TP1 Python
Introduction au langage

I)Généralités

I.1)Variables et type

Les variables sont notées à l'aide d'une lettre ou d'un mot, suivi éventuellement d'un chiffre ou d'un caractère spécial.

Quelques types de variables à connaître :

– le type **int** (integer=entier)
Exemple : a=2, temps=4, y1=−5

– le type **float** (=décimal)
Exemple : valeur_finale=3.5, resultat=10/3

– le type **str** (string=chaîne de caractères)
Exemple : a="Bonjour"

– le type **bool** (=booléen)
Utilisé pour des variables dont la valeur est **True** ou **False**.
Exemple : a=1 < 2. La variable a prend la valeur True car l'inégalité 1 < 2 est vraie.

> Remarque
> Python fait la différence entre majuscule et miniscule.

I.2)Opérations

Entre des variables numériques a et b, on définit :

– l'addition a+b et la soustraction a-b

– la multiplication a*b et la division a/b

– la puissance a**n

– l'opération a//b, quotient de la division euclidienne de a par b,

– l'opération a%b, reste de la division euclidienne de a par b.

I.3)Fonction print

La fonction print(...) sert à afficher du texte ou des valeurs à l'écran.

Exemple

```
x=2
y=x+1
print(y)
print("la valeur de x est", x)
```

A l'écran, s'affiche 3 et << la valeur de x est 2 >>.

I.4)Fonction input

La fonction input(...) sert à demander une information à l'utilisateur.
Ce que l'utilisateur tape au clavier est ensuite récupéré par le programme et restitué à l'écran.

Exemple

```
mot=input("entrez un mot")
```

Python affiche le message << entrez un mot >> et enregistre la réponse de l'utilisateur dans une variable mot de type string.

Exemple

```
x=int(input("entrez un nombre entier"))
```

Python affiche le message << entrez un nombre entier >>, puis enregistre la réponse de l'utilisateur dans une variable de type string qu'il convertit en nombre entier par la commande int.

I.5)Incrémentation

L'incrémentation (respectivement la décrémentation) consiste à augmenter (respectivement diminuer) la valeur d'une variable numérique x d'une quantité donnée.

Exemple

```
x=5
x=x+1
```

A la fin du programme, x vaut 6.

Exemple

```
x=5
x=x-2
```

A la fin du programme, x vaut 3.

II)Instructions conditionnelles

Les instructions conditionnelles (ou tests) sont de trois types :
if ... ou **if...else...** ou **if...elif...else...**

La condition placée après **if**, **else** ou **elif** utilise les opérateurs suivants :

and (et)

or (ou)

== (égal à)

<, >, <= et >=,

! = (différent de)

 Après la condition, il faut **mettre deux points**, passer à la ligne, puis **indenter** de 4 espaces les instructions du test.

II.1)Le test if...

Le test **if** ... demande au programme d'exécuter un bloc d'instructions si et seulement si une condition est vraie.

if condition **:**
    bloc d'instructions

Exemple

```
n=int(input("entrer n"))
if n%2==0:
    print("n est pair")
```

Exemple

```
age=int(input("entrez votre age"))
if age >= 18:
    print("vous êtes majeur")
```

II.2)Le test if...else...

Le test **if**...**else**... demande au programme de choisir entre deux actions selon qu'une condition est vraie ou fausse.

if condition **:**
    bloc1 d'instructions
else :
    bloc2 d'instructions

Exemple

```
age=int(input("entrez votre age"))
if age >= 18:
    print("vous êtes majeur")
else:
    print("vous êtes mineur")
```

II.3)Le test if...elif...else...

Le test **if**...**elif**...**else** permet de choisir entre plusieurs possibilités.

if... teste la première condition,
elif ... teste la seconde condition si la première est fausse,
else... s'exécute si aucune des conditions précédentes n'est vraie.

if condition1 **:**
    bloc1 d'instruction(s)
elif condition2 **:**
    bloc2 d'instruction(s)
else :
    bloc3 d'instruction(s)

Remarque
Dès qu'une condition est réalisée, on exécute le bloc et on sort du test.

