> : cette section donne quelques informations générales sur la page comme son titre, l'encodage (pour la gestion des caractères spéciaux), etc. Cette section est généralement assez courte. Les informations que contient l'en-tête ne sont pas affichées sur la page, ce sont simplement des informations générales à destination de l'ordinateur. Elles sont cependant très importantes !

Le corps <body> : c'est là que se trouve la partie principale de la page. Tout ce que nous écrirons ici sera affiché à l'écran. C'est à l'intérieur du corps que nous écrirons la majeure partie de notre code.

5) L'encodage (charset) :

```
<meta charset="utf-8" />
```

Cette balise indique l'encodage utilisé dans votre fichier .html.

Sans rentrer dans les détails, car cela pourrait vite devenir compliqué, l'encodage indique la façon dont le fichier est enregistré. C'est lui qui détermine comment les caractères spéciaux vont s'afficher (accents, idéogrammes chinois et japonais, caractères arabes, etc.).

6) Le titre principal de la page :

```
<title>
```

C'est le titre de votre page, probablement l'élément le plus important ! Toute page doit avoir un titre qui décrit ce qu'elle contient. Il est conseillé de garder le titre assez court (moins de 100 caractères en général).

Le titre ne s'affiche pas dans votre page mais en haut de celle-ci (souvent dans l'onglet du navigateur). Enregistrez votre page web et ouvrez-la dans votre navigateur. Vous verrez que le titre s'affiche dans l'onglet. Il faut savoir que le titre apparaît aussi dans les résultats de recherche, comme sur Google.

7) Les commentaires :

Un commentaire en HTML est un texte qui sert simplement de mémo. Il n'est pas affiché, il n'est pas lu par l'ordinateur, cela ne change rien à l'affichage de la page. Cela sert à vous et aux personnes qui liront le code source de votre page. Vous pouvez utiliser les commentaires pour laisser des indications sur le fonctionnement de votre page.

Quel intérêt ? Cela vous permettra de vous rappeler comment fonctionne votre page si vous revenez sur votre code source après un long moment d'absence. Un commentaire est une balise HTML avec une forme bien spéciale :

```
<!-- Ceci est un commentaire -->
```

Vous pouvez le mettre où vous voulez au sein de votre code source : il n'a aucun impact sur votre page, mais vous pouvez vous en servir pour vous aider à vous repérer dans votre code source (surtout s'il est long).

```
<!DOCTYPE html>
<html>
   <head>
      <!-- En-tête de la page -->
      <meta charset="utf-8" />
      <title>Titre</title>
   </head>

   <body>
      <!-- Corps de la page -->
   </body>
</html>
```

Attention, tout le monde peut voir vos commentaires… et tout votre code HTML !

Exercice 04 : insérer un commentaire dans votre page html.

III – Organiser son texte :

1) Les paragraphes :

La plupart du temps, lorsqu'on écrit du texte dans une page web, on le fait à l'intérieur de paragraphes. Le langage HTML propose justement la balise <p> pour délimiter les paragraphes.

```
<p>Bonjour et bienvenue sur mon site !</p>
```

3. Quel type de balise est-ce ? ……

Exercice 05 : insérer plusieurs paragraphes dans votre page html.

2) Aller à la ligne :

Pour éviter un saut de ligne, il existe une balise « Aller à la ligne » ! :

```
<br />
```

4. Quel type de balise est-ce ? ……

Vous devez obligatoirement mettre cette balise à l'intérieur d'un paragraphe.

Exercice 06 : insérer des sauts de ligne dans votre page html.

3) Les titres :

Lorsque le contenu de votre page va s'étoffer avec de nombreux paragraphes, il va devenir difficile pour vos visiteurs de se repérer. C'est là que les titres deviennent utiles.

<h1> </h1> : signifie « titre très important ». En général, on s'en sert pour afficher le titre de la page au début de celle-ci.
<h2> </h2> : signifie « titre important ».
<h3> </h3> : pareil, c'est un titre un peu moins important (on peut dire un « sous-titre »).
<h4> </h4> : titre encore moins important.
<h5> </h5> : titre pas important.
<h6> </h6> : titre vraiment, mais alors là vraiment pas important du tout.

Exercice 07 : insérer des titres dans votre page html.

4) Mise en valeur :

Au sein de vos paragraphes, certains mots sont parfois plus importants que d'autres et vous aimeriez les faire ressortir. HTML vous propose différents moyens de mettre en valeur le texte de votre page.

  : en italique par défaut (balise <i> déconseillée mais non interdite)
  : en gras par défaut (idem pour la balise <b>)
<u> </u> : souligner (mais il faudra privilégier le css)

Exercice 08 : mettre en valeur des zones de texte dans votre page html.

Le HTML n'a pas été conçu pour gérer la mise en page (c'est possible, mais c'est une mauvaise pratique). Le HTML s'occupe uniquement du contenu et de la sémantique ; pour tout ce qui concerne la mise en page et l'aspect « décoratif » (on parle du « style » de la page), on utilisera le CSS : voir chapitre 02 !

5) Les listes :

Exercice 09 : recopier les deux séries de code ci-dessous.

```
<ul>
   <li>Maths</li>
   <li>Physique-Chimie</li>
   <li>SVT</li>
</ul>
```

```
<ol>
   <li>Français</li>
   <li>Histoire</li>
   <li>anglais</li>
</ol>
```

4. Quelle est l'utilité de la balise ul ? .....
5. Quelle est l'utilité de la balise ol ? .....

IV – Insérer des images ou des vidéos :

1) Généralités :

Le format le plus répandue sur le Web est le JPEG (*Joint Photographic Expert Group*), pouvant comporter plus de 16 millions de couleurs différentes. Les extensions possibles sont .jpg ou .jpeg.

Le JPEG détériore un peu la qualité d'une image, généralement de façon imperceptible : c'est ce qui le rend si efficace pour réduire le poids des photos. Pour une meilleure qualité, il faut privilégier le format PNG.

Pour éviter des problèmes de chargement, le nom des fichiers doit être en minuscule, sans espace, ni accent, comme mon_image.jpg (les espaces sont remplacés par *underscore*).

2) Insérer une image :

<img /> : balise orpheline
src="mon_image.jpg" : attribut obligatoire pour le chemin de l'image
alt="blabla" : attribut facultatif du texte alternatif si l'image ne se charge pas
title="Titre de mon image" : attribut facultatif pour avoir une infobulle
height="21" ou width="21" : attribut facultatif pour la taille de l'image

Exercice 10 : insérer une image dans votre page html.

3) Insérer une vidéos YouTube :

- ✓ Lire une vidéo.
- ✓ Clic droit.
- ✓ Copier le code d'intégration.
- ✓ Coller le code dans votre page HTML.

Exercice 11 : insérer une vidéo dans votre page html.

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 02
Le langage CSS

Les possibilités esthétiques du HTML sont assez rudimentaires mais il est possible de programmer d'assez jolies pages en utilisant des images, des tableaux avec des bordures invisibles…

Cette méthode fut très en vogue dans les années 1990. Cependant, elle est simple à mettre en place pour des petits sites mais cela demande d'énormes travaux en cas de modification des pages (il faut modifier les pages les unes après les autres).

Pour éviter cette lourdeur de programmation, l'idée de séparer la forme et le fond est né en 1996 avec le langage CSS (plutôt appelé les feuilles de style CSS : Cascading Style Sheet).

I – Premier exemple :

Ouvrir le site www.csszengarden.com/tr/fr et changer de style en cliquant sur les liens de la colonne de droite.

6. Le contenu a-t-il changé ? ……

7. Le style a-t-il changé ? ……

A retenir :

Le fichier HTML sert à décrire …… alors que le fichier CSS sert à décrire ……

II – Premier exemple :

Exercice 12 : créer avec NPP deux fichiers, l'un exercice_12.html et l'autre exercice_12.css

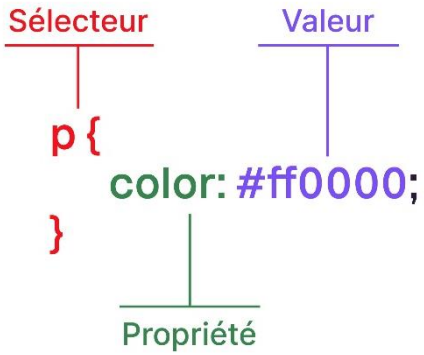
- **Fichier HTML :**

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" href="exercice_12.css" />
</head>

<body>
   <h1>Ceci est un titre</h1>
   <h2>Ceci est un sous-titre</h2>
   <p id="para_1">Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris ?</p>
</body>
</html>
```

La ligne indique quel fichier de style CSS il faut appliquer à cette page.

➢ **Fichier CSS :**

```
h1
{
   text-align: center;
   background-color: red;
}

h2
{
   font-family: Verdana;
   font-style: italic;
   color: green;
}
```



8. Que constatez-vous ?

Pour h1 : ……

Pour h2 : ……

➢ **Modifier le fichier CSS en rajoutant les lignes suivantes :**

```
#para_1
{
   font-style: italic;
   color: blue;
}
```

9. Que constatez-vous pour le premier paragraphe ? ……

Il est donc possible de cibler un paragraphe et pas un autre en utilisant l'id du paragraphe (en CSS l'id se traduisant par le signe #).

Il est possible d'utiliser l'attribut class à la place de l'id. Dans le CSS on utilisera le point **. à la place du #**. L'attribut "class" permet de donner le même nom à plusieurs reprises dans une même page.

Si nous avions eu un deuxième paragraphe, nous aurions pu avoir :

```
<p class="para_1">Voici un deuxième paragraphe</p>
```

mais nous n'aurions pas pu avoir :

```
<p id="para_1">Voici un troisième paragraphe</p>
```

car l'id para_1 a déjà été utilisé pour le 1er paragraphe.

III – Pour aller plus loin :

```
p.exemple { font-family: sans-serif; }
```

Un paragraphe <p class="exemple"> subira ici la règle CSS.

```
table p { text-indent: 0; }
```

Un paragraphe contenu dans un tableau subira ici la règle CSS.

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 03
Les événements HTML

Un événement est une action qui se produit lorsque l'utilisateur réalise une action comme le clic d'une souris, le survol d'une image ou la frappe d'une touche du clavier.

Le premier événement que nous allons étudier est le système hypertexte, avec les liens.

Attention, nous sommes encore dans le Web 1.0 (statique) ;
Le Web 1.5 concerne le Web dynamique (avec JS/PHP)
tandis que le 2.0 est pour le Web participatif.

I – Généralités :

Comme vous le savez, un site web est composé de plusieurs pages. Comment faire pour aller d'une page vers une autre ? À l'aide de liens : il s'agit d'un texte sur lequel on peut cliquer pour se rendre sur une autre page.

On peut faire un lien d'une page a.html vers une page b.html, mais on peut aussi faire un lien vers un autre site (par exemple, http://www.google.com). Dans les deux cas, nous allons voir que le fonctionnement est le même.

II - Un lien vers un autre site :

```
<a href="http://www.google.com">Google</a>
```

Exercice 13 : insérer un lien absolu dans votre page html.

III - Un lien vers une autre page de son site :

Exercice 14 : créer un dossier « exercice_14 », dans lequel se trouvera le dossier « racine » dans lequel se trouvera le dossier « includes ». Créer plusieurs pages html, « page1.html » et « page2.html » dans le dossier « racine », « page3.html » dans le sous-dossier « includes » et « page4.html » dans le dossier parent « exercice_14 ». Ecrire à chaque fois un contenu permettant de les identifier.

Recopier le code suivant dans le fichier page1.html :

```
<a href="page2.html">Page 2</a>
<a href="includes/page3.html">Page 3</a>
<a href="../page4.html">Page 4</a>
```

IV - Personnalisation des liens :

Exercice 15 : recopier les deux liens ci-dessous.

```
<a href="http://www.leschartreux.com" title="Vous ne le regretterez pas !">Chartreux</a>
<a href="http://www.leschartreux.com" target="_blank">Chartreux</a>
```

10. Quel est l'utilité de l'attribut « title » ? .....
11. Quel est l'utilité de l'attribut « target » ? .....

V – Lien sur une image :

Il est possible de faire un lien sur une image en imbriquant les balises <img> et <a>.

```
<a href="page2.html"><img src="mon_image.jpg" /></a>
```

Exercice 16 : insérer une image dans votre fichier.

VI – Mapping :

On peut également faire plusieurs zones de liens sur une image. Nous allons dessiner des zones et attribuer un lien chacune d'entre elles. C'est ce qu'on appelle une « image map » ou « image cliquable » en français.

Exercice 17 : récupérer l'image exercice_17.gif et écrire le code ci-dessous au fur et à mesure.

Nous allons d'abord préciser dans notre balise image que l'on utilise un "map" avec l'attribut usemap en précisant son nom.

```
<img src="exercice_17.gif" usemap="#panneaux" />
```

Nous allons ensuite ajouter une balise <map> avec le nom correspondant. C'est ici que nous définirons nos zones cliquables. Cette balise peut être placée n'importe où dans le corps de la page.

```
<map name="panneaux">
...
</map>
```

Enfin nous allons ajouter à notre map autant de balises <area> que nous avons de zones cliquables. Pour chacune d'elle nous allons préciser les attributs suivants :

- ✓ shape, la forme de la zone cliquable qui peut-être :
  - un cercle (shape="circle"),
  - un rectangle (shape="rect"),
  - un polygone (shape="poly").
- ✓ coords, une suite de coordonnées dessinant la forme choisie.
- ✓ href, le lien vers la page que l'on souhaite afficher lors du clique sur la zone

Nous allons choisir de faire 3 zones cliquables sur mon image :

- ✓ Pour dessiner le rectangle, il faut donner les positions en x (horizontale) et en y (verticale) du point supérieur gauche, et du point inférieur droit. Il y aura donc 4 chiffres pour l'attribut coords.

```
<area shape="rect" coords="x1,y1,x2,y2" href="page-contact.html" />
```

- ✓ Pour dessiner le cercle, il faut donner les positions en x (horizontale) et en y (verticale) de son centre, puis son rayon. Il y aura donc 3 chiffres pour l'attribut coords.

```
<area shape="circle" coords="x,y,r" href="page-faq.html" />
```

- ✓ Pour dessiner le polygone, je vais donner les positions en x (horizontale) et en y (verticale) de tous les points que je souhaite. Il faut au moins 3 points pour dessiner un polygone (un triangle), et nous pouvons en mettre autant que l'on en souhaite. Il y aura donc de 6 à N chiffres pour l'attribut coords.

```
<area shape="poly" coords="x1,y1,x2,y2,x3,y3" href="page-home.html" />
```

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 04
Le langage JavaScript

*Le JavaScript permet une **interaction***
avec l'utilisateur dans une page Web.

I – Historique :

Ce langage a été inventé en 1995 par Brendan Eich (né en 1961, informaticien américain, un des créateurs notamment de la Mozilla Foundation). Il a été mis en place pour Netscape, le tout premier navigateur Web (et ancêtre de Firefox), le JavaScript ayant pour objectif de rendre le contenu d'une page Web dynamique et interactive.

Après l'explosion de la bulle Internet en 2000, l'arrivée du Web 2.0 en 2003 a contribué à la popularisation du JavaScript.

Attention, il ne faut pas confondre le JavaScript avec le langage de programmation orienté objet Java, également présenté en 1995. C'est un langage facilement portable sur plusieurs OS mais il a l'inconvénient d'être un peu lourd (en mémoire et en processeur) à cause de sa machine virtuelle.

II – Blockly Games :

Pour commencer à programmer il faut se familiariser avec certains types de raisonnement, ce que nous allons faire en jouant aux jeux du site Blockly Games :

https://blockly-games.appspot.com

Exercice 18 : puzzle.

Il s'agit dans ce premier jeu de se familiariser avec les manipulations de base utilisées par la suite.

Exercice 19 : labyrinthe.

Réalisez les 10 étapes de ce jeu pour apprendre à utiliser des boucles.

Exercice 20 : oiseau.

Réalisez les étapes de 1 à 7 seulement. Ici vous devez vous familiariser avec les tests et les structures conditionnelles pour guider votre oiseau en fonction des coordonnées X et Y qu'il atteint.

III – JavaScript :

1) Page HTML de référence :

Exercice 20 : nous allons créer une page HTML nommée index.html.

Méthode 1	Méthode 2
<!doctype html> <html lang="fr"> <head> <meta charset="utf-8" /> <title>Programmation JavaScript</title> <script src="mon_programme.js" type="text/javascript" ></script> </head> <body> </body> </html>	<!doctype html> <html lang="fr"> <head> <meta charset="utf-8"> <title>Programmation JavaScript</title> </head> <body> <button onclick="mon_programme()">Cliquez ici</button> </body> </html>

Dans la méthode 1, nous avons ajouté la balise <script>. Cette balise accepte un attribut "src" qui correspond au chemin du fichier JavaScript (extension .js) qui doit être exécuté. Dans notre exemple, notre fichier "JavaScript" sera dans le même dossier que notre fichier "HTML" et se nommera "mon_programme.js". Nous n'aurons plus à modifier cette ligne (sauf si vous décidez de modifier le nom du fichier JavaScript).

Dans la méthode 2, on utilise un bouton HTML qui lancera une fonction mon_programme() qui sera incluse dans le code du fichier HTML.

2) Création du fichier JavaScript :

Exercice 21 :

Méthode 1	Méthode 2
Dans le fichier "mon_programme.js" : document.write("Hello World !");	Dans l'head du fichier HTML : <script> function mon_programme() {document.write("Hello World !");} </script>

Ouvrez le fichier HTML que nous avons créé à l'aide du navigateur Firefox. Observez le résultat.

12. Que constatez-vous ? ……

Il est important de noter qu'en JS, une ligne de code doit se terminer par un **point-virgule**.

3) Introduction des variables :

En informatique, une variable est un symbole qui associe un nom à une valeur. Le nom doit être unique. En JavaScript, on utilise le mot clé « var » afin de déclarer une variable.

Exercice 22 : recopier le code ci-dessous.

```
var point_de_vie;
point_de_vie=15;
document.write(point_de_vie);
```

Enregistrer le fichier et ouvrir la page HTML.

13. Est-ce le même fonctionnement qu'en Python ? ……

Exercice 22 bis : modifier le programme ainsi.

```
var point_de_vie=15;
document.write(point_de_vie);
```

Enregistrer le fichier et ouvrir la page HTML.

14. Que constatez-vous ? ……

Exercice 22 ter : on peut aussi améliorer le message avec le code suivant.

```
var point_de_vie=15;
document.write("point_de_vie a pour valeur ", point_de_vie);
```

15. Quelle est la différence entre une chaîne de caractère et une variable ?

……

……

En JavaScript, les variables peuvent être des strings (chaîne de caractère), des boolean (vrai/faux), des numbers (des nombres entiers ou décimaux).

Le JavaScript est un langage faiblement typé, le programmeur n'a pas besoin de préciser le type de variable.

La fonction typeof() renvoie le type de la variable qui a été passée en argument.

Exercice 23 : recopier le code ci-dessous.

```
var a=4;
document.write ("a a pour valeur ", a, ". Elle est de type ", typeof(a), "<br />");
var b="Hello";
document.write ("b a pour valeur ", b, ". Elle est de type ", typeof(b), "<br />");
var c=true;
document.write ("c a pour valeur ", c, ". Elle est de type ", typeof(c), "<br />");
var d=3.14;
document.write ("d a pour valeur ", d, ". Elle est de type ", typeof(d), "<br />");
var e;
document.write ("e a pour valeur ", e, ". Elle est de type ", typeof(e), "<br />");
```

16. Que remarquez-vous pour e? ……

Exercice 23 bis : recopier le code ci-dessous.

```
var a=4;
document.write ("a a pour valeur ", a);
document.write ("b a pour valeur ", b);
var c="Hello";
document.write ("c a pour valeur ", c);
```

17. Que remarquez-vous ? ……

Exercice 23 ter : faire un programme JavaScript qui crée 2 variables contenant pour l'une le texte « Bonjour, j'ai » et pour l'autre le nombre 17. Ensuite le programme doit afficher le contenu de ces variables pour qu'on voit : « Bonjour, j'ai 17 ans bientôt ! ».

4) Les boites de dialogue :

JavaScript propose des boîtes de dialogue par défaut qui permettent d'interagir avec l'utilisateur. Il en existe 3 types. Les **alert**() qui permettent d'afficher un message. Les **confirm**() qui permettent de récupérer une valeur booléenne et les **prompt**() qui permettent de récupérer une valeur textuelle.

Exercice 24 : recopier le code ci-dessous.

```
alert("Une alerte simple");
var myText="Une alerte avec le message dans une variable";
alert(myText);
var myNumber=13;
alert("La variable contient la valeur " + myNumber); // La virgule ne fonctionne pas
```

18. Lors de la dernière ligne, on effectue une concaténation. Donner la définition :

……

La boîte de dialogue « confirm() » ressemble à alert() mais elle propose deux boutons.

Exercice 24 bis : modifier le programme 24 en insérant une confirm().

19. Indiquez le nom des deux boutons : ……

Si on clique sur le bouton de gauche, cela renvoie le booléen « true » alors que le bouton de droite renvoie « false ». On peut ainsi commencer par mettre en place une condition (un if) :

Exercice 24 ter : recopier le code ci-dessous.

```
if (confirm("Vous désirez vraiment quitter?") == true) {
        alert("oui");
}
else {
        alert("non");
}
```

Nous allons maintenant étudier la méthode qui permet à l'utilisateur de rentrer des valeurs au clavier : la méthode prompt(). Nous allons utiliser une structure de la forme : var maVariable = prompt (message). En réponse à la méthode prompt, le navigateur affichera une fenêtre avec : un bouton OK, un bouton Annuler, un message et une zone de saisie. L'utilisateur va alors saisir (au clavier) du texte dans la zone de saisie. La validation avec le bouton OK permettra d'attribuer le texte entré par l'utilisateur à la variable maVariable. Au cas où l'utilisateur ne rentrerait rien ou appuierait sur Annuler, on aurait alors maVariable = null (pas de valeur).

Exercice 25 : voici un premier exemple

```
var prenom=prompt("Quel est votre prénom ?");
document.write ("Bonjour ", prenom, ", vous allez bien ?");
```

20. Quel mot utilise-t-on en Python à la place du « prompt » ? ……

Exercice 25 bis : améliorer le programme afin qu'une action s'exécute lorsque l'utilisateur clique sur annuler (if/else).

Exercice 26 : une machine à additionner

```
document.write ("Nous allons additionner 2 nombres, a et b <br />");
var a=prompt("Entrer a ");
var b=prompt("Entrer b ");
var resultat=a+b;
document.write ("Résultat ",a," + ",b," = ",resultat);
```

21. Que remarquez-vous (un typeof() peut vous aider) ? ……

……

Pour que notre programme fonctionne, il faut "transformer" notre chaîne (variable de type string) en nombre (variable de type integer). Nous allons faire du transtypage.

22. Qu'est-ce que le transtypage ? ……

……

Exercice 26 bis : utilisation de la méthode parseInt()

```
document.write ("Nous allons additionner 2 nombres, a et b </br>");
var as=prompt("Entrer a ");
var bs=prompt("Entrer b ");
var a=parseInt(as);
var b=parseInt(bs);
var resultat=a+b;
document.write ("Résultat ",a," + ",b," = ",resultat);
```

La méthode parseFloat transtype en flottant, tout en supprimant un caractère non numérique.
Par exemple, parseFloat(« 12.34Bonjour56)=12.34

5) Les tests conditionnels : if {} else {} :

si (condition vraie) fait { instructions }
sinon (sous entendu la condition est fausse) fait { instructions }

Exercice 27 : un premier exemple

```
var a=5;
if (a==5)
{
	document.write("a est égale à 5");
}
else
{
	document.write("a n'est pas égale à 5");
}
```

Voici la liste des opérateurs de comparaison :

Égalité	==	Différent de	!=
Strictement supérieur	>	Strictement inférieur	<
Supérieur ou égal	>=	Inférieur ou égal	<=
Égalité stricte (sans transtypage)	===		

Notons qu'il est aussi possible de faire des combinaisons de tests avec le ET logique (&&) ou le OU logique (||) comme le montre l'exemple suivant :

Exercice 27 bis : avec le ET logique

```
var valeur=prompt("Entrez un chiffre entre 5 (exclu) et 10 (inclus)");
if ((valeur>5) && (valeur<=10))
{
	document.write("Bravo, vous avez réussi !");
}
else
{
	document.write("Raté");
}
```

Exercice 27 ter : modifier le programme 27 bis afin qu'il vérifie si la valeur est plus petite que 5 OU plus grande ou égale à 10.

Exercice 27 quater : écrire un programme en JavaScript demandant en entier naturel n non nul (prompt) et affichant la somme des entiers de 1 à n (alert).

6) Les fonctions en JavaScript

Pour définir une fonction en JavaScript il faut utiliser le mot clé function. Nous allons avoir une structure de la forme :

```
function maFonction(paramètre1, paramètre2,…)
{
	…instructions..
	return y ;
}
```

Exercice 28 : un premier exemple

```
function CALCUL(x)
{
	var y;
	y=3*x+2;
	return y;
}
document.write ("y=3x+2 avec x = 4 <br />");
document.write ("Résultat : y = ", CALCUL(4));
```

Il est d'usage en programmation de décaler les contenus des fonctions afin de rendre le code plus lisible. Ceci s'appelle **l'indentation**.

CALCUL(4) appelle la fonction CALCUL avec un paramètre égal à 4. Dans le document.write de la dernière ligne, CALCUL(4) est "remplacé" par la valeur retournée par la fonction CALCUL.

Exercice 28 bis : un exemple plus évolué

```
function CALCUL(x)
{
	var y;
	y=3*x+2;
	return y;
}
var valeur = prompt ("y=3x+2, entrez la valeur de x");
document.write ("y=3x+2 avec x = ", valeur, "</br>");
document.write ("Résultat : y = ", CALCUL(valeur));
```

Ici la valeur du paramètre de la fonction calcul est entrée par l'utilisateur, elle porte le nom de « valeur ». Pour le reste, le code est identique au précédent.

Comme déjà dit précédemment, une fonction peut très bien attendre plusieurs paramètres.

Exercice 28 ter : un exemple avec deux paramètres

```
function CALCUL (x,b)
{
	var y;
	y=3*x+b;
	return y;
}
var valeur1 = prompt ("y=3x+b, entrez la valeur de x");
var valeur2 = prompt ("y=3x+b, entrez la valeur de b");
valeur1=parseInt(valeur1);
valeur2=parseInt(valeur2);
document.write ("y=3x+b avec x = ", valeur1, " et b = ", valeur2, "</br>");
document.write ("Résultat : y = ", CALCUL(valeur1, valeur2));
```

23. Que se passe-t-il, à votre avis, si on enlève les deux lignes avec parseInt() ?

……

Évidemment, le paramètre de la fonction peut-être de type string. Il faut aussi savoir que le return n'est pas obligatoire (une fonction peut ne rien renvoyer, juste "faire quelque chose").

Exercice 29 : une fonction sans return

```
function afficheNom(nom)
{
	document.write("Bonjour", nom);
}
afficheNom("Toto");
```

Un simple afficheNom("Toto"); vous permet d'appeler la fonction afficheNom, qui ne renvoie rien, mais qui affiche du texte.

Pour terminer cette partie sur les fonctions et le JavaScript, étudions la différence entre les variables locales et les variables globales (on parle de la portée d'une variable).

Exercice 30 : exemple avec une variable locale

```
function maFonction()
{
        var i=10;
}
document.write ("Voici la valeur de la variable i : ", i);
```

En effet, la variable i a été définie dans une fonction, donc elle n'existe pas en dehors de cette fonction, c'est une variable locale. Une variable définie en dehors d'une fonction est appelée variable globale, elle est accessible partout dans le programme. Cette confusion peut être à l'origine de bugs difficilement détectables.

Exercice 30 bis: exemple avec une variable globale

