Aide-mémoire Python

1. Variables, entrées, sorties

```
a = valeur           # affecte (ou réaffecte) une valeur à une variable
var2 = int(var1)     # convertit (si possible) var1 en nombre entier
var2 = float(var1)   # convertit (si possible) var1 en nombre flottant
var2 = str(var1)     # convertit (si possible) var1 en chaîne de caractères
a = input()          # affecte une valeur saisie au clavier
                     # à une variable, qui sera nécessairement de type str
a = input("texte")   # même instruction, avec affichage d'un message
a = int(input())     # convertit une valeur saisie au clavier en type int,
                     # puis l'affecte à la variable a
print(var)           # affiche la valeur d'une variable dans la console
print(var1, var2)    # affiche les valeurs de plusieurs variables
print("texte")       # affiche la chaîne "texte"
print("texte", var)  # affiche la chaîne "texte" suivie de la valeur de var
```

2. Opérations sur les nombres

```
a + b        # somme des deux nombres a et b
a - b        # différence des deux nombres a et b
a * b        # produit des deux nombres a et b
a / b        # quotient du nombre a divisé par le nombre b
a ** b       # nombre a élevé à la puissance b
a // b       # quotient dans la division euclidienne du nombre a par le nombre b
a % b        # reste dans la division euclidienne du nombre a par le nombre b
```

3. Listes

```
liste = []               # crée une liste vide
liste = [e1, e2, e3]     # crée une liste contenant les éléments e1, e2, e3
a = liste[i]             # affecte à a la valeur du terme de rang i (en commençant à 0)
liste[i] = a             # remplace par la valeur de a le terme de rang i
liste = liste + [e]      # ajoute l'élément e (en dernière position) à la liste
longueur = len(liste)    # affecte à la variable longueur le nombre d'éléments
                         # de la liste
liste2 = sorted(liste1)  # affecte à liste2 les éléments de liste1 triés
                         # dans l'ordre lexicographique
```

4. Opérateurs de comparaison

a et b désignent toutes les deux des variables de même type (nombre ou bien chaîne).

```
a == b               # a est égal à b
a != b               # a est différent de b
a < b                # a est inférieur à b
a > b                # a est supérieur à b
a <= b               # a est inférieur ou égal à b
a >= b               # a est supérieur ou égal à b
```

```
and                  # et
or                   # ou
not                  # non
```

5. Instructions conditionnelles (si alors sinon)

```
if condition:
    bloc d instructions à exécuter si condition est True (vraie)
suite du programme
```

```
if condition:
    bloc d instructions à exécuter si condition est True (vraie)
else:
    bloc d instructions à exécuter si condition est False (fausse)
suite du programme
```

```
if condition1:
    bloc d instructions si condition1 est True
elif condition2:
    bloc d instructions si condition1 est False et condition2 est True
else:
    bloc d instructions si condition1 et condition2 sont False
suite du programme
```

6. Boucle bornée (pour)

```
for compteur in range(n):
    # la variable compteur varie de 0 (inclus) à n (exclu)
    bloc d instructions à répéter
suite du programme
```

```
range(n)        # génère les entiers de 0 (inclus) à n (exclu)
range(m,n)      # génère les entiers de m (inclus) à n (exclu)
range(m,n,p)    # génère les entiers de m (inclus) à n (exclu) avec un pas p
```

```
for element in sequence:
    # sequence peut être une chaîne, une liste...
    bloc d instructions à répéter
suite du programme
```

7. Boucle non bornée (tant que)

```
while condition:
    bloc d instructions à exécuter tant que condition est True
suite du programme
```

8. Fonctions personnalisées

```
# pour une fonction renvoyant une valeur :
def nom_de_la_fonction(paramètre1, paramètre2):     # les paramètres sont facultatifs
    bloc d instructions
    return val_de_retour
```

```
# pour une fonction ne renvoyant pas de valeur (procédure) :
def nom_de_la_fonction(paramètre1, paramètre2):     # les paramètres sont facultatifs
    bloc d instructions
```

9. Fonctions du module `random`

```
from random import *    # importe la bibliothèque
random()            # sans argument, retourne un réel de l'intervalle [0;1[
uniform(a,b)        # retourne un réel de l'intervalle [a;b]
randint(n,p)        # retourne un entier choisi entre n et p (inclus)
choice(seq)         # retourne un élément de la séquence seq (liste ou chaine)
shuffle(seq)        # ne retourne rien, mais mélange les éléments de seq
```

10. Fonctions du module `math`

```
from math import *       # importe la bibliothèque
abs(x)                # retourne la valeur absolue de x
exp(x)                # retourne l'exponentielle de x
log(x)                # retourne le logarithme népérien de x
pow(x,y)              # retourne x puissance y (comme x**y)
sqrt(x)               # retourne la racine carrée de x
sin(x)                # retourne le sinus de x (x en radians)
cos(x)                # retourne le cosinus de x (x en radians)
tan(x)                # retourne la tangente de x (x en radians)
factorial(n)          # retourne la factorielle de n
pi                    # sans parenthèses, retourne la valeur de pi
e                     # sans parenthèses, retourne la valeur de e
```

