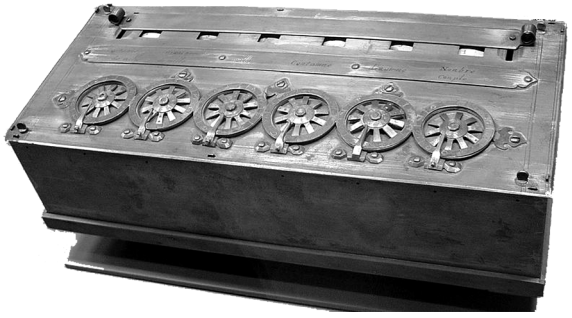
ISN Terminale S
Activités
et
130 Exercices de Programmation en Python

Frédéric Laroche

Table des Matières



La Pascaline, première machine à calculer mécanique.
Construite par Blaise Pascal en 1642.

Musée des Arts et Métiers, Paris.
Photographie : David Monniaux.

Avant-propos

Depuis la rentrée de l'année scolaire 2012-2013 un enseignement d'informatique est proposé en tant que Spécialité aux élèves de Terminale S. Cet enseignement est structuré autour de deux directions principales et complémentaires : un apprentissage de la programmation et un questionnement autour du rôle de l'informatique dans la société moderne.

C'est précisément l'organisation de cet ouvrage : une première partie principalement tournée vers des explications techniques sur le fonctionnement matériel et immatériel des machines ainsi que sur diverses questions « sociétales », et une deuxième partie orientée sur la programmation avec l'apprentissage du langage Python.

L'essentiel de la formation ISN tourne autour de la notion de projet et de la mise en activité permanente des élèves. Il faut bien néanmoins que chacun acquière un minimum de connaissances et d'éléments de base pour la conduite des projets, ce qui est le but de la deuxième partie de ce livre. Par ailleurs il n'y a pas de parcours prévu de l'ouvrage : les chapitres sont numérotés parce qu'il faut bien qu'ils le soient mais on peut très bien commencer au chapitre 5 tout en consultant le 4 et en lisant de temps en temps des parties du 1 ou du 2… Chacun suivra ses propres désirs.

Enfin la Spécialité ISN fait l'objet d'une évaluation finale consistant en une interrogation orale de vingt minutes donnant lieu à une note pour le Baccalauréat, coefficient 2. Le dernier chapitre du livre rappelle l'organisation de cette interrogation et fournit quelques pistes et conseils pour vous aider à préparer au mieux cette épreuve.

Le langage Python est un langage interprété (les instructions du programme sont lues au fur et à mesure de l'exécution par l'interpréteur Python et on peut s'arrêter où l'on veut), orienté objet, libre de droits et gratuit… Une bibliothèque de modules très fournie permet de faire à peu près ce qu'on veut.

Son utilisation dans l'enseignement permet une mise en action très rapide des élèves et une grande facilité de correction pour l'enseignant. Grâce à cet apprentissage assez rapide on peut réaliser des programmes conséquents dans le laps de temps imparti sur le temps scolaire : le livre donne d'ailleurs un certain nombre de pistes et de solutions pour démarrer des projets. On peut critiquer l'utilisation de Python en comparaison de langages plus « professionnels » comme C++ ou Java, mais l'objectif de cet enseignement n'est pas de former des spécialistes de la programmation, plutôt de donner quelques bases aussi bien techniques que citoyennes.

Utiliser Python permet de se focaliser sur les questions plus algorithmiques et surtout de ne pas « perdre » trop d'élèves au cours de l'année. Par ailleurs la réalisation d'un projet final, pouvant être aussi bien rudimentaire que sophistiqué, permet à chacun de progresser à son rythme tout en apprenant au fur et à mesure les concepts et méthodes de base. C'est d'ailleurs une des principales sources de satisfaction de cet enseignement.

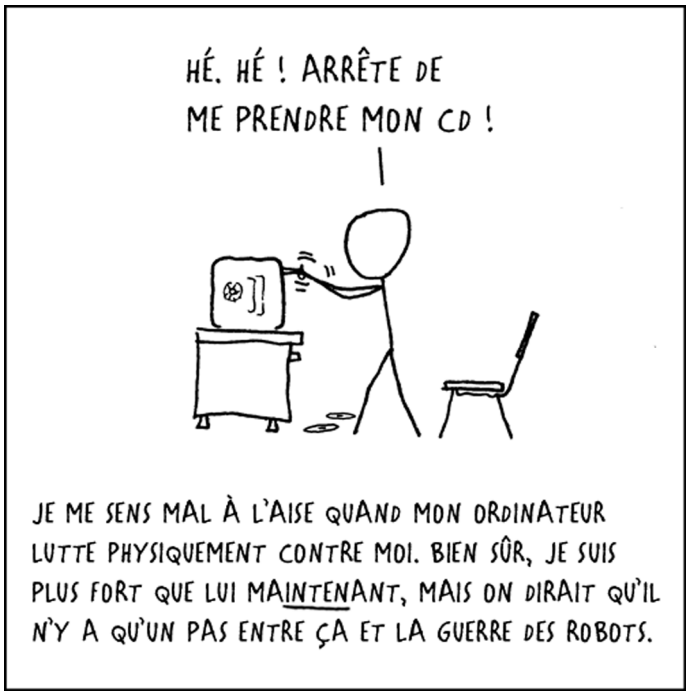
Une partie des exercices du chapitre 5 sont corrigés et commentés dans le livre : vous pouvez récupérer les corrections (sans les commentaires) sur le site http://laroche.lycee.free.fr/ ainsi que divers fichiers de données utilisés dans le livre.