```
var i=5 ;
function changerLaValeurDei (x)
{
        var i=5*x;
}
var a=prompt("Entrer une valeur pour x");
changerLaValeurDei(a);
document.write ("Voici la nouvelle valeur de la variable i : ", i);
```

24. Que remarquez-vous ? ……

Afin d'indiquer que nous souhaitons utiliser une variable globale dans une fonction, il faut enlever le mot clé « var ».

Exercice 30 ter : modifier l'exercice 30 bis afin qu'il fonctionne.

7) Les commentaires en JavaScript :

Les programmes étant de plus en plus complexes, il convient de joindre des commentaires. Ainsi, une autre personne peut suivre votre code et vous pouvez également vous même remanier celui-ci des mois après son écriture.

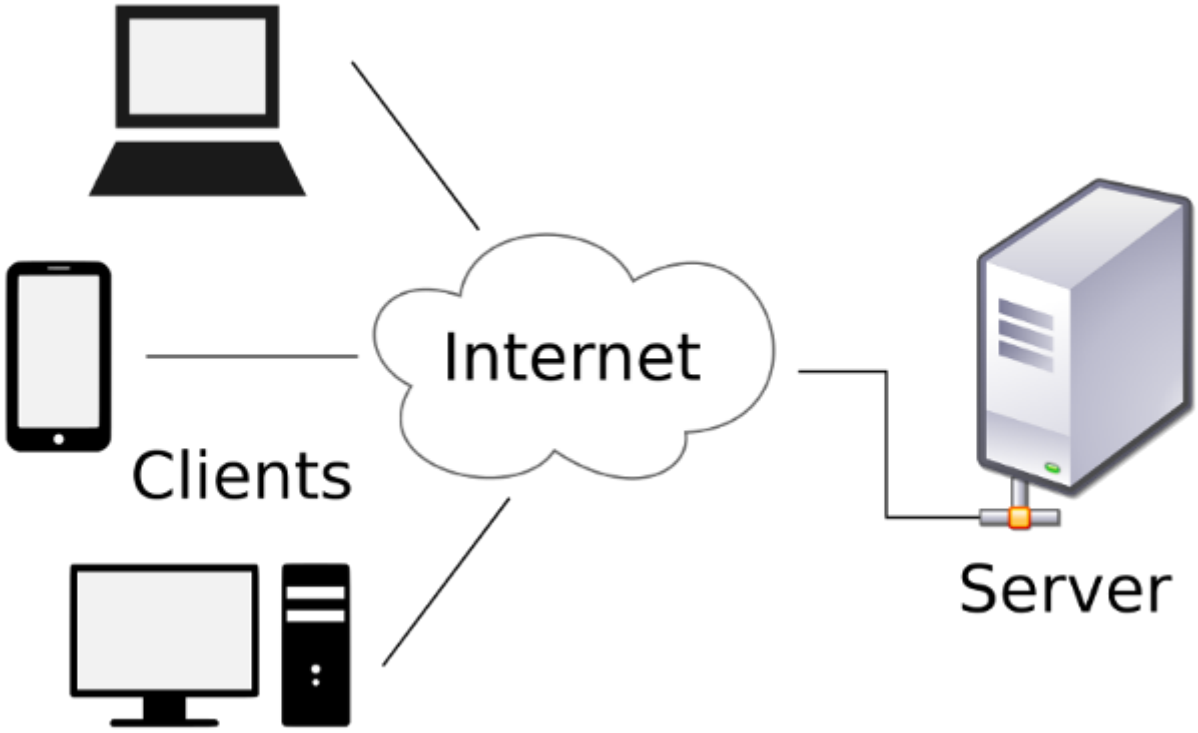
En JavaScript, il existe 2 façons d'écrire des commentaires :

```
// Ceci est un commentaire sur une ligne

/* Ceci est un commentaire
 sur plusieurs lignes */
```

Exercice 30 quater : rajouter deux commentaires dans le programmes 30 ter.

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 05
Interaction client-serveur

I – Définitions :



- ✓ **Requête** : instruction contenant une ou plusieurs demandes.
- ✓ **Client** : généralement l'ordinateur sur lequel on se connecte à Internet, c'est lui qui envoi les requêtes.
- ✓ **Serveur** : machine sur laquelle est la page web, c'est elle qui répond aux requêtes.
- ✓ **Requête exécutée au niveau client** : par exemple, avec des événements JavaScript (alert, confirm ou prompt), tout se passe au niveau client. Par conséquent, le serveur (là où est hébergé la page web) n'a aucune connaissance de l'issue des événements. La réponse est rapide car il n'y a pas de délais entre l'instant où on exécute la requête et l'instant où on reçoit la réponse.
- ✓ **Requête exécutée au niveau serveur** : quand on valide un formulaire (voir chapitre 06), c'est le serveur qui traite les informations avant d'exécuter un code-réponse qui va être envoyé au client.

II – Les requêtes clients HTTP / HTTPS :

1) Syntaxe générale :

Méthode **En-tête de requête** **<saut de ligne>** **Corps de requête**

2) Méthode :

Il existe plusieurs méthodes pour une requête http. Les plus importantes sont :

- ✓ GET : c'est la méthode la plus courante pour demander une ressource au serveur. Cette requête ne modifie pas la ressource, et il doit être possible de la répéter.
- ✓ HEAD : cette méthode ne demande que des informations sur la ressource, sans demander la ressource elle-même.
- ✓ POST : cette méthode est utilisée pour soumettre des données en vue d'un traitement, comme pour un formulaire.

Il y en a d'autres : DELETE, PUT…

3) L'en-tête :

C'est un peu le même principe que les balises « <meta> » de l'HTML : elles peuvent prendre beaucoup de valeur, comme par exemple :

- ✓ Host : <nom du site>
- ✓ Connection : Close (fermer la connexion après la réponse) ou Keep-Alive (conserver la connexion après la réponse).
- ✓ Content-type : <type de média du corps de la requête>
- ✓ Content-Length : <longueur du corps de la requête>

4) Le corps :

Il peut contenir par exemple le contenu d'un formulaire HTML.

Exercice 31 : détailler les requêtes ci-dessous.

HEAD /fichier.html http/1.1 Host : www.site.com Connection : Close <saut de ligne>	GET /fichier.html http/1.1 Host : www.site.com User-Agent : Mozilla/5.0 Accept : text/html Connection : Close <saut de ligne>	POST /fichier.php http/1.1 Host : www.site.com Connection : Keep-Alive Content-Type : application/x-www-form-urlencoded Content-Lengh : 20 <saut de ligne> nom=PLESSY&prenom=JB
…… …… …… …… ……	…… …… …… …… ……	…… …… …… …… ……

5) Cas du HTTPS :

Certaines données sont sensibles (par exemple, le numéro de carte bancaire). Il est donc important, quand on souhaite les transmettre via le protocole HTTP, que ces données soient chiffrées, c'est-à-dire transformées, pour ne pas être comprises par d'éventuel(le)s pirates informatiques, qui les intercepteraient entre le client et le serveur.

Les pages chiffrées sont reconnaissables en regardant la barre d'adresse. L'URL commence toujours par HTTPS, signifiant **HyperText Transfert Protocol Secure** ; c'est la combinaison du HTTP et d'une couche de chiffrement (comme le SSL, devenu le TLS, deux protocoles de sécurisation des échanges sur Internet).

III – Les réponses serveurs HTTP :

1) Syntaxe générale :

Ligne de statut
En-tête de réponse
<saut de ligne>
Corps de réponse

2) Ligne de statut :

La ligne de statut est composée :

- ✓ De la version HTTP du serveur.
- ✓ Du code retourné (par exemple 200 si le document a été trouvé, 404 si le document n'a pas été trouvé).

- ✓ Du texte associé à l'erreur (par exemple OK pour 200 et NOT FOUND pour 404).

3) L'en-tête :

Les en-têtes peuvent contenir ici aussi beaucoup d'informations, comme :

- ✓ L'horodatage de génération de la réponse.
- ✓ Le modèle de serveur HTTP.
- ✓ Le type de média.
- ✓ La longueur du corps.

Exercice 32 : exemple de réponse

```
HTTP/1.1 200 OK
Date: Thu, 15 feb 2019 12:02:32 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4
Connection: close
Transfer-Encoding: chunked <!-- les données sont transférées par blocs sans connaître la taille totale du corps de la réponse -->
Content-Type: text/html;
charset=ISO-8859-1

<!doctype html> <html lang="fr"> <head> <meta charset="utf-8"> <title>Voici mon site</title> </head> <body> <h1>Hello World! Ceci est un titre</h1> <p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris ?</p> </body> </html>
```

PARTIE A - INTERACTION ENTRE L'HOMME ET LA MACHINE SUR LE WEB
Chapitre 06
Formulaire d'une page Web

I – Généralités :

Un formulaire est un des principaux moyens d'interactions entre un utilisateur et un site Web. Un formulaire HTML utilise généralement le JavaScript (pour vérifier l'entrée des données avant l'envoi) et le PHP (pour traiter les données).

Ici, nous nous contenterons uniquement d'étudier la mise en place du formulaire HTML (sans JS et PHP) avant de voir la différence entre les méthodes GET et POST.

II – Formulaire HTML :

Exercice 33 : ouvrir le code du fichier exercice_33 et répondre aux questions

25. Quelle balise permet de créer un formulaire HTML ? ……
……
……
26. Que fait le mot clé « required » ? ……
……
……
27. Que fait l'option « pattern » ? ……
……
……
28. Que fait l'option « placeholder » ? ……
……
……
29. Que fait le mot clé « checked » ? ……
……
……
30. Que fait le mot clé « selected » ? ……
……
……
31. Que fait le type « date » ? ……
……
……

III – Transmission des informations au serveur :

La plupart du temps, c'est un fichier PHP qui traite les informations. Nous allons simplement étudier la différence entre les méthodes GET et POST qui appellent le fichier PHP (non étudié en NSI).

Exercice 34 : requête GET

32. Ouvrir le site https://get.nsi.xyz
33. Saisir deux nombres et cliquer sur « additionner ».
34. Quelle est alors l'adresse du site web ?

    ……
35. Le résultat du calcul est-il juste ? ……
36. En affichant le code source (CTRL + U), vois-tu des scripts PHP ? ……
37. Où est le code PHP ? ……
38. Arrives-tu à saisir un mot dans les cases à remplir ? ……
39. Pour quelle raison ce n'est pas possible ? ……
40. En réinitialisant la page, effectuer les additions suivantes (interdiction de faire du calcul mental – chut, ne pas le dire à ton professeur de maths) :

1er nombre	8	42	255	1394	1991	8086	1.61803
2ème nombre	128	80	404	1337	4004	2001	3.14159
Somme							

41. Pour tenter de contourner cette restriction, est-il possible de passer directement par la barre d'adresse ? ……

Exercice 35 : requête POST

42. Ouvrir le site https://post.nsi.xyz
43. Saisir deux nombres et cliquer sur « additionner ».
44. Quelle est alors l'adresse du site web ?

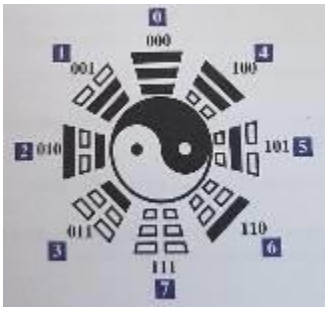
    ……
45. Quelle requête faut-il favoriser quand on manipule des données sensibles ?

    ……

    ……

PARTIE B – REPRESENTATION DES DONNEES (TYPES SIMPLES)
Chapitre 07
Bases 2, 10 et 16

I – Prémices du binaire :



Trois millénaires avant J.-C., l'empereur Chinois **Fou-Hi** avait pour symbole un octogone à trigramme contenant les 8 premiers nombres représentés sous forme binaire par des traits uniforme ou discontinu à la place des 0 et des 1… Ce ne sera qu'en 1679 que Leibniz découvre et analyse ce symbole s'en servant pour mettre au point un algorithme binaire. C'est la plus ancienne forme logique s'apparentant à l'informatique telle qu'on la connait.

II – Différents systèmes :

Vers la fin des années 1930, Claude **Shannon** (père fondateur de la théorie de l'information) démontra qu'à l'aide de « contacteurs » (interrupteurs) fermés pour « vrai » et ouverts pour « faux » il était possible d'effectuer des opérations logiques en associant le nombre 1 pour « vrai » et 0 pour « faux ».

Ce codage de l'information est nommé « **base binaire** » (ou « **base 2** »). C'est avec ce codage que fonctionnent les ordinateurs. Il consiste à utiliser deux états (représentés par les chiffres 0 et 1) pour coder les informations : ce sont les bits (de binary digit).

Un octet (byte en anglais) est un ensemble de ……

Exemple : ……

Le LSB (Least Significant Bit ou bit de poids le plus faible) est le bit situé le plus à droite tandis que le MSB (Most Significant Bit ou bit de poids le plus fort) est le bit situé le plus à gauche.

L'homme calcule depuis 2000 ans avant Jésus-Christ avec 10 chiffres (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), on parle alors de « **base décimale** » (ou « **base 10** »). Toutefois dans des civilisations plus anciennes ou pour certaines applications actuelles d'autres bases de calcul ont et sont toujours utilisées :

- ✓ Base sexagésimale (60), utilisée par les Sumériens. Cette base est également utilisée pour ……
  ……
- ✓ Base vicésimale (20), utilisée par les Mayas ;
- ✓ Base duodécimale (12), utilisée par les anglo-saxons dans leur système monétaire jusqu'en 1960 : un « pound » représentait vingt « shilling » et un « shilling » représentait douze « pences ». Cette base est également utilisée dans ……
- ✓ Base quinaire (5), utilisée par les Mayas, notamment avec ……
- ✓ Base quaternaire (4), utilisé par les Shadocks (0-3) et par des héros de Walter …… et de Peyo (……).
- ✓ *Cette base fut surnommée « bi-binaire » par le mathématicien et chanteur Bobby Lapointe (1922-1972).*

III – Trois bases en informatique :

- ✓ Base 2 (binaire) : ……
- ✓ Base 10 (décimal) : ……
- ✓ Base 16 (hexadécimal) : ……

On voit un premier problème apparaître : si les bases ont des chiffres communs, comment savoir en quel base est écrit un nombre ? Quand vous écrivez dans une base autre que la base 10, il est important de bien le préciser, par exemple en l'indiquant en indice : $00101010_2 = 42_{10} = 42 = 2A_{16}$.

IV – Quelques conversions :

1) Tableau récapitulatif :

Décimal	0	1	2	3	4	5	6	7
Binaire								
Hexadécimal								
Décimal	8	9	10	11	12	13	14	15
Binaire								
Hexadécimal								
Décimal	16	17	30	31	32	33	62	63
Binaire								
Hexadécimal	1 0							
Décimal	64	65	126	127	128	129	254	255
Binaire								
Hexadécimal								

2) Conversion binaire → décimal :

On souhaite convertir 00011010 en décimal. Nous allons utiliser le tableau suivant :

Valeur (décimale)	128	64	32	16	8	4	2	1
Puissance de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Bits								

A chaque valeur 1 trouvée, il suffit de prendre la valeur associée en décimal et de faire la somme des valeurs obtenues. On obtient : ……

Exercice 01 : convertir 01101101 en décimal, puis vérifier votre résultat sous Python avec la fonction int('nombre',2)

……

……

3) Conversion décimal → binaire :

Il existe deux méthodes : la division par 2 avec le reste (il sera possible de l'aborder en Python plus tard) et les puissances descendantes de 2 avec la soustraction.

On souhaite convertir 122 en binaire. Nous allons utiliser le tableau suivant :

Valeur (décimale)	128	64	32	16	8	4	2	1
Puissance de 2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Bits								

A chaque décimal utilisée, on associe le bit « 1 ». On obtient : ……

Exercice 02 : convertir 93 puis vérifier votre résultat sous Python avec la fonction bin(nombre)

……

……

4) Conversion hexadécimal → décimal :

On souhaite convertir C921 en décimal. Nous allons utiliser le tableau suivant :

Valeur (décimale)	4 096	256	16	1
Puissance de 16	16^3	16^2	16^1	16^0
Caractère hexa				

Pour chaque colonne, on additionne la valeur en décimal multipliée par le caractère noté dans la 3e ligne. On obtient : ……

Exercice 03 : convertir 18A2F en décimal, puis vérifier votre résultat sous Python avec la fonction int(nombre,16)

……

……

5) Conversion décimal → hexadécimal :

On souhaite convertir 493 en hexadécimal. Nous allons utiliser le tableau suivant :

Valeur (décimale)	4 096	256	16	1
Puissance de 16	16^3	16^2	16^1	16^0
Caractère hexa				

Il faudra effectuer des divisions successives en ignorant le reste du résultat après la virgule. On obtient : ……

Exercice 04 : convertir 15554 en hexadécimal, puis vérifier votre résultat sous Python avec la fonction hex(nombre)

……

……

PARTIE B – REPRESENTATION DES DONNEES (TYPES SIMPLES)
Chapitre 08
Représentation binaire des entiers relatifs

I – Le binaire signé :

Une première idée serait d'utiliser un bit (celui de gauche) pour le signe et les autres bits pour la valeur absolue.

Ainsi, le bit de poids fort sera un 0 pour un entier positif et 1 pour un entier négatif.

*Exemple : 89_{10} s'écrit **0**1011001_2 et -89_{10} s'écrit **1**1011001_2*

Pour une mémoire de 8 bits (256 possibilités), on peut donc coder les nombres de -127 à 127. Un premier problème se pose car 0 est codé à la fois par 0000 0000 et par 1000 0000 !

Exercice 05 : donner le binaire du nombre -56

……

……

Exercice 06 : en 2014, le compteur Youtube de Gangnam Style (Psy) s'est bloqué à 2 147 483 647 de consultations. Pourquoi ? Quelles solutions ont été envisagées par Google® ?

……

……

……

……

……

……

Exercice 06 bis : combien de bits sont utilisés pour les teintes chez Castorama© ?

……

Cependant, cette première idée est très peu utilisée, car :

- ✓ Le traitement spécifique du signe coûte cher en circuit électronique.
- ✓ Le temps de calcul est rallongé.
- ✓ Les additions sont alors fausses.

Exercice 07 : l'addition en binaire s'effectue comme en base 10 avec une retenue (1 + 1 = 10, soit 0 de résultat et on retient 1). Calculer, en binaire et en décimale, l'addition 4 + (-2)

Décimal :	Binaire (8 bits)
4	……
+ (-2)	+ ……
----------	----------
= ……	= ……

II – Le complément à deux :

Voici la méthode pour trouver le complément à 2 de n :

- ✓ **Si n > 0, on utilise le binaire signé.**
- ✓ **Si n<0, on code |n| en binaire signé, puis on prend le complément à 1 (on remplace tous les 0 par 1 et tous les 1 par 0) avant d'additionner 1 (cette dernière étape s'appelle le complément à 2).**

Exemple pour -57_2 :

- ✓ *57 en binaire s'écrit 00111001_2.*
- ✓ *On prend le complément à 1 et on obtient 11000110_2.*
- ✓ *On ajoute 1 au binaire obtenu, soit 11000111_2 !*

Pour les extrêmes, 0 en décimal est codé 0000 0000 et 127 est codé 0111 1111.
Pour les nombres négatifs, nous pouvons aller de -1 (1111 1111) à -128 (1000 0000).

Cette nouvelle méthode résout le problème du double zéro et des additions.

Exercice 08 : réaliser l'addition 113 -90

Décimal :	Binaire :
113	……
+ (-90)	+ ……
----------	----------
= ……	= ……

Si la machine sait additionner, elle peut alors soustraire (a – b = a + (-b)), multiplier (n x p = n + n +n + … + n) et diviser (n / p = n – p – p – p - … - p jusqu'à obtenir 0 et le quotient est le nombre de soustractions).

Exercice 09 : répondre aux questions suivantes avec une mémoire de 8 bits

1. Trouver l'écriture en base 10 du nombre 11010100 (binaire complément à 2) :

   ……

   ……

   ……

2. Effectuer l'addition -84 + 29 (en complément à deux) :

   ……

   ……

   ……

   ……

   ……

   ……

PARTIE B – REPRESENTATION DES DONNEES (TYPES SIMPLES)
Chapitre 09
Représentation approximative des nombres réels

I – Exemples :

Exercice 09 sous Python, écrire a = 0.1, b = 0.2 et c = 0.3 puis tester l'égalité a + b == c

3. Que remarquez-vous ? D'où peut provenir le souci ?

……

……

Le 25 février 1991, lors de la première guerre du Golfe, un missile Patriot américain a raté l'interception d'un missile Scud irakien, ce dernier provoquant la mort de 28 personnes.
La commission parlementaire a mis en évidence l'erreur : l'horloge interne du missile mesurait le temps en dixième de seconde et a été codé sur un registre de 24 bits.
Or, 0,1 seconde en base 2 s'écrit 0,000 1100 1100 1100 1100 1100, ce qui a induit une erreur binaire de 0,000 0000 0000 0000 0000 0000 1100 1100…, soit 0,000 000 095 seconde en base 10 pour 0,1 seconde. En multipliant ce décalage par 100 heures (temps entre la mise en marche du système et le lancement du missile), on obtient un décalage de l'horloge interne avec le temps réel de 0,34 seconde.
Or, un missile Scud vole à environ 1 676 m/s, soit donc plus de 500 mètres en 0,34 seconde !
En virgule flottante simple (double) précision, le décalage aurait été de 9 m (30 nm).

II – Représentation des nombres réels à virgule fixe :

1) Partie décimale d'un nombre :

Tout nombre décimal possède une partie décimale pouvant s'écrire comme somme de produits des chiffres de 0 à 9 par des puissances de 10 avec exposants entiers négatifs.

Exemple : $12,5024 = 1.10^{1} + 2.10^{0} + 5.10^{-1} + 0.10^{-2} + 2.10^{-3} + 4.10^{-4}$

Procédons de manière identique en base 2 pour la partie décimale sur un exemple simple :

*Exemple : 0,**4375** = 0,25 + 0,125 + 0,0625 =1/4 + 1/8 + 1/16, ce qui donne $0,(\mathbf{0111})_2$*

Exercice 10 : convertir 0,6875 en base 2

……

2) Nombre à virgule fixe :

Certains anciens calculateurs sur 8 bits ou plus récemment de petits processeurs dont le CPU (processeur) est dépourvu de FPU (unité de calcul en virgule flottante), ne peuvent travailler sur des nombres à virgule flottante étudiés dans la troisième partie qui suit, ou alors au prix d'un fort ralentissement de leur fonctionnement : ils utilisent alors la virgule fixe, qui permet en réalité de ne travailler qu'avec des entiers et ainsi gagner en rapidité.
Le principe revient simplement à coder séparément, la partie Entière et la partie Décimale, en binaire naturel, puis à les juxtaposer.

Exemple :

- ✓ *5 est codé (101)*
- ✓ *0,1875 est codé 0,(0011)*
- ✓ *$5{,}1875_{10} = 101{,}0011_2$*

Exercice 11 : convertir 13/4 puis 123,34375 en base 2

……

……

……

III – Représentation des nombres réels à virgule flottante :

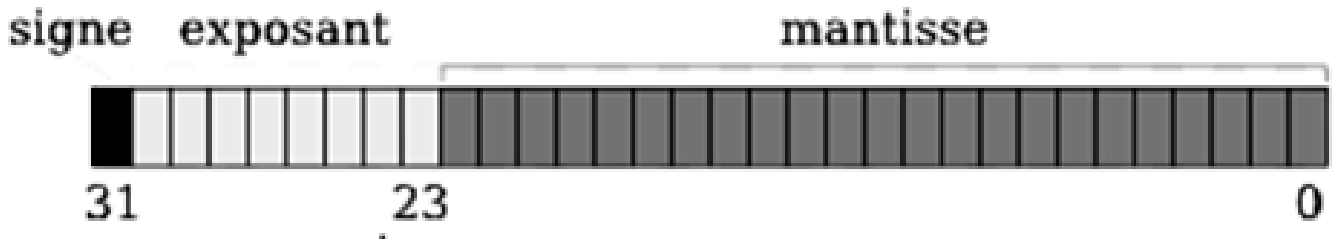
1) Principe :

Compte tenu de la taille finie de sa mémoire, l'ordinateur ne peut manipuler et stocker toute la précision de certains réels comme par exemple les irrationnels, mais aussi ceux dont le nombre de caractères est trop important (comme pour 0,1). Il possède donc sa propre représentation interne des réels en notation scientifique, désormais appelés nombres à virgule flottante. Cette norme standardisée est référencée : IEEE-754 (qui date de 1985).

Elle se décline en 2 versions : 32 bits pour la simple précision ou 64 bits pour le codage double précision.

Nous allons étudier sur un exemple le codage 32 bits simple précision, dans lequel chaque nombre est décomposé en 3 parties distinctes qui sont :

1 bit de signe (S) / 8 bits d'exposant (E) / 23 bits de mantisse (M) = total 32 bits :



2) Méthode :

Rappels sur l'écriture scientifique en puissance de 10 :

- ✓ 3 = ……
- ✓ 527 = ……
- ✓ 0,0000000123 = ……

De même, on peut utiliser, en binaire, une écriture « scientifique » utilisant les puissances de 2 :

- ✓ 3 = 11 = ……
- ✓ 527 = ……
- ✓ 0,34375 = ……

Tous les nombres binaires peuvent s'écrire sous la forme S 1,MMMMM… x 2^E

3) Premier exemple (hors programme) :

On cherche à coder le nombre réel (-35,625) sur 32 bits (simple précision) :

1) Le bit de signe est mis à 1, puisque négatif : S = 1
2) On sépare le nombre en deux : la partie entière (35) et la partie décimale (0,625)
3) On convertit la partie entière en nombre binaire : $(35)_d = (100011)_b$
4) On convertit la partie décimale en nombre binaire : $(0{,}625)_d = 0,(101)_b$

En effet, 0,625 = $1x2^{-1}$ + $0x2^{-2}$ + $1x2^{-3}$

35,625 s'écrit donc en écriture à virgule fixe : $(35,625)_d = (100011,101)_b$

5) On décale alors la virgule à gauche ici de 5 rangs pour écrire le nombre sous la forme 1,M :

L'écriture du nombre binaire devient 1, 00011101.
L'exposant de la puissance de 2 vaut 5 : E = 5
La mantisse est M = (00011101000…) (on supprime le 1 pour gagner 1 bit)

On obtient donc le codage du nombre $(35,625)_d = (100011,101)_b = (1,00011101)_b x2^5$

Pour respecter la norme IEEE 754 en simple précision :

6) On ajoute 127 à l'exposant avant de le coder **: E = 5 + 127 = $(132)_d = (10000100)_b$**
Plutôt que d'utiliser le complément à deux, on applique simplement un décalage fixé à l'avance
7) On complète la mantisse à 23 bits par des 0 : **M = $(00011101000000000000000)_b$**

Enfin, Le nombre $(-35,625)_d$ s'écrit avec la norme IEEE-754 en simple précision :

1	10000100	00011101000000000000000
signe	**exposant**	**mantisse**

4) Deuxième exemple où on tourne en « rond » (hors programme mais à comprendre) :

Les nombres réels peuvent être finis (comme 3, -4,25 ou 0,1) ou infinis (π, 1/3 ou $\sqrt{2}$). Lorsque ces nombres sont codés en binaire, les nombres infinis sont donc arrondis mais il arrive aussi que cela se produise pour certains nombres finis. C'est le cas notamment de 0,1 ou 0,2.

0,2 = 3,2 x 2^{-4} = 3x2^{-4} + 0,2x2^{-4} = 3x2^{-4} +(3x2^{-4} + 0,2x2^{-4})x2^{-4} = …
On peut donc montrer que 0,2 = 3x2^{-4} + 3x2^{-8} + 3x2^{-12} + 3x2^{-16} + …
Son écriture en binaire est donc 0,0011 0011 0011 0011 …

C'est le même cas pour 0,1 = 0,2x2^{-1} , qui s'écrit 0,0001 1001 1001 1001 1001…

5) Avantage de la virgule flottante :

En 32 bits, la virgule fixe peut coder 16 bits pour la partie entière et 16 bits pour la partie décimale.
- ✓ **Domaine de définition** : on peut coder de -32 768 à +32 767 (2^{16})
- ✓ **Pas de quantification** : la précision du codage (partie décimale) est de 16 bits.
(soit en décimal 5 chiffres après la virgule, 2^{16})

En 32 bits, la virgule flottante a comme caractéristiques :
- ✓ **Domaine de définition** : On peut coder de ≈ -3,4.10^{38} à ≈ +3,4.10^{38} -1
(8 bits de Mantisse donne 256 possibilités pour les puissances, donc de -2^{128} à +2^{128} – 1)
- ✓ **Pas de quantification** : la précision du codage (partie décimale) est de 23 bits.
(soit en décimal 7 chiffres après la virgule, 2^{23}).

PARTIE B – REPRESENTATION DES DONNEES (TYPES SIMPLES)
Chapitre 10
Les booléens

I – Transistor :

1) Un peu d'histoire :

Inventés par les américains John Bardeen, William Shockley et Walter Brain (chercheurs des laboratoires Bell) en 1947, ces chercheurs ont reçu pour cette invention le prix Nobel de physique en 1956. A noter que John Bardeen est le seul physicien à avoir reçu deux prix Nobel de physique !

Un transistor est un composant électronique qui est utilisé dans la plupart des circuits électroniques, notamment les circuits logiques, à la base du calcul booléen.

Un transistor est un interrupteur qui permet de réunir ou de séparer deux fils. L'état d'un transistor (passant ou non), est commandé par un troisième fil.

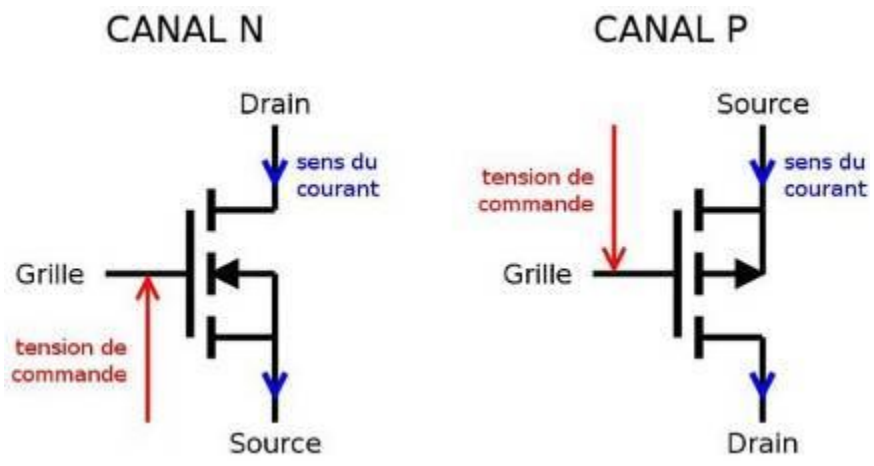
Il existe deux types de transistors (technologie MOS) : les transistors de type N ("non-passant par défaut"), pour lesquels le transistor est non passant quand le fil de commande est dans l'état 0, et les transistors de type P ("passant par défaut"), pour lesquels le transistor est passant quand le fil de commande est dans l'état 0.

2) Quelques détails :

Un transistor possède trois broches appelées drain, grille et source pour la technologie MOS (ou collecteur, base et émetteur pour la technologie bipolaire). Le drain et la source jouent des rôles presque symétriques et sont pratiquement interchangeables.

Pour un transistor n-MOS, si la grille est mise à une tension de 3 V, la source et le drain sont connectés. Si, au contraire, la grille est mise à une tension de 0 V, le circuit entre la source et le drain est ouvert.

Le fonctionnement d'un transistor p-MOS est l'inverse.



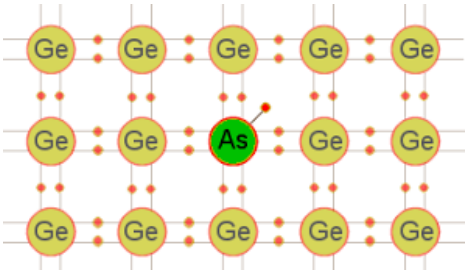
3) Pour aller plus loin :

Un transistor appartient à la catégorie des semi-conducteurs (composés essentiellement de silicium) : matériau qui a la caractéristique d'un isolant mais pour lequel la probabilité qu'un électron puisse contribuer à un courant électrique, quoique faible, est suffisamment important. La conductivité électrique d'un semi-conducteur est donc intermédiaire entre celle des métaux conducteurs et celle des isolants.

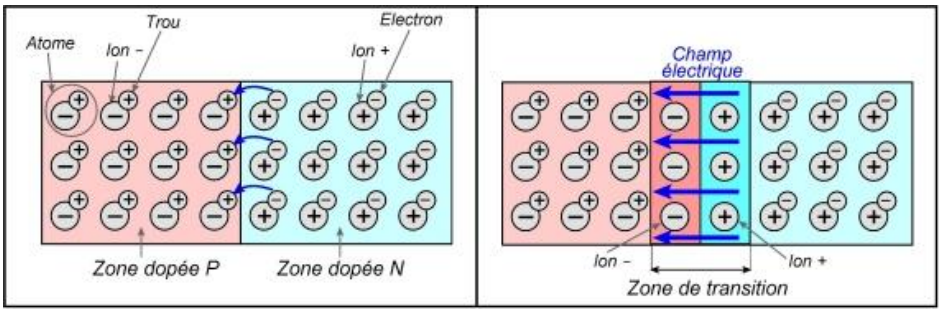
Pour arriver à la création d'un transistor, voici les différentes étapes du processus :

1/ Il faut « doper » du silicium (4 électrons de valence) afin d'augmenter la densité en électrons (dopage N = Négatif) : pour cela, on introduit des atomes ayant 5 électrons de valence (phosphore, arsenic ou antimoine).

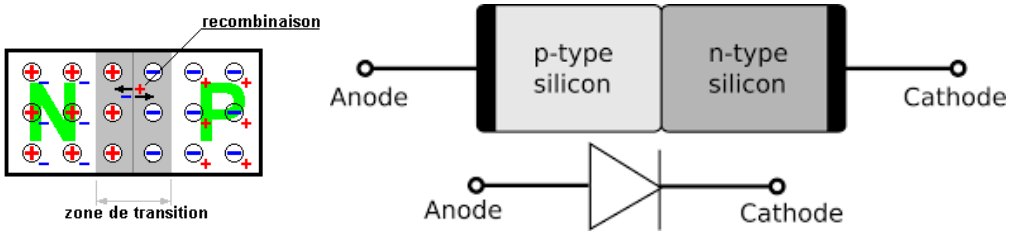
2/ De même, Il faut « doper » du silicium afin de diminuer la densité en électrons (dopage P = Positif) : pour cela, on introduit des atomes ayant 3 électrons de valence (bore). On peut parler de l'augmentation du nombre de "trous" (zones libérées par les électrons dans la bande de valence).



3/ On associe un semi-conducteur dopé N avec un semi-conducteur dopé P pour créer une « jonction PN ».



4/ S'il y a un contact métallique entre la zone P et la zone N de la jonction, on obtient une diode, qui permet le passage d'un courant électrique dans un seul sens. En appliquant une tension positive côté P et négative côté N, des électrons de N iront « boucher » des trous côté P : le courant circule. Inversement, si une tension positive est appliquée côté N, on va creuser le différentiel électrons / trous : la diode sera bloquante.



5/ Enfin, en dopant une 3e région, on peut obtenir des doubles jonctions NPN ou PNP, qui forment les transistors bipolaires. La technologie à effet de champ (JFET en 1952 puis la variante MOSFET en 1960) est basée sur une grille fortement dopée alors que les deux autres pôles sont faiblement dopés (donc appellation unique de N et P). Une autre différence est qu'un transistor bipolaire se commande en courant alors qu'un transistor à effet de champ se commande en tension.

II – Algèbre de Boole de base :

1) OUI (suiveur) :

Exercice 12 : remplir la table de vérité ci-dessous.

Entrée a	Sortie OUI(a)
0	
1	

4. Quel transistor est utilisé ici ? ……

2) NON (inverseur) :

Exercice 13 : remplir la table de vérité ci-dessous.

Entrée a	Sortie NON(a)
0	
1	

5. Quel transistor est utilisé ici ? ……

3) OU :

Exercice 14 : remplir la table de vérité ci-dessous.

Entrée a	Entrée b	Sortie a OU b
0	0	
0	1	
1	0	
1	1	

4) ET :

Exercice 15 : remplir la table de vérité ci-dessous.

Entrée a	Entrée b	Sortie a ET b
0	0	
0	1	
1	0	
1	1	

5) XOR (OU exclusif) :

Exercice 16 : remplir la table de vérité ci-dessous.

Entrée a	Entrée b	Sortie a XOR b
0	0	
0	1	
1	0	
1	1	

7) NAND (NON ET – hors programme) :

Exercice 17 : remplir la table de vérité ci-dessous.

Entrée a	Entrée b	Sortie a NAND b
0	0	
0	1	
1	0	
1	1	

6. D'un point de vue transistor, quelle est la différence entre AND et NAND ?

   ……

   ……

8) NOR (NON OU – hors programme) :

Exercice 18 : remplir la table de vérité ci-dessous.

Entrée a	Entrée b	Sortie a NOR b
0	0	
0	1	
1	0	
1	1	

7. D'un point de vue transistor, quelle est la différence entre OR et NOR ?

   ……

   ……

Exercice 19 : réduire (ou seulement modifier) les équations suivantes en s'aidant de l'algèbre de Boole

Aide : Règle de priorités : NON > ET > OU.
Si on obtient que des 1, on note « 1 ». De même pour que des 0.

A = a OU (a ET b) = ……

B = a ET (a OU b) = ……

C = a OU NON(a) ET b = ……

C' = (a OU NON(a)) ET b = ……

D = (a OU b) ET (a OU NON (b)) = ……

D' = (a OU b) OU (a OU NON (b)) = ……

E = NON(a) ET b OU a ET NON(b) OU a ET b = ……

F = (a OU b) ET (a OU c) = ……

G = (a OU b) ET (NON(a) OU c) = ……

H = (a OU NON(b)) ET (b OU NON(c)) ET (c ou NON(a)) = ……

I = (NON(a) ET NON(b) ET NON(c)) OU (a ET NON(b) ET c) OU (a ET NON(b) ET NON(c)) OU (a ET b ET c)

I = ……

I' = (NON(a) ET NON(b) ET NON(c)) ET (a ET NON(b) ET c) ET (a ET NON(b) ET NON(c)) ET (a ET b ET c)

I' = ……

I'' = (a ET b ET NON(c)) ET (a ET NON(b) ET c) ET (a ET NON(b) ET NON(c)) ET (a ET b ET c)

I'' = ……

PARTIE B – REPRESENTATION DES DONNEES (TYPES SIMPLES)
Chapitre 11
La grande épopée des encodages texte

I – Au commencement était l'ASCII (1961) :

Le premier encodage historique est l'**ASCII**, soit l'*American Standard Code for Information Interchange* (en français, le *code américain normalisé pour l'échange d'informations*). C'est une norme américaine qui avait pour but d'organiser le bazar informatique à l'échelle nationale. Ce n'est pas le premier encodage utilisé mais on peut oublier les précédents.

Le jeu de caractères ASCII utilise **7 bits** et dispose donc de **128 caractères uniquement, numérotés de 0 à 127**. En effet, il est paru à une époque où des regroupements par 7 au lieu de 8 étaient encore assez fréquents (le 8e bit était utilisé pour corriger les erreurs, assez fréquentes dans les premières mémoires électroniques).

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[	123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	|
29	1d	035	GS	61	3d	075	=	93	5d	135	]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

Exercice 20 : coder en hexadécimal le texte " Et l'ASCII survint. "

……

Exercice 20 bis : décoder le texte " 46 61 63 69 6c 65 2c 20 6e 6f 6e 20 3f "

……

Les lettres majuscules sont séparées de leurs homologues minuscules par un intervalle de 32. Cela signifie qu'il suffit de modifier un bit (le 6e) pour passer des unes aux autres, ce qui simplifie les traitements.

Dès 1966, des informaticiens ont réalisés des dessins en « ASCII Art » !

Exercice 21 : sous Python, donner la signification des commandes suivantes

print(ord("k")) # ……

print(chr(52)) # ……

II – La révolte gronde :

La norme ASCII convient bien à la langue anglaise, mais pose des problèmes dans d'autres langues, par exemple le français. En effet l'ASCII ne prévoit pas d'encoder les lettres accentuées.

C'est pourquoi des extensions de l'ASCII sont apparues. Certaines ont gardé la base ASCII et utilisent le 8e bit laissé libre afin d'avoir plus de caractères à disposition (deux fois plus, 256), d'autres sont restés sur 7 bits, et ont modifié carrément les 128 caractères de l'ASCII pour leur propre besoin.

Par exemple, les japonais ont mis en place la norme JIS C 6226 (1978) tandis que les chinois ont utilisé la norme GB2312 (1980) !

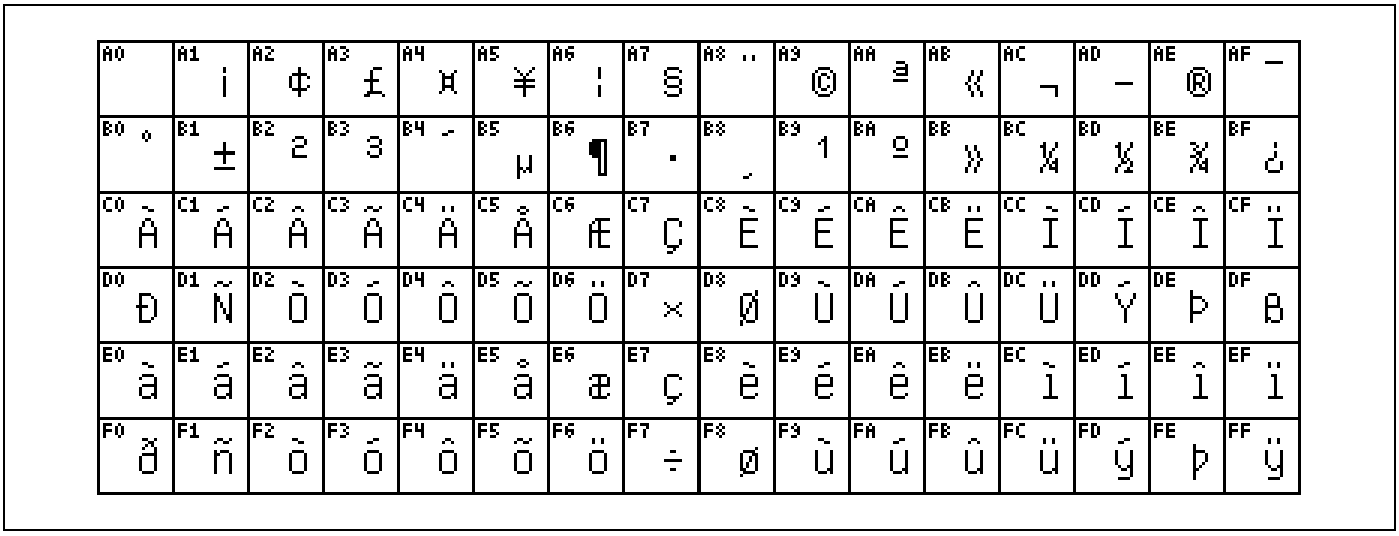
Vous imaginez la pagaille monstrueuse que cela a été, lorsque chaque pays ou groupe linguistique s'est mis à éditer sa propre page de code. Ça fonctionnait bien tant que les documents ne quittaient pas la zone où leur propre encodage était en usage, mais les échanges internationaux étaient sujets à problèmes. Comme un même code signifiait des caractères différents d'un jeu à l'autre, le récepteur ne lisait pas la même chose que le destinataire. Par exemple, le symbole du dollar ($) aux États-Unis devenait celui de la livre (£) au Royaume-Uni : dégâts assurés !

III – La naissance de l'ISO 8859 (1986) :

Une norme mieux pensée a fait son apparition : la norme **ISO 8859**. Cette fois, elle utilise **8 bits donc 256 caractères au maximum**. Le standard ISO 8859 comporte en fait plusieurs « parties », c'est-à-dire des pages de code indépendantes, nommées ISO 8859-*n* où *n* est le numéro de la partie.

ISO 8859 a été pensée afin que les parties soient le plus largement compatibles entre elles. Ainsi, elle englobe l'ASCII (codes 0 à 127) comme base commune, et les codes 128 à 255 devaient accueillir les caractères propres à chaque version, en s'arrangeant pour que des caractères identiques ou proches d'une page à l'autre occupent le même code.

Il existe au total 16 parties pour la norme ISO 8859. 10 sont des versions pour l'Europe Occidentale (dont 8859-1 dit latin-1 et 8859-15 dit latin 9 avec l'apparition du symbole € en 1998) mais aussi une version cyrillique (8859-5), l'arabe (6), le grec (7), l'hébreu (8) et même le thaï (11).



Le caractère A0 est un espace insécable, souvent utilisé avec les signes de ponctuation.

Exercice 22 : coder en latin-1 (hexadécimal) « Ô âme oubliée ! »

……

IV – Pendant ce temps-là, en Asie… :

Les langues latines s'en sont plutôt bien sorties finalement. Elles ont réussi à ne pas dépasser la limite fatidique de l'octet, ce qui restait quand même le plus pratique pour les traitements (et la consommation mémoire). Mais les langues asiatiques comme le japonais, le coréen ou le chinois disposent de bien trop de caractères pour que tout tienne sur 8 bits. Les encodages mis au point en Asie de l'Est ont donc franchi le saut du **multi-octet**. Certains utilisaient 2 octets, ce qui permet 65 536 (2^{16}) codes différents.

Comme pour les langues latines, un standard a été mis au point pour les organiser, on l'appelle **ISO 2022** (1973). Ainsi, les chinois ont eu la version ISO 2022-CN, les coréens la version ISO 2022-KR et les japonais la version ISO 2022-JP (cette dernière étant encore assez utilisé).

V – Unicode, la solution ultime (1991) :

Avec des normes comme **ISO 8859** ou **ISO 2022**, on commençait à s'en tirer pas trop mal. Les problèmes sont atténués, mais subsistent (et si vous rédigez un document en français mais voulez y insérer de l'arabe ?). Finalement, des illuminés se sont dit : « Et si on créait un jeu de caractères unique pour tout le monde ? » De cette idée toute simple sont nés deux monuments : le standard ISO 10646 (Jeu Universel de Caractère « JUC ») et surtout l'**Unicode**.

L'Unicode se décline en 3 versions :

✓ L'**UTF-32** code chaque caractère sur un mot de 32 bits (4 octets) mais il est rarement employé.

8. Combien d'octets sont utilisés pour la phrase de l'exercice 22 en UTF-32 ?

……

✓ L'**UTF-16** code les 65 536 premiers caractères sur 2 octets et les suivants sur 4 octets.

9. Combien d'octets au minimum sont utilisés pour la phrase de l'exercice 22 en UTF-16 ?

……

✓ L'**UTF-8** est plus économe en mémoire : les caractères ASCII sont codés sur un octet, les suivants sur 2, 3 ou 4 octets. Ainsi, les caractères UTF-16 commençant par 00 sont automatiquement réduits.
Exemple : le symbole @ est codé par 00-40 en UTF-16 et il devient 40 en UTF-8 (écriture hexadécimale).

Exercice 23 : convertir l'UTF-16 en UTF-8 ci-dessous