11. Fonctions du module `turtle`

```
from turtle import *     # importe la bibliothèque
setup(largeur, hauteur) # définit les dimensions de la fenêtre en pixels
clear()               # efface la zone de dessin
reset()               # efface la zone de dessin, remet la tortue à l'origine
                      # et réinitialise ses paramètres
exitonclick()         # attend un clic de la souris pour fermer la fenêtre
up()                  # relève le crayon (pour avancer sans dessiner)
down()                # abaisse le crayon (pour recommencer à dessiner)
forward(distance)     # avance d'une distance donnée
backward(distance)    # recule
left(angle)           # tourne à gauche d'un angle donné (en degrés)
right(angle)          # tourne à droite
goto(x, y)            # déplace la tortue au point (x,y)
circle(rayon)         # dessine un cercle de rayon donné
position()            # retourne le couple de coordonnées de la tortue
                      #   position()[0] est l'abscisse
                      #   position()[1] est l'ordonnée
write(texte)          # affiche un texte (chaîne de caractères)
color(couleur)        # définit la couleur du tracé ("red", "blue", etc.)
width(épaisseur)      # définit l'épaisseur du tracé
shape("turtle")       # utilise une icône tortue
```

12. Fonctions de la bibliothèque `matplotlib`

```
from matplotlib.pyplot import *
                      # importe la bibliothèque
grid()                # crée un quadrillage
plot(xA, yA)          # crée le point de coordonnées (xA,yA)
plot(xA, yA, "ro")    # l'argument supplémentaire indique la couleur et la forme :
                      # "ro" pour un rond rouge, "g*" pour une étoile verte,
                      # "b+" pour une croix bleue...
plot([xA, xB...], [yA, yB...])
                      # crée la ligne polygonale passant par les points (xA,yA), (xB,yB)...
plot([xA, xB...], [yA, yB...], "ro")
                      # crée le nuage de points de coordonnées (xA,yA), (xB,yB)...
                      # là encore "ro" indique leur couleur et leur forme
plot([xA, xB...], [yA, yB...], "ro-", lw=5)
                      # "ro-" permet d'obtenir les points ET la ligne polygonale
                      # l'argument facultatif lw permet de préciser
                      # l'épaisseur du trait (exprimée en points)
scatter([xA, xB...], [yA, yB...] , s=[sA, sB...], c=[cA, cB...], alpha=opacite)
                      # crée un nuage de bulles centrées en (xA,yA), (xB,yB)...,
                      # de surfaces proportionnelles aux valeurs sA, sB...,
                      # et de couleurs cA, cB... (facultatif)
                      # le paramètre facultatif alpha indique l'opacité des bulles
                      # (de 0 : transparent, à 1 : opaque)
hist(liste, n)        # crée l'histogramme des valeurs de liste, regroupées dans n classes
show()                # affiche le graphique créé au préalable
```

13. Exemple de programme Python

La suite (u_n) est définie sur $\mathbb{N}^*$ par :

$$u_n = \sum_{k=1}^{n} \frac{1}{k^2}.$$

Cette suite (u_n) converge vers $\ell = \dfrac{\pi^2}{6}$.

Le programme suivant permet de déterminer le plus petit rang p pour lequel u_p approche ℓ avec une précision arbitraire, et d'afficher la valeur de u_p :

```
from math import *

def u(n):
    """ Cette fonction calcule la somme des 1/k^2, pour k de 1 à n """
    s = 0
    for k in range(1, n+1):
        s = s + 1 / k**2
    return s

# on saisit la précision souhaitée
precision = float(input("Quelle précision (réel positif) ?"))

# on s'assure que cette précision est bien positive
if precision <= 0:
    print("La précision doit être strictement positive !")
else:
    # on définit limite et on initialise p
    limite = pi**2 / 6
    p = 1

    # les mathématiques nous disent que cette boucle n'est pas infinie...
    while abs(u(p) - limite) > precision:
        p = p + 1

    # on affiche le résultat
    print("u(", p, ") vaut environ", u(p))
    print("Il approche la limite à", precision, "près.")
```

14. Références

- **Le site officiel de Python** (en anglais)
  ▶ http://www.python.org
- La « bible » de **Gérard Swinnen**. *Apprendre à programmer avec Python 3*. Eyrolles, 2012 (3e édition). 436 p. ISBN 978-2-212-13434-6.
  Disponible librement sur :
  ▶ http://www.inforef.be/swi/python.htm
- Le livre de **Vincent Maille**. *Apprendre la programmation par le jeu*. Ellipses, 2015 (2e édition). 224 p. ISBN 978-2-340-00459-7.
- **Le site d'ÉduPython**
  ▶ http://www.edupython.tuxfamily.org/