Langage et programmation
1 - Présentation

1 - Exécution d'un programme

Le seul langage que comprend le microprocesseur au cœur de l'ordinateur est son ***langage machine*** spécifique (aussi appelé *code machine*, *code natif* ou encore *code binaire*). C'est un codage purement numérique représentant des opérations très basiques et/ou des données.
Il peut être exprimé dans un langage symbolique textuel d'un peu plus haut niveau, appelé ***langage d'assemblage***, à partir duquel un programme ***assembleur*** permet d'interpréter les symboles pour générer le programme exécutable en code machine.
Exemple[1] de programme assembleur sous Windows (avec les commentaires) :

```
ORG 100h                                ; Fichier .com, explications dans le chapitre 3.

MOV AH, 09h                             ; Indique le numéro de la fonction à utiliser par l'interruption
MOV DX, message                         ; Indique le paramètre utilisé par la fonction
INT 21h                                 ; Appel de l'interruption 33 (21h), fonction numéro 9 : afficher une
                                            chaîne à l'écran.
RET                                     ; On termine l'application proprement.

message db "Bonjour le monde !", '$'    ; On définit la chaîne de caractères 'message', terminée par '$'
```

Pas très adapté à l'être humain, on a besoin de données de plus haut niveau, d'exprimer des concepts plus abstraits, de comprendre facilement un code écrit… de là sont nés de (très[2]) nombreux ***langages de programmation*** (Lisp, Fortran, Cobol, Algol, APL, Basic, PL/1, Simula, C, Smaltalk, Prolog, Pascal, Forth, SQL, Ada, C++, Eifel, Perl, Haskell, Python, Ruby, Lua, Javascript, Java, PHP, C#, Rust, Go…). La plupart expriment les programmes avec du texte, mais on peut trouver certains langages graphiques (LabView, Pd, NodeBox…).

Chaque langage a ses avantages et inconvénients, est plus ou moins adapté suivant le type de problème que l'on veut résoudre. Ils permettent d'exprimer les algorithmes en offrant différents paradigmes[3] de programmation (logiques de fonctionnement) parmi lesquels on citera :

- La ***programmation impérative***, où l'exécution d'une séquence d'instructions va modifier l'état des données (la plupart des langages sont impératifs).
- La ***programmation objet*** à base de classes, qui est une façon d'organiser et de regrouper les données et les traitements qui y sont liés.
- La ***programmation fonctionnelle*** déclarative, qui se rapproche des théories mathématiques.

On distingue historiquement deux modes d'exécution : « *compilation* » et « *interprétation* » :

- La ***compilation*** : un programme intermédiaire, le *compilateur*, lit le texte du programme en langage de haut niveau et le traduit en code machine directement exécutable par le processeur.
  Ensuite, le *code machine est directement exécuté* par le processeur dans le cadre d'un processus du système d'exploitation.
- L'***interprétation*** : un programme intermédiaire, l'*interpréteur*, lit le texte du programme de haut niveau et en assure lui-même l'interprétation ligne par ligne.

Les techniques actuelles mixent allègrement ces deux modes d'exécution, utilisant parfois des représentations intermédiaires, appelées ***byte-code***, ou générant directement le code machine par de la ***compilation à la volée*** juste avant de l'exécuter.

[1] Source : https://fr.wikiversity.org/wiki/Assembleur/Le_langage_assembleur
[2] Voir une historique sur https://www.levenez.com/lang/
[3] Voir https://fr.wikipedia.org/wiki/Paradigme_(programmation)

2 - Langage et interpréteur

Tout au long de l'année nous utiliserons un langage de programmation, **Python**. L'écriture des programmes en Python et leur exécution se feront avec le logiciel **Pyzo** qui est un IDE (Integrated Development Environment) pour Python.

Recherches internet

Quand et par qui le langage Python a-t-il était inventé ?

Inventé par Guido van Rossum, première publication publique en 1991.

Pourquoi s'appelle-t-il Python ?