```
texte :            «____ ____ Ô____  ____ â____ m____ e____  ____ o____ u____ b____ l____ i____ é____ e____ ____ !____ ____ »____
valeur en UTF-16 :  00-AB 00-A0 00-D4 00-20 00-E2 00-6D 00-65 00-20 00-6F 00-75 00-62 00-6C 00-69 00-E9 00-65 00-A0 00-21 00-A0 00-BB
```

……

10. Dans ce cas, quel est le taux de compression ? ……

VI – Et aujourd'hui ?

Windows utilise essentiellement l'**UTF-16** mais supporte assez bien l'**UTF-8** tandis que Mac OS X, Linux et Internet utilisent, le plus souvent, **UTF-8**.

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 12
A la découverte d'un 8e continent, Python

Introduction :

Nous utiliserons Anaconda : c'est une distribution Python (version 3.x) qui installe Python ainsi qu'une multitude de packages (comme par exemples IPython Jupyter, Matplotlib, Numpy, pip, scipy et Spyder).

Nous utiliserons essentiellement Spyder, qui est un IDE pour Python (*Integrated Development Environnement*, environnement de développement intégré).

Python est un langage de programmation interprété : les instructions que vous lui envoyez sont « transcrites » en langage ..... au fur et à mesure de leur lecture ;

Par opposition, on a des langages compilés (comme le C) : le code source doit être transformé en un code ..... à l'aide d'un compilateur

I – Calculs :

Exercice 24 : saisir les commandes ci-dessous sur la première ligne, comprendre leur rôle et l'expliquer après le symbole #.

permet, sous Python, de mettre des commentaires du programmeur.

1+1 # Effectue le calcul 1+1
7*5 # .....
4/2 # .....
7/3 # .....
abs(-3) # .....

*La grandeur entre parenthèse s'appelle ici un **argument**.*

7//2 # Récupère le quotient de la division .....
round(4/3,1) # Arrondit à .....
round(4/3,2) # .....
10**2 # .....(moins efficace que .....)
10**5 # .....
2**8 # .....
2**1000 # .....
23%10 # Récupère le reste .....
23%7 # .....
21%7 # .....
divmod(17,3) # Récupère .....

Pour certaines commandes, il faut d'abord importer le module « math » :

```
import math          # Importation du module ……
math.pow(5,2)        # Equivaut à ……
math.sqrt(2)         # sqrt signifie square root, soit ……
math.floor(3.1)      # ……
```

Pour générer des nombres pseudo-aléatoires, il faut d'abord importer le module « random » :

Le terme pseudo-aléatoire est utilisé en informatique pour désigner une suite de nombres qui s'approche d'un aléa statistiquement parfait. De par les procédés algorithmiques utilisés pour la créer et les sources employées, la suite ne peut être complètement considérée comme aléatoire.

```
import random as rd      # Importation du module ……
rd.random()              # Génère un nombre aléatoire entre ……
rd.randint(1,10)         # ……
a = 0                    # Bien respecter les espaces pour une meilleure lisibilité
b = 50
rd.randint(a,b)          # ……
        ### randint est en décalage avec les usages de Python, il vaut mieux utiliser rd.randrange ###
rd.randrange(1,10)       # ……
rd.randrange(1,6,2)      # ……
```

II – Variables et type de données :

```
a = 123                  # Initialise la variable ……
print(a)                 # ……
print(type (a))          # Affiche le type ……
b = 1.23**10             # ……
print(b)                 # ……
print(type(b))           # ……
c = (5 < 3)              # ……
print(c)                 # ……
print(type(c))           # ……
d = "Hello World !"      # ……
print(d)                 # ……
print(type(d))           # ……
```

« Hello World ! » fait référence à une tradition dans le monde informatique : c'est un des premiers messages mis en place pour tester un programme et cela appartient désormais à l'histoire de l'informatique (1974).

III – Initialisations groupées :

```
a, b, c = 1, 2, 3       # .....
d = e = f = 6           # .....
```

IV – Variables booléennes :

```
print(5 == 3)           # Teste si .....
print(5 != 3)           # Teste si .....
b1, b2, b3 = (5 > 3), (9 < 7), (-1 < 0)
print(b1)               # Indique si la condition booléenne est vraie ou fausse : .....
print(not b1)           # .....
print(b1 or b2)         # .....
print(b1 and b3)        # .....
print(not not b1)       # .....
print(not (b1 and b2))  # .....
```

V – Entrée et sorties :

La commande input() permet une saisie en mode texte (string = chaîne de caractère) que l'on peut ensuite transtyper : .....

```
a = input("Saisissez votre message : ")          # .....
b = float(input("Saisissez un nombre réel : "))  # .....
c = int(input("Saisissez un nombre entier : "))  # .....
print (a)                                        # .....
print (b)                                        # .....
print (c)                                        # .....
print ("Somme : ",b+c)                          # .....
print ("Différence : ",b-c)                      # .....
print ("Produit : ",b*c)                         # .....
```

VI – Fonctionnement de l'affectation des variables sous Python :

Quelques éléments de réflexion (hors programme)

x = 300 correspond à une affectation. Lors d'une affectation, Python crée à la fois la valeur (ici 300) et la variable (ici x) qui pointe vers cette valeur (différent du C avec une boîte x qui contient la valeur 300).

On peut faire une analogie : Python utilise comme un cahier avec un index :

- ✓ **x = 300** : dans une page vierge (par exemple la page 17), on écrit « objet de type entier de valeur 300 ». Dans la page index, on écrit « x : page 17).
- ✓ **y = x** : dans la page index, on lit « x : page 17 », donc on écrit « y : page 17 ».
- ✓ **x = x + 1** : on lit « x : page 17 », on récupère la valeur à la page 17, on lui ajoute 1, on prend une page vierge (par exemple la page 35) où on écrit « objet de type entier de valeur 301 ». Il ne reste qu'à remplacer dans l'index « x : page 35 ».
- ✓ **del y** : on gomme dans l'index « y : page 17 ». Le « ramasse-miette » vérifiera si l'entier 300 n'est plus utilisé avant d'effacer la page 17 pour libérer de la mémoire.

Commenter les lignes de code ci-dessous :

```
a = 69
b = a
print(id(a))
a = a + 42
print(id(a))
print(id(b))
```

VI – PEP 8 :

Le langage Python a été créé par Guido van Rossum (*dictateur bienveillant à vie*) mais grandit et évolue grâce à sa communauté. Chaque proposition d'amélioration est publique et publiée sous le nom de « *Python Enhancement Proposal* » (proposition d'amélioration de Python) et portent chacune un numéro.

La PEP 8 a pour objectif de définir des règles de développement communes entre développeurs. En effet, vous avez pu remarquer que le langage est assez flexible. Par exemple, il est possible d'utiliser 2 espaces ou 4 pour l'indentation. Vous pouvez également ne pas mettre d'espace entre les opérateurs. Le code fonctionnera toujours ! Mais ce n'est pas parce que vous pouvez que vous devez le faire !

Voici les règles les plus importantes à respecter :

- ✓ Maximum 80 caractères par ligne.
- ✓ 4 espaces pour l'indentation.
- ✓ Saut d'une ligne entre deux fonctions et espace entre deux paramètres (chapitre 15).
- ✓ Naming sans accent (variable, fonction) : ……
- ✓ Docstring obligatoire pour toutes les fonctions.
- ✓ Une ligne par bibliothèque, toujours en début d'un script.
- ✓ Pas d'espace avant les deux points mais après (voir dic du chapitre 23) : ……
- ✓ Opérateurs (+, -, *, /, =, >…) : un espace avant et après : ……
- ✓ Commentaires : même indentation que le code qu'il commente.
- ✓ Variables et fonctions en SnakeCase : ……
- ✓ Constantes en Screaming SnakeCase : ……

Une recommandation facultative : le nom des variables doit comporter au minimum 3 caractères

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 13
Premiers algorithmes

I – C'est parfois un peu long :

Exercice 25 : on considère l'algorithme de calcul suivant

- ✓ Choisir un nombre entier n.
- ✓ Lui soustraire 4.
- ✓ Multiplier le résultat obtenu par le nombre n choisi.
- ✓ Ajouter 4 à ce produit.
- ✓ Afficher le résultat.

1. Faire fonctionner cet algorithme (ligne par ligne) pour les entiers n compris entre 0 et 3 :

   n = 0 : ..... n = 1 : .....

   n = 2 : ..... n = 3 : .....

2. Traduire cet algorithme en pseudo-code sur papier :

   Règles du bon pseudocode : https://tinyurl.com/bmw4ps3b

   .....

   .....

   .....

   .....

   .....

II – Pour aller plus vite :

Pour gagner du temps, il est possible d'écrire plusieurs lignes en même temps et de faire un input afin que l'utilisateur puisse choisir le n.

Exercice 26 : améliorer l'algorithme de calcul de l'exercice 25.

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 14
Structures conditionnelles et itératives

I – Instruction if (instruction conditionnelle) :

Exercice 27 : une personne doit traverser une route. Que se dit-elle avant de traverser la route ?

……

……

Les instructions conditionnelles vont permettre d'exécuter différents traitements selon la valeur d'une condition. Ce sont des tests qui vont donner un peu de complexité et permettre à l'ordinateur de prendre des (bonnes ?) décisions.

```
# Instruction précédente au même niveau d'indentation que le if

if condition 1 (par exemple a > 0):
        # Instruction 1.1
        # Instruction 1.2

else:
        # Instruction 2.1
        # Instruction 2.2

# Instruction suivante au même niveau d'indentation que le if
```

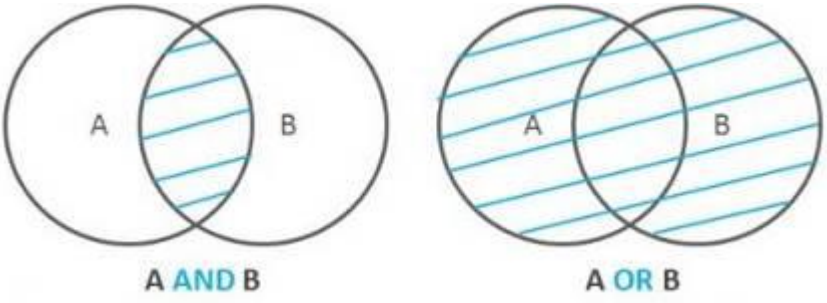
Remarques :

- ✓ Les : à ne pas oublier à la fin du **if** et du **else**.
- ✓ Bien penser à **indenter** les blocs correctement.
- ✓ Dans les conditions, il est possible d'utiliser les mots **or** et **and**.
- ✓ Pas de **end** à la fin du bloc, un retour à la ligne sans tabulation est suffisant.

Exercice 28 : une personne doit traverser une route. De plus, il y a des flaques d'eau. Que se dit-elle avant de traverser la route ?

……

……



Exercice 29 : que se passe-t-il à la fin de ces programmes ? Donner une réponse sans les coder.

Programme 1

```
a=3
b=5
if a<b:
a=a+2
b=7
print(a)
print(b)
```

Programme 2

```
a=5
b=3
if a<b:
        a=a+2
        b=7
print(a)
print(b)
```

Programme 3

```
a=3
b=5
if a<b:
        a=a+2
        b=7
else :
        a=a+3
        b=9
        print(a)
        print(b)
```

Programme 4

```
a=3
b=5
if a<b:
        a=a+2
        b=7
else :
        a=a+3
        b=9
print(a)
print(b)
```

Programme 1 : ……

Programme 2 : ……

Programme 3 : ……

Programme 4 : ……

Exercice 30 : écrire un programme qui demande à l'utilisateur un nombre entier et affiche « pair » si le nombre est pair et « impair » si le nombre est impair.

Rappel : 11%2=1 et 10%2=0

Il est possible de faire des choix multiples en utilisant la structure elif :

```
# Instruction précédente au même niveau d'indentation que le if

if condition 1:
        # Instruction 1.1
        # Instruction 1.2

elif condition 2:
        # Instruction 2.1
        # Instruction 2.2

else:
        # Instruction 3.1
        # Instruction 3.2

# Instruction suivante au même niveau d'indentation que le if
```

Exercice 31 : écrire un programme qui demande un nombre de photocopies à faire et affiche le prix total des photocopies sachant que le prix d'une photocopie est 0,50 euro si le nombre de photocopies est inférieur ou égal à 20, 0,25 euros s'il est compris entre 21 à 50 inclus, 0,10 euros de 51 à 100 inclus et 0,05 € au-delà.

3. Que remarquez-vous pour 18 et 22 photocopies ? ……

4. Quel choix commercial pourrait être fait (système des impôts) ? ……

……

Exercice 32 : écrire un programme qui demande une première valeur, une opération (+ - * ou /), une deuxième valeur et affiche le résultat de l'opération sur les deux valeurs. La gestion de la division par zéro doit être envisagée.

II – Boucle for (boucle itérative bornée) :

Exercice 33 : un robot doit monter un escalier de 13 marches. Comment faire pour permettre au robot de grimper jusqu'en haut ?

……

……

```
# Instruction précédente au même niveau d'indentation que le for

for i in range(indice_debut,indice_fin_non_compris): # Pas de 1
        # Instruction 1
        # Instruction 2

# Instruction suivante au même niveau d'indentation que le for
```

Exercice 34 : deux exemples de boucles « for » sont proposés ci-dessous. On se propose de les écrire et de les tester. Analyser les résultats.

Cas 1 :

```
prenom = ['Kylian','Antoine','Karim','Hugo']

for i in prenom :
    print(i)
    print(prenom)
```

Cas 2 :

```
prenom = ['Kylian','Antoine','Bruno','Hugo']

for i in range(4) :
    print(i)
    print(prenom[i])
```

Les listes seront étudiés lors du 4e thème

5. Quelles sont les valeurs que prend la variable itérative dans chacun des exemples ?

   Cas 1 : ……

   Cas 2 : ……

6. Que renvoient les variables suivantes ?

   prenom : ……

   prenom[i] : ……

7. Dans le cas 2, quelle est la première valeur de i ? ……

8. Dans le cas 2, quelle est la dernière valeur de i ? ……

9. Dans le cas 2, quel est le nombre d'éléments contenus dans la liste ? ……

Exercice 35 : écrire un programme qui affiche les 20 premiers résultats de la table de 7 (ou d'un autre entier naturel) comme ci-dessous.