Les dessins humoristiques sont dus au talent de Randall Munroe (xkcd.com) ; certains ont été traduits par les auteurs du site lapin.org.

J'espère que ce livre correspondra à vos attentes et surtout que vous prendrez autant de plaisir à programmer que j'en ai depuis que j'ai réalisé mon premier programme en BASIC vers 1980…

Tous mes remerciements à mon épouse Christine qui supporte mes humeurs depuis de nombreuses années, Corinne Baud des Éditions Ellipses qui m'empêche de faire trop de bêtises tout en soutenant mes divers projets et surtout à tous les élèves qui m'ont écouté patiemment depuis tant d'années… et m'ont donné envie de réaliser tous mes livres.

Montpellier, juin 2013.



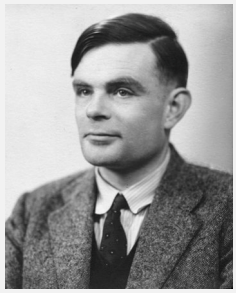
xkcd 251

1. Découvrir l'informatique

Cédric Villani, médaille Fields[1] 2010 : « Tout le monde devrait apprendre à programmer pour sentir ce qu'est un programme ! »

1.1 Introduction

On peut dire sans grand risque de se tromper que nous sommes à l'aube de grands changements pour l'humanité : il y a vingt ans un micro-ordinateur milieu de gamme coûtait l'équivalent de trois mois de salaire moyen, actuellement c'est plutôt de l'ordre de dix jours de travail et les téléphones sont devenus des quasi-ordinateurs ; quand aux capacités des machines elles continuent imperturbablement à suivre la *loi de Moore* : le nombre de transistors des processeurs d'entrée de gamme double tous les deux ans… Les puissances de calcul sont démultipliées, les liaisons entre les machines ont créé une foultitude d'applications reliant les êtres humains, les futures applications de l'informatique sont dans les tuyaux de demain : quel monde nous préparons-nous ?



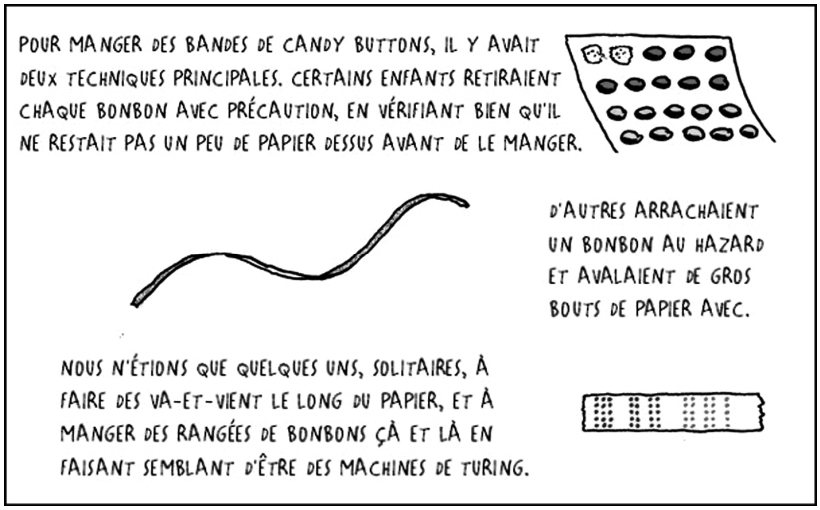
John Von Neumann et Alan Turing[2] peuvent être considérés comme les pères fondateurs de l'informatique. Ils ont défini ce que doit faire une machine et comment elle peut le faire. À l'heure actuelle 99,9 % des ordinateurs fonctionnent selon les principes qu'ils ont énoncés.

[1] Équivalent du prix Nobel en mathématiques. En informatique la plus haute récompense mondiale est le Prix Turing. Joseph Sifakis en 2007 est le premier Français à l'avoir obtenu depuis sa création, en 1966.

[2] Alan Turing fut un brillant mathématicien anglais, actif dans les années 1940-1950 ; il participa au décryptage de la machine Enigma pendant la guerre. Homosexuel, il fut poursuivi et « soigné » par les autorités anglaises, ce qui le poussa au suicide ; fan du film Blanche-Neige (Walt Disney), il se suicida en croquant dans une pomme imprégnée de cyanure : c'est devenu le logo de la firme Apple !

La maîtrise minimale des divers aspects de l'informatique est indispensable au citoyen et on ne peut que regretter qu'il ne s'agisse pas encore d'un enseignement obligatoire pour tous[3] : même si certains concepts de l'informatique sont fortement liés aux mathématiques et à la physique et par là-même difficiles d'accès pour une frange non négligeable de la population, il n'empêche que l'essentiel de son objet en fait un sujet abordable pour la grande majorité des élèves moyennant un peu de logique, d'organisation et surtout de persévérance.

