ALGO 01 – VARIABLES ET BIBLIOTHÈQUES

Pour rendre les programmes plus lisibles (pour vous ou pour les personnes qui vous corrigent), on peut **commenter les lignes** de codes grâce au symbole #. Python ignore alors tout ce qui suit le symbole # sur la ligne de code.

Entrée [1]:
```
1 + 1 #Ici, je réalise l'opération 1+1
```

Out [1]:
```
2
```

I Réaliser un calcul

Python permet de réaliser simplement les **calculs** usuels sur les nombres (`int` ou `float`).

Opération	Synthaxe Python	Exemple
Addition	`+`	La commande `2+5` renvoie `7`
Soustraction	`-`	La commande `2-5` renvoie `-3`
Multiplication	`*`	La commande `2*5` renvoie `10`
Division	`/`	La commande `2/5` renvoie `0.4`
Puissance	`**`	La commande `2**5` renvoie `032`

- En Python, les nombres décimaux s'écrivent avec un point et non pas une virgule.
- N'hésitez pas à mettre des **espaces** entre les nombres et les opérations pour aérer le code.
- Python respect les priorités d'opérations mais n'hésitez pas à mettre des **parenthèses** pour éviter les erreurs.

Exercice 1 Réaliser les calculs suivants à l'aide de Python.

$$8^3 \times \frac{1}{4^2}, \qquad \frac{2}{4} - \frac{1}{3}, \qquad \frac{36}{25} \times \frac{15}{12} \times 5.$$

Entrée [2]:
```
#Premier calcul
8**3*(1/4**2)

```

Out [2]:
```
32
```

Entrée [3]:
```
#Deuxième calcul
(2/4)-(1/3)

```

Out [3]:
```
0.16666666666666669
```

Entrée [4]:
```
#Troisième calcul
(36/25)*(15/12)*5

```

Out [4]:
```
9
```

II Déclarer une variable

En programmation, une **variable** est une case mémoire dans laquelle on stocke une information. En Python, cette information peut être de plusieurs types :

- chaîne de caractères : `str` (string)
- nombre entier : `int` (integer)
- nombre flottant (nombre à virgule) : `float`
- booléen (`True` ou `False`) : `bool`
- liste : `list`

L'**affectation des variables** se fait grâce au signe "=", en utilisant la syntaxe suivante :

```
nomdelavariable = valeur
```

Pour **afficher la valeur d'une variable**, on utilise la commande `print`. Attention aux noms des variables :

- Un nom de variable ne peut être composé que de lettres (majuscule ou minuscule), de chiffres et de tirets du bas. Pas d'accent, pas d'espace ! Le nom d'une variable doit toujours débuter par une lettre.
- Python distingue les minuscules des majuscules : la variable `x` n'est pas la même que la variable `X`.
- Il faut privilégier l'usage de **noms** de variables **descriptifs**, qui permettent de ce rappeler de l'information que contient la variable. Si on veut créer une variable contenant le nom d'élèves en ECG1, on l'appelle `nbreelevesECG1` plutôt que `x`.

Entrée [5]:
```
a = 2 #On affecte à la variable a la valeur 2
print(a) #On demande d'afficher la valeur de la variable a
```

Out [5]:
```
2
```

Entrée [6]:
```
message = 'Bonjour' #Ici, la variable est une chaine de caractères
print(message)
```

Out [6]:
```
Bonjour
```

Entrée [7]:
```
a, b = 3, 0.1 #Affecte simultanément 3 à a et 0.1 à b
print(b)
```

Out [7]:
```
0.1
```

On peut effectuer les opérations de calculs (addition, soustraction,...) sur les variables.

Entrée [8]:
```
A = 10
B = 12
C = 3
2*A-3*B
```

Out [8]:
```
-16
```

Entrée [9]:
```
C*(A+2*B)
```

Out [9]:
```
102
```

Enfin, on peut **incrémenter** la valeur d'une variable, c'est-à-dire définir la nouvelle valeur d'une variable à partir de son ancienne valeur.

Entrée [10]:
```
a = 1
print(a)
a = a + 1
print(a)
```

Out [10]:
```
1
1
```

Exercice 2 Pour tous les programmes donnés ci-dessous, noter sur votre feuille, dans chaque case du tableau les valeurs successives prises par les variables, et s'il y un affichage, écrire la valeur affichée.