```
1 * 7 = 7
2 * 7 = 14
[…]
```

Exercice 36 : on lance un dé à 6 faces non pipé. Déterminer à l'aide d'un programme la fréquence d'apparition d'un chiffre pair lorsqu'on effectue N lancers.

Exercice 36 bis : fréquences d'apparition de chaque nombre pour la somme de deux dés sur N lancers.

```
# Instruction précédente au même niveau d'indentation que le for

for i in range(indice_debut,indice_fin_non_compris,valeur_pas): # Le pas peut être différent de 1
        # Instruction 1
        # Instruction 2

# Instruction suivante au même niveau d'indentation que le for
```

Exercice 37 : écrire un programme qui demande un nombre n multiple de 3 à l'utilisateur et affiche les nombres de n à 1, 3 par 3.

Exemple, avec n = 18 :

```
18 17 16
15 14 13
12 11 10
[...]
3 2 1
```

Exercice 37 bis : améliorer le programme 37 afin qu'il s'arrête à 1 si n n'est pas un multiple de 3, en utilisant obligatoirement un pas différent de 1.

Exemple, avec n = 20 :

```
20 19 18
17 16 15
[...]
8 7 6
5 4 3
2 1
```

Exercice 38 : écrire un programme qui calculera la somme de tous les entiers multiples de 9, de 0 à n inclus (choisi par l'utilisateur).

Exercice 39 : écrire un programme qui recherche le nombre de lettres « e » dans une phrase.

Rappel : for x in blabla permet de parcourir « blabla »

III – Boucle while (boucle conditionnelle non bornée) :

Exercice 40 : un robot doit monter un escalier. Comment faire pour permettre au robot de grimper jusqu'en haut ?

……

……

```
# Instruction précédente au même niveau d'indentation que le while

while condition:
        # Instruction 1
        # Instruction 2

# Instruction suivante au même niveau d'indentation que le while
```

Exercice 40 bis : que font les programmes ci-dessous ?

……

……

……

```
k = 0
while k < 10 :
    k = k + 1
    print(k)
```

```
s = 0
i = 100

while i >= 0:
        s = s + 1
        i = i - 1
print(s)
```

```
s = 0
i = 100

while i >= 0:
        s = s + i
        i = i - 1
print(s)
```

Exercice 41 : écrire un programme qui calcule et affiche le plus petit entier n tel que la somme des entiers compris entre 1 et n soit au moins égale à 1000.

Exercice 41 bis :

1. Demander un réel M à l'utilisateur et déterminer combien de fois il faut plier en deux sur elle-même une feuille de papier de 0,05 mm d'épaisseur pour atteindre ou dépasser l'épaisseur M.
2. Tester le programme, par curiosité, avec une valeur de M égale à la distance Terre-Lune (environ 380 000 km).

Exercice 42 : proposer un programme permettant de trouver le PGCD de 2 nombres a et b.
https://www.mathematiquesfaciles.com/calculer-le-pgcd-d-un-nombre-avec-cours_2_75657.htm

Le mot clé **break** permet de sortir prématurément de la boucle while ou for suite à une instruction conditionnelle (uniquement de la dernière boucle).
Le « **while True** » est déconseillé car il n'y a pas de variant de boucle qu'on peut interrompre.

Exercice 43 : que font les programmes ci-dessous ?

```
s = 0
maxi = 50
i = 0

while i < 100:
        s = s + 1
        i = i + 1
print("s vaut",s)
print("i vaut",i)
```

```
s = 0
maxi = 50
i = 0

while i < 100:
        s = s + 1
        if s > maxi:
                break
        i = i + 1
print("s vaut",s)
print("i vaut",i)
```

IV – Boucle « for » ou boucle « while » ?

- ✓ Si on connait à l'avance le nombre de répétitions à effectuer, la boucle « for » est préférable.
- ✓ Si la décision d'arrêter la boucle ne peut s'exprimer que par un test, c'est la boucle « while » qui est préférable.

Exercice 44 : déterminer quelle boucle est adaptée à l'écriture de programmes traitant les problèmes ci-dessous.

a) Le calcul du total à payer à une caisse enregistreuse :

……

b) La recherche du jour le plus pluvieux de l'année :

……

c) Le calcul du périmètre d'un polygone :

……

d) Le calcul du périmètre d'un octogone :

……

e) Le calcul de la durée d'une émission de radio en connaissant la date de début & fin :

……

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 15
Notion de fonction

I – Fonction sans paramètre :

Dans un programme, il est souvent utile de répéter plusieurs fois les mêmes instructions.

```
print ("Le train à destination de Grenoble partira à 8h00")
print ()
print ("----------------------------------------------")
print ()
print ("Le train à destination de Paris partira à 8h30")
print ()
print ("----------------------------------------------")
print ()
print ("Le train à destination de Genève partira à 9h00")
print ()
print ("----------------------------------------------")
print ()
```

Les instructions (ci-dessous) sont répétées trois fois. On peut donc décider de les regrouper pour définir une fonction « TirerUnTrait ».

```
print ()
print ("----------------------------------------------")
print ()
```

```
def nom_fonction():
    # instruction 1
    # instruction 2
    return valeur
```

Si return, cela permet de **renvoyer** (pas retourner) une valeur en dehors de la fonction (qui est interrompue) ; la valeur récupérée sera typée, contrairement à un print.

Remarques :

- ✓ Le mot clé **def** est obligatoire.

L'appel d'une fonction ne peut avoir lieu qu'après sa définition, par : nom_fonction()

Ainsi, notre exemple devient :

```
def TirerUnTrait () :
    '''Permet d'afficher un saut de ligne, un trait et un nouveau saut de ligne'''
    print ()
    print ("----------------------------------------------")
    print ()

print ("Le train à destination de Grenoble partira à 8h00")
TirerUnTrait()
print ("Le train à destination de Paris partira à 8h30")
TirerUnTrait()
print ("Le train à destination de Genève partira à 9h00")
TirerUnTrait()
```

II – Fonction avec paramètres :

Une fonction peut contenir des paramètres (qui deviendront des arguments au moment de l'appel de la fonction), permettant d'améliorer notre fonction.

```
def AnnoncerUnTrain(destination,horaire):
        '''Affiche la destination (str) d'un train et son heure de départ (str) '''
        print ("Le train à destination de ",destination," partira à ",horaire)
        print ()
        print ("----------------------------------------------")
        print ()
# Au lieu du Docstrings, on peut écrire :
# def AnnoncerUnTrain(destination:str,horaire:str) -> None:
```

```
AnnoncerUnTrain ("Grenoble","8h00")
AnnoncerUnTrain ("Paris","8h30")
AnnoncerUnTrain ("Genève","9h00")
```

Exercice 45 : écrire une fonction avec paramètre « premier(n) » qui indiquera si n est un nombre premier. Ci-dessous le résultat que vous devriez obtenir.

```
print(premier(7))
True
print(premier(10))
False
```

III – Portée des variables :

Exercice 46 : exécuter les 6 programmes ci-dessous et les commenter.

def triple(x): x=3*x x=2 triple(x) **print**(x)	x=2 **def** triple(x): x=3*x triple(x) **print**(x)	x=2 **def** triple(): global x x=3*x triple() **print**(x)	**def** triple(x): global x x=3*x x=2 triple(x) **print**(x)	**def** triple(): x=2 x=3*x triple() **print**(x)	**def** triple(): x=2 x=3*x **print**(x) triple()

Programme 1 : .....

Programme 2 : .....

Programme 3 : .....

Programme 4 : .....

Programme 5 : .....

Programme 6 : .....

- ✓ Si une variable est déclarée dans une fonction, alors sa portée est limitée à cette fonction. On parle alors aussi de variable **locale** (excepté pour les list et les dict).
- ✓ Si deux variables locales dans deux fonctions différentes ont le même nom, cela n'est pas gênant, car la portée de chacune est limitée à la fonction correspondante. Elles peuvent même avoir des types différents.
- ✓ Si dans une fonction une variable portant le même nom que dans le programme est utilisée, il risque d'y avoir un conflit :
  - si elle est déclarée dans le programme avant la déclaration de la fonction, car alors Python ne sait plus si la variable est **globale** ou **locale**. C'est donc une situation à éviter absolument.
  - si elle est déclarée dans le programme après la fonction, alors Python considère que la variable de même nom dans la fonction est une variable **locale** et n'affecte donc pas la variable du corps du programme.
- ✓ Si l'on doit absolument modifier une variable définie dans le corps du programme dans une fonction, on utilisera le mot clé **global** (à éviter).

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 16
Diversité et unité des langages

I – Introduction :

Jusqu'à présent, nous avons surtout utilisé le langage de programmation Python. Il existe beaucoup d'autres langages de programmation : Java, C++, Ruby, PHP... Tous ces langages sont différents, mais ils ont aussi des points communs, on peut même dire qu'ils ont plus de points communs que de différences (voir https://rosettacode.org/wiki/Binary_search)

Afin d'entrer un peu dans les détails, nous allons nous intéresser à un langage qui tient une place à part dans l'histoire de l'informatique, le langage C. Le but n'est pas de faire de vous des "programmeurs C", mais de vous montrer que même si le langage Python et le langage C ont des différences, ils ont aussi de nombreux points communs.

Le langage C a été créé par Dennis Ritchie (1941-2011) et Ken Thompson (né en 1943) en 1972. Le langage C est une évolution du langage B (langage B créé par Ken Thompson à la fin des années 60). Le langage C est encore très utilisé aujourd'hui (dans le top 10 des langages de programmation les plus utilisés). Par exemple, le noyau du système d'exploitation Linux est écrit en C. Tout informaticien qui se respecte doit avoir, un jour ou l'autre (au moins pendant ses études), écrit des programmes en C.

II – C, un langage compilé :

Le C est un langage compilé, c'est-à-dire qu'un programme appelé "compilateur" transforme le code source (le code écrit par le programmeur) en langage machine. Cette opération, appelée "compilation", doit être effectuée à chaque fois que le programmeur modifie le code source, cette phase de compilation peut prendre des dizaines de minutes pour de très gros programmes

Exercice 47 : voici le code d'un premier programme qui sera compilé afin qu'il fonctionne. L'objectif est de comprendre la compilation, nous détaillerons la syntaxe par la suite.

```
#include <stdio.h>

int main(void)
{
	int somme;
	int i;
	somme = 0;
	for (i=1;i<=20;i++)
	{
		somme=somme+i;
	}
	printf("%d\n",somme);
}

/* pour compiler : gcc exercice_47.c -o exercice_47 */ (-o pour out)
/* pour exécuter : ./exercice_47
```

10. Que va réaliser ce programme ? ……

……

11. Qu'est-ce que « gcc » ? ……

12. Si on veut faire la somme de 1 à 10, que faut-il faire ?

……

Nous allons écrire quelques programmes en langage C, mais afin de faciliter cette première approche, nous n'allons pas directement manipuler un compilateur, nous allons utiliser un "service en ligne qui s'occupe de tout" (compilation puis exécution du code machine) :

https://repl.it/languages/c

III – Premier programme en C :

A l'ouverture, vous pouvez constater la présence du programme suivant :

```
#include <stdio.h>

int main(void) {
        printf("Hello World\n");
        return 0;
}
```

Exercice 48 : cliquer sur « play » afin de compiler puis d'exécuter le programme ci-dessus.

- ✓ **stdio.h** (Standard Input/Output Header » est une bibliothèque standard, permettant notamment l'utilisation du printf.
- ✓ **int main(void)** est la fonction principale nommée « **main** ». La fonction doit renvoyer un entier (**int**) et ne prend pas de paramètres (**void**).
- ✓ **printf** (print formated) est une commande permettant l'affichage d'une chaîne de caractère.
- ✓ **return 0 ;** permet de renvoyer l'entier 0 pour signifier à la fonction main qu'elle est terminée.

13. Que se passe-t-il si on supprime le « ; » à la fin du printf ?

……

Exercice 49 : exécuter le programme ci-dessous.

```
#include <stdio.h>

printf("Hello World\n");
```

14. Que se passe-t-il ?

……

En C, la fonction « main » est obligatoire : en effet, le système cherche cette fonction afin d'exécuter les instructions qui se trouvent à l'intérieur.

IV – Les variables en C :

Exercice 50 : exécuter le programme ci-dessous.

```
#include <stdio.h>

int main(void) {
        int i ;
        i=15 ;
        printf("La valeur de i est %d\n",i);
        return 0;
}
```

En C, la déclaration du type de variable est obligatoire afin de lui réserver de la mémoire au moment de la compilation.

En Python, le typage est dit dynamique : inutile de le déclarer, le système s'en occupe au moment où la variable est affectée.

15. Est-il possible de déclarer et d'affecter une variable en même temps ? ……

16. Que signifie \n ? ……

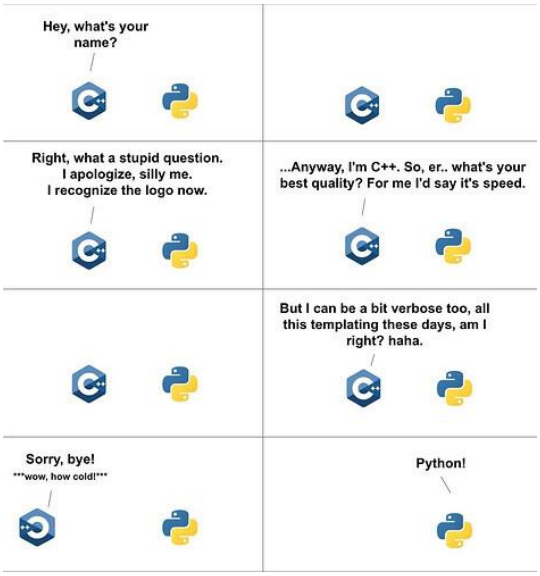
17. Que signifie %d ? ……

V – Les boucles en C :

Exercice 51 : exécuter le programme ci-dessous.

```
#include <stdio.h>

int main(void) {
        int i;
        i=0;
        while (i<10){
           printf("La valeur de i est %d\n",i);
           i=i+1;
        }
        return 0;
}
```

18. Que se passe-t-il avec le while ? ……



VI – Les conditions en C :

Exercice 52 : exécuter le programme ci-dessous.

```
#include <stdio.h>

int main(void) {
	int i;
	i=19;
	if (i<18){
	   printf("Vous êtes mineur");
	}
	else {
	   printf("Vous êtes majeur");
	}
	return 0;
}
```

19. Que va afficher ce programme ? ……

En C, l'indentation n'est pas obligatoire mais elle rend le code plus lisible 😉

VII – Les fonctions en C :

Exercice 53 : exécuter le programme ci-dessous.

```
#include <stdio.h>

int somme(int x, int y){
	int s;
	s=x+y;
	return s;
}

int main(void) {
	int res;
	int a;
	int b;
	a=5;
	b=4;
	res=somme(a,b);
	printf("La somme de %d et de %d vaut %d\n",a,b,res);
	return 0;
}
```

20. Que va afficher ce programme ? ……

21. Si on place la fonction « somme » après main, que se passe-t-il ? ……

Nous verrons dans le prochain chapitre qu'il est possible de placer une fonction après la fonction « main » mais pour cela, nous découvrirons le prototypage.

Nous constatons comme pour la fonction « main » qu'il est nécessaire d'indiquer le type de la valeur renvoyée par la fonction (ici « int » car notre fonction « somme » renvoie bien un entier). À la différence de notre fonction « main », la fonction « somme » prend deux paramètres : x et y (tous les deux de type entier). Il est nécessaire d'indiquer le type des paramètres, ici « int » pour x et y. Si vous omettez le type d'un paramètre, vous aurez le droit à une erreur au moment de la compilation.

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 17
Spécification

I – Définitions :

- ✓ **Spécification** : en génie logiciel, cela consiste à décrire ce que le logiciel doit faire. Pour cela, on utilise les propriétés de prototypage, de précondition, de postcondition et d'assertion.
- ✓ **Prototypage** : en C, les fonctions doivent être déclarées en indiquant au compilateur le nom de la fonction, le type de la valeur de retour et le type des paramètres.
- ✓ **Précondition** : condition à vérifier avant le début d'un calcul ou de l'appel d'une fonction afin de pouvoir en garantir le résultat.
- ✓ **Postcondition** : condition à vérifier à la fin d'un calcul ou de l'appel d'une fonction afin de vérifier que le résultat est correct.
- ✓ **Assertion** : elle vérifie si une condition est vérifiée et si cette dernière est fausse, elle affiche les informations relatives au test et stoppe le programme proprement.

II – Prototypage en C :

1) Présentation :

Il est possible de placer une fonction (par exemple « somme ») après la fonction « main » mais il faut fournir au compilateur le prototype de la fonction.

Exercice 54 : exécuter le programme ci-dessous.

```
#include <stdio.h>

int somme(int x, int y);

int main(void) {
        int res;
        int a;
        int b;
        a=5;
        b=4;
        res=somme(a,b);
        printf("La somme de %d et de %d vaut %d\n",a,b,res);
        return 0;
}

int somme(int x, int y){
        int s;
        s=x+y;
        return s;
}
```

22. Où est le prototype de la fonction « somme » ?

.....

L'utilisation des prototypes est une "bonne pratique" de programmation, il est donc très vivement recommandé d'utiliser les prototypes en C.

Quand les programmes C comportent de nombreuses fonctions, il est judicieux de placer les prototypes des fonctions dans un fichier à part. Ces fichiers portent l'extension ".h" (exemple : "stdio.h"), ils sont appelés "header" ("en tête" en français).

2) Avantage ou inconvénient ?

On pourrait penser que toutes les contraintes imposées par le C par rapport au Python (indiquer le type des variables, le type des paramètres d'une fonction, le type de la valeur renvoyée par une fonction) est un handicap pour le programmeur.

En fait, pas du tout, car ces exigences obligent le programmeur à une plus grande rigueur et permettent de détecter beaucoup plus facilement certaines erreurs. Certains programmeurs n'aiment pas programmer en Python parce qu'ils le trouvent trop "laxiste" avec le type des variables.

Exercice 55 : écrire le programme ci-dessous en Python.

```
def somme(x,y):
        s = x + y
        return s
```

23. Que renvoie Python si on teste la fonction avec 2 et 5 ?

    ……

24. Que renvoie Python si on teste la fonction avec "2" et "5" ?

    ……

Exercice 56 : écrire le programme ci-dessous en C.

```
#include <stdio.h>

int somme(int x, int y);

int main(void) {
  int res;
  res=somme(2,5);
  printf("La somme vaut %d\n",res);
  return 0;
}
int somme(int x, int y){
  int s;
  s=x+y;
  return s;
}
```

25. Ce programme fonctionne-t-il ? ……

26. Modifier 2 et 5 par "2" et "5". Que se passe-t-il ?

    ……

    ……

    ……

Ce qui, au départ, aurait pu paraitre comme une contrainte inutile peut rendre de grand service au programmeur au cours du débogage du programme, alors qu'en Python, l'erreur pourrait passer inaperçue.

III – Les assertions en Python :

Il existe un moyen en Python d'éviter ce genre de problème : l'utilisation des assertions.

Exercice 57 : écrire le programme ci-dessous en Python.

```
def somme(x,y):
        assert isinstance(x,int), "phrase_réponse"          # correct in PEP 8
        assert type(y) == int, "phrase_réponse"             # wrong in PEP 8
        s = x + y
        return s
```

27. Que renvoie Python si on teste la fonction avec 2 et 5 ?

    ……

28. Que renvoie Python si on teste la fonction avec "2" et "5" ?

    ……

Dans le second cas, on dit qu'une exception est levée.

"isinstance" permet de vérifier le type d'une variable. Dans l'exemple ci-dessus "isinstance(x,int)" renvoie True si x est de type entier (int) et False si x n'est pas de type entier. Si ce qui se trouve dans les parenthèses suivant le mot-clé "assert" est False, le système lève une exception et le programme s'arrête. En résumé, le "assert(isinstance(x,int))" permet de lever une exception si la variable x n'est pas de type entier.

Ces assertions permettent de pallier, au moins en partie, les insuffisances de Python en termes de typage des paramètres d'une fonction (donner le type des paramètres d'une fonction).

L'exemple donné ici est volontairement très simple (voire même simpliste), mais ce genre de problème peut se poser dans des programmes extrêmement complexes, ou parfois, il peut se passer des choses inexplicables, très difficiles à comprendre, durant l'exécution. Alors que les erreurs qui entrainent ces comportements erratiques auraient été facilement identifiées si une exception avait été levée.

IV – Les conditions :

1) Préconditions :

Exercice 58 : modifier le programme en ajoutant une précondition

```
import math

def f(x):
    y=math.sqrt(x)-x**2+1
    print(y)

racine(9)
```

2) Postconditions :

Exercice 59 : modifier le programme en ajoutant une postcondition

```
def carre(x):
    y=pow(x,2)
    print(y)

carre(3)
```

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 18
Mise au point de programmes

Une petite ligne de code mal écrite et c'est tout le programme qui plante
et qui peut faire perdre une fortune considérable à une société.
Il est important de trouver ses erreurs en utilisant des jeux de tests !

I – Programme qui plante :

Lorsqu'un programme Python plante, c'est en général en raison d'une exception non gérée. Voici le genre d'affichage que cela produit dans la console :

```
Traceback (most recent call last):
  File "test.py", line 6, in <module>
    Test()
  File "test.py", line 3, in test
    print(table[4])
IndexError: list index out of range
```

Ces erreurs affichées par Python se lisent toujours de **bas** en **haut**. La ligne la plus importante est la **dernière** sur laquelle on trouve 2 informations :

- ✓ La première, c'est le type d'erreur qui s'est produit. Dans cet exemple, il s'agit d'une IndexError.
- ✓ La seconde, c'est le message contenu dans cette exception. Au travers de ce message, Python donne des indications sur ce qui a provoqué l'erreur (ici un « list index out of range »).

Les lignes qui précèdent la dernière sont ce que l'on appelle le "traceback". C'est ce qui permet de trouver rapidement où s'est produite l'erreur. Ces lignes vont en général 2 par 2 :

- ✓ La première ligne indique, le fichier dans lequel on se trouve, la ligne de ce fichier à laquelle s'est produite l'erreur, et le nom de la fonction que l'on est en train d'exécuter.
- ✓ La seconde ligne présente l'instruction qui a fait planter le programme.

Exercice 60 : pour chacun de ces scripts, trouvez l'erreur (ou les erreurs) et corrigez-la (les).

```
a = 5
if a > 0:
print("a est strictement supérieur à 0")
```

.....
.....
.....

```
def puissance(a,b)
   a**b
print(puissance(2,2))
###deux erreurs###
```

.....
.....
.....

```
nom = input(comment t'appelles-tu ?)
```

.....
.....
.....

```
import mathematiques
print(mathematiques.sqrt(4))
```

.....
.....
.....

```
def poids(m):
    return m*g
print(poids(80))
```

.....
.....
.....

```
def conversionFahrenheitCelsius(f):
    return 0.55(f-32)
print(conversionFahrenheitCelsius(375))
```

.....
.....
.....

```
def pairImpair(n):
    if n%2 = 0:
        print("{} est pair".format(n))
    else:
        print("{} est impair".format(n))
```

.....
.....
.....

```
def degreeEnRadian(d):
    pi = 3,14
    return pi*d / 180
print(degreeEnRadian(10))
```

.....
.....
.....

```
def nombrePremier(n):
    premier = True
    for k in range(2,n+1):
        if n%k == 0:
            premier = False
            print("{} non 1er car divisible par {}".format(n,k))
    return premier
print(nombrePremier(1))
print(nombrePremier(17))
### 2 erreurs ###
```

.....
.....
.....

```
def factorielle(n):
    for k in range(n+1):
        nombre = nombre * k
    return nombre
print(factorielle(5))
### Deux erreurs ###
###Factorielle : 5 ! = 5*4*3*2*1 = 120###
```

.....
.....
.....

```
def recherche(nombre_mystere):
    booleen = true
    while booleen == true:
        n = int( input (" Devinez un entier entre 0 et 10: "))
        if n == nombre_mystere:
            print(" Bravo !")
            booleen = false
recherche(7)
###Plusieurs erreurs mais même catégorie###
```

.....
.....
.....

```
def divisible_par(n,p):
    if n%p == 0 and p != 0:
        print("{} est divisible par {}".format(n,p)
        print("c'est super!")

divisible_par(7,5)
```

.....
.....
.....

```
def cube(n):
    return n**3

def volumeSphere(r):
    return 4 * 3.1416 * cube(r) / 3

r = input("Entrez la valeur du rayon : ")
printf("Le volume de cette sphère vaut", volumeSphere(r))
### 2 erreurs ###
```

.....
.....
.....

```
def parcours_liste(L):
    for i in range(len(L) + 1):
        print(i == L[i])

L=[0,1,2,3,4,5,6]
parcours_liste(L)
```

.....
.....
.....

```
def mystere():
    L.sort()

def autre_fonction():
    assert L == [3,2,1]

L = [3,2,1]
mystere()
autre_fonction()
```

Bug lié à un « effet de bord » :

.....
.....
.....

```
def assert_egal(a, b):
    return a == b

print(assert_egal(0.2*0.2, 0.04))
# Affiche False, pourquoi ?
```

.....
.....
.....

II – Programme qui ne plante pas mais… :

Parfois, un programme fonctionne lors d'un premier test mais une erreur persiste. Il est donc important de réaliser plusieurs tests, notamment avec les fameux « cas particuliers » qu'on rencontre souvent en maths.

Exercice 61 : identifier l'erreur de ce programme et la corriger

```
def permute(liste) :
    #Fonction qui permute le premier élément et le dernier élément d'une liste
    #Avec [1,2,3,4], ce programme renvoie bien [4,2,3,1]
    copie=liste
    copie[0],copie[-1] = copie[-1],copie[0]
    return copie

print(permute([1,2,3,4]))
```

.....

.....

.....

PARTIE C – LANGAGES ET PROGRAMMATION
Chapitre 19
Utilisation de bibliothèques

Aucune connaissance exhaustive d'une bibliothèque particulière n'est exigible.
Vous allez travailler sur différents « grands » projets, chacun utilisant une bibliothèque différente.
Attention, ne pas dire « librairie » (un anglicisme de « librairy »)

I – Module Turtle :

Le module Turtle est un ensemble d'outils permettant de dessiner à l'aide d'instructions simples en déplaçant une « tortue » (en option) à l'écran.

1) Un premier exemple :

Exercice 62 : taper le programme ci-dessous ligne par ligne. Vous exécuterez la première ligne, puis les 2 premières lignes… cela permettra de suivre ce qui se passe au fur et à mesure.