En général la création d'applications informatiques ne nécessite pas de connaissances très élaborées et, mis à part les chercheurs, les « petits génies de l'informatique », comme se plaisent à les appeler les journalistes, n'ont pas grand chose de génial au sens propre du terme, hormis souvent un désir redoutable d'atteindre les buts qu'ils se sont fixés.



xkcd 205

Voir aussi sur Youtube (How to eat Candy Buttons like a Recreational Mathemusician)

1.1.1 Des usages et de l'apprentissage de l'informatique

L'informatique est dorénavant omniprésente dans les entreprises et les administrations, la société en général et la vie de tous les jours ; elle est la forme contemporaine de l'industrialisation du 19e siècle. Voici quelques exemples en vrac :

- production de biens manufacturés ou agricoles ; automatisation de plus en plus poussée des processus de production ; apparition des imprimantes 3D permettant de fabriquer n'importe quel objet ; utilisation massive de robots soulageant le travail des ouvriers mais faisant disparaître bon nombre de métiers ; prévisions météo et détection de l'apparition de maladies par analyse d'images satellites ; gestion des flux de production (stockage, prévisions à plus ou moins long terme) ;

- création de nouveaux produits ou amélioration de produits anciens par l'introduction de puces et de logiciels dans la plupart des objets ou machines, afin d'assurer des fonctions de plus en plus nombreuses avec plus de précision et de fiabilité que ne pouvaient en donner les hommes ou les mécanismes traditionnels. Ceci est particulièrement visible dans les transports, mais tous les pans de l'activité sont désormais touchés ;

---

[3] L'organisation d'ISN peut laisser cependant espérer de l'apparition d'un enseignement d'informatique généralisé au lycée, voire au collège et même au primaire d'ici quelques temps… Voir [AcadSciences2013] et [SIF2013] ainsi que http://www.epi.asso.fr.

2. Éléments sur les langages de programmation

« Un langage de programmation est une convention pour donner des ordres à un ordinateur. Ce n'est pas censé être obscur, bizarre et plein de pièges subtils. Ça, ce sont les caractéristiques de la magie. » Dave Small

Il existerait à peu près 4000 langages de programmation, l'article *liste des langages de programmation* sur Wikipedia en répertorie 750… certains de très bas niveau : langages machine ou assembleur, d'autres de haut niveau (Pascal, Java, Perl, Fortran, ML, …) ; certains sont compilés (traduits en langage machine par un compilateur de manière à être exécutés rapidement comme C, C++, objective C, Pascal), d'autres sont interprétés par un interpréteur comme Visual Basic, Ruby, Python, ActionScript (Flash) ou encore intermédiaires comme Java[17].

Il existe également des langages de description comme XHTML, HTML, CSS ou encore des langages interprétés par un outil destiné initialement à un autre langage (JavaScript ou PHP avec les navigateurs web).

Bref, pour chaque usage on aura un langage plus ou moins compliqué à apprendre et maîtriser ; le choix de l'apprentissage d'un langage ne peut donc se faire à la légère. Ceci dit ils se ressemblent quand même tous un peu et les différences portent plutôt, au moins pour commencer, sur leur plus ou moins grande facilité d'utilisation.

2.1 L'assembleur

2.1.1 Un langage spécifique à chaque processeur

Le langage **machine** est le **seul** langage qu'un processeur puisse exécuter, or chaque famille de processeur utilise un jeu d'instructions différent. Par exemple, un processeur de la famille x86 (Intel) reconnaîtra une instruction du type

[17] Le code compilé par Java est transcrit dans le langage d'une machine virtuelle universelle qui a son correspondant local implanté dans chaque machine, ce qui permet une portabilité maximale, aussi bien sur des ordinateurs différents (processeurs Intel, Motorola, AMD, ARM, etc.) que des téléphones portables, des tablettes ou tout autre appareil programmable. Le passage par la machine virtuelle ralentit souvent les processus mais pour de nombreuses applications ça n'est pas très gênant.

```
10110000 01100001
```

En langage **assembleur**, cette instruction est représentée par un équivalent « plus facile » à comprendre (*mnémonique*) pour le programmeur : (10110000 = movb %al ; 01100001 = $0x61)

```
movb $0x61,%al
```

ce qui signifie :

```
« mettre la valeur hexadécimale 61 dans le registre "AL" ».
```

Le langage assembleur est donc une représentation exacte du langage machine et est spécifique à chaque architecture de processeur. De plus, plusieurs groupes de mnémoniques ou de syntaxes de langage assembleur peuvent exister pour un seul ensemble d'instructions, créant ainsi des macro-instructions.

2.1.2 Réversibilité du langage machine

Contrairement à un langage de haut niveau, il y a une correspondance univoque entre le code assembleur et le langage machine. Ainsi il est théoriquement possible de traduire le code dans les deux sens sans perte d'information. La transformation du code assembleur en langage machine est accomplie par un programme nommé **assembleur**, dans l'autre sens par un programme appelé **désassembleur** (attention à la protection intellectuelle…). Les opérations s'appellent respectivement *assemblage* et *désassemblage*.

En pratique le désassemblage est un peu plus complexe que cela car lors de la création du code en assembleur on peut affecter des noms aux positions en mémoire, commenter son code, utiliser des macro-instructions ou générer du code conditionnel au moment de l'assemblage. Tous ces éléments n'apparaissent pas forcément clairement lors du désassemblage.