Exemple
```
age=int(input("entrez votre age"))
if age >= 18:
    print("vous êtes majeur")
elif age <15:
    print("vous avez moins de 15 ans")
else:
    print("vous avez 15,16 ou 17 ans")
```

Remarque
Entre if et else, il est possible de mettre plusieurs elif.

III) Boucles

III.1) Boucle for

La boucle for permet de répéter plusieurs fois une même instruction.
Elle est très pratique lorsqu'on connaît à l'avance le nombre de répétitions ou bien lorsqu'on veut parcourir une liste ou une chaîne de caractères.

for k **in** collection **:**
bloc d'instructions

Cette boucle exécute le bloc d'instructions autant de fois que de valeurs de k parcourant la collection (liste, objet de classe range, ...)

Exemple
```
for k in [1,2,3]:
    print(k)
```
A l'écran s'affichent 1 2 et 3.

Exemple
```
for k in range(5):
    print(k)
```
A l'écran s'affichent 0,1,2,3 et 4.

Exemple
```
for k in range(2,7):
    print(k)
```
A l'écran s'affichent 2,3,4,5 et 6.

Remarque importante :
range(n) génère une suite de nombres entiers allant de 0 à n-1.
range(p,n) génère une suite de nombre entiers allant de p à n-1.
Attention donc au décalage à la fin !!!

III.2)Boucle while

La boucle while permet de répéter des instructions tant qu'une condition est vraie.
Contrairement à la boucle for, on ne sait pas forcément à l'avance combien de fois la boucle va s'exécuter.

while condition **:**
bloc d'instructions

Exemple

```
i=0
while i<100:
    print(i)
    i=i+1
```

A l'écran s'affichent tous les entiers consécutifs de 0 à 99.

 A la fin de la ligne for ou while, il faut **mettre deux points**, passer à la ligne, puis **indenter** de 4 espaces le bloc d'instructions de la boucle.

IV)Fonctions

Une fonction est un bloc d'instructions qui réalise une tâche précise (un peu comme une boîte à outils). Elle évite d'avoir à répéter plusieurs fois les mêmes instructions dans le programme principal.

def nom_de_la_fonction(arguments) **:**
bloc d'instructions
return resultat

Exemple

```
def carre(x):
    return x**2
```

La commande carre(3) renvoie alors 3^2, c'est-à-dire 9.

Exemple

```
def carre(x):
    y=x**2
    return y
```

C'est le même programme, mais avec l'ajout d'une variable locale y.

La présence d'arguments (= paramètres d'entrée) ou de paramètre de sortie (= ce qu'on retourne) dans une fonction n'est pas obligatoire.

Exemple

```
def compteur(stop):
    i= 0
    while i<stop:
        print(i)
        i=i+1
```

La commande compteur(4) affiche à l'écran 0,1,2,3.
Cette fonction ne possède pas de paramètre de sortie.

Exemple

```
def hello():
    print("bonjour")
```

La commande hello() affiche bonjour.
Cette fonction ne possède ni argument ni paramètre de sortie.

V)Exercices

Exercice 1 ★ ☆ ☆☆
Sur la console, déteminer la valeur et le type des variables ci-dessous :
a=2**10 b=2*3+4/5 c=1515/5 d=1515//5 e=37%5
f=6 > 5 and 4 < 3 g=1*2*3*4*5==120

Exercice 2 ★ ☆ ☆☆
1)A l'issue du programme suivant, quelles valeurs prennent a,b,c ?

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

2)Ecrire un programme effectuant successivement les tâches suivantes :
– affecter 1 à la variable x, puis 2 à la variable y,
– échanger les valeurs de x et y (sans utiliser d'affectations parallèles).