```
>>>import turtle as tt      # Permet d'importer le module Turtle et ses fonctions.
>>>tt.forward(120)         # ……
>>>tt.left(90)             # ……
>>>tt.color("red")         # ……
>>>tt.forward(80)          # ……
```

2) Fonctions principales de Turtle :

```
✓ tt.reset()                # On efface tout et on recommence
✓ tt.goto(x,y)              # Aller à l'endroit de coordonnées x et y
✓ tt.forward(distance)      # Avancer d'une distance donnée
✓ tt.backward(distance)     # Reculer
✓ tt.up()                   # Lever le crayon (pour pouvoir avancer sans dessiner)
✓ tt.down()                 # Baisser le crayon (afin de pouvoir recommencer à dessiner)
✓ tt.color(couleur)         # Couleur prédéfinie ('red') ou HTML ('#330000')
✓ tt.left(angle)            # Tourner à gauche d'un angle donné (exprimé en degré)
✓ right(angle)              # Tourner à droite
✓ tt.width(épaisseur)       # Choisir l'épaisseur du tracé
✓ tt.begin_fill()           # Remplir un contour fermé à l'aide de la couleur sélectionnée
                            (si il y a 2 couleurs, la première est le trait, la seconde le remplissage)
✓ tt.end_fill()             # Pour terminer le remplissage du contour fermé
✓ tt.write('texte')         # Texte doit être une chaîne de caractère délimitée avec des '
                            # Pour personaliser : write('texte',font=('police',15))
✓ tt.speed(vitesse)         # Pour aller plus vite, on peut mettre un nombre de 1 à 9 (6 = normal)
✓ tt.circle(rayon)          # Tracer un cercle avec un rayon défini
```

- ✓ tt.bye() # Ferme automatiquement
- ✓ tt.setheading(angle) # Angle absolu 0 vers bord droit et 90 vers bord haut
- ✓ print(tt.xcor()) # Position x
- ✓ tt.setup(640, 480, 100, 100) #Largeur / Hauteur / pos x / pos y de la fenêtre en pixels
- ✓ tt.exitonclick() puis tt.mainloop() # Ferme avec un clic gauche (Python 3)

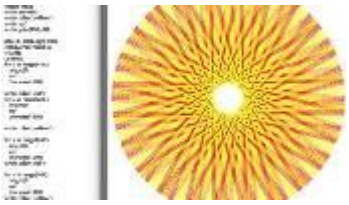
3) Exercices :

Exercice 63 : taper le programme suivant, puis l'exécuter :

```
import turtle as tt
a = 0
while a < 12 :
        a = a+1
        tt.forward(100)
        tt.left(150)
```

Exercice 64 : écrire un programme permettant de tracer un triangle équilatéral à l'aide d'une boucle for.

Exercice 65 : écrire un programme qui demande la largeur du côté d'un carré, puis qui dessine le carré à l'aide d'une boucle while.



II – Modification d'image avec la bibliothèque Pillow :

Pour accéder aux fichiers, RV dans le lecteur K « VM » (Virtual Machine)

1) Généralités :

Nous allons travailler sur des images et plus précisément sur des **pixels** (= Picture Element). Les images seront aujourd'hui en **niveau de gris**, 0 pour le noir et 255 pour le blanc. Le format par défaut utilisé est .**png** (Portable Network Graphique).

La bibliothèque que nous allons utiliser est Pillow. Le fichier python doit être dans le même dossier que l'image étudiée.

- ✓ Import PIL # Importation de la bibliothèque gérant les images
- ✓ Im = PIL.Imahge.open("nom.png") # Im est le nom de la variable de stockage
- ✓ PIL.Im.show() # Pour visualiser une image (non enregistrée)
- ✓ (co,li) = PIL.Im.size # Pour obtenir la taille de l'image
- ✓ PIL.Im.getpixel((36,12)) # Récupère la valeur du pixel en 36, 12
- ✓ PIL.Im.putpixel((36,12),0) # Remplace la valeur du pixel en 36,12 par 0
- ✓ Interdiction d'appeler son image (Image) (mot réservé) mais (image) possible.

2) Exercices :

Exercice 66 : avec l'image « phare_bruit », faire les actions suivantes :

- ✓ Ouvrir l'image puis l'afficher.
- ✓ Obtenir la valeur d'un pixel avec input et vérifier en zoomant sur l'image.
- ✓ Modifier la valeur d'un pixel.
- ✓ Insérer un carré noir de 100 pixels de côté au milieu.

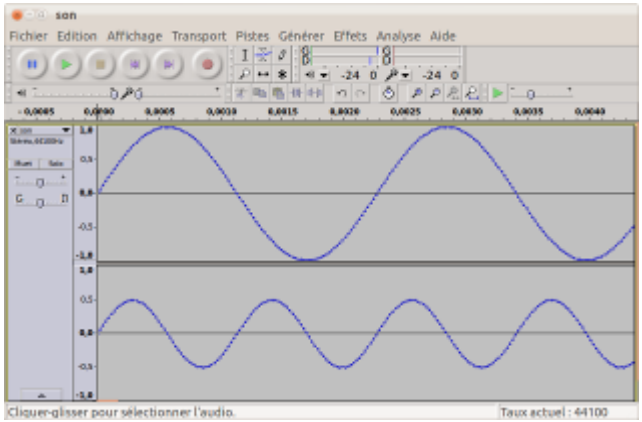
Exercice 67 : on souhaite nettoyer l'image avec un filtrage médian :

- ✓ Pour chaque pixel, il faut récupérer 9 valeurs : la sienne et les 8 voisins.
- ✓ Les 9 valeurs sont injectées dans une liste, puis elle est triée avec la fonction sort.
- ✓ La valeur médian (la 5e) remplacera la valeur du pixel dans une nouvelle image.
- ✓ Gestion des angles et des côtés en option.

Exercice 68 : avec l'image « riz », on souhaite afficher l'image en noir et blanc, ce qui équivaut à du seuillage : il faut déterminer une valeur seuil permettant d'indiquer la limite entre le noir et le blanc.

III – Création de fichiers sons avec le module wave :

1) Présentation :



Un fichier audio contient des données numériques pour stocker des sons, notamment de la musique et de la voix humaine. Le programme qui transforme un signal sonore en fichier et vice-versa s'appelle un codec, abréviation de Coder-DECoder.

Le format WAV a été standardisé par Microsoft et IBM, grâce notamment à Windows : il permet un encodage sans perte de qualité en 16 bits (2 octets par échantillon) à une fréquence d'échantillonnage de 44 100 Hz (44 100 échantillons par seconde).

Sous Python, on utilisera le module **wave**.

2) Exercices :

Exercice 69 : ouvrir le fichier de l'exercice et l'exécuter avec une fréquence de 440 Hz à gauche, 880 Hz à droite, niveau de son à gauche de 1 et à droite de 0,5 et une durée de 2,5 secondes.

Exercice 70 : un fichier audio WAV possède les propriétés suivantes : stéréo, 16 bits, 44100 Hz, taille du fichier 105 884 octets.

29. Quelle est la taille de l'en-tête (en octets) ? .....
30. Quelle est la taille des données (en octets) ? .....
31. Quelle est la durée du son de ce fichier (en ms) ?

.....

.....

Exercice 71 : écrire le script d'un bruit blanc (son généré de manière aléatoire).

On partir sur un fichier mono, codé sur 1 octet, une fréquence de 44 100 Hz et d'une durée de 2 secondes.

Exercice 72 : écrire un script qui ajoute un écho au fichier helpme.wav avec un retard de x secondes.

Il faudra compléter le code où le début et la fin sont déjà écrits.

PARTIE D – REPRESENTATION DES DONNEES (TYPES CONSTRUITS)
Chapitre 20
Les p-uplets (ou tuples)

I – Généralités :

Quand le nombre de données est important ou s'il est intéressant d'en regrouper certaines (comme pour les coordonnées d'un point par exemple), alors nous avons besoin de définir des variables dont les valeurs sont des ensembles de valeurs.

Un p-uplet est une liste non modifiable. Le mot anglais est « *tuple* », qui provient du suffixe quintuple ou sextuple (avec une coupure erronée puisque le réel suffixe est -ple, comme dans triple).

II – Commandes générales :

✓	mon_tuple = ()	# Création d'un tuple vide
✓	mon_tuple = (1, « ok », « jb »)	# Création d'un nouveau tuple avec valeurs
✓	mon_tuple += (3,)	# Création d'un nouveau tuple par concaténation
✓	mon_tuple[0]	# Affiche une valeur d'un tuple
✓	len(mon_tuple)	# Longueur du tuple

III – Utilités d'un tuple :

Les tuples sont moins utilisés que les listes (voir chapitre 21). Ils peuvent s'avérer très pratiques pour récupérer plusieurs valeurs renvoyées par une fonction (avec un return). Ils sont également utilisés pour définir des constantes qui n'ont pas vocation à changer.

Pour comprendre la notion de « non-modifiable » (immuable), commenter les lignes de code ci-dessous :

```
mon_tuple = (69,)

mon_tuple.append(42)

mon_tuple[0] = 42

mon_tuple = mon_tuple + (42,)
```

IV – Exercices :

Exercice 73 : recopier les lignes suivantes et indiquer ce que l'on obtient.

```
t = divmod(7,3)
print(type(t))
```

……

……

Exercice 74 : lire le programme ci-dessous puis répondre aux questions.

```
1  def triplets(n) :
2      t =()
3      for a in range (1,n) :
4          for b in range (a+1,n) :
5              for c in range (b+1,n) :
6                  if c**2 == a**2 + b**2 :
7                      t = t+((a,b,c),)
8      return t
9  print(triplets(100)[-1])
```

1. Quelle est l'utilité de ce programme ?

   ……

   ……

2. Que va-t-il afficher ?

   ……

3. Quelle est l'utilité de la dernière virgule en ligne 7 ?

   ……

4. Pourquoi optimise-t-on le code avec « a+1 » et « b+1 » ?

   ……

5. Optimiser le code pour diminuer le nombre de tests de la ligne 6

PARTIE D – REPRESENTATION DES DONNEES (TYPES CONSTRUITS)
Chapitre 21
Les listes

I – Généralités :

Toujours dans l'objectif d'économiser de la mémoire (si on utilise une variable à chaque fois), une liste sous python est, contrairement à un p-uplet, modifiable !

- ✓ ma_liste=[] # Création d'une liste vide
- ✓ ma_liste[5]=17 # Modifie l'élément en 6e position de la liste par l'entier 17
- ✓ ma_liste.append(38.5) # On ajoute le réel 38,5 à la fin de la liste
- ✓ ma_liste.insert(3,'blabla') # On insère le string blabla à l'indice 3 (4e position)
- ✓ del ma_liste[5] # Supprime l'élément qui se trouve à l'indice 5 (6e position)
- ✓ ma_liste.remove(32) # Supprime l'entier 32 présent dans ma liste (une seule fois)
- ✓ ma_liste.index(53) # Donner la position du premier item prenant la valeur 53
- ✓ len(ma_liste) # Donne la longueur d'une liste
- ✓ for element in ma_liste: # Parcours chaque élément de la liste
- ✓ ma_liste.reverse() # Inverse l'ordre de la liste
- ✓ ma_liste.sort() # Trie la liste par ordre croissant

Exercice 75 : réaliser les manipulations simples ci-dessous et indiquer les résultats

- ✓ Définir une liste (nommée liste) contenant les nombres 1, 7, 26, 42, 63, 73, 74, 43, 38, 15 et 3 dans cet ordre.
- ✓ Trier par ordre croissant et afficher la liste ainsi obtenue : ……
- ✓ Ajouter l'élément « 69 » à la fin de la liste : ……
- ✓ Renverser et afficher la liste : ……
- ✓ Afficher l'indice de l'élément 42 : ……
- ✓ Enlever l'élément 42 et afficher la liste : ……
- ✓ Afficher la sous-liste du 3e au 4e élément : ……
- ✓ Afficher la sous-liste du début au 3e élément exclu : ……
- ✓ Afficher la sous-liste du 4e élément exclu à la fin de la liste : ……
- ✓ Afficher le dernier élément en utilisant un indiçage négatif : ……

Exercice 75 bis : étudier la différence entre ma_liste.sort() et sorted(ma_liste), tout d'abord directement puis en utilisant une nouvelle variable.

……

……

……

Exercice 75 ter : étudier la différence entre b = a et b = a.copy() pour dupliquer une liste. Il serait intéressant d'ajouter un élément lors d'un test.

……

……

……

……

Exercice 75 quater : soit la liste stock = ['Ordinateur fixe', 50, 'Ordinateur portable', 100, 'Caméra', 310, 'Haut-parleurs', 27, 'Télévision', 1000, 'Cartes mères', 20, 'Souris', 15, 'Clavier',45, 'Barrettes de mémoire', 500]

Créer une liste composée des objets et une autre composée des effectifs, en utilisant une première fois « for element in » et une seconde fois en utilisant « for i in range ».

Exercices 75 quinquies : écrire un algorithme recherchant le maximum dans une liste sans utiliser la fonction native « max() » de Python. Cet exercice est proche de ce qui peut être demandé lors du premier exercice de l'Epreuve Pratique en terminale NSI.

II – Construire une liste par compréhension :

Construire une liste par compréhension signifie simplifier le code pour le rendre plus lisible et plus rapide.

Voici la syntaxe à utiliser (le if n'est pas obligatoire).

✓ ma_liste = [<fonction sur item> for <item> in <liste> if <condition>]

Exercice 76 : écrire les valeurs de la nouvelle liste.

```
nombres = [2,5,11,3,17,15,20]
nouvelle_liste = [2*n for n in nombres]
# Réexécuter en ajoutant un str ou un booléen à la fin de 'nombres'
```

……

Exercice 76 bis : écrire les valeurs de la nouvelle liste.

```
nombres = [2,5,11,3,17,15,20]
nouvelle_liste = [2*n for n in nombres if n%2 == 0]
# Modifier avec 'if nombres.index(n) % 2 == 0'
```

……
……

Exercice 77 : transformer ce code afin qu'il construise une liste par compréhension

```
liste = [1,4,2,7,1,9,0,3,4,6,6,6,8,3]
new_liste = []
for i in liste :
    if i > 5 :
        new_liste.append(i)
```

……

……

PARTIE D – REPRESENTATION DES DONNEES (TYPES CONSTRUITS)
Chapitre 22
Les tableaux

I – Généralités :

Pour travailler avec un **tableau** (deux dimensions, ligne puis colonne, également appelé **matrice**), nous allons utiliser une astuce : nous allons créer une **liste** de **listes**.

✓ grid=[[0]*nbr_co for x in range(nbr_li)] # Création d'un tableau par compréhension

✓ grid[3][5]=8 # Insérer 8 en ligne 3 et colonne 5

Si vous imprimer votre tableau, le résultat sera peu agréable à la lecture. Vous pouvez utiliser la fonction suivante :

```
def affiche(grid):
    for i in range(len(grid)):
        print(grid[i])
```

II – Exercices :

Exercice 78 : créer un tableau par compréhension de la matrice nulle (un tableau carré ne contenant que des 0) avec un input pour la taille de la matrice.

Exercice 79 : créer un tableau de la matrice identité (un tableau carré contenant des 1 sur la diagonale et des 0 ailleurs) avec un input pour la taille de la matrice.

Exemple de la matrice identité de taille 4 :

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Exercice 79 bis : créer un tableau de la matrice ne contenant que des 1 sur l'autre diagonale avec un input pour la taille de la matrice.

Exemple de la matrice de taille 4 :

0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

Exercice 80 : créer un tableau de la matrice ci-dessous avec un input pour la taille de la matrice et avec une boucle for (trois méthodes différentes possibles).

Exemple de la matrice de taille 4 :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

PARTIE D – REPRESENTATION DES DONNEES (TYPES CONSTRUITS)
Chapitre 23
Les dictionnaires

I – Généralités :

Un dictionnaire ressemble à une liste. La principale différence est que les indices ne sont pas obligatoirement des entiers (0, 1, 2, …) et peuvent être du type str, float, tuple. Ces indices ne sont pas ordonnés et s'appellent des clés. A chaque clé correspond une valeur.

Un dictionnaire est donc un ensemble de couples clé-valeur.

II – Syntaxe :

✓ a = {}	# Initialisation d'un dictionnaire
✓ a[clé] = valeur	# Ajoute/remplace une valeur au dictionnaire a
✓ a.get(clé) ou a[clé]	# Récupère la valeur à partir de la clé
✓ del a[clé]	# Supprime une entrée à partir de la clé
✓ a.**keys**()	# Récupère toutes les clés
✓ a.**values**()	# Récupère toutes les valeurs
✓ a.**items**()	# Récupère l'ensemble des couples clé-valeur
✓ len(a)	# Nombre de couples
✓ a = b	# Copie liée des deux dictionnaires
✓ b = a.copy()	# Copie indépendante des deux dictionnaires
✓ clé in a	# Apporte une réponse True or False
✓ for clés in a	# Parcourt toutes les clés comme in « a.keys() »
✓ for valeur in a.values()	# Parcourt toutes les valeurs

III – Exercices :

Exercice 81 : tableau physico-chimique

	T_{eb}	Tf	Z	A
Au	2 970	1 063	79	197
Ga	2 237	29,8	31	69

Affecter les données de ce tableau à un dictionnaire « dico » de façon à pouvoir écrire par exemple :

```
print(dico["Au"]["Z"]) (affiche : 79)
```

Exercice 82 : Base de données

L'entreprise *KiStokTout* possède une base de données dans laquelle elle enregistre les informations personnelles de ses clients :

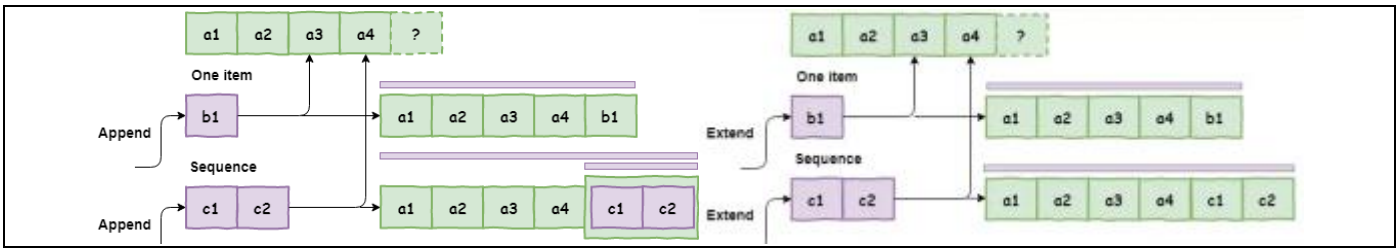
nom, prénom, adresse, âge ; taille (en cm), poids (en kg)

Elle initialise cette base de données par la ligne suivante :

```
registre = [[1,{'nom':'Mbappé','prenom':'Kylian','adresse':'Camp des Loges,
Paris','age':23,'taille':178,'poids':73}]]
```

où 1 est le numéro de la ligne dans la base de données.

6. Quel est le type de la variable registre ? ……
7. Quel est le type de registre[0] ? ……
8. Quel est le type de registre[1] ? ……
9. Quel est le type de registre[1][`nom'] ? ……



10. Rechercher l'instruction permettant d'ajouter les informations de Gianluigi Donnarumma, 24 ans, même adresse, 1m96 pour 90 kg à la ligne 2 de la base de données (en une seule étape) :

    ……

    ……

    ……

11. Compléter le programme afin qu'il permette à une personne d'entrer au clavier les informations tant qu'il y en a :

```
Registre = []
Ligne = …
rep = …

while rep == … :
    nom = input("Nom : ")
    prenom = input("Prenom : ")
    adresse = input("Adresse : ")
    age = …(input("Age : "))
    taille = …(input("Taille : "))
    poids = …(input("Poids : "))
    Ligne …
    Registre.…([Ligne,{"nom":nom,"prenom":prenom,"adresse":adresse,"age":age,"taille":taille,
    "poids":poids}]) #Sur la même ligne que la précédente
    rep = input("Voulez-vous poursuivre la saisie (O = oui) ? ")
```

PARTIE E – TRAITEMENT DE DONNEES EN TABLES
Chapitre 24
Indexation de tables

I – Introduction au CSV :

1) Généralités :

Le CSV est un format texte particulier pour structurer des données simplement.

> Comma-separated values, connu sous le sigle CSV, est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules. Ce format n'a jamais vraiment fait l'objet d'une spécification formelle.
> Un fichier CSV est un fichier texte par opposition aux formats dits « binaires ». Chaque ligne du texte correspond à une ligne du tableau et les virgules correspondent aux séparations entre les colonnes. Les portions de texte séparées par une virgule correspondent ainsi aux contenus des cellules du tableau. Une ligne est une suite ordonnée de caractères terminée par un caractère de fin de ligne, la dernière ligne pouvant en être exemptée.
>
> **Wikipédia.org**

Ce format est très utilisé pour l'échange de données car il est relativement facile à lire et à générer notamment par les tableurs ou les bases de données. Finalement, les développeurs de site Web les utilisent fréquemment. Par exemple, quasiment toutes les applications Webmail permettent d'importer et d'exporter ses contacts grâce à ce format.

En France, la virgule est utilisée pour séparer la partie entière et de la partie décimale dans l'écriture des nombres contrairement aux anglo-saxons qui utilisent le point. Comme les séparateurs ne sont pas standardisés, le point-virgule est utilisé pour séparer les données dans les fichiers CSV.

2) Site data.gouv.fr :

Depuis 2011, le gouvernement permet un accès libre et une réutilisation gratuite d'un très grand nombre de données publiques. L'ouverture de ce portail interministériel unique répond à la volonté de renforcer la transparence de l'action de l'Etat, comme des collectivités locales auprès des citoyens.

Découvrir le site data.gouv.fr et rechercher un fichier contenant des informations au format csv (résultats des Chartreux au bac, résultats des dernières élections, liste des auteurs dans les programmes du secondaire…). Ouvrir ce fichier avec un éditeur de texte (BlocNote ou NotePad++) puis avec un tableur (si nécessaire, modifier le séparateur à l'ouverture du fichier).

1. Comment sont organisées les données pour chaque lycée/candidat/auteur ? ……
2. Pour un même lycée/candidat/auteur, quel est le séparateur qui permet de séparer les données entre-elles ? ……

La première ligne (titre des colonnes) est appelée « descripteurs ». Les lignes suivantes contiennent les « valeurs » associées aux descripteurs.

II – Importation d'une table :

Pour lire le fichier, il faut utiliser la fonction open(). Cette fonction s'utilise de la manière suivante :

```
fichier = open(nomfichier,mode)
```

Il existe d'autres argument dont, par exemple, le troisième (facultatif) qui est le l'encodage du fichier (utf-8, iso8859-1, ascii, ...).

Le premier argument est une chaine de caractères comportant le chemin (relatif ou absolu) et le nom du fichier. Le deuxième est une chaine de caractères précisant le mode d'ouverture du fichier :

- ✓ **'r'** (read) indique un mode en lecture (mode par défaut si argument omis).
- ✓ **'w'** (write) indique un mode en écriture.
- ✓ **'a'** (append) indique un mode en ajout.

Le programme devra fermer le fichier grâce à la méthode **close()**. Un fichier ne doit pas rester ouvert car si un autre programme souhaite l'utiliser il ne pourra pas le faire. Il est possible d'utiliser le « **with** » qui génère un block d'instruction qui, lorsqu'il est quitté, ferme le fichier.

Exercice 83 : compléter les codes ci-dessous afin d'ouvrir le fichier exercice_83.csv.

Avec une boucle while Méthode .append()	Avec une bouche for Méthode .append()	Avec une boucle for Utilisation de « with »

fichier = open('…', '…')
ligneListe=[]
ligne = fichier.readline()
# explication : 	
while ligne:
  ligneListe. …(…)
  ligne = fichier. …
print(ligneListe)
fichier.close()
	fichier = open('…', '…')
	ligneListe=[]
	for ligne in …:
	  ligneListe. …
	print(ligneListe)
	fichier.close()
		with open('…', '…') as fichier:
		  ligneListe=[]
		  for ligne in …:
		  ligneListe. …
		print(ligneListe)
		
		# Déconseillé par certains collègues car plus délicat.

*Pour les utilisateurs de **Visual Studio Code**, cela ne fonctionne pas. Il faut tester avec le module « **csv** ».*

En python, les chaînes non vides sont toujours vraies ce qui permet de mettre la lecture d'une ligne comme condition du **while**.

Cependant, en regardant l'affichage du résultat, on peut voir deux problèmes :

3. a) Que constatez-vous en fin de ligne ? ……

   ……

   b) Est-ce facile d'accéder à une donnée ? ……

   ……

4. a) Quel est le type de ligneListe ? ……

   ……

   b) Et de ligne ? ……

   ……

Il existe deux solutions : une méthode classique avec l'utilisation de deux « **if** » et une méthode par compréhension en utilisant la méthode **strip()** et la méthode **split(';')** :

- ✓ La méthode **strip()** enlève les espaces éventuels au début et à la fin de la chaîne de caractères, ainsi que le caractère blanc ' ' et le caractère de retour à la ligne '\n'.
- ✓ La méthode **split(';')** découpe une chaîne en une liste de plusieurs éléments à chaque délimiteur « ; » rencontré (qui est lui supprimé).

Exercice 84 : compléter les codes ci-dessous afin d'ouvrir le fichier exercice_83.csv et supprimer les retours à la ligne et les points-virgules : le résultat doit être sous la forme d'un tableau (liste de listes).

Avec deux « if »

```
fichier = open('…', '…')
Liste=[]
for ligne in …:
    donnee = ""
    ligneListe=[]
    for caract in …:
        if caract != …:
            if caract != …:
                donnee += caract
            else:
                ligneListe….(…)
                donnee = ""
    ligneListe….(…)
    Liste….(…)
print(Liste)
fichier.close()
```

Méthode par compréhension à l'aide des méthodes strip et split

```
fichier = open('…', '…')
for ligne in …:
    Liste….(ligne….()….(…))
print(Liste)
fichier.close()
```

PARTIE E – TRAITEMENT DE DONNEES EN TABLES
Chapitre 25
Recherche dans une table

I – Rechercher la présence d'une valeur dans une table :

Exercice 85 : écrire un programme qui affiche « True » ou « False » selon qu'il trouve un pays d'Océanie ou pas dans la liste des pays commençant par la lettre A (projet 24).

Option : rajouter un test permettant de vérifier que le pays commence bien par A.

II – Afficher toutes les instances d'une valeur :

Exercice 86 : écrire un programme affichant la liste des pays d'Europe commençant par A.

III – Recherche de doublons :

L'objectif est de vérifier que certaines données ne sont pas répétées afin de supprimer éventuellement les lignes en double.

Exercice 87 : sans ouvrir avec NotePad++ le fichier exercice_87.csv (dans lequel vous trouverez la réponse), avec deux boucles for, créer une nouvelle liste dans laquelle il n'y a qu'une seule ligne par pays.

Le programme doit afficher en sortie la liste des pays du CSV d'origine et la nouvelle liste après traitement sans le pays en doublon.

5. Quel pays était en doublon ? ……

PARTIE E – TRAITEMENT DE DONNEES EN TABLES
Chapitre 26
Tri d'une table

I – Tri suivant la première colonne d'une table :

Exercice 88 : trier le fichier exercice_88.csv en utilisant dans un premier temps le code print(sorted(table)) et dans un second temps la méthode table.sort().

6. Quel code n'affecte pas la table initiale ? ……

7. Quel code modifie la table elle-même ? ……

II – Les fonctions « lambda » :

Python permet une syntaxe intéressante qui vous laisse définir des mini-fonctions d'une ligne à la volée (emprunté au langage Lisp de 1958). Cela peut simplifier la lisibilité du code et nous permettra de trier une colonne autre que la première d'une table.

Fonction classique	Fonction « lambda »
def f(x) : return x*2 print(f(3))	g = lambda x : x*2 print(g(3))

Exercice 89 : écrire une fonction « lambda » permettant de faire la somme de deux arguments x et y.

III – Tri suivant une colonne quelconque :

Pour réaliser ce tri, sorted et sort() admettent un paramètre optionnel appelé *key*. C'est grâce à cet argument que l'on va pouvoir sélectionner la colonne sur laquelle on va effectuer le tri, et ce à l'aide d'une fonction lambda.

Exercice 90 : compléter le code ci-dessous.

```
Liste….(key = lambda …: …[…])
print(…)
```

Le mot utilisé pour choisir la colonne n'a pas d'importance, cela peut être « colonne » ou « col » par exemple.

PARTIE E – TRAITEMENT DE DONNEES EN TABLES
Chapitre 27
Fusion de tables

On peut distinguer deux situations :
- ✓ *L'ajout d'enregistrements : on parle de concaténation.*
- ✓ *L'ajout de champs : on parle de jointure entre tables.*

I – Concaténation de tables :

Concaténer provient du latin « cum » (ensemble) et « catena » (chaîne), c'est donc l'action de relier deux chaînes informatiques pour en créer une nouvelle.

Nous disposons d'une deuxième table (exercice_91.csv) du même type que la première (exercice_88.csv), c'est-à-dire avec des noms, des prénoms, des dates de naissance et des sexes de personnes identiques et différentes.

Exercice 91 : concaténer les deux fichiers en utilisant le signe « + », en sachant que dans la seconde liste, il y a un doublon et un jumeau d'une personne de la première liste.

II – Jointure de tables :

La jointure est l'opération permettant d'associer plusieurs tables n'ayant pas les mêmes champs. Le résultat de l'opération est une nouvelle table.

Exercice 91 bis : tester le code ci-dessous.