2.1.3 Instructions machine

En général on a les opérations de base suivantes dans la plupart des jeux d'instructions assembleur :

- déplacement :
  - * chargement d'une valeur dans un registre ;
  - * déplacement d'une valeur entre un emplacement mémoire et un registre et inversement ;
- calcul :
  - * addition (+, –) des valeurs de deux registres et chargement du résultat dans un registre ;
  - * opération booléenne entre les valeurs de deux registres (ou opération bit à bit) ;
- modification du déroulement du programme :
  - * saut à un autre emplacement du programme (instructions exécutées séquentiellement) ;
  - * saut à un autre emplacement après sauvegarde de l'instruction suivante afin de pouvoir y revenir (point de branchement) ;
  - * retour au dernier point de branchement ;
- comparaison :
  - * comparer les valeurs de deux registres.

On trouve également des instructions spécifiques pour des opérations qui peuvent nécessiter beaucoup d'instructions élémentaires. Exemples :

- * déplacement de grands blocs de mémoire ;

3. Informatique et société

L'informatique, sous des formes très diverses, a envahi notre monde d'humains : depuis les gigantesques calculateurs destinés à simuler des explosions nucléaires jusqu'au plus petit microprocesseur utilisé dans un programmateur de machine à laver, depuis les machines de la NSA décryptant et traquant toutes les conversations dans le monde jusqu'à votre compteur électrique « intelligent », depuis les monstrueux fichiers de données accumulées par Google, Microsoft ou Facebook jusqu'à votre liste de contacts, il y a de l'électronique, du traitement d'information et de la programmation partout : il faut s'interroger, sans tomber dans l'angélisme béat ni dans la paranoïa destructrice, sur les usages et les à-côtés de l'informatique actuelle. Quelques prévisions peuvent également être envisagées même s'il s'agit d'un exercice délicat…

3.1 L'informatique, le réseau et l'avenir (possible)

Les premiers systèmes informatiques étaient très centralisés : les *mainframes* assuraient le travail d'un grand nombre de consoles de travail ; l'ordinateur central effectuait alors tous les travaux des consoles et gérait tous les périphériques.

Avec l'arrivée des circuits intégrés (1970) et l'avènement de la micro-informatique (1980), l'informatique est devenue *distribuée*. L'Oric-1, le ZX80, l'IBM-PC, l'Apple 2, l'Amstrad 1512, etc. furent les premiers exemples de généralisation de l'informatique personnelle : le poste de travail devint autonome, gérant ses logiciels, ses connexions et ses périphériques.

Un regroupement des deux types d'organisation informatique est apparu avec l'architecture Clients / Serveurs : un poste de type micro-ordinateur peut être Client d'un serveur de données ou de traitements par l'intermédiaire d'une connexion réseau. Le Client peut également assurer ses traitements propres avec les logiciels et les données en local.

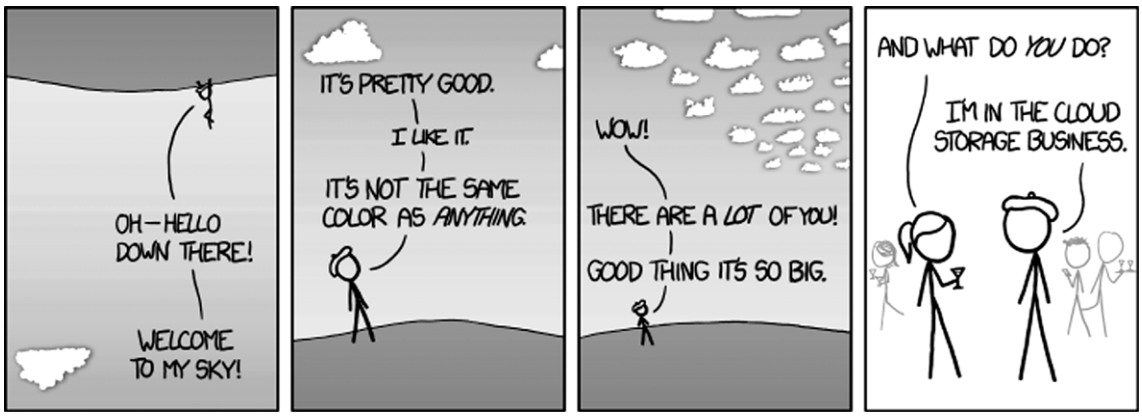
L'apparition d'Internet a bouleversé le fonctionnement de cet univers centralisé : en permettant l'interconnexion de réseaux de types et de structures différents par l'intermédiaire du protocole IP (*Internet Protocol*), il est devenu la source d'une nouvelle informatique.

L'intégration progressive des systèmes de communication téléphonique au réseau global a également modifié la donne : un téléphone n'est plus uniquement destiné à la voix, il sert à quasiment tout et si il ne dispose pas des outils ad-hoc, par simple commutation vers une machine plus puissante, il peut exécuter pratiquement toutes les activités d'un micro. La gestion des données dans le « nuage » (*cloud*) évite également les problèmes de stockage : visionner un film ne demande plus de disposer physiquement du film.