(i) Premier programme.

```
s = 0
s = s + 1
s = s + 2
```

	Valeur de s
Ligne 1	0
Ligne 2	1
Ligne 3	3

(ii) Deuxième programme.

```
a=1
b=2
a=a-b
b=a+b
```

	a	b
Ligne 1	1	
Ligne 2	1	2
Ligne 3	-1	2
Ligne 4	-1	1

(iii) Troisième programme.

```
a, b, c = 1, 2, 1
a=2*c+3*b
c=b
b=a
```

	a	b	c
Ligne 1	1	2	1
Ligne 2	8	2	1
Ligne 3	8	2	2
Ligne 4	8	8	2

(iv) Quatrième programme.

```
a,b,c = 1,2,1
a = 2*c + 3*b
b=c
a=b
```

	a	b	c
Ligne 1	1	2	1
Ligne 2	8	2	1
Ligne 3	8	1	1
Ligne 4	1	1	1

(v) Cinquième programme.

```
1 x = 2
2 y = x * 3
3 print(y)
4 print(x + 5)
5 x = y
```

	x	y	Affichage
Ligne 1	2		
Ligne 2	2	6	
Ligne 3	2	6	6
Ligne 4	2	6	7
Ligne 5	6	6	

(vi) Sixième programme.

```
1 a,b = 1,6
2 a = 2 * b - 1
3 b = 2 * a
4 a = a - 2
```

	a	b
Ligne 1	1	6
Ligne 2	11	6
Ligne 3	11	22
Ligne 4	9	22

(vii) Septième programme.

```
1 x,y = 0,2
2 y = y - 1
3 print(x + y)
4 z = x + y + 2
5 x, y = y, x
6 print(x, y)
```

	x	y	z	Affichage
Ligne 1	0	2		
Ligne 2	0	1		
Ligne 3	0	1		1
Ligne 4	0	1	3	
Ligne 5	1	0	3	
Ligne 6	1	0	3	1,0

Exercice 3 Remets les lignes du programme suivant dans l'ordre de telle sorte qu'à la fin la variable x ait la valeur 46.

```
1 y = y - 1
2 y = 2 * x
3 x = x + 3 * y
4 x = 7
```

Correction non détaillée : 4 - 2 - 1 - 3

Exercice 4 Écrire un programme qui échange la valeur de deux variables. Par exemple, si au départ, la variable a vaut 2 et b vaut 5, à la fin, la la variable a doit valoir 5 et b doit valoir 2.

Entrée [11]:
```
#Correction sans la double affectation
a = 2 #Le programme doit fonctionner si on change la valeur de a ici
b = 5 #...ou celle de b ici

u = a #Variable auxiliaire qui sauvegarde la valeur de a
a = b
b = u

print(a) #Cette commande doit afficher 5 (dans notre exemple)
print(b)  #Cette commande doit afficher 5 (dans notre exemple)
```

Out [11]:
```
5
2
```

Entrée [12]:
```
#Correction avec la double affectation
a = 2 #Le programme doit fonctionner si on change la valeur de a ici
b = 5 #...ou celle de b ici

a, b = b,a

print(a) #Cette commande doit afficher 5 (dans notre exemple)
print(b)  #Cette commande doit afficher 5 (dans notre exemple)
```

Out [12]:
```
5
2
```

Exercice 5 Que permet de réaliser le programme suivant ?

```
a = ... #on suppose que l'utilisateur entre une valeur ici
b = ... #on suppose que l'utilisateur entre une valeur ici
a = a + b
b = a - b
a = a - b
```

Ce code permet d'échanger les valeurs de deux variables.

Exercice 6 Une pièce en forme de carré a été mesurée avec quatre bâtons de longueur respectives 17m, 7m, 5m et 2m. La longueur d'un côté de la pièce est égale à cinq fois la longueur du premier bâton, plus deux fois la longueur du second, plus une fois la longueur du troisième, plus deux fois la longueur du quatrième. Écrire un programme qui affiche la surface et le périmètre de la pièce.