```
def fusion(table1, i1, table2, i2) :
    rep = table1
    for ligne1 in table1:
        for ligne2 in table2:
            if ligne1[i1] == ligne2[i2]:
                for i in range(len(ligne2)):
                    if i != i2:
                        ligne1.append(ligne2[i])
    return rep

t1 = [['France', 68000000, 672000],['Espagne', 47000000, 506000]]
t2 = [['français', 'France'],['italien','Italie']]

print(fusion(t1,0,t2,1))
```

PARTIE F – ARCHITECTURE MATERIELLE ET SYSTEME D'EXPLOITATION
Chapitre 28
Architecture de Von Neumann

I – Généralités :

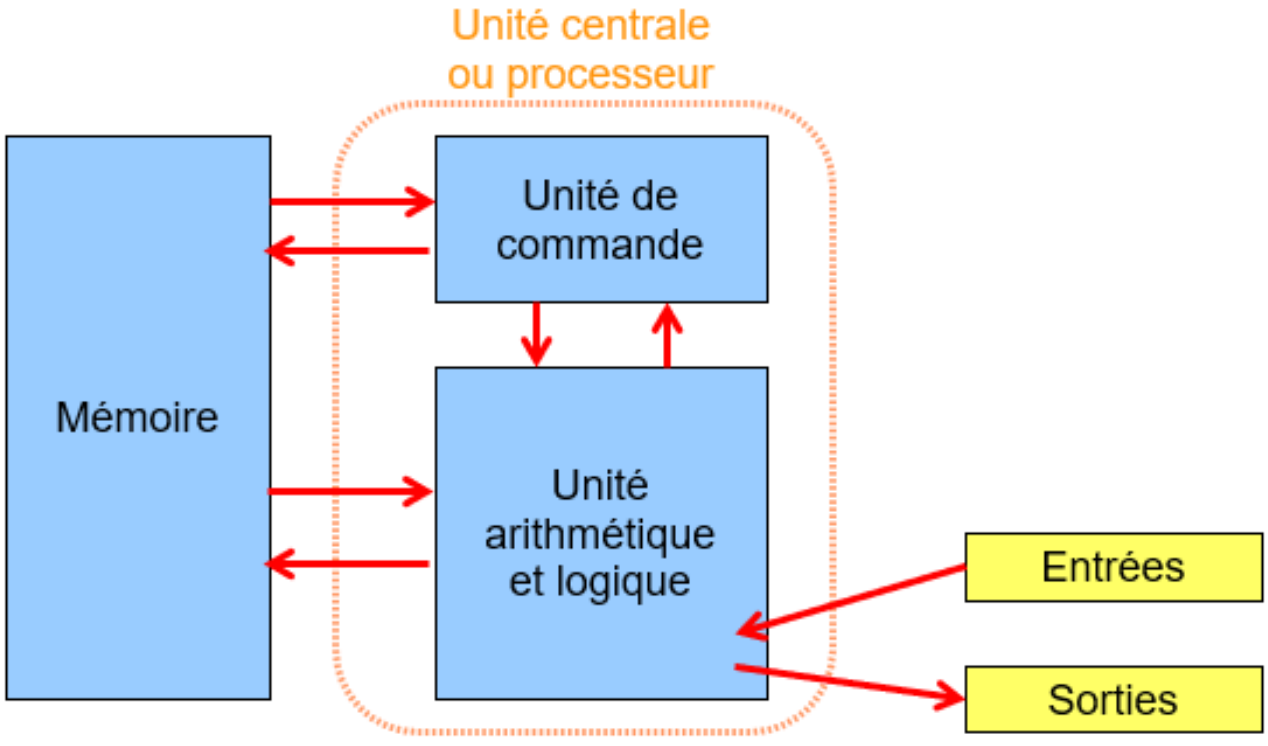
Un ordinateur est une machine électronique qui fonctionne par la lecture séquentielle d'un ensemble ..... organisées en programmes, qui lui font exécuter des opérations ..... et ..... sur des nombres binaires.

Un ordinateur :

- ✓ est électronique (par opposition à .....)
- ✓ est numérique (par opposition à .....)
- ✓ est programmable : .....
- ✓ peut exécuter les quatre opérations élémentaires : .....
- ✓ peut exécuter des programmes enregistrés en mémoire : .....

II – Le modèle de Von Neumann (1945) :

L'architecture générale des ordinateurs a été imaginée par John Von Neumann, mathématicien et physicien américano-hongrois

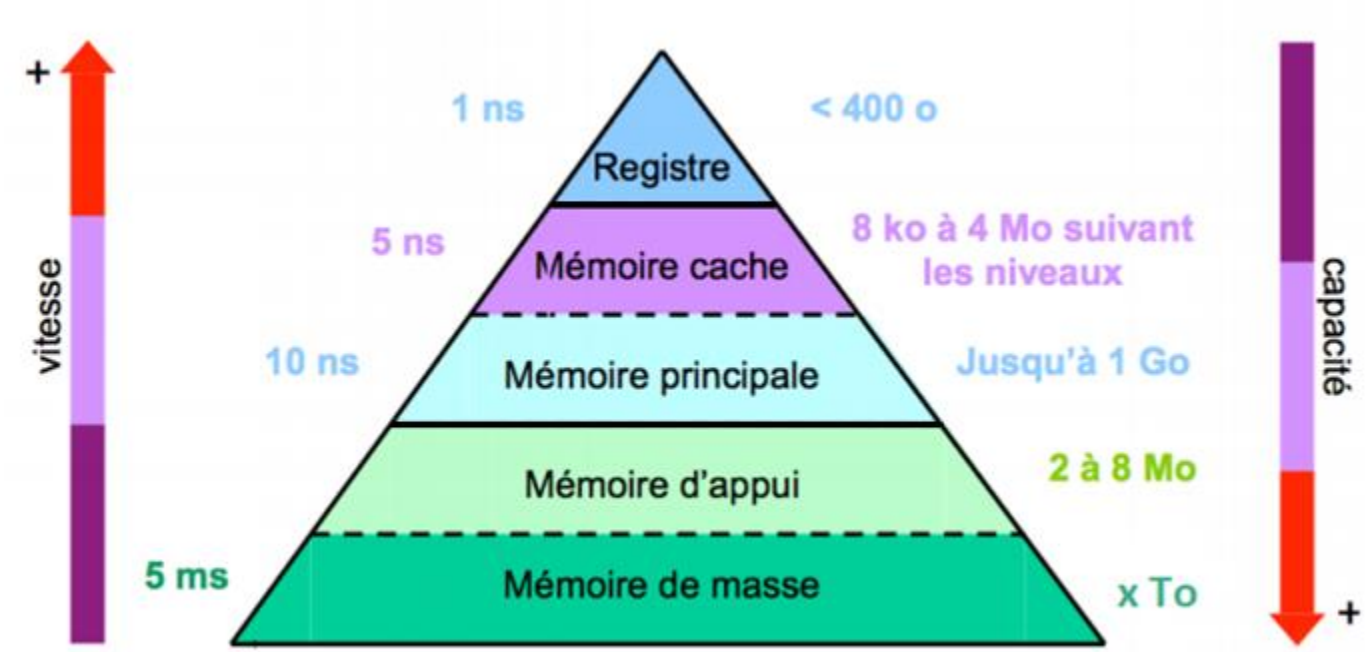


1) Mémoire :

Quelques définitions :

- ✓ Mémoire vive ou **RAM** (Random Access Memory) : mémoire volatile (perte dès que l'alimentation est coupée) accessible en lecture et en écriture (accès en 10 ns).
  Exemple : ……
- ✓ Mémoire morte ou **ROM** (Read Only Memory) : mémoire à l'origine non volatile utilisées pour stocker les programmes et informations de démarrage (accès en 150 ns).
  Exemple : ……
- ✓ Mémoire **flash** : compromis entre RAM (lecture & écriture) et ROM (non volatilité)
  Exemple : ……

On peut hiérarchiser les mémoires d'un ordinateur de cette façon :



- ✓ Registre : mémoire volatile du processeur permettant un accès ultra rapide à de courtes instructions ou des opérandes.
- ✓ Mémoire principale (ou centrale) : mémoire volatile permettant de mémoriser les données et les programmes lors de l'exécution de programmes.
- ✓ Mémoire de masse (ou physique) : mémoire non volatile (magnétique comme les HDD, optique comme les CD/DVD, flash comme les clés USB ou ROM comme le BIOS de la carte mère) permettant un stockage à long terme.
- ✓ Mémoire cache et d'appui : mémoire intermédiaire tampon sauvegardant les instructions et les données les plus souvent utilisées, permettant de réduire les temps d'accès.

2) Unité de commande (ou de contrôle) du processeur :

C'est elle qui donne les ordres à toutes les autres parties de l'ordinateur.

Voici son fonctionnement :

- ✓ Elle cherche la prochaine instruction en ……
- ✓ Elle décode cette instruction codée en langage machine (codée en ……)
- ✓ Elle fait exécuter cette instruction par ……

3) Unité Arithmétique et Logique (UAL) du processeur :

Une UAL de base permet de réaliser les opérations suivantes :

- ✓ Les fonctions logiques : ……
- ✓ Les instructions arithmétiques : ……
- ✓ Les opérations de comparaison (contrôle de séquence) : >, < et ……
- ✓ Transfert de données : ……

4) Entrée et sorties (unités d'échange) :

Ces périphériques permettent d'interfacer la machine avec l'environnement externe.

Exemples : ……

5) Les bus de communication :

On distingue généralement 3 bus :

Bus	D'adresse	De données	De contrôle
Description	Il indique la position d'une case dans la mémoire	Il fait circuler les données et les programmes entre les différentes unités	Il assure la synchronisation des flux sur les autres bus
Unidirectionnel ou bidirectionnel ?			

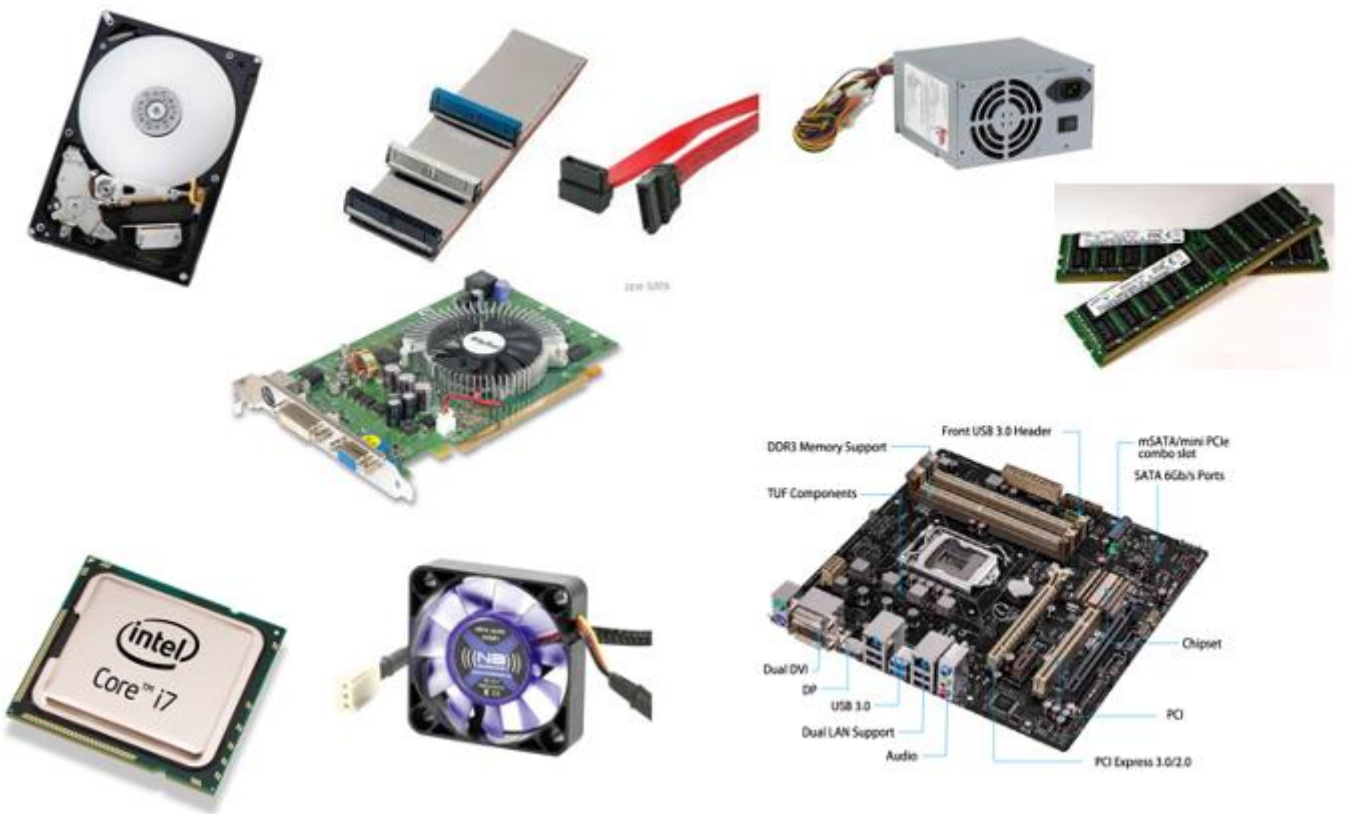
III – Evolution vers les modèles actuels :

Plus de soixante ans après son invention, le modèle d'architecture de Von Neumann régit toujours l'architecture des ordinateurs. Par rapport au schéma initial, on peut noter deux évolutions :

- ✓ Depuis le début des années 1960 les entrées/sorties sont sous le contrôle de processeurs autonomes. Associée à la multiprogrammation (partage de la mémoire entre plusieurs programmes mis en place lors de la 3e génération d'ordinateur 1963-1971), cette organisation a notamment permis le développement des systèmes en temps partagé (amélioration de la multiprogrammation).
- ✓ Les ordinateurs comportent maintenant plusieurs processeurs ou plusieurs "cœurs" dans une même puce. Ceci permet d'atteindre une puissance globale de calcul élevée sans augmenter la vitesse des processeurs individuels, limitée par les capacités d'évacuation de la chaleur.

Ces deux évolutions ont pour conséquence de mettre la mémoire au centre de l'ordinateur plutôt que le processeur, et d'augmenter le degré de parallélisme dans le traitement et la circulation de l'information. Mais elles ne remettent pas en cause les principes de base que sont la séparation entre traitement, commande et la notion de programme enregistré.

Exercice 92 : légender le schéma ci-dessous.



PARTIE F – ARCHITECTURE MATERIELLE ET SYSTEME D'EXPLOITATION
Chapitre 29
Langage assembleur

I – Les langages de programmation :

✓ **L'homme utilise des langages de haut niveau, de plus en plus proches du langage naturel.**

Exemples de langages : ……

Exemples de mots-clés : ……

✓ **Le compilateur ou l'interpréteur traduit le code source de haut niveau (d'abstraction) en code assembleur. Il contient des instructions assez rudimentaires.**

Exemples de mots-clés : cmp pour comparer, ……

✓ **Ce code assembleur est ensuite traduit bit à bit en langage machine, seul langage que connait le processeur.**

Exemples de mots-clés : ……

II –AmilWeb (Assembleur Miniature pour l'Informatique de Licence) :

Vous allez utiliser le simulateur ici : http://www.physiquechimie.org/amilweb
(si le lien ne fonctionne pas : http://www.chimiephysique.free.fr/amilweb)
!!! Sur Github ou lipn.univ-paris13.fr, l'application JS contient des bugs !!!

Exercice 93 : simulation du copier / coller

Ecrire un programme qui lit un nombre en entrée (en mémoire) et le restitue en sortie (en mémoire). La valeur du registre sera effacée.

Exercice 94 : calcul d'une fonction affine

Lire dans l'ordre la valeur de 'a', 'b' et 'x', et rendre ensuite le résultat y=ax+b.

Exercice 95 : simulation d'une inversion

Saisir (valeur) deux valeurs dans r0 et r1 et échanger leurs valeurs en se servant de r2, puis en se servant de la mémoire au lieu de r2.

Exercice 96 : simulation du if

Ecrire 3 à l'emplacement mémoire 12. Lire un nombre placé en mémoire 10. Si ce nombre est plus grand ou égal à 3, l'écrire à l'emplacement mémoire où le 3 est écrit.

Exercice 97 : simulation du if puis du else

Modifier le programme de l'exercice 96 afin qu'il réponde 0 (faux) ou 1 (vrai) à la condition.

Exercice 97 bis : simulation du if, du elif et du else

Modifier le programme de l'exercice 97 afin qu'il affiche 0 en cas d'égalité entre 3 et le nombre placé en mémoire 10, 1 s'il est supérieur et -1 s'il est inférieur.

Exercice 98 : calcul d'une puissance

Insérer une valeur x en r0 et une valeur n en r1. Le programme doit écrire en ligne 10 la valeur de x à la puissance n.

PARTIE F – ARCHITECTURE MATERIELLE ET SYSTEME D'EXPLOITATION
Chapitre 30
Introduction à Linux et aux commandes

I – Présentation :

Qu'est-ce qu'UNIX ?

Unix est un système d'exploitation créé en 1969 par des ingénieurs de Bell Labs.

Pourquoi UNIX ?

Les ingénieurs souhaitaient un OS fiable, multi-tâches (révolutionnaire à l'époque), multi-utilisateur (révolutionnaire également) qui puissent effectuer des calculs scientifiques tout en tournant 24h/24.

Quels sont les différents formats ?

Format propriétaire : spécifications contrôlées par des intérêts privées (brevets) et souvent payant.
Format Open Source : l'utilisateur a accès aux codes sources mais modification interdite.
Format libre : l'utilisateur peut exécuter, modifier, améliorer et redistribuer le programme.

Qu'est-ce que GNU/Linux ?

GNU (1984 par Richard Stalman) et Linux (1991 par Linus Torvalds) dont l'objectif était de créer un UNIX libre (**G**NU is **N**ot **U**nix).

Quelle est la différence entre GNU et Linux ?

Linux est le **noyau** : il s'occupe des basses besognes (gestion de la mémoire, accès aux périphériques, le partage du microprocesseur…).
GNU est un ensemble de **programmes utilitaires** : le compilateur C gcc, l'archivage tar, les modes d'emplois man…
GNU/Linux (surnommé Linux 😉) est un **système d'exploitation** : il permet de travailler, tout comme Windows 11 (noyau Windows NT), MacOS X (noyau XNU), Android (noyau Linux)…

Et Ubuntu ?

GNU/Linux étant gratuit, différentes sociétés l'ont repris et complété afin de distribuer un OS à leur goût : c'est ce qu'on appelle les **distributions**.

II – Lancement d'un terminal :

Un **terminal** permet de taper des commandes UNIX ou des noms de vos propres scripts (en binaires exécutables) car c'est un interpréteur de commande (aussi appelé **Shell**).

Par défaut, vous vous retrouvez dans votre « home directory », à la racine des dossiers de votre compte utilisateur.

Pour avoir un accès à un GNU/Linux, vous avez 5 possibilités :

1. Vous avez un PC ou un portable avec un dualBoot Ubuntu : utilisez-le !
2. Si vous êtes interne, vous pouvez accéder à Ubuntu : lors de l'écran de connexion au réseau avec VMware, il faut quitter VMware puis se connecter avec son login classique sous Linux.
3. Télécharger et installer Oracle VM VirtualBox (gratuit). Nouvelle > Linux & Ubuntu (64-bit) : il construit une machine virtuelle (environ 10 Go de place).
4. Utiliser le simulateur en ligne https://repl.it/languages/bash (Bash est l'acronyme de Bourne-Again-SHell, qui a été le premier shell d'Unix, écrit par Stephen Bourne). Attention, tout l'exercice 99 ne fonctionne pas online (comme man).
5. Sous W10, activer le support Linux (Paramètres > Programmes > Activer des fonctionnalités Windows > Cocher « Sous-système Windows Linux »). Redémarrer la machine. Aller dans le Microsoft Store et rechercher WSL (Ubuntu 20.04 LTS par exemple).

III – Syntaxe des commandes UNIX :

Commande (-option) (argument1) (argument2) (argument3)

Exemples :

- ✓ **ls** liste le contenu du dossier courant (0 option et 0 argument).
- ✓ **du -k** affiche la place occupée par les fichiers et dossiers du dossier courant (1 option et 0 argument).
- ✓ **cp -R monRepertoire nouveauRepertoire** copie un dossier de manière récursive (avec les sous-dossiers) vers un autre dossier (1 option et 2 arguments).
- ✓ **cd Documents** (*Change* Directory) se positionne dans le sous-dossier Documents (0 option et 1 argument).
- ✓ **ls -al** liste tous les fichiers, mêmes cachés (all), avec les détails comme son propriétaire ou ses droits (longue) (2 options et 0 argument).

Attention, les arguments sont parfois appelés « paramètres » sur Internet.

IV – Les caractères spéciaux pour les noms de fichiers :

Caractère	Signification	Exemples
/	Séparateur de dossier	/ : dossier racine /usr/bin : sous-dossier bin du dossier usr
~	Racine des dossiers	~/toto : fichier ou dossier toto à la racine
.	Dossier courant	. : dossier courant (ou alors ./) ./toto : fichier ou dossier toto du dossier courant
..	Dossier supérieur	.. : dossier parent (ou alors ../) ../bin : bin est le fils du dossier parent
?	N'importe quel caractère mais une seule fois	to?o : toto, toyo, toro…
*	N'importe quel caractère Mais 0, 1 ou plusieurs fois	to*o : too, toto, toyo, toro, tofgrto…

V – Principales commandes UNIX :

Nom	Description	Options	Arguments
cal	Affichage du calendrier		1983
cd	Se positionne sur le dossier désigné		dossier
chmod	change les permissions d'écriture	-R récursivité sur tous les sous-dossiers	Mode Fichier/dossier
cp	Copie	-i : demande confirmation -r : copie récursive	Fichier source Dossier destination
env	Affiche variables d'environnement		
printenv	Affiche une variable d'environnement		Variable
file	Retourne info sur fichier spécifié		Fichier
find	Recherche		Nom recherché
gcc	Compilateur C	-c : compilation seule -o : nom fichier sortie	Fichier source à compiler
gunzip	Décompression		Fichier à décompresser
gzip	Compression	-9 : niveau compression max	Fichier à compresser
hexdump	Affiche au format hexadécimal	-C : affiche aussi au format texte	Fichier
ls	Liste le contenu d'un dossier	-a : avec dossiers cachés -l : info détaillées	Dossier
man	Mode d'emploi d'une commande		Nom de la commande
mkdir	Crée un dossier		Dossier
more	Liste le contenu d'un fichier page par page		Fichier
mv	Déplace	-i : demande de confirmation	Fichier ou dossier source Dossier de destination
	Renomme	-i : demande de confirmation	Fichier ou dossier source Nouveau nom
pwd	Nom absolu du dossier courant		
rm	Suppression	-f : forçage -r : récursif	Fichier/dossier
rmdir	Suppression dossier vide		Dossier vide
top	Infos sur utilisation ressources système		

VI – A vous de travailler :

Exercice 99 : pour chacune des actions suivantes, indiquer la commande UNIX utilisée et donner le résultat de la commande.

Si erreur d'enregistrement pour fichier lors de la 9e étape : cd /partages/priv puis nettoyage_linux

N°	Actions	Commande	Résultat
01	Indiquez le dossier où vous vous trouvez		
02	Affichez l'aide de la commande *mkdir* (sortie de l'affichage avec q) *(sous Bash, utilisez commande --help)*		
03	Créez un nouveau dossier « calendriers »		
04	Allez dans ce dossier		
05	Affichez les calendriers de 1998 et 2018 et trouvez les jours du 12/07/98 et 15/07/18		
06	Exécutez et expliquez la commande suivante : cal 1984 | more		
07	Créez les fichiers calendriers à l'aide de *cal aaaa > mon_nom.txt* **(bien respecter les espaces)**		
08	Listez tous les fichiers de ce dossier		
09	Avec un éditeur de texte, créez le fichier auto.sh comportant la seule ligne : *cal $1 | gzip > $1.txt.gz*		
10	Affichez l'aide de la commande *chmod*		
11	Rendez auto.sh exécutable avec la commande : *chmod u+x auto.sh*		
12	Utilisez le programme que vous venez de créer en utilisant la commande : *./auto.sh 2000*		
13	Listez tous les fichiers de ce dossier		
14	Décompressez le fichier 2000.*txt.gz* qui vient d'être créé		

15	Avec l'explorateur, affichez le contenu du fichier annee.txt et trouvez le jour du 02/07/00	
16	Expliquez ce qui s'est passé avec *./auto.sh* (*$1* « rapatrie » un argument extérieur)	
17	Listez uniquement les fichiers commençant par *20* (*ne pas utiliser ls*)	
18	Copiez le fichier *auto.sh* dans le dossier supérieur	
19	Déplacez le fichier *2000.txt* dans le dossier supérieur, puis le renommer	
20	Effacez tous les fichiers du dossier courant	
21	Remontez dans le dossier supérieur	
22	Supprimez le dossier vide *calendriers*	
23	Listez tous les fichiers du dossier	
24	Affichez l'ensemble des variables d'environnement	
25	Affichez uniquement le contenu de la variable *USER*	
26	Affichez le type des fichiers *auto.sh* et *2000.txt*	
27	Affichez les informations sur l'utilisation des ressources par le système	
28	Listez tous les fichiers du dossier avec des infos détaillés	rw- : lect/écri proprio r-- : lect seule groupe r-x : lect/exé autres
29	Exécutez et expliquez la commande suivante : *chmod a+x annee.txt*	
30	Vérifiez votre réponse en faisant un clic droit sur le fichier > Permissions	

PARTIE F – ARCHITECTURE MATERIELLE ET SYSTEME D'EXPLOITATION
Chapitre 31
Modèle OSI des réseaux

I – Un peu d'histoire :

Le réseau ARPANET, ancêtre d'Internet, date de **1969**. C'est le premier réseau qui a utilisé un système à base de **paquets** pour le transfert de données. La théorie mathématique a été développée par Leonard Kleinrock qui est considéré comme l'un des pères d'Internet avec Vinton Cerf (TCP/IP), Robert Kahn (TCP/IP) et Larry Roberts (implémentation d'ARPANET).

Le premier message est envoyé le 29 octobre 1969 entre l'université UCLA de Californie (équipe dirigée par L. Kleinrock) et l'institut de recherche de Stanford (équipe de Douglas Engelbart) : suite à l'envoi du mot « login », les deux premières ont bien été reçues mais le système a planté sur la 3e lettre.

II – Modèle OSI des réseaux :

1) Les différentes couches :

Conçu par l'ISO en 1984, l'Open Systems Interconnections (OSI) a créé un cadre à la problématique d'interconnexion des réseaux. Ils ont différencié 7 couches indépendantes, mais par souci de simplification, nous n'étudierons pas les couches 5 et 6.

Couche	Transmission	Exemples d'action	Matériel (1 à 3) ou passerelle (4 à 7)	Exemple de normes
1/ Physique	Bit à bit	Connectique	Répéteur Concentrateur (hub, désuet)	ADSL, câble coaxial, USB, Wi-Fi…
2/ Liaison	Trames	Transfert de données entre deux stations (même réseau)	Commutateur (switch)	Ethernet (avec adresse MAC)
3/ Réseau	Paquets	Adressage logique, interconnexion entre deux réseaux	Routeur	Internet Protocol (IP)
4/ Transport	Segments	Acheminer des segments et vérifier la bonne réception.	Relais de transport	Transmission Control Protocol (TCP)
~~5/ Session~~	~~Données de session~~	~~Etablir une session entre deux utilisateurs distants~~	~~Convertisseur de session~~	~~RPC~~
~~6/ Présentation~~	~~Données de présentation~~	~~Format de l'information transmise~~	~~Convertisseur de format~~	~~ASCII, UTF-8, JPEG, MPEG~~
7/ Application	Donnée d'application	Protocoles spécifiques aux applications	Proxy	HTTP, FTP, SMTP, IMAP, SSH…

Les règles de communications (**protocoles**) entre ordinateurs doivent se soumettre à certaines contraintes pour que les réseaux soient compatibles entre eux :

- ✓ **Communication** : chaque couche ne peut communiquer qu'avec la couche immédiatement inférieure ou supérieure.

- ✓ **Encapsulation** : chaque tâche est encapsulée dans une couche.

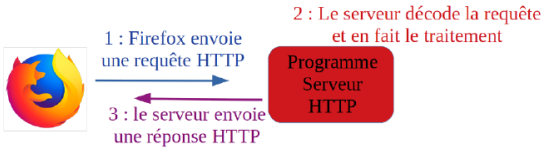
A part cela, les couches sont indépendantes : on peut changer le code interne d'une couche sans risques !

2) La couche application :

Son rôle est principalement de choisir le mode de transmission (ce sont des protocoles comme http, https, ftp, smtp…).

Nous retrouvons ici les notions du chapitre 5 : GET / POST…

Exemple : Votre navigateur Web (par exemple Firefox) veut communiquer avec le serveur HTTP servant le site www.physiquechimie.org : pour cela, les deux programmes (le client HTTP et le serveur HTTP) respectent un protocole commun : le http.



Les serveurs web utilisent une « fréquence d'écoute » pour dialoguer avec la machine distante appelée « **port** » : les plus connus sont les ports 80 (http) et 443 (https).

Mais ce n'est pas le programme FIREFOX lui-même qui va directement envoyer le message au serveur. Non, il va simplement envoyer son message (mis en forme en respectant HTTP) à la couche du dessous : la couche TRANSPORT.

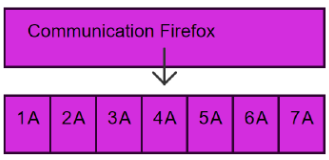
3) La couche transport :

Une fois choisi le mode de transport, cette couche est chargée de le mettre en œuvre. En gros, deux protocoles sont disponibles : UDP (User Datagram Protocol) et TCP (Transmission Control Protocol).

- ✓ Le plus connu, **TCP**, est un protocole **fiable**, qui permet l'acheminement sans erreur de données issues d'une machine à une autre machine. Son rôle est de fragmenter le message à transmettre de manière à pouvoir le faire passer sur la couche internet. A l'inverse, sur la machine destination, TCP replace dans l'ordre les fragments transmis sur la couche internet pour reconstruire le message initial.
- ✓
- ✓ **UDP** est en revanche un protocole plus **simple** que TCP. Son utilisation présuppose que l'on n'a pas besoin de la conservation de l'ordre de remise des paquets. Il n'y a pas vérification de l'arrivée de tous les paquets (très utile pour la transmission de vidéos...).

Voici ce que réalise le protocole TCP :

- ✓ Premièrement, elle **découpe le message** en plusieurs sous-messages si le message de base est trop gros.



✓ Ensuite, elle identifie l'émetteur (SRC) et le récepteur (DST) avec un identifiant, qui sera le PORT, un simple numéro encodé sur 2 octets (donc entre 1 et 65535).

✓ Enfin, elle rajoute un numéro pour ordonner chaque morceau (séquence).

Ces 3 informations représentent **l'en-tête** du segment. L'en-tête associée avec le sous-message s'appelle un **segment**.

A présent on a plein de segments dont on connaît l'expéditeur et le destinataire. Mais comment trouver la bonne machine ? C'est simple : la couche TRANSPORT ne sait pas le faire. Alors elle délègue à la couche RESEAU qu'on nomme également couche INTERNET.

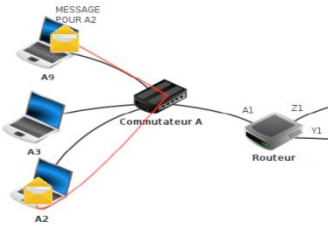
4) La couche réseau (ou internet) :

Cette couche réalise l'interconnexion des réseaux et ce à l'aide du protocole IP (Internet Protocol). Elle permet d'acheminer les données au bon destinataire dans le réseau, en laissant aux couches supérieures le soin de les réordonner (TCP) et de les interpréter (Application).

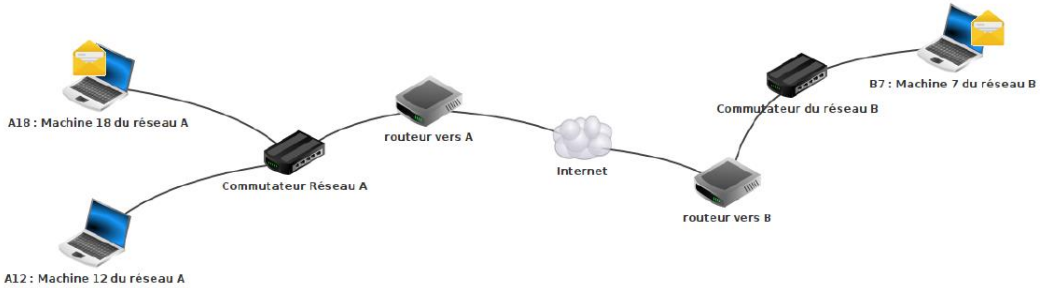
Cette couche est considérée comme un aiguilleur. Elle se charge de savoir si le message est à destination :

✓ de la **même machine** (ici A9 à un message pour A9) : elle va envoyer le message vers la couche APPLICATION.

✓ d'une machine qui **appartient au même réseau** que la machine elle-même (ici A9 à un message pour A2): on sait alors qu'on peut envoyer le message à la couche RESEAU du destinataire via le réseau interne.



✓ d'une machine qui **n'appartient pas au même réseau** (ici A9 à un message pour B7): on sait qu'il faut envoyer le message vers un réseau externe.



Ainsi, la couche **RESEAU** reçoit les segments que la couche **TRANSPORT** lui a fournis. Elle ne les envoie pas directement : les segments ne contiennent pas l'adresse IP du destinataire !
On prend donc le segment et on lui rajoute un **en-tête IP** : cela devient un **paquet IP**.

L'en-tête IP est un ensemble d'information qu'on va placer devant le segment TCP. Comme l'en-tête TCP, il s'agit **d'informations encodées** sur un nombre spécifique d'octets et décodables facilement. Le contenu exact de l'en-tête va dépendre du système d'adressage utilisé : IPv4 ou IPv6 par exemple.

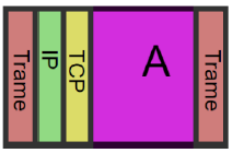
Cet en-tête contient :

- ✓ L'adresse IP (adresse logique) destination (4 octets en IPv4, 16 octets en IPv6). On la place en premier car c'est la première chose qu'un routeur doit lire.
- ✓ L'adresse IP source (4 octets en IPv4, 16 octets en IPv6). Sans cela, on ne pourrait pas répondre au message.
- ✓ Un compteur nommé TTL (Time To Live) en IPv4 ou Hop Limit en IPv6 (1 octet dans les deux cas) : c'est un compteur qui décroit de 1 à chaque fois que le paquet est transféré par un routeur. Arrivé à 1, le paquet n'est plus déplacé et est envoyé à la poubelle.

5) La couche liaison (ou accès réseau) :

Le principe est de rajouter encore un en-tête devant le paquet IP pour créer ce qu'on nomme une trame. Mais on va aussi rajouter des choses à la fin cette fois, de façon à entourer le paquet IP.

Le paquet IP contient les informations qu'on veut faire transiter d'un ordinateur à un autre.
La trame est le « carton d'emballage ».



L'en-tête de la trame contient essentiellement :

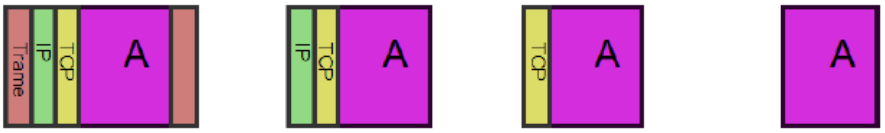
- ✓ L'adresse **MAC** (Media Access Control) liée à la carte réseau utilisée (adresse physique) sous 6 octets x 2 (source et destination).
- ✓ Le paquet IP est entouré d'une séquence signalant **le début et la fin** du signal envoyé. Comme cela, le récepteur sait qu'il reçoit bien un vrai signal et pas juste du bruit.

6) La couche physique :

La couche physique (par exemple du cuivre pour l'ADSL) transforme la trame reçue composée de 0 et 1 numérique **en signaux électrique** pour l'envoyer sur le câble. La trame quitte la carte réseau du PC.

7) La décapsulation :

Une fois qu'on parvient à joindre la bonne machine de destination, on remonte dans les couches en lisant et supprimant un par un les en-têtes.
On passe ainsi de la couche LIAISON à RESEAU en supprimant l'en-tête de la trame, puis on remonte à la couche TRANSPORT (suppression de l'en-tête IP) et enfin à la couche APPLICATION (suppression de l'en-tête TCP).



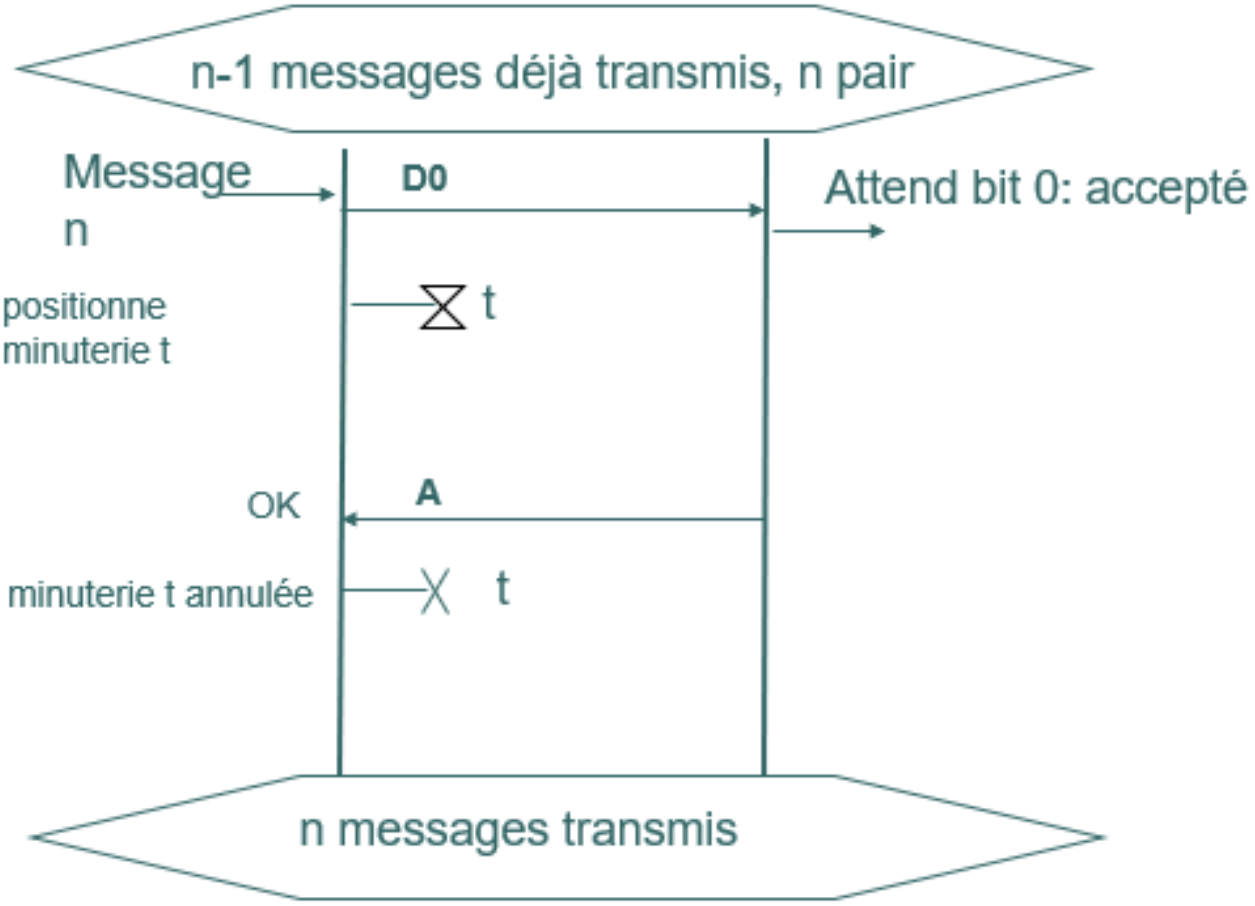
III – Technique de détection et de correction des erreurs de transmission :

1) Protocole du bit alterné (couche liaison) :

C'est un des plus simples protocoles de liaison de données, décrit par Bartlett et Scantleburry en 1969 ; il réussit à récupérer certaines erreurs de transmission mais a cependant certaines limites…

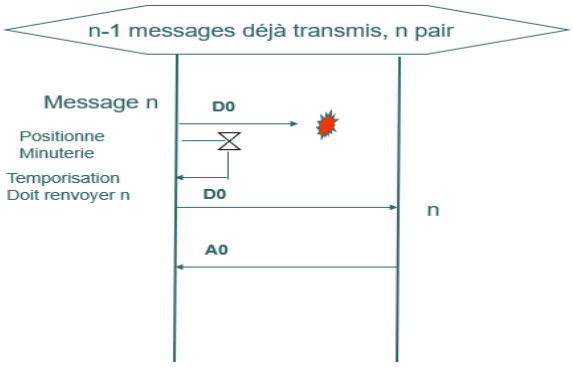
Pour ne pas gaspiller les bits pour identifier les messages, un compteur 1 bit est mis en place : 0 ou 1 (appelé FLAG = drapeau). L'expéditeur envoie des messages D et le récepteur renvoie un acquittement A pour chaque message reçu (ACK = ACKNOLEDGMENT = accusé de réception).

✓ **Cas général :**

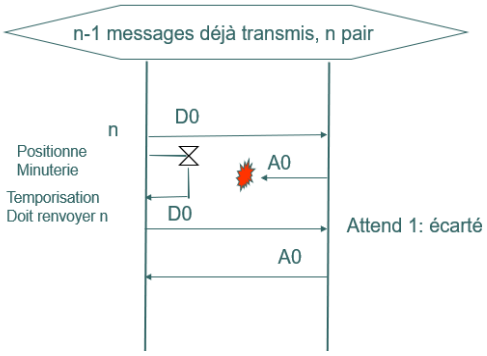


✓ **Correction d'erreurs :**

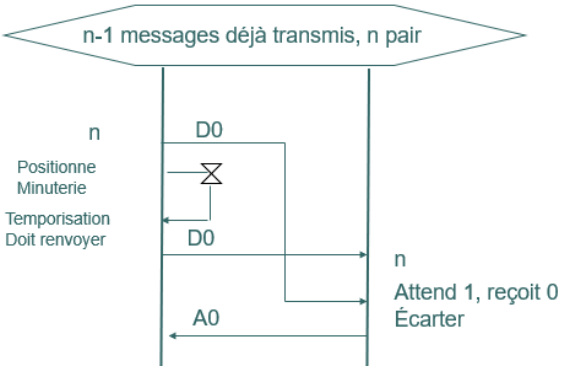
Cas de perte de message



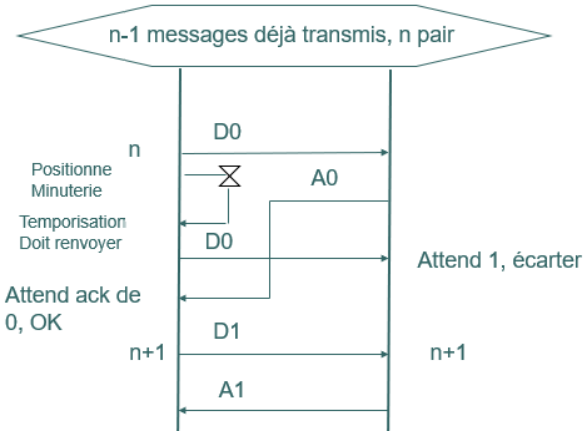
Cas de la perte d'acquittement



Chevauchement de messages



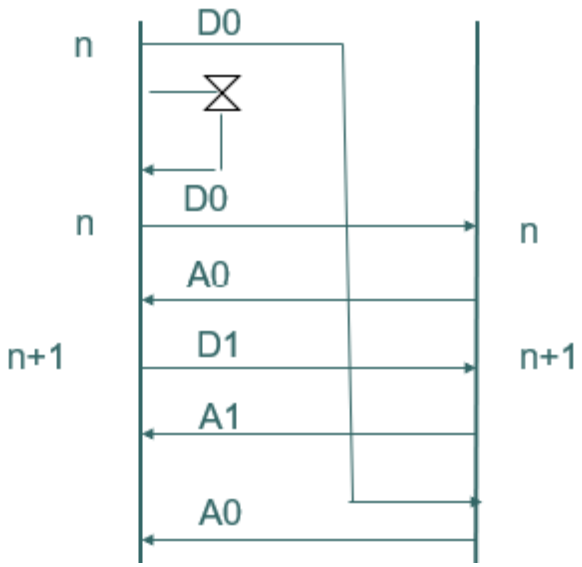
Retard d'acquittement (à compléter)



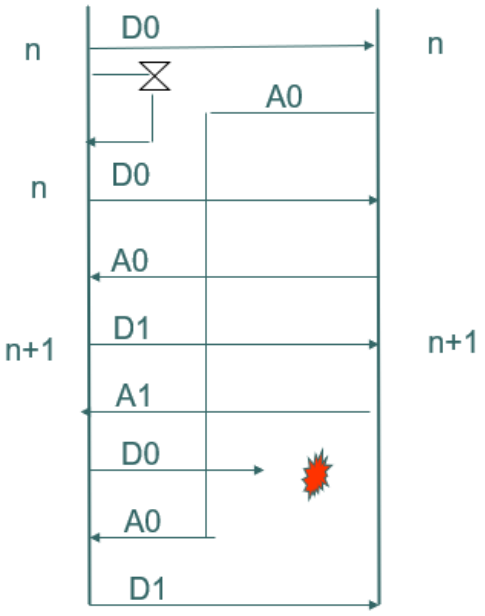
✓ **Limites :**

Exercice 100 : décrire les problèmes rencontrés.

Double chevauchement de messages



Double chevauchement d'acquittements



2) Une nouvelle approche (naïve), la répétition :

Une méthode simple pour détecter et corriger une erreur dans une suite de bits est d'y introduire une forme de redondance, en répétant chacun des bits plusieurs fois. Ainsi, au lieu de transmettre sur un réseau la suite de bits 10110110, on transmet la suite de bits 111000111111000111111000, où chaque bit est répété trois fois. Pour retrouver le message original, il suffit de lire les bits reçus trois par trois, en remplaçant les triplets 000 par des 0 et les triplets 111 par des 1.

Si l'un des triplets reçus n'est ni 111 ni 000, par exemple si c'est 010 ou 001, on peut être certain qu'une erreur s'est glissée. On peut même corriger l'erreur : les 0 étant majoritaires, le triplet original était sans doute 000, et l'on peut interpréter le triplet 100, 010 ou 001 par 0. Ce code permet donc de détecter et corriger toutes les erreurs, à condition qu'il y ait au plus une erreur par triplet. En revanche, si plusieurs erreurs sont commises sur le même triplet, elles peuvent passer inaperçues ou être mal corrigées. Cette méthode fonctionne assez bien.

Quel est le principal inconvénient de cette méthode ? ……

3) Code de parité croisée :

Dans certains cas, détecter les erreurs sans les corriger est suffisant. Par exemple, quand on transmet un message sur un réseau, la machine qui reçoit le message a souvent la possibilité de le redemander à celle qui l'a envoyé, s'il est erroné. Une manière peu coûteuse pour détecter des erreurs dans une suite de bits transmise est d'ajouter un bit de contrôle tous les 100 bits transmis, indiquant si le nombre de 1 dans ce paquet de 100 bits est pair (0) ou impair (1). La longueur des messages est ainsi augmentée de 1 % seulement. Si une erreur se produit lors de la transmission des 101 bits, c'est-à-dire si un 0 est remplacé par un 1, ou un 1 par un 0, la parité du nombre de 1 est changée et l'erreur est détectée.

Dans d'autres cas, il est nécessaire de corriger les erreurs. Par exemple, quand on lit un DVD et que l'on détecte une erreur, on veut pouvoir la corriger au vol et non demander au spectateur d'aller acheter un autre DVD pour voir la fin du film. Une méthode de correction des erreurs moins coûteuse que le triplement des bits décrit ci-avant consiste à ajouter seulement 20 bits de contrôle tous les 100 bits, de la manière suivante : on organise le paquet de 100 bits en une liste de 10 lignes sur 10 colonnes et on ajoute un bit de contrôle par ligne et un bit de contrôle par colonne, soit 20 bits au total. Ce bit indique simplement si le nombre de 1 dans la ligne ou la colonne est pair ou impair. Quand on reçoit le message, si on détecte une erreur dans la ligne l et une erreur dans la colonne c, on sait que le bit erroné est celui qui se trouve dans la liste à la ligne l et à la colonne c ; il suffit, pour corriger le message, de remplacer ce bit par un 1 si c'est un 0 ou par un 0 si c'est un 1. Si on détecte une erreur dans une ligne, mais aucune erreur dans les colonnes, ou le contraire, c'est que l'erreur porte sur le bit de contrôle lui-même et il n'y a donc rien à corriger dans le message. Cette méthode demande donc d'allonger le message de 20 % et elle permet de corriger toutes les erreurs à condition qu'une erreur au plus se produise dans chaque suite de 120 bits.

Exercice 101 : répondre aux questions du code de parité croisée 24, 16 :

Colonne de contrôle

0	**0**	**1**	**1**	
0	**1**	**0**	**1**	
1	**1**	**0**	**1**	
0	**1**	**1**	**1**	

Ligne de contrôle

1. Compléter les 8 bits de contrôle.
2. Est-il possible de reconstituer le message original si on reçoit la séquence suivante : 0??1 0?01 1101 0?11 001? 1100 : ……

0	**……**	**……**	**1**	0
0	**……**	**0**	**1**	0
1	**1**	**0**	**1**	1
0	**……**	**1**	**1**	……
1	1	0	0	

3. Montrer que le message suivant : 1010 0111 1001 0010 0001 0100, transmis suivant la même méthode, est incohérent. Expliquer cette incohérence. Comment y remédier ?

……

……

1	**0**	**1**	**0**	0
0	**1**	**1**	**1**	0
1	**0**	**0**	**1**	0
0	**0**	**1**	**0**	1
0	1	0	0	

4. Si un message comporte une erreur sur un bit de contrôle, quelle différence y-a-t-il avec le message précédent ?

……

……

IV – Adresse IP et masque de sous-réseau (couche 3) :

Une adresse IP (avec IP pour Internet Protocol) est un numéro d'identification qui est attribué de façon permanente ou provisoire à chaque périphérique relié à un réseau informatique qui utilise l'Internet Protocol. L'adresse IP est à la base du système d'acheminement (le routage) des paquets de données sur Internet.

Structure d'une adresse IP (IPv4)

- Une adresse IP est codée sur 32 bits (4 octets ou bytes) (IPv4)
- Une adresse IP est souvent présentée en 4 blocs de 4 octets : A.B.C.D, par exemple : IP = 192.168.1.55
- La plus petite adresse est : 0.0.0.0 et la plus grande :255.255.255.255
- Il y a 2^{32} adresses possibles.
- Une partie représente l'adresse de la machine et une autre celle du sous-réseau.
- En 2011, les adresses IPv4 sont épuisées : le développement des adresses en IPv6 (sur 128 bits) est accéléré par l'IANA (*Internet Assigned Numbers Authority*).

Elle est accompagnée du masque de sous-réseau qui est aussi de 4 octets : A'.B'.C'.D'
Par exemple : MASQ=255.255.255.0
C'est avec le masque de sous-réseau que l'on peut déterminer quelle est l'adresse du sous-réseau et quelle est l'adresse de la machine.

L'adresse de sous-réseau est obtenue avec l'adresse machine et le masque de sous-réseau :
L'adresse du réseau est donnée par : *IP* & *MASQ* (& est le AND bit à bit)

Remarque :
Un masque ne peut pas contenir n'importe quel nombre. Il doit forcément n'avoir que des 1 à gauche et que des 0 à droite.
255.255.240.0 = 11111111.11111111.11100000.00000000 est un masque valide
240.255.240.0 = 11110000.1111111.11110000.00000000 n'est pas un masque valide

Voici une IPv4 90.98.100.3 et son masque de sous réseau 255.255.255.0
C'est à dire en binaire :

IP	0101 1010.	0110 0010.	0110 0100.	0000 0011
& MASQ	1111 1111.	1111 1111.	1111 1111.	0000 0000
Adresse sous-réseau	0101 1010.	0110 0010.	0110 0100.	0000 0000
	90.	98.	100.	0

Exercice A :

Ecrire une fonction Python **strToInt(ipstr)** qui convertit une adresse IP écrite sous la forme d'une chaîne de caractères "A.B.C.D" en un entier : $Ax2^{24}+Bx2^{16}+Cx2^{8}+D$.
On peut utiliser la méthode **split(".")** pour casser la chaîne suivant les points.

Exercice B :

Écrire une fonction Python **intToStr(ipint)** qui fait le contraire de la fonction précédente.

Exercice C :

A l'aide des fonctions précédentes, écrire une fonction **adresseReseau(ip , masque)** qui renvoie l'adresse réseau sous la forme 'A.B.C.D' correspondant à cet IP et ce masque. Attention les entrées doivent pouvoir être écrites sous la forme 'A.B.C.D', c'est à dire sous la forme d'une chaîne de caractères.

Pour une IP et un masque donnés, il y a deux adresses réservées :

- Une pour le sous-réseau
- Une pour le broadcast (adresse de diffusion qui permet d'envoyer une trame à toutes les machines du sous-réseau).

On peut savoir combien il y a de machines dans un sous-réseau en comptant le nombre de bits à zéro dans le masque.
On obtient l'adresse broadcast en remplaçant les zéro du masque par des 1 dans l'adresse de réseau.
Par exemple, pour l'adresse 90.98.100.3/255.255.255.0, il y a 8 bits à zéro dans le masque donc il y a 2^8-2 machines possibles. L'adresse réseau est 90.98.100.0 et l'adresse broadcast est 90.98.100.255.

Exercice D :

Ecrire une fonction Python **NbrAdresses(masque)** qui renvoie le nombre d'adresses dans un sous-réseau.
Il faut compter le nombre de zéros dans le masque.

Remarque :
Il existe une autre notation pour les masques de sous-réseau : on donne l'IP suivie d'un slash et du nombre de bits à gauches utilisés pour l'adresse du sous réseau (notation CIDR).
Par exemple, 90.98.100.3/21 indique que le masque est 255.255.248.0 c'est-à-dire en binaire : 11111111.11111111.11111000.00000000.

Plages d'adresses possibles dans un sous-réseau

• La première adresse possible dans un sous-réseau est l'adresse du sous réseau.
• La dernière adresse possible dans un sous-reseau est l'adresse de broadcast.
• Donc les adresses machines sont toutes les adresses possibles entre ces deux valeurs.

Exemple :
Pour 115.250.207.3/255.255.255.248.
L'adresse réseau est 115.250.207.0, l'adresse de broadcast est 115.250.207.7 donc les adresses possibles de machines sont :
115.250.207.1
115.250.207.2
115.250.207.3
115.250.207.4
115.250.207.5
115.250.207.6

Exercice E :

Ecrire une fonction Python **plageAdresses(ip ,masque)** prenant en paramètres une adresse IP et le masque de sous-réseau et renvoyant la liste des adresses possibles pour les machines de ce réseau.

PARTIE F – ARCHITECTURE MATERIELLE ET SYSTEME D'EXPLOITATION
Chapitre 32
Interface Homme-Machine : PoppyTorso

I – Définitions :

Pour un ordinateur, on parle généralement d'entrée / sorties (voir Von Neuman). De manière générale, on parle de capteurs et d'actionneurs pour les systèmes embarqués comme des robots ou des objets connectés (IHM).

Un actionneur est chargé d'agir sur le monde réel.

Exemples : ……

Un capteur est un dispositif d'acquisition. Il obtient des informations du monde qui l'entoure et il permet de détecter un événement extérieur provoquant un changement d'état.

Exemples : ……

II – Utilisation du simulateur V-REP (1) et d'un notebook (2) :

V-REP (pour Virtual Robot Experimentation Platform) est un simulateur de robot, très utilisé dans le monde de l'ingénierie. Il est gratuit pour un usage éducatif.

Pour faire fonctionner notre robot, il faut utiliser Python mais pas seulement. Nous allons aussi avoir besoin de ce que l'on appelle une bibliothèque. La bibliothèque qui permet d'utiliser notre robot s'appelle Pypot et elle est entièrement écrite avec le langage Python. Cette bibliothèque a été construite par des chercheurs de l'INRIA (organisme public de recherche, dédié aux sciences et technologies du numérique) et nous allons simplement apprendre à l'utiliser : pip install poppy_torso (cmd d'Anaconda).

La première chose à faire est d'aller chercher dans la bibliothèque Pypot, les bons modules.