Alors faisons un peu de prospective :

- **la communication des appareils** (domestiques ou non) va se poursuivre de manière accélérée : la télévision va évidemment se connecter complètement à Internet et devenir une console comme une autre permettant de surfer, télécharger, commander en ligne, etc. Mais on devrait voir d'autres appareils se connecter : la machine à laver, le grille-pain, la cafetière pourront se connecter grâce au réseau électrique, actuellement sous-exploité ; la voiture, le scooter, voire le vélo électrique vont se connecter : ils ne pourront démarrer que sur ordre, la consommation sera gérée automatiquement et optimisée, l'alimentation par induction pourra être gérée de manière décentralisée (flottes de véhicules), on pourra faire ses courses depuis la voiture…

- **les robots** : très utilisés de manière industrielle, ils sont en train de s'imposer dans le domaine médical et on devrait les voir apparaître de manière substantielle dans notre quotidien… Machine à laver triant le linge, le lavant suivant les constituants lus sur des étiquettes RFID[23], séchant, repassant, rangeant ; véhicules sachant suivre un trajet grâce à son positionnement GPS, munie de capteurs de chocs, de détecteurs de risques de collisions, d'analyseurs de scènes, communiquant avec son environnement, capable de prendre les commandes en cas de défaillance du conducteur… On peut imaginer encore des milliers d'applications du même style, la puissance de calcul embarquée risquant de devenir phénoménale rapidement.



xkcd 1117

- **les données** : c'est le nerf de la guerre pour lequel se battent les États ainsi que toutes les entreprises commerciales ; Google, Facebook, Microsoft, Apple, IBM sont en train de bâtir leur puissance actuelle ou future sur l'accumulation de données personnelles qui permettront de proposer des produits aux clients grâce à leurs habitudes de consommation et à la géolocalisation, d'anticiper les besoins de chacun, de gérer les transports, les consommations d'eau, d'énergie, etc.

Les États sont également partie prenante pour l'espionnage et la surveillance mais également en tant que gestionnaires de nombreux domaines : l'accès aux données administratives, fiscales, sociales (Open Data) donné aux citoyens rend l'action des responsables politiques plus transparente et doit permettre aux administrés d'avoir un regard critique sur l'action politique.

Dans un monde connecté les données sont l'équivalent de la nourriture, de l'eau et de l'air pour notre corps : notre société ne survivrait probablement pas longtemps à un arrêt brutal du fonctionnement des machines…

---

[23] Étiquettes à champ magnétique, ne nécessitant pas d'apport énergétique, elles devraient remplacer les code-barre sur les produits ; on en trouve déjà qui servent de clé de serrure.

4. Programmer

Les principes de base de la réalisation d'applications sont globalement intangibles :

- définir grossièrement le projet, son utilité, ses ressources, ses caractéristiques ; à cet effet un début de documentation peut être rédigé de manière à en préciser les contours ;

- un synopsis du programme peut être élaboré avec quelques dessins d'écran ;

- en général un programme est construit autour d'un thème central que nous appellerons le *cœur* (communiquer des informations, gérer une base de données, réaliser un calcul scientifique, jouer aux échecs…) : on commence donc à écrire le code des algorithmes du cœur en utilisant éventuellement des éléments d'une bibliothèque externe ; ce programme peut être partie intégrante d'un programme plus vaste ;

- le programme commence à prendre forme et des idées d'amélioration apparaissent : les noter pour ne pas les oublier et laisser le codage pour plus tard ;

- une fois le cœur écrit, il faut le vérifier et le tester : c'est le **débogage** qui prend en général au moins autant de temps que le codage du cœur, donc pas d'énervement et ne rien lâcher tant que ce n'est pas nickel ;

- terminer par les modifications éventuelles (bien déboguer et **documenter** au fur et à mesure), l'habillage (menus, boîtes de contrôle, messages divers), l'écriture de la documentation et la distribution du programme (payant, shareware, freeware, open source, licence GNU,…).

La tentation, lorsqu'on débute, est de s'attacher davantage à l'apparence de l'application qu'à son comportement : on ne peut que recommander de démarrer par la programmation du cœur en réalisant succinctement les entrées-sorties, quitte à améliorer l'apparence ultérieurement.

Dorénavant, pour éviter les quiproquos, nous écrirons tous les nombres à virgule avec un point : 3,415 sera 3.415, la multiplication avec * et les puissances de 10 avec ***n*e*k*** : 2.5e3 vaut 2500. Par ailleurs il vaut mieux éviter les caractères accentués dans les programmes et surtout dans les noms de fichiers (comme séparateur utiliser le « tiret du 8 » :'_' également appelé « underscore »).

Nous allons nous appuyer dans la suite de cette section essentiellement sur Python avec quelques aperçus sur Java.

4.1 Bases de la programmation

En Python, on peut travailler en mode *console* : on écrit les instructions avec le *shell* (>>>…), qui fonctionne comme une grosse calculatrice, les résultats s'affichant alors dans l'écran standard (éventuellement avec des *print*) ; on travaille néanmoins le plus souvent en mode *script* (on écrit le programme dans un fichier texte que l'on va exécuter une fois le texte finalisé), ce qui permet d'utiliser facilement diverses ressources externes (bibliothèques comme *Tkinter*, *mathplotlib*, *pygame*,…), même si ce n'est pas une obligation. Un éditeur comme **pyscripter** fait parfaitement l'affaire (Windows), sinon les distributions Python disposent de leur propre éditeur (**IDLE**, un peu rustique), mais on peut avoir facilement des outils élaborés comme par exemple **komodo** qui semble bien adapté à toutes les plate-formes ; voir diverses solutions sur

http://wiki.python.org/moin/IntegratedDevelopmentEnvironments.