Exercice 3 ★ ☆ ☆☆
Que va afficher ce programme ?

```
x=10
y=15
z=x+y
x=y
y=z
print(x+y+z)
```

Exercice 4 ★ ☆ ☆☆
On considère le programme ci-dessous :

```
a=int(input("entrer a"))
b=int(input("entrer b"))
c=int(input("entrer c"))
if a>=b and a>=c:
    d=a
elif b>=a and b>=c:
    d=b
else:
    d=c
print(d)
```

1)L'utilisateur entre pour a,b,c les valeurs 2,3 et 4. Que va afficher Python ?
2)Même question s'il entre les valeurs 5,8 et 2.
3)Plus généralement, quel est le but de ce programme ?

Exercice 5 ★ ☆ ☆☆

1)On exécute le programme ci-dessous. Que va afficher Python ?

```
for i in range(5):
    print(i**2)
```

2)On exécute le programme ci-dessous. Que va afficher Python ?

```
c=1
while c<100:
    print(c)
    c=2*c
```

Exercice 6 ★ ☆ ☆☆

1)A l'aide de tests, écrire une fonction Python d'en-tête `def f(x):` d'argument x et renvoyant la valeur de f(x) défini par :

$$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ x & \text{si } 0 < x < 1 \\ 1 & \text{si } x \geq 1 \end{cases}$$

2)Vérifier les valeurs de $f(-1)$, $f(0,7)$ et $f(2)$.

Exercice 7 ★ ★ ☆☆

On considère la suite $(u_n)_{n\geq 0}$ définie par $u_0 = 10$ et $\forall n \in \mathbf{N}$, $u_{n+1} = \frac{1}{2}u_n - 3$.

1)Ecrire un programme qui pour une valeur de n entrée par l'utilisateur affiche u_1, ..., u_n.
On vérifiera que $u_{10} \approx -5.98$.

2)Conjecturer la limite de la suite $(u_n)_{n\geq 0}$.

3)Modifier le programme précédent pour qu'il affiche le plus petit entier n tel que $|U_n + 6| \leq 10^{-3}$. On trouvera $n = 14$.

Exercice 8★ ★ ☆☆

A l'aide d'une boucle, écrire un programme qui pour une valeur de n entrée par l'utilisateur affiche la valeur de la somme $\sum_{k=1}^{n} k^3$.

On vérifiera que $\sum_{k=1}^{100} k^3 = 25502500$.

Exercice 9★ ★ ☆☆

1)Ecrire un programme qui calcule la somme de tous les entiers de 1 à 2025 qui sont divisibles par 3.

2)Ecrire un programme qui calcule la somme de tous les entiers de 1 à 2025 qui sont divisibles par 3 ou par 5.

Exercice 10★ ★ ☆☆
1)Ecrire une fonction d'en-tête `def d(n):`
prenant en argument $n \in \mathbf{N}^*$ et renvoyant le nombre de diviseurs de n.

2)En déduire un programme permettant de connaître si un nombre entier r entré par l'utilisateur est premier ou non.

Exercice 11★ ★ ★☆ (suite de Syracuse)
On considère la suite $(u_n)_{n\geq 0}$ définie par $u_0 = 2026$ et pour tout $n \in \mathbf{N}$:

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

1)Ecrire un programme permettant d'afficher u_1, u_2, ..., u_{200}.
Exécuter ce programme. Que constate t-on ?

2)Changer la valeur de u_0. Observe t-on toujours la même chose ?

3)Ecrire un programme qui affiche le temps de vol, c'est-à-dire le plus petit entier n tel que $u_n = 1$.

4)Ecrire un programme qui renvoie l'altitude, c'est-a-dire la plus grande valeur de u_n pour les entiers n inférieurs ou égaux au temps de vol.

Exercice 12 ★ ★ ★☆
1)Ecrire une fonction d'en-tête `def factorielle(n):`
d'argument $n \in \mathbf{N}^*$ et qui renvoie $n!$

2)Ecrire une fonction d'en-tête `def sommechiffres(n):`
d'argument $n \in \mathbf{N}^*$ et qui renvoie la somme des chiffres de n.

Indication : on pourra faire intervenir le reste et le quotient d'une division euclidienne par 10.

3)En déduire la somme des chiffres de 1000 !