```
from pypot.creatures import PoppyTorso
```

Ensuite, vous allez créer un objet s'appelant nom_du_robot et étant un robot de type PoppyTorso. Vous pouvez donner le nom que vous souhaitez à votre robot. Il vous suffit d'écrire :

```
nom_du_robot = PoppyTorso(simulator='vrep') # En gras, partie modifiable
```

Attention, il faut désactiver rapidement le message d'erreur sinon le robot se « bloque ».

Comme toute chose en langage Python, notre robot poppy est un objet qui contient d'autres objets qui sont ses moteurs.

Vous devez donc accéder aux moteurs de poppy (qui se nomment "motors") et qui se trouve à l'intérieur de Poppy pour cela tapez :

```
nom_du_robot.motors
```

Tous les mouvements sont basés sur des rotations des moteurs situés aux articulations. Il suffit de fixer l'angle que l'on désire pour un moteur. Pour cela, nous pouvons utiliser la méthode :

goto_position(**angle_en_degrées,temps**)

Dans la syntaxe ci-dessus, angle_en_degrées doit être remplacé par une valeur entre 0 et 180. Le temps doit être remplacé par une durée en seconde que vous désirez donner au mouvement. Une durée longue (5) entraine un mouvement lent une durée courte (0.5) entraine un mouvement rapide.

Exercice 102 : A présent, choisissez un moteur au hasard dans la liste de moteurs obtenues précédemment et faites-le bouger pour le localiser sur le robot. Vous devez remplir le tableau suivant avec les noms des 10 moteurs :

Nom du moteur 01 : …… ; Nom du moteur 02 : ……

Nom du moteur 03 : …… ; Nom du moteur 04 : ……

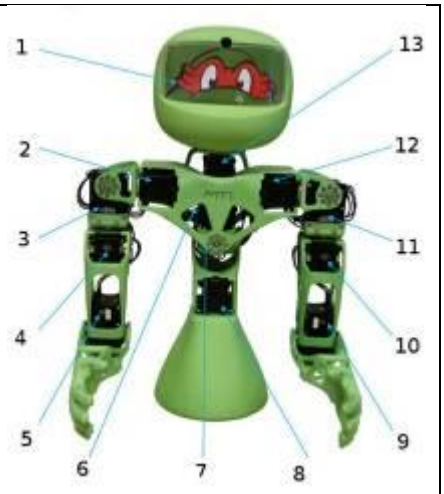
Nom du moteur 05 : …… ; Nom du moteur 06 : ……

Nom du moteur 07 : …… ; Nom du moteur 08 : ……

Nom du moteur 09 : …… ; Nom du moteur 10 : ……

Nom du moteur 11 : …… ; Nom du moteur 12 : ……

Nom du moteur 13 : ……



Si lors de vos essais, vous faîtes tomber votre robot, il est important de connaitre l'instruction qui permet de remettre la simulation à zéro :

nom_du_robot.reset_simulation()

Pour éviter que cela se reproduise, il faut fixer la base du robot :
- ✓ Double clic gauche sur la goutte à gauche de « base respondable ».
- ✓ Cliquer sur « Show dynamic properties dialog ».
- ✓ Décocher « Body is dynamic ».

Si votre robot ne répond plus et que vous ne comprenez pas pourquoi, le programme de contrôle du robot ou l'interface Python est peut-être hors service ; dans ce cas, il faut redémarrer l'interface Python et tout recommencer au début.
Une solution moins longue : il faut stopper la simulation (VREP) puis cliquer sur Kernel > Restart kernel avec le NoteBook.

Enfin, voici un script pour remettre tous les moteurs à zéros :