Avec Java, une interface simple pour débuter est **JavaScool** (http://javascool.gforge.inria.fr/), didactique et bien documentée ainsi que l'éditeur de **Processing** (http://processing.org/) qui dispose de diverses fonctions de base destinées à la manipulation du multimédia.

La plupart des programmes que vous écrirez sont structurés en 3 parties principales :

- (déclaration et) initialisation des bibliothèques, des variables et des constantes ;
- contrôle du flux : traitement fait aux variables, boucles, branchements, appels de fonctions ;
- retour du programme au programme appelant, entrées-sorties diverses (lecture de fichiers, affichage, son, etc.).

On peut évidemment mixer ces divers éléments, mais c'est peu recommandé car on risque de perdre pied rapidement et ne plus retrouver les éléments importants. Insistons lourdement de nouveau sur la **nécessité absolue** de documenter votre programme : il vous faudra trouver le juste milieu entre l'explication du moindre détail et le survol général. Une bonne idée pourrait être de vous demander ce que vous comprendriez encore de votre programme dans trois mois ou de soumettre les commentaires à un-e camarade.

4.1.1 Les variables et les constantes

Une **variable** est une donnée que l'ordinateur va stocker dans un espace mémoire. C'est comme un compartiment dont la taille n'est valable que pour un seul type d'information. Elle est caractérisée par un nom qui permettra d'y accéder facilement ; on a donc tout intérêt à donner des noms explicites aux variables : pas dans un premier temps sous peine de se transformer en machine à écrire mais lorsque le programme sera finalisé, on donnera des noms significatifs en utilisant les fonctions de recherche et remplacement de l'éditeur. Quelques remarques pour l'écriture du code :

- en Python les blocs d'exécution sont identifiés par un **décalage de 4 espaces sur la droite** ; faites bien attention en cas d'imbrications multiples à ce que chaque bloc se termine comme vous l'avez prévu. Par ailleurs on peut mettre autant d'espaces que l'on veut dans chaque ligne (écrire plutôt z = a – b/c que z=a-b/c) et même mettre des lignes blanches ;
- en Java les blocs d'exécution sont identifiés par des accolades ; il y a peu de risques de se tromper, l'habitude étant de passer à la ligne avant la dernière accolade.

Il existe différents types de variables : des nombres entiers (*int* ou *long*), des nombres à virgule (*float*), du texte (*char* ou *String*), des valeurs *vrai/faux* (*True/False*, *boolean*). Un nombre à décimales, comme 3.14159, n'étant pas un nombre entier, serait donc du type *float*. Si vous voulez obliger un

5. Exercices en langage Python

Les exercices sont rangés par type et par ordre de difficulté plus ou moins croissante ; ils n'ont qu'un seul objectif, vous aider à apprendre à programmer.

L'avantage de la programmation (contrairement aux mathématiques) est de savoir en général à la fin du programme si on a réussi ou pas… et il est très formateur de ne pas lâcher le morceau tant que ce n'est pas fini. Alors accrochez-vous bien avant de regarder la solution… et rappelez vous :

Règle numéro 5 : les débutants doivent à tout prix éviter le copier/coller. Sauf, évidemment, s'ils veulent copier le code d'un programme qu'ils ont écrit.

On peut utiliser aussi bien la console que les scripts pour réaliser de petits programmes. La console est plus rapide à manipuler et permet de faire des changements immédiats. Les scripts sont plus lents et nécessitent le chargement des modules prédéclarés, ce qui ralentit la mise en œuvre.

Dans tout ce qui suit, un argument précédé de * est facultatif : par exemple **`range`**`(*a, b, *c)` pourra s'écrire `range(b)`, `range(a, b)` ou `range(a, b, c)` mais pas `range(b, c)`.

Aides diverses

Si on a quelques trous de mémoire sur les diverses fonctions utilisables, tapez **`dir`**`(objet)` dans la console… En général dans les scripts un cartouche apparaît lorsqu'on passe la souris sur une fonction ; on peut alors cliquer dans le cartouche sur le module où la fonction est implémentée : si le module n'est pas chargé il le sera alors. Un petit mémento comme *Python Pocket Reference* de Mark Lutz peut être très utile, sinon la référence générale est *The Python Standard Library* :

http://docs.python.org/2/library/index.html#library-index

Deuxième option d'aide : taper dans la console (__ = **deux** tirets *underscore*) :

```
>>> print (NomDeLaFonction.__doc__)
```

Si la fonction a été documentée vous aurez le commentaire associé (la *docstring* enregistrée dans son module) : vous pouvez vous même créer des *docstrings* en plaçant un texte entre apostrophes juste après le nom de la fonction que vous implémentez :

```
def split(ligne):
      r'''Décompose une chaine de caractères constituée de champs séparés par des ;
      le résultat est constitué du nombre total de champs et d'une liste des contenus.
      http://docs.python.org/2/library/stdtypes.html?highlight=split#str.split '''
   instructions
```

Les paramètres d'usage de la fonction considérée sont en général détaillés dans la *docstring*.

Insertion de commentaires dans vos programmes :

: commentaires sur la ligne ;

'''……'''ou """……""" ou ***……*** : commentaires en bloc.