Entrée [13]:
```
cote = 5*17 + 2*7 + 1*5 + 2*4
surface = cote**2
perimetre = 4*cote
print("La surface de la piece est", surface, "m2")
print("Le perimetre de la piece est", perimetre, "m")
```

Out [13]:
```
La surface de la piece est 12554 m2
Le perimetre de la piece est 448 m
```

III Importer une bibliothèque

Certains objets spécialisés (certaines fonctions mathématiques comme l'exponentielle ou le logarithme, les matrices,...) ne sont pas accessibles directement dans Python. Pour les obtenir, il faut **importer la bibliothèque**/la librairie dans laquelle ils se trouvent grâce à la synthaxe suivante :

```
import (nom de la bibliotheque) as (raccourci)
```

Cette commande permet d'importer la totalité des fonctionnalités de la librairie et que les objets de la librairie pourront être utilisés précédés de ce qui a été choisi pour raccourci (les raccourcis sont imposés par le programme). Le programme permet d'utiliser les librairies suivantes.

- `numpy` (raccourci `np`) : dédiée au calcul mathématique, elle donne accès aux principales fonctions mathématiques, aux constantes e et π et permet de faire des calculs sur des tableaux (et donc des vecteurs et des matrices)

Entrée [14]:
```
import numpy as np
```

Fonction usuelle	Synthaxe Python
Exponentielle	`np.exp`
Logarithme	`np.log`
Racine carrée	`np.sqrt`
Valeur absolue	`np.abs`
Partie entière	`np.floor`

Constante	Synthaxe Python
Nombre e	`np.e`
Nombre π	`np.pi`

- `matplotlib.pyplot` (raccourci `plt`) : dédiée au tracé des courbes et des surfaces et à la visualisation de données sous formes de graphique

Entrée [15]:
```
import matplotlib.pyplot as plt
```

- `pandas` (raccourci `pd`) : dédiée à l'analyse de données

Entrée [16]:
```
 import pandas as pd
```

Exercice 7 Réaliser les calculs suivants à l'aide de Python

$$\frac{\sqrt{5+\mathrm{e}}}{\pi} \qquad \ln(|1-\exp(6)|) \qquad \frac{\mathrm{e}^5+2}{5\times 10^6}$$

Entrée [17]:
```
#Premier calcul
np.sqrt(5+np.e)/np.pi
```

Out [17]:
```
0.8843220287568424
```

Entrée [18]:
```
#Deuxième calcul
np.log(np.abs(1-np.exp(6)))
```

Out [18]:
```
5.99751817063104
```

Entrée [19]:
```
#Troisième calcul
(np.exp(5)+2)/(5*10**6)
```

Out [19]:
```
3.008263182051532e-05
```

Exercice 8 Dans cet exercice, on cherche à résoudre grâce à Python des équations du second degré.

1. Résoudre au préalable à la main, les trois équations du second degré suivantes :

$$2x^2 - 10x + 8 = 0 \qquad 3x^2 + 5x - 2 = 0 \qquad x^2 - x - 1 = 0$$

Correction non détaillée.

- L'équation $2x^2 - 10x + 8 = 0$ admet deux solutions données par 1 et 4.
- L'équation $3x^2 + 5x - 2 = 0$ admet deux solutions données par -2 et $1/3$.
- L'équation $x^2 - x - 1 = 0$ admet deux solutions données par

$$\frac{1}{2} - \frac{\sqrt{5}}{2} \qquad \text{et} \qquad \frac{1}{2} + \frac{\sqrt{5}}{2}$$

2. Ecrire un programme qui, étant donnés les coefficients a, b et c d'un trinôme dont le discriminant est strictement positif, envoie les deux racines de ce trinôme.

Entrée [20]:
```
import numpy as np

a = 2
b = -10
c = 8

Delta = b**2-4*a*c
x1 = (-b-np.sqrt(Delta))/(2*a)
x2 = (-b+np.sqrt(Delta))/(2*a)

print("Les solutions de l'équation sont", x1, "et", x2)
```

Out [20]:
```
Les solutions de l'équation sont 1.0 et 4.0
```

Exercice 9 Calculer et afficher les cinq premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = \frac{1}{2}$ et

$$\text{pour tout } n \in \mathbb{N}, \qquad u_{n+1} = \frac{\exp(u_n)}{n+2}$$

Entrée [21]:
```
#Calcul de u0
n = 0
u = 1/2

#Calcul de u1
u = np.exp(u)/(n+2)
n = n + 1

#Calcul de u2
u = np.exp(u)/(n+2)
n = n + 1

....
```