Par référence à la série télévisée Monty Python's Flying Circus.

Donner d'autres noms de langages informatiques.

Java, Pascal, Ada, C, C++, javascript, PHP.

Quels sont les avantages de Python ?

Python est multi plateforme et open source, facile à apprendre, permet de créer des fonctions avec moins de lignes de codes, couramment utilisé dans la science des données, beaucoup utilisé dans les entreprises.

Donner d'autres noms de logiciels IDE pour Python ?

Pycharm, Wing IDE, Eric, Cloud9, Ninja, Sublime Text, GNU Emacs, Visual Studio Code, Spyder, Thonny, ...

Quels sont les avantages de Pyzo ?

Quelques un : Multiplateforme, facile à installer, peut être installé sur une clé, possède de nombreux modules par défaut, inclut le notebook Jupyter,...

Python

Python exécute ses programmes avec une compilation du texte « source » vers une représentation en byte-code intermédiaire, qui est ensuite interprété par un interpréteur « machine virtuelle ». Sa syntaxe très lisible lui permet de passer simplement de l'algorithme au programme et donc de pouvoir rapidement tester l'exécution du programme sur des jeux de données.

Pyzo

Voici à quoi ressemble Pyzo lorsqu'il est ouvert.
La partie gauche est la *fenêtre d'édition* dans laquelle les programmes sont écrits,
la partie droite comporte plusieurs fenêtre,
- en haut c'est le *shell* (l'interpréteur de commande, ou shell, est l'interface de communication entre l'utilisateur et le système d'exploitation. C'est un exécutable chargé d'interpréter les commandes écrites par l'utilisateur et de les exécuter.),
- en bas c'est le gestionnaire de fichier.

Les programmes sont écrits dans l'éditeur mais on peut écrire des lignes de commande dans le shell pour les tester.