Installation : pour AmiensPython se référer à la documentation du site qui permet aussi d'installer l'éditeur sur une clé USB ainsi que sous Linux. Pour Mac utiliser IDLE fourni avec la distribution standard ou Komodo.

Liens indispensables

Association française de Python	http://www.afpy.org/
Site officiel de Python	http://python.org/
Bibliothèques Tkinter et PIL	http://www.pythonware.com/
Python pour le lycée :	http://amienspython.tuxfamily.org/
Ressources en Français	http://pythonfacile.free.fr/python/ressources.html

5.1 Éléments de syntaxe

En général les instructions Python seront notées en **gras**, les noms de variable génériques en *italique*, le reste en caractères standard.

5.1.1 Données et opérateurs

Typage des données (exemples en commentaires)

chaîne de caractères : **str**()	#st="abcdef" ou 'abcdef' / on utilise aussi **unicode**()		
entier : **int**() ou **long**()	#n=125L, pas de max.	booléen : **bool**()	#True ou False
pseudo-réel : **float**()	#12.5, max=1e302	complexe : **complex**()	#5+2j ; i pas utilisé.
indexation : *liste*[i]	#L[i]	segmentation : *liste*[i:j]	#L[i:j]
Attention : a = b n'est pas un operateur math. mais une **assignation** de variable.			

Opérations

Mathématiques :		**Evaluations** :	
+ - * /	# division flottante	a == 10	# égalité mathématique
//	# division entière (version <=2.7)	a != b	# différent de
%	# modulo	a > b	# plus grand que
**	# puissance	a < b	# plus petit que

6. Correction des exercices

6.1 Séries 5.2 & 5.3

Exercice 5.2.1

```
>>> largeur = 20  #définition de la variable largeur
>>> hauteur = 5 * 9.3
```

définition de la variable hauteur comme produit de deux nombres (un entier et un réel)

```
>>> largeur * hauteur
```

#calcul du produit des deux variables, le résultat est un réel (float) : 930.0.

Exercice 5.2.2

Utilisez les commandes suivantes :

```
>>> a, b, c = 13, 25, 57
>>> a - b/c
```

Le résultat est 13 avec les versions de Python antérieures à 3.0. La division / est la division des entiers. On peut contraindre Python à utiliser des réels avec par exemple :

```
>>> a, b, c = 13.0, 25.0, 57.0
```

Quand on opère avec un script, le résultat est correct grâce à l'instruction

```
from __future__ import division
```

Exercice 5.2.3

Attention il ne faut pas perdre de valeur en cours de route…

```
>>> A, B, C = 1, 2, 3
>>> Z=C ; C=B ; B=A ; A=Z
>>> A, B, C
(3, 1, 2)
```

Exercice 5.2.4

`>>> r , pi = 12, 3.14159`
`>>> s, t = pi * r**2`
`>>> t= 'surface = '`
`>>> print t, s`
`>>> print type(r), type(pi), type(s), type(t)`
	Définition d'un entier r et d'un réel pi.
	Définition d'un réel calculé s et d'une chaîne de caractères t.
	Affichage de `surface = 452.38896`.
	On a : <type 'int'> <type 'float'> <type 'float'> <type 'str'>

La fonction `type()` donne des informations sur les variables utilisées.

On peut les tester avec un `isinstance(objet, classe)` :

```
r=300.0
u=isinstance(r, float)
print(u)
```

renvoie `True`

Exercices 5.2.5 & 5.2.6

`reussi=False`
`while not reussi:`
`  u=raw_input('Donnez un nombre')`
`  try:`
`  x=float(u)`
`  except:`
`  print 'on vous demande un nombre !'`
`  else:`
`  reussi=True`
 	
`if x>0:`
`  st='positif'`
`elif x<0:`
`  st='négatif'`
`else:`
`  st='nul'`
	On utilise une boucle booléenne :
	tant qu'on n'a pas *reussi*, on demande un nombre à l'utilisateur ;
	traitement de la saisie avec une exception et conversion de type (Str vers float) :
	si ce n'est pas un nombre on recommence en engueulant le client… ;
	si c'est bon le booleen *reussi* devient vrai (True) et on sort de la boucle.
	Traitement des cas : remarquez qu'il est inutile de construire toute la phrase de réponse immédiatement, c'est fait à la fin avec la concaténation ; lorsqu'on sépare les objets par des virgules dans **print** on a des espaces qui apparaissent, sinon il faut les traiter soi-même…

`print 'Le nombre',x,'est '+st+', son carré est '+str(x*x)+'.'`	

Exercices 5.2.8 & 5.2.9

`h,m,s=-1,-1,-1`
`while h<0 or h>24:`
`  h=int(raw_input('Heure 0-24'))`
`while m<0 or m>60:`
`  m=int(raw_input('Minutes 0-60'))`
`while s<0 or s>60:`
	On donne volontairement des valeurs incorrectes aux variables h, m et s pour entrer dans les boucles **while** et contrôler la saisie a minima (voir ci-dessus : en fait ce serait bien d'utiliser une fonction spéciale contrôlant ce qu'on veut obtenir).

7. L'épreuve terminale

L'épreuve d'ISN se passe à l'intérieur de votre établissement sous forme d'un oral de vingt minutes réalisé par votre enseignant accompagné par un collègue non nécessairement enseignant d'ISN.

