1 Fondamentaux

- Déclaration/Affectation de variable: `var = 1` sera un entier, `var = 1.0` sera un réel, `var = "1"` sera une chaine de caractère, `var = [var1,var2,var3]` sera une liste.
- Changer le type d'une variable: `var = int("1")` sera un entier, `float( )` pour réel, `str( )` pour chaine de caractères.
- Variable saisie au clavier: `var = input("texte s'affichant à l'écran")`. Le contenu de la variable saisie au clavier est par défaut une chaine de caractère, il faudra la convertir si on veut des nombres.
- Affichage: `print()` éventuellement avec options `print(var, sep='; ', end=' ')` ou `print('Les variables {0} et {1}'.format(var1, var2))`
- Permuter des données: `var1, var2 = var2, var1`
- Opérateurs spéciaux: `//` , `%` (Quotient et reste de la division euclidienne) `**` (puissance)

2 Conditions et boucles

- Opérateurs logiques: `==` `<=` (et similaire) `x<y<z` `not` `and` `or`
- Condition si (if):

```
if (condition1):
        instruction1
elif (condition2):
        instruction2
else:
        instruction final
```

- Boucle pour (for):

```
for i in range(1, 10, 1):
    instruction à répéter pour i
    allant de 1 à 9 par pas de 1.
```

```
for fruit in [mûre, kiwi, yuzu]:
    instruction à répéter pour
    chaque fruit.
```

- Boucle tant que (while):

```
while (condition):
        instruction tant que condition est vraie
```

3 Les listes

La position des éléments dans la liste est donnée par un indice commençant à 0.

- Création d'une liste L:
  - ➔ Création directe : `L = [el1 ,el2 , el3]`
  - ➔ Liste des nombres de 2 à 20 (exclu) par pas de 3 : `L = range (2, 20, 3)`
  - ➔ Compréhension de liste : `carre =[x**2 for x in range(10)]`
- Sélections de liste:
  - ➔ Sélection de l'élément d'indice 2 (le 3eme) : `L[2]`
  - ➔ Sélection de 2eme éléments en partant de la fin : `L[-2]`
  - ➔ Sélection de l'élément d'indice 2 jusqu'à la fin de L : `L[2:]`
  - ➔ Sélection du début jusqu'à l'élément 2 d'indice 3 : `L[:3]`
  - ➔ Sélection du 2eme au 7eme (exclu) éléments par pas de 3 : `L[1:6:3]`
  - ➔ Sélection du premier élément de L et prendre le dernier élément de celui ci : `L[0][-1]`
- Opérations sur les listes:
  - ➔ Concaténation/Duplication de deux listes : `L1+L2` / `3*L`
  - ➔ Ajouter un élément à la fin d'une liste : `L.append(élément à ajouter)`
  - ➔ Insérer l'élément x dans la liste avant l'élément à la position i (le reste est décalé) : `L.insert(i,x)`
  - ➔ Enlever l'élément à la position i (le reste est décalé) : `L.pop(i)`
  - ➔ Enlever l'élément x (première occurence trouvée) : `L.remove(x)`
  - ➔ Trier une liste / avoir le maximum d'une liste : `L.sort()` / `max(L)`
  - ➔ Inverser l'ordre des items d'une liste : `L.reverse()`
  - ➔ Indice de la première occurrence de l'élément x : `L.index(x)`
  - ➔ Nombre d'occurence d'un élément dans une liste : `L.count(élément à compter)`
  - ➔ Connaitre la longueur/le maximum/le minimum d'une liste : `len(L)` / `max(L)` / `min(L)`
  - ➔ Savoir si le nombre 16 est dans la liste L : `16 in L`

4 Les fonctions

- Importer un module:
  `from math import *` importe toutes les fonctions du module math
- Modules courants: maths, random, time, tkinter (fenêtres), numpy, matplotlib (graphique), sympy (calcul formel), httplib (connections http), os (système), ...
- Définition d'une fonction:

```
def  nomDeLaFonction(param1, param2=valParDefault2, etc... ):
        """ description de la fonction, Docstring """
        Actions
        return var1, var2
```

Les variables sont locales SAUF pour les listes qui sont définitivement modifiées!

- Exemple d'un programme incluant une fonction:

```
#!/usr/bin/python3
#-*- coding: Utf-8 -*-

#Created on 16/09/2014
#Author:ASS

######Les fonctions
def factorielle(n):
    """ Fonction factorielle """
    facto = 1
    for i in range(1, n+1):
        facto *= i
    return facto

######Corps du programme
print('Ce programme affiche les n premieres factorielles')
fin = int(input("Combien de factorielles souhaitez-vous afficher ? "))
for i in range(1, fin+1):
    print('factorielle de {0} vaut {1}'.format(i, factorielle(i)))
######fin du programme
```

5 Gestions de fichiers

- Opérations sur les fichiers
  - ➔ Ouverture d'un fichier : `s=open('nomFichier','mode')` (Le fichier est pointé par l'objet s. Le 'mode' prend les valeurs 'w','r' ou 'a' suivant que écriture, lecture ou ajout en fin de fichier)
  - ➔ Fermeture du fichier pointé par l'objet s : `s.close()`
  - ➔ Écriture dans le fichier pointé par l'objet s : `s.write('chaine de caractère')`
  - ➔ Lecture du fichier pointé par l'objet s : `s.read()` ou `s.readlines()`
- Opérations sur le répertoire de travail
  - ➔ import du module os : `import os` (ne pas faire from os import * qui surcharge opérateur open)
  - ➔ Obtenir le répertoire de travail : `os.getcwd()`
  - ➔ Changer le répertoire de travail : `os.chdir('/home/')`
  - ➔ voir le site python.developpez.com (module os) pour plus d'infos.
- Gestion de fichier non-existant

```
filename = input("Veuillez entrer un nom de fichier : ")
try:
    f = open(filename, "r")
except:
   print("Le fichier", filename, "est introuvable")
```

6 Règles d'écriture d'un programme

On respectera les conventions suivantes lors de l'écriture du code :

- Bien commenter le code en indiquant régulièrement le role des instructions ;
- Choisir des noms de variables explicites ;
- Placer les fonctions en début de fichier ;
- Bien aérer le code. En particulier :
  - ➔ Avoir une taille fixe pour les indentations (4 espaces sont préconisées) ;
  - ➔ avoir des lignes pas trop longues (79 caractères maximum) ;
  - ➔ toujours placer une espace de chaque côté d'un opérateur (ex a = 3 plutot que a=3) ;
  - ➔ toujours placer une espace après une virgule, un point-virgule ou deux-points ;
  - ➔ ne jamais placer d'espace avant une virgule, un point-virgule ou deux-points ;
  - ➔ ne pas placer d'espace entre le nom d'une fonction et sa liste d'arguments.

Voir ci-contre pour un exemple.