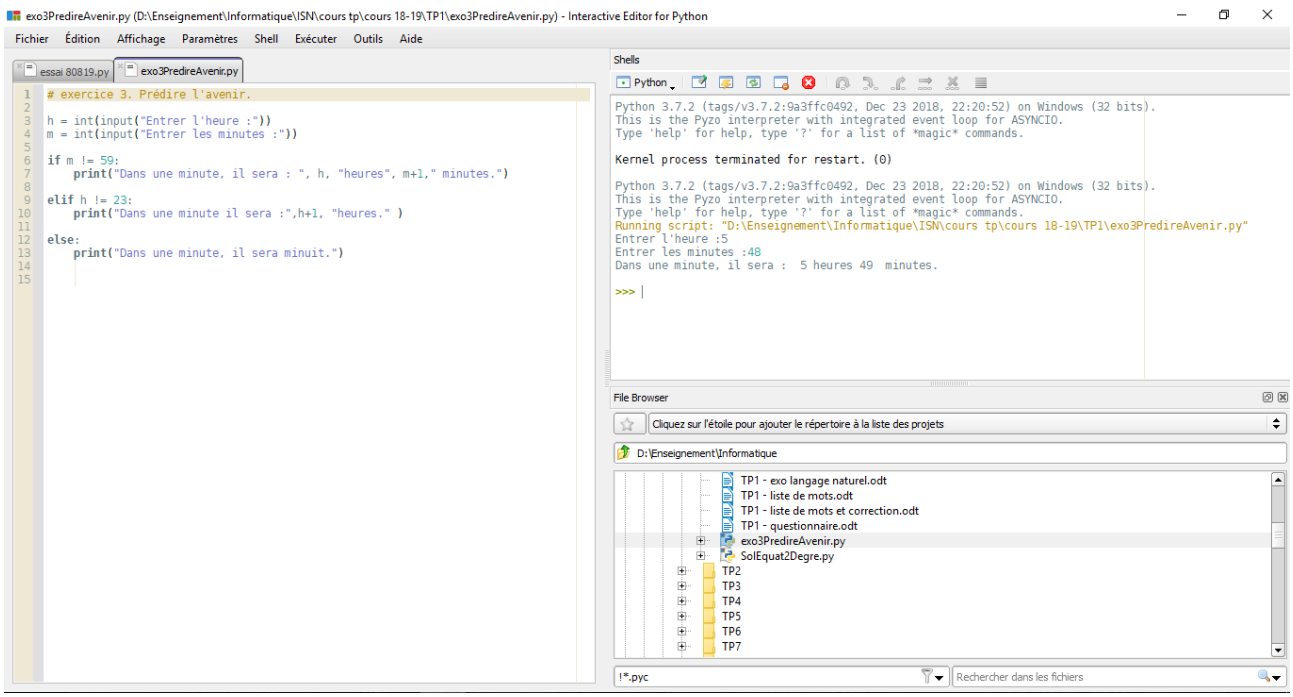

Figure 1: interpréteur Pyzo

3 - Mots clés du langage Python

Dans les programmes Python, ces mots clés (keywords) sont interprétés avec une signification particulière : and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, while, with, yield.

4 - Structure d'un programme

```
""" Écrivez une fonction qui renvoie la longueur de la plus
 longue suite d'éléments consécutifs égaux d'une liste.
 Par exemple, pour la liste [2,3,3,3,2,2,2,5,5,5,5,3,3]
 votre fonction devra répondre 4 (la plus longue sous suite
 d'éléments égaux consécutifs (chiffre 5) a 4 éléments)."""


def taille_soussequence(liste):
    i=0
    count=1
    resultat=0
    max=0

    while i < len(liste):

        while i < len(liste)-1 and liste[i]==liste[i+1]:
            count+=1
            i+=1
        resultat=count
        count=1
        i+=1
        if resultat>max:
            max=resultat

    return max


# Puis les tests :
assert taille_soussequence([2,3,3,3,2,2,2,5,5,5,5,3,3]) == 4
assert taille_soussequence([2,3,2,2,2,5,5,5,5,3,5,5,3]) == 4
assert taille_soussequence([2,3,2,2,2,3]) == 3
assert taille_soussequence([]) == 0
assert taille_soussequence([42]) == 1
assert taille_soussequence([1, 1, 2, 2, 2, 4, 4, 4, 4]) == 4


# Vous pouvez faire des affichages :
l = [1,2,2,2,3,3,3,3,4,4,4,4,4,4,4,4,2]
print("Longueur ss séquence de", l, ":", taille_soussequence(l))
```

Figure 2: un exemple de script Python

Les programmes Python sont simplement des fichiers textes, appelés aussi *fichiers scripts Python*, avec une ***extension .py***. Ils sont formés de lignes qui contiennent les instructions à exécuter, une par ligne. Le caractère **#** permet d'introduire des *commentaires* dans le code du programme (du # à la fin de la ligne).
Si l'instruction sur une ligne contient une expression non terminée — un (ou [ou { non fermé — alors elle se continue sur les lignes suivantes tant que l'expression n'est pas terminée par le) ou] ou } fermant. Un caractère **** tout en fin de ligne permet aussi de poursuivre l'instruction commencée sur la ligne suivante.
L'exécution est réalisée ligne par ligne en parcourant le programme de haut en bas, avec des ruptures et des reprises possibles grâce aux ***structures de contrôle*** (ici *while* et *if)* et aux ***fonctions*** (*def*).
Il utilise l'***indentation*** (décalage du texte) pour délimiter les ***blocs d'instructions*** liées à ces structures de contrôles et fonctions. La structure visuelle du programme est donc l'image de sa structure fonctionnelle.
Un fichier script Python constitue un ***module***, que l'on peut importer pour le réutiliser dans un autre programme.

5 - Installation et utilisation de Python

https://www.python.org/
https://docs.python.org/fr/3/tutorial/index.html

6 - Installation de Pyzo

https://pyzo.org/

Sources entre autres : **https://pixees.fr/informatiquelycee/n_site/nsi_prem_pythonbase.html**