Ces 20 minutes sont constituées d'une première partie de 8 minutes où vous présentez votre projet, lequel projet fait l'objet d'un dossier de 5 à 10 pages[42]. Après ces 8 minutes le jury vous torturera pendant 12 minutes sur votre projet mais peut également vous interroger sur n'importe quelle question ayant trait à ce que vous avez fait dans l'année.

Au cas où vous ne présenteriez pas de dossier le jury vous le signifiera et devra mettre 0 à la première partie… Si vous êtes candidat libre, vous serez convoqué dans un établissement proche de votre domicile et vous n'êtes pas obligé de fournir un dossier. Ceci dit vous avez tout intérêt à fournir ce dossier un peu à l'avance ce qui mettra le jury dans de bonnes dispositions à votre égard.

La notation ressemble un peu à ce qui se passe pour les TPE : point n'est besoin d'un projet hyper sophistiqué ou d'une application de plusieurs centaines de lignes de code… L'essentiel sera dans l'investissement que vous y aurez mis et de la compréhension de ce que vous présentez.

7.1 Les grandes lignes du programme

> « L'objectif de l'enseignement de spécialité ISN en classe terminale de la série S n'est pas de former des experts en informatique, mais plutôt de fournir aux élèves quelques notions fondamentales et de les sensibiliser aux questions de société induites. Il s'agit d'un enseignement d'ouverture et de découverte des problématiques actuelles, adapté à la société d'aujourd'hui, qui valorise la créativité et contribue à l'orientation. »

Le programme tel qu'il a été écrit essentiellement à partir des idées des informaticiens de l'INRIA (G. Dowek, T. Vieville, M. Nivat, J. P. Archambault, G. Berry, S. Abiteboul, C. de la Higuera, etc.) se décompose en quatre grandes parties :

---

[42] Aucun texte ne précise quand doit être remis le dossier : par correction envers les examinateurs il semble que 48 heures soit un minimum. Une situation qui s'est rencontrée quelques fois : le dossier est très insuffisant et/ou le projet a été intégralement « pompé » sur Internet… En général le jury s'en apercevra et vous posera plein de questions piègeuses… Il vaut mieux travailler sur son projet de manière régulière afin d'en maîtriser tout les aspects. Dernier point : lorsque l'exposé est basé sur un diaporama (Powerpoint par exemple) on considère qu'une diapo équivaut à une demi-page A4. Si un autre support est choisi (film, BD, etc.) mettez-vous bien d'accord avec votre professeur à l'avance.

Représentation de l'information	Représentation binaire, Opérations booléennes, Numérisation, Formats, …
Algorithmique	Algorithmes simples : rechercher un élément dans un tableau trié par une méthode dichotomique ; trier un tableau par sélection ; ajouter deux entiers exprimés en binaire. Algorithmes plus avancés : tri par fusion ; recherche d'un chemin dans un graphe par un parcours en profondeur (DFS) ; recherche d'un plus court chemin par un parcours en largeur (BFS).
Langages et programmation	Types de données, Fonctions, Correction d'un programme, Langages de description (HTML).
Architectures matérielles	Éléments d'architecture, jeu d'instructions. Réseaux : Transmission point à point, Adressage, Routage…
On peut rajouter à titre optionnel une initiation à la robotique.	

Un deuxième objectif consiste à faire prendre conscience des enjeux de la numérisation accélérée de nombreux pans de la société : sur la base d'exposés d'élèves un dialogue doit s'installer permettant de cibler quelques problématiques actuelles. La propriété intellectuelle, le « cloud computing », le « fichage » des populations, les Anonymous, les bugs et les tests, etc. sont des sujets actuels qui permettent des discussions animées ainsi que des prises de conscience des enjeux de l'informatique.

Par ailleurs, au cœur du dispositif se trouve la mise en activité de l'élève :

> « Afin de refléter le caractère scientifique et technique propre à la discipline et de développer l'appétence des élèves en faveur de cet enseignement nouveau pour eux, il convient de les mettre en situation d'activité aussi souvent que possible. Une pédagogie de projet est à privilégier pour favoriser l'émergence d'une dynamique de groupe. Dans ce cadre, le professeur joue un rôle central : il impulse et coordonne les projets, anime les débats et met en place l'évaluation et ses modalités. »

Quelques détails :

> « Les projets réalisés par les élèves, sous la conduite du professeur, constituent un apprentissage fondamental tant pour la compréhension de l'ISN que pour l'acquisition de compétences variées. Ils peuvent porter sur des problématiques issues d'autres disciplines et ont essentiellement pour but d'imaginer des solutions répondant à l'expression d'un besoin. »

L'expérience montre que ce programme est beaucoup trop ambitieux et en grande partie inadapté à des élèves n'ayant **jamais** fait d'informatique sérieusement et pour qui l'algorithmique reste une notion assez nébuleuse. Avec 2 heures de cours par semaine, soit un total approximatif de 50 heures annuelles, on peut alors espérer acquérir raisonnablement quelques notions basiques de programmation (18 heures), explorer quelques thèmes sociétaux sous forme d'exposés (4 heures), réaliser un mini projet (6 heures) et un projet complet (22 heures).

Maintenant la possibilité de réaliser un projet longue distance doit permettre à chacun de trouver sa place et aux néophytes aussi bien qu'aux experts de prendre plaisir à cet enseignement.