```
for m in bob.motors:
    m.goto_position(0,1)
```

Exercice 103 : vous devez mettre les bras de votre robot à l'horizontale.

Vous avez sans doute remarqué que les mouvements de tous les moteurs s'exécutent en même temps, en simultané. Il peut être utile de décomposer les mouvements. Par exemple, pour mettre les bras à l'horizontale : bouger d'abord les épaules puis ensuite les coudes. Pour faire cela, il faut rajouter à la méthode goto_position() un argument wait='True' :

nom_du_robot.**nom_du_moteur**.goto_position(**angle_en_degrées**,**temps**,wait=True)

Pour faire parler le robot, voici la commande :

```
import os
os.system('espeak -p 70 -a 200 -s 100 - v fr-fr "Mon blabla ici"')
```

Voici les paramètres que vous pouvez modifier :

- ✓ -p : son aigue / grave
- ✓ -a : puissance
- ✓ -s : débit
- ✓ -v : détection de la langue

Pour que le robot puisse jouer un fichier mp3, voici la commande :

```
import os
os.system('mpg321 mon_fichier.mp3 &')
```

- ✓ Le fichier doit se trouver dans le même dossier que votre notebook.

Exercice 104 : vous devez mettre les bras de votre robot à l'horizontale en bougeant d'abord les épaules, puis ensuite les coudes.

Exercice 105 : les bras sont à l'horizontale, remettez les dans leur position de départ, c'est-à-dire avec les angles des moteurs à 0 degrés.

Pour terminer la simulation, il faut arrêter le robot :

```
nom_du_robot.close()
```

III – Utilisation d'un véritable robot :

Tout le code développé à l'aide du simulateur doit normalement être valide sur un véritable robot.

Il suffit d'instancier la class robot sans l'argument du simulateur :

```
nom_du_robot = PoppyTorso()
```

Attention dans le cas du contrôle d'un véritable PoppyTorso, le code doit être exécuté dans une interface Python qui pointe sur le nom réseau du robot et non pas sur localhost (ordinateur local) comme pour le simulateur.

Si la connexion Wifi rencontre des difficultés, voici une solution :

- ✓ Ouvrir le logiciel Tftpd32 et ouvrir l'onglet DHCP.
- ✓ Rechercher l'adresse IP du robot (MAC C2.22.09…).
- ✓ Par défaut, elle devrait être 192.168.1.101

Avec le nouveau routeur en X106, voici l'adresse IP fixe du robot :

- ✓ 10.11.150.25

PARTIE G – ALGORITHMIQUE
Chapitre 33
Parcours séquentiel d'un tableau quelconque

I – Définitions :

- ✓ **Algorithme** : succession d'instructions permettant d'aboutir à un résultat souhaité.
- ✓ **Coût ou complexité** : nombre d'opérations élémentaires (arithmétiques ou logiques) ainsi que d'affectations nécessaires à son exécution.
- ✓ **Coût ou complexité linéaire** : si la liste est de taille n, le coût sera proportionnel à n.
- ✓ **Coût ou complexité quadratique** : si la liste est de taille n, le coût sera proportionnel à n^2.

II – Recherche d'une occurrence :

Exercice 106 : créer une liste avec 100 valeurs tirées au hasard entre 1 et 100. Ecrire un algorithme qui recherche si la valeur 50 est présente dans la liste et renvoie le booléen True dans ce cas ainsi que le premier 50 rencontré, et qui renvoie False sinon (sans utiliser la méthode « .index() »).

1. En supposant que la dernière valeur de la liste est la valeur trouvée (pire cas) et si on utilise un booléen faux qui devient vrai, combien y-a-t-il d'affectations ? ……
2. Combien y-a-t-il d'itérations dans la boucle ? ……
3. Combien y-a-t-il de comparaisons ? ……
4. Quel est le coût de cet algorithme ? ……
   ……

	Meilleur cas	Cas moyen	Pire cas
Coût en temps / Complexité temporelle			

III – Recherche d'un extremum :

Exercice 107 : en utilisant la même liste, rechercher un extremum (sans 'min' et 'max').

5. Combien y-a-t-il d'itérations dans la boucle ? ……
6. Combien y-a-t-il de comparaisons ? ……
7. Combien y-a-t-il d'affectations au maximum ? ……
8. Quel est le coût de cet algorithme ? ……
   ……

	Meilleur cas	Cas moyen	Pire cas
Coût en temps / Complexité temporelle			

PARTIE G – ALGORITHMIQUE
Chapitre 34
Tri par sélection

I – Définitions :

- ✓ **Terminaison** : on prouve que les boucles sont finies (l'algorithme se termine).
- ✓ **Variant de boucle** : expression dont la valeur varie à chacune des itérations de la boucle.
  Exemple : quand on calcule la somme S des entiers de 1 à n, S est un variant de boucle.
- ✓ **Invariant de boucle** : propriété qui, si elle est vraie avant une itération, demeure vraie après l'exécution de l'itération.
  Mode d'emploi : on utilise souvent le raisonnement par récurrence (on vérifie si la propriété est vraie au rang 1, ensuite on suppose que la propriété est vraie au rang n et on vérifie qu'elle est encore vraie au rang n+1).
- ✓ **Correction d'un algorithme** : on prouve qu'un algorithme produit bien le résultat attendu, souvent en utilisant un « invariant de boucle ».

II – Principe :

L'idée du tri consiste à chaque étape à rechercher le plus petit élément non encore trié et à le placer à la suite des éléments déjà triés. On peut l'utiliser pour trier les livres d'une bibliothèque.

A une étape i, les i – 1 plus petits éléments sont en place, et il nous faut sélectionner le $i^{ème}$ élément à mettre en position i.

III – Terminaison, correction et coût :

1) Terminaison :

9. Pourquoi la première boucle se termine-t-elle forcément ? ……
10. Pourquoi la seconde boucle se termine-t-elle forcément ? ……

2) Correction :

11. Pour prouver la correction, utiliser l'invariant de boucle : « pour chaque i, la liste est une permutation de la liste initiale et tous les éléments de la liste liste[i+1:n] sont supérieurs ou égaaux à tous les éléments de la liste liste[0:i+1] :

    ……

    ……

    ……

3) Coût :

12. Quel est le nombre de comparaisons ?

    ……

    ……

    ……

PARTIE G – ALGORITHMIQUE
Chapitre 35
Tri par insertion

I – Principe :

C'est le tri du joueur de cartes (notamment quand M. Fontaine joue au « chinois »).

On fait comme si les éléments à trier étaient donnés un par un, le premier élément constituant, à lui tout seul, une liste triée de longueur 1. On range ensuite le second élément pour constituer une liste triée de longueur 2, puis on range le troisième élément pour avoir une liste triée de longueur 3 et ainsi de suite... Le principe du tri par insertion est donc d'insérer à la $n^{ième}$ itération le $n^{ième}$ élément à la bonne place.

II – Terminaison, correction et coût :

1) Terminaison :

13. Pourquoi la première boucle se termine-t-elle forcément ? .....
14. Pourquoi la seconde boucle se termine-t-elle forcément ? .....

2) Correction :

15. Pour prouver la correction, utiliser l'invariant de boucle : « pour chaque i, la liste est une permutation de la liste initiale et la liste liste[0:i+1] est triée :

    .....

    .....

    .....

    .....

3) Coût :

16. Si les valeurs sont dans l'ordre décroissant, quel est le nombre de comparaisons ?

    .....

    .....

    .....

    .....

PARTIE G – ALGORITHMIQUE
Chapitre 36
Algorithme des k plus proches voisins

I – Principe :

L'algorithme des k plus proches voisins (« *k nearest neighbbors* » ou KNN) est un algorithme d'apprentissage automatique, souvent appelé, même en français, algorithme de machine learning. L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème.

Cette idée d'apprentissage automatique ne date pas d'hier, puisque le terme de machine learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959.

Pourquoi le machine learning est tant "à la mode" depuis quelques années ? Simplement parce que le nerf de la guerre dans les algorithmes de machine learning est la qualité et la quantité des données (les données qui permettront à la machine d'apprendre à résoudre un problème). Or, avec le développement d'internet, il est relativement simple de trouver des données sur n'importe quel sujet (on parle de "big data").

À noter aussi l'importance des stratégies mises en place par les GAFAM (Google, Apple, Facebook, Amazon et Microsoft) afin de récupérer un grand nombre de données concernant leurs clients. Ces données sont très souvent utilisées pour "nourrir" des algorithmes de machine learning (comment, d'après vous, Amazon arrive à proposer à ces clients des "suggestions d'achats" souvent très pertinentes ?)

II – Exemple étudié :

En 1936, le botaniste américain Edgar Anderson (1897-1969) a collecté des données sur 3 espèces d'iris : "iris setosa", "iris virginica" et "iris versicolor".

Iris setosa

Iris virginica

Iris versicolor

	A	B	C
1	petal_length	petal_width	species
2	1.4	0.2	0
3	1.4	0.2	0
4	1.3	0.2	0
5	1.5	0.2	0
6	1.4	0.2	0
7	1.7	0.4	0
8	1.4	0.3	0
9	1.5	0.2	0
10	1.4	0.2	0
11	1.5	0.1	0

Extrait du fichier csv

Pour chaque iris étudié, Anderson a mesuré (en cm) :

- ✓ La largeur des sépales
- ✓ La longueur des sépales
- ✓ La largeur des pétales
- ✓ La longueur des pétales

Par souci simplification, nous nous intéresserons uniquement à la largeur et à la longueur des pétales. Pour chaque iris mesuré, Anderson a aussi noté l'espèce ("iris setosa", "iris virginica" ou "iris versicolor").

Dans le fichier CSV, vous trouverez 150 de ces mesures, avec :

- ✓ La longueur des pétales
- ✓ La largeur des pétales
- ✓ L'espèce de l'iris (au lieu d'utiliser les noms des espèces, on utilisera des chiffres : 0 pour "iris setosa", 1 pour "iris virginica" et 2 pour "iris versicolor")

III – Représentation graphique :

Exercice 108 : après avoir récupéré les fichiers, tester le code du fichier python.

17. Qu'obtient-t-on ?

    ……

    ……

18. Que fait plt.scatter ?

    ……

    ……

19. Observe-t-on des nuages de points bien distincts ?

    ……

    ……

IV – Première promenade en forêt :

Au cours d'une promenade, vous trouvez un iris. N'étant pas un spécialiste, il ne vous est pas vraiment possible de déterminer l'espèce. En revanche, vous êtes capable de mesurer la longueur et la largeur des pétales de cet iris.

Cet iris a un pétale de 0,5 cm de largeur et 2 cm de longueur.

Exercice 109 : rajouter la ligne "plt.scatter(2.0, 0.5, color='k')"

20. Quelle est l'espèce de cet iris ?

    ……

    ……

V – Seconde promenade en forêt :

Vous retournez cette fois en forêt, accompagné de votre crush, et vous avez l'occasion de cueillir un second iris pour ses beaux yeux. Cet iris a un pétale de 0,75 cm de largeur et 2,5 cm de longueur.

21. Quelle est l'espèce de cet iris ?

    ……

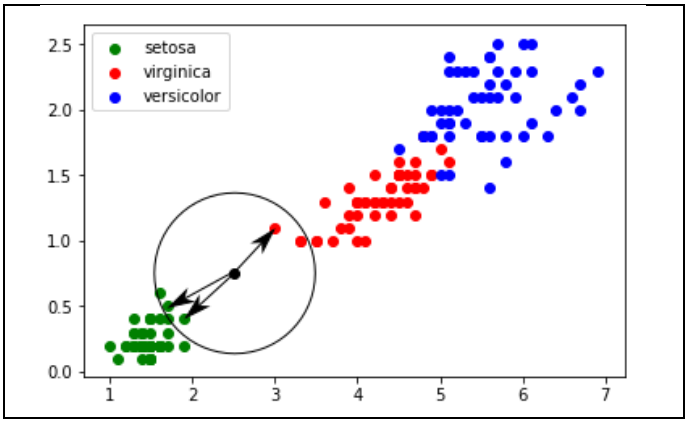
    ……

VI – Algorithme des k plus proches voisins :

Voici en quoi consiste cet algorithme :

- ✓ On calcule la distance entre notre point (largeur du pétale = 0,75 cm ; longueur du pétale = 2,5 cm) et chaque point issu du jeu de données "iris".
- ✓ On sélectionne uniquement les k distances les plus petites (les k plus proches voisins).
- ✓ Parmi les k plus proches voisins, on détermine quelle est l'espèce majoritaire. On associe à notre "iris mystère" cette "espèce majoritaire parmi les k plus proches voisins".

Prenons par exemple k = 3 :



Les 3 plus proches voisins sont signalés ci-dessus avec des flèches : nous avons deux "iris setosa" et un "iris virginica". D'après l'algorithme des "k plus proches voisins", notre "iris mystère" appartient à l'espèce "setosa".

La bibliothèque Python **Scikit Learn** propose un grand nombre d'algorithmes lié au machine learning (c'est sans aucun doute la bibliothèque la plus utilisée en machine learning). Parmi tous ces algorithmes, **Scikit Learn** propose l'algorithme des k plus proches voisins.

Exercice 110 : ouvrir le fichier et répondre aux questions.

22. Que fait la ligne 28 « d=list(zip(x,y)) » ?
    Elle permet de passer de deux listes à ……
23. Que fait la ligne 30 « train_test_split » ?
    Elle permet de séparer …… (size=0.5) les données qui seront utilisées pour la phase d'apprentissage et les données qui seront utilisées pour tester l'algo.
24. Que fait la ligne 31 « model.fit(d,lab) » ?
    Permet d'associer chaque tuple avec …… (phase d'apprentissage)
25. Que fait la ligne 32 « model.score» ? *(changer le size)*
    Permet de tester la machine sur les données de test …… (phase de test)
26. Quel est le type de la variable « prediction » ? (phase de prévision)
    ……
27. Tester différents k. Que remarquez-vous pour k = 5 ?
    ……
    ……
    ……
    ……
28. Tester l'algorithme avec un iris de votre choix (si vous n'avez pas d'iris sur vous, vous pouvez toujours inventer des valeurs !).
    ……
    ……
    ……

PARTIE G – ALGORITHMIQUE
Chapitre 37
Recherche dichotomique dans un tableau trié

I – Principe du problème :

Nous voulons savoir si l'élément 7 est dans la liste [1, 2, 5, 9, 10, 14, 17, 24, 41] et si oui, sa position.

Avec une recherche séquentielle (recherche par balayage), on parcourt la liste du début à la fin en comparant chaque valeur à l'élément recherché.

29. Dans le pire des cas, combien de comparaisons aura-t-on ? ……

Comme la liste de départ est triée, la **recherche dichotomique** permet d'améliorer la performance de la recherche.

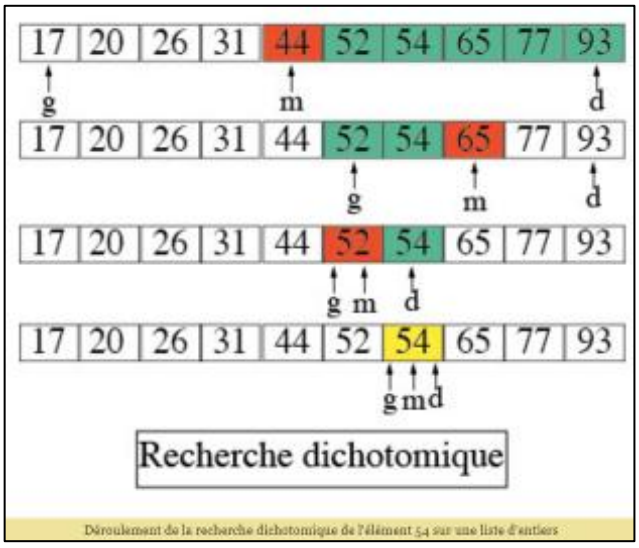
II – Principe de la recherche dichotomique :

Le fait que le tableau soit trié facilite la recherche d'un élément.

En effet, il suffit de comparer la valeur recherchée avec la valeur située au milieu du tableau. Si elle est plus petite, on peut restreindre la recherche à la moitié gauche du tableau. Sinon, on la restreint à la moitié droite du tableau.

En répétant ce procédé, on divise la zone de recherche par deux à chaque fois : c'est une application du principe « **diviser pour régner** » (méthode récursive) qui sera vu en terminale.

Très rapidement, on parviendra soit à la valeur recherchée, soit à un intervalle devenu vide.



Le projet 14 (le jeu du plus petit ou du plus grand) illustre également ce principe.

30. Compléter le nombre de tours de boucle maximum afin de trouver un élément dans une liste triée :

Taille de la liste	1	2	4	8	16	32	64	128	N
Recherche séquentielle									
Recherche dichotomique									Log_2 N + 1

La recherche dichotomique a donc un coût **…… binaire.**

Ne pas confondre avec le log en base 10 de la calculette avec $\log_{10}(x) = y \Leftrightarrow x = 10^y$ et avec le log népérien (base e ≈ 2,71828) avec $\ln(x) = y \Leftrightarrow x = e^y$

III – Recherche dichotomique :

Exercice 111 : écrire une fonction qui prend deux paramètres (liste, élément recherché) et qui renvoie True si l'élément recherché est dans la liste ou False si ce n'est pas le cas.

IV – Terminaison de l'algorithme :

Pour démontrer que la boucle conditionnelle « tant que » se termine, on utilise l'(le) invariant/variant de boucle : *diff = d – g*

Dans notre algorithme, g et d sont des entiers naturels, par conséquent, la variable diff est aussi un entier naturel.

A chaque itération, diff décroit car soit ……

Il existe donc une itération pour laquelle diff ne sera pas positif ou nul: la boucle « tant que » s'arrêtera alors, ce qui prouve la terminaison de l'algorithme.

V – Correction de l'algorithme (à comprendre mais à ne pas apprendre) :

1) Accès à la liste :

L'accès à la liste se fait à l'indice médian dans la boucle while. Cet indice est calculé par la moyenne (arrondie vers le bas) de g et d, dont on sait qu'ils vérifient la relation g ≤ d car on est dans la boucle.

De plus, on a comme invariants de boucle 0 ≤ g et d < len(liste).

On a donc toujours 0 ≤ g ≤ moyenne ≤ d < len(liste)

Cela garantit donc un accès légal à la liste. Lorsque g ou d sont modifiés, les invariants sont bien préservés.

2) Renvoie du True si l'élément recherché est trouvé :

Si on place en premier if liste[moyenne] < element puis elif liste[moyenne] > element, lorsqu'on arrive dans le else, le return True n'est effectué que lorsqu'on a l'égalité liste[moyenne] == element.

3) Correction de l'algorithme :

La correction garantit qu'un algorithme donnera toujours la bonne solution si on fait varier l'entrée.

Cette notion est moins évidente : nous pourrions avoir raté des éléments du tableau par accident, par exemple si vous avions écrit g = moyenne + 2.

Pour montrer la correction, on identifie l'invariant de boucle : « la valeur recherchée ne peut se trouver qu'entre liste[g] et liste[d] (inclus dans les deux cas).

C'est vrai initialement car g est initialisé à 0 et d à len(liste) – 1.

Lorsque g est modifié, on peut vérifier que cet invariant est préservé : liste[moyenne] < element et on effectue g = moyenne + 1. Or, le tableau est trié, l'élément n'est donc pas dans la partie de gauche du tableau et ne peut se trouver que dans la partie droite du tableau.

On raisonne de la même façon si d = moyenne – 1.

PARTIE G – ALGORITHMIQUE
Chapitre 38
Algorithmes gloutons

I – Définitions :

En informatique, on rencontre souvent des problèmes d'optimisation comme celui du rendu de monnaie des caisses automatiques (comment rendre le moins de pièces possibles) ou comme celui des GPS (détermination de la route la plus courte en temps). Nous nous intéresserons aujourd'hui au problème du sac à dos.

- ✓ **Force brute** : on énumère toutes les solutions possibles puis on choisit la solution optimale.
- ✓ **Algorithme glouton** : on procède étape par étape en faisant, à chaque étape, le choix qui semble le meilleur, sans jamais remettre en question les choix passés.

31. Dans la vie de tous les jours, donner un exemple de ces deux méthodes :

Force brute : ……

Algorithme glouton : ……

II – Contexte :

En vacances chez votre Mémé à Etretat, vous décidez de faire une journée « crapahutage » (déplacement dans un milieu difficile) le long des falaises d'Etretat.

Au détour d'une crique, vous apercevez, sous la végétation, l'entrée d'une grotte. Curieux comme Tintin, vous décidez d'y pénétrer, malgré un panneau « danger, éboulements ». Vous avez bien fait, vous découvrez le trésor d'un flibustier.

Le coffre contient :

- ✓ Un petit lingot d'or de 750 grammes d'une valeur de 30 000 euros.
- ✓ Un diamant de 5 carats (1 gramme) d'une valeur de 40 000 euros.
- ✓ Un rubis de 10 carats (2 grammes) d'une valeur de 27 000 euros.
- ✓ Une horloge gousset de 500 grammes de platine d'une valeur de 25 000 euros.
- ✓ Une chevalière de 20 grammes en palladium d'une valeur de 3 000 euros.
- ✓ Un poignard avec une lame de 650 grammes en titane d'une valeur de 3 euros.
- ✓ Une chaîne de 800 grammes en laiton d'une valeur de 2 euros.
- ✓ Un collier de 40 perles de 24 grammes d'une valeur de 9 200 euros.
- ✓ Une statuette en bronze de 1 kg d'une valeur de 1 000 euros.
- ✓ 300 grammes de truffes (sous vide) pour une valeur de 1 800 euros.

Malheureusement, votre sac est déjà bien lourd et vous décidez d'emporter au maximum 1 kg du trésor.

L'objectif est d'écrire 4 algorithmes afin de remplir votre sac à dos, de telle sorte que vous emportiez la valeur maximale.

Vous utiliserez pour cela deux listes, une liste « valeur » et une liste « poids ».

Etant donné qu'il faudra récupérer en même temps le poids et la valeur de l'objet, vous aurez besoin de l'indice de sa position : il est recommandé d'utiliser « for i in range » et non « for element in ma_liste ».

III – Premier algorithme :

1. Votre premier algorithme doit travailler sur la liste « valeur ». Compléter l'algorithme suivant :

```
valeur = [30000,40000,27000,25000,3000,3,2,9200,1000,1800]
poids = [750,1,2,500,20,650,800,24,1000,300]

valeur_tot = … # valeur totale mise dans le sac à dos à initialiser
poids_tot = … # poids total mis dans le sac à dos à initialiser

while valeur != []:
    valeur_max = … # valeur de l'objet la plus grande trouvée dans la liste
    pos_max = …

    for i in range (1,len(valeur)):
        if valeur[i] > … :
            valeur_max = …
            pos_max = i

    if 1000-poids_tot-poids[pos_max] … 0 :
        break

    valeur_tot = …
    poids_tot = …

    valeur.remove(valeur_max)
    del poids[pos_max]

print("le poids emporté est de", … ,"grammes")
print("la valeur emportée est de", … ,"euros")
```

IV – Deuxième algorithme :

2. Votre deuxième algorithme doit travailler sur la liste « poids ».

V – Troisième algorithme :

3. Votre troisième algorithme doit travailler sur une nouvelle liste « valeur_massique ».

VI – Quatrième algorithme :

Afin d'écrire votre quatrième algorithme, recopier le bout de code suivant :

```
# dans le CMD, importer la bibliothèque : pip install more-itertools
import itertools

indice=[0,1,2,3,4,5,6,7,8]
a=0

for p in itertools.permutations(indice,3) :
    a=a+1
    print (p)
print(a)
```

4. Qu'affiche le print (p) ?
5. Qu'affiche le print (a) ?
6. Quelle est donc l'utilité de la fonction permutations de la bibliothèque itertools ?
7. a) En utilisant ce code, écrire un 4e algorithme permettant de trouver la meilleure solution au problème. Vous devez obtenir le poids emporté, la valeur emportée mais aussi la liste des objets emportés.

   b) Pourquoi cette méthode s'appelle « force brute » ?
8. Quel algorithme propose la meilleure solution ? Expliquer.
9. BONUS : en utilisant la bibliothèque itertools, déterminer la formule permettant de donner le nombre de permutations différents de 9 éléments.

VII- Comparaison :

	Résolution 1	Résolution 2	Résolution 3	Résolution 4
Résolution	Valeur décroissante	Poids croissant	Valeur massique décroissante	
Poids emporté				
Valeur emportée				
Algorithme				
Optimal	Non	Non	Non	Non car .permutations
Exact ou approché	Approché	Approché	Approché	Exact
Complexité temps	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n!)$ $O(2^n)$
Temps de calcul	ms	ms	ms	s

PROJETS

I – Généralités :

Un enseignement d'informatique ne saurait se réduire à une présentation de concepts ou de méthodes sans permettre aux élèves de se les approprier en développant des projets applicatifs.

II – Quotas horaire :

Une part de l'horaire de l'enseignement d'au moins un quart du total en classe de première doit être réservée à la conception et à l'élaboration de projets conduits par des groupes de deux à quatre élèves.

III – Organisation :

Suite à la conférence de la SIF (Société Informatique de France) du 10 juin 2020, la mise en place sous forme de mini-projets est une organisation très pertinente.

Références

I – Netographie (2019) :

- ✓ http://www.apprendre-en-ligne.net (pour l'histoire de l'informatique).
- ✓ http://sebsauvages.net (introduction à Linux).
- ✓ https://slideplayer.fr (diaporama de L. Dutertre sur Linux).
- ✓ http://formation.upyupy.fr (mapping).
- ✓ http://ens-info.irem.univ-mrs.fr (JavaScript).
- ✓ http://sdz.tdct.org (encodages)
- ✓ https://pixees.fr (NSI première)
- ✓ http://cahier-de-prepa.fr de la TSI2 du lycée Coubertin de Meaux (erreurs Python).
- ✓ https://python-django.dev (dictionnaires).
- ✓ http://www.supinfo.com (préconditions, postconditions et invariants de V. Penando, 2017)
- ✓ http://www.courstechinfo.be (mémoire informatique).
- ✓ http://imss-www.upmf-grenoble.fr (diaporama sur la gestion de la mémoire).
- ✓ http://nsi.xyz (get et post)

II – Bibliographie (2019) :

- ✓ Cours de G. Oudinet, ISEN Toulon (Institut Supérieur de l'Electronique et du Numérique).
- ✓ Cours de T. Mas, NSI 1ère, lycée du Sacré-Cœur à Aix-en-Provence.
- ✓ Cours de M. Seroussi, NSI 1ère, lycée Yavné à Marseille.
- ✓ Cours de F. Boursier, NSI 1ère, lycée Saint Joseph à Ollioules.
- ✓ Cours de H. Bartolo, NSI 1ère, lycée Saint Louis à Orange.
- ✓ Cours de B. Xerri, NSI 1ère, lycée Saint Louis à Annecy.
- ✓ Cours de M. Pingault, NSI 1ère, lycée Saint-Gabriel de Valréas.
- ✓ Cours de P. Chassilian, NSI 1ère, lycée Saint Joseph à Avignon.
- ✓ Cours de X. Trouillot, ISN Tale, Institution des Chartreux.
- ✓ Cours de X. Ouvrard-Brunet, ISN Tale, Lycée international de Ferney.
- ✓ Cours de F. Sincère, ISN Tale, lycée Algoud-Laffemas de Valence.
- ✓ Cours d'architecture de G. Buffard, IUT Lyon 1
- ✓ Accard Cyprien, Apprendre le développement web au lycée, éditions Ellipses, 2018.
- ✓ Bays Serge, Spécialité NSI 1ère, éditions Ellipses, 2019.
- ✓ Pasquet Stéphane, NSI 1ère, Nathan, collection « les vrais exos donnés dans les lycées, 2019.
- ✓ Dowek G., ISN Tale, éditions Eyrolles, 2013.
- ✓ Bouchaour H. C., Mémoire de Magister, « Abstraction des circuits par isomorphisme de…», 2011.
- ✓ Ouled Zaid A., « La couche liaison et ses problèmes de contrôle de flux de données ».

III – Logiciels utilisés :

- ✓ Linux (Ubuntu)
- ✓ Editeur de texte comme Notepad++
- ✓ Navigateur Web comme Mozilla Firefox
- ✓ IDLE Python (Integrated Development Environment for Python)
- ✓ Lecteur d'images comme IrfanView
- ✓ Analyseur de paquet réseau Wireshark
- ✓ Logisim pour la simulation de circuits logiques (version utilisée 2.7.1)
- ✓ Analyseur de couleur comme Just Color Picker
- ✓ Amilweb pour découvrir le langage assembleur, par P. Boudes, Maître de conférences (Paris 13)