Python

I. Instructions élémentaires 1

Les exercices de cette section sont à effectuer sans utiliser Python, seulement sur feuille.

Exercice 1
Pour chacune des expressions données plus bas,

- Indiquer ce que sera le résultat obtenu ;
- Indiquer le type du résultat :
  `int` (entier), `float` (flottant), `str` (chaîne de caractère) ou `bool` (booléen).

1. `2*5`
2. `2.0*4`
3. `4/3`
4. `4//3`
5. `2 == 5`
6. `"3"*2`
7. `str(4) == 4`
8. `15%4`
9. `2**3`

Exercice 2 (Vrai/Faux)
Pour chacune des affirmations suivantes, dire si elle est vraie ou fausse.
Justifier si possible la réponse.

1. Après les instructions :

```
x=3
y=5
x=y
y=x
```

   la valeur de x est 5 et celle de y est 3.

2. Après les instructions :

```
x=3
y=5
y==x
x=y
```

   la valeur de x est 5 et celle de y est 5.

3. Une instruction de la forme `if ...` est une boucle conditionnelle.
4. Avec l'instruction `for i in range(10):`, la variable `i` prend 9 valeurs puisque la dernière est 9.

Exercice 3 (QCM)
Pour chaque question, une seule réponse parmi celles proposées est exacte.

1. Le langage Python a été créé en :

(a) 1971 (b) 1991 (c) 2001 (d) 2011

2. Parmi les propositions suivantes, laquelle n'est pas une expression ?

(a) a<b (b) a!=b (c) a=b (d) a>=b

3. On considère les instructions suivantes :

```
a=8
b=5
a==b+1
b=b+1
a==b+1
b=b+1
print(a==b+1)
```

Quel est le résultat affiché ?

(a) 8 (b) une erreur (c) False (d) True

4. Combien de fois la fonction `print` est-elle appelée dans le code en Python suivant ?

```
n=4
for i in range(2,n):
    print(i)
```

(a) Jamais (b) une fois (c) deux fois (d) trois fois

5. Quelle est la valeur finale de x après l'exécution du code Python suivant ?

```
x=1
for i in range(4):
    x=x+i
```

(a) 6 (b) 7 (c) 10 (d) 11

6. Quelles sont les valeurs finales de x et y après l'exécution de ce code ?

```
x=4
while x>0:
    y=0
    while y<x:
        y=y+1
        x=x-1
```

(a) x=-1, y=0 (b) x=0, y=0 (c) x=0, y=1 (d) boucle infinie

Exercice 4

Dans chacun des deux cas, quelle est la valeur de `x` à la fin de l'exécution du code ?

```
x=1
n=1
while n>1:
    x=x+n
    n=n-1
```

```
x=0
for i in range(2):
    x=x*i
    for j in range(3):
        x=x+j
```

Exercice 5
Dans chacun des cas suivants, indiquer l'affichage obtenu lors de l'exécution du code.

1.
```
for n in range (10):
  print(n)
```

2.
```
for n in range(2,7):
  print(n)
```

3.
```
for n in range(100,110,2):
  print(n)
```

Exercice 6
Que vaut f à la fin de l'exécution des d'instructions suivantes ?

```
n=5
f=0
i=1
while i<n+1:
  f=f+i
  i=i+1
```

Exercice 7
Que fait cette suite d'instructions ?

```
s=0
for i in range(5) :
  x=int(input('Entrez un nombre : '))
  s=s+x
```

II. Instructions élémentaires 2

À partir d'ici, les codes sont à écrire et tester en Python, sauf indication contraire.

Exercice 8
Une séance de cinéma est interdite aux moins de 16 ans. Le prix du billet varie avec l'âge : les seniors (à partir de 65 ans) et les mineurs (moins de 18 ans) paient un tarif réduit, les autres un tarif plein.
Écrire un programme qui demande à un utilisateur de saisir son âge et qui lui donne une réponse quant à son autorisation de voir le film ainsi que le tarif du billet.
Exemple : si l'utilisateur saisit 22 , le programme doit afficher « Vous pouvez-voir ce film, le billet est au tarif plein. »

Exercice 9
Une année est bissextile si elle est divisible par 4 mais non divisible par 100. Les années divisibles par 400 sont également bissextiles.
Écrire un programme qui demande à l'utilisateur de saisir une année et qui affiche un message pour préciser si cette année est bissextile ou non.

Exercice 10
Écrire un programme qui fait faire 10 tours de manège en affichant un message à chaque tour :

```
C'est le tour n°1.
C'est le tour n°2.
...
C'est le tour n°10.
```

Aide : Utiliser l'argument `sep=""` dans la fonction `print` pour ne pas avoir d'espace autour du numéro du tour.

Exercice 11
Écrire un programme qui affiche tous les nombres entre 1 et 10, et indique pour chacun si celui-ci est pair ou impair :

```
1 est impair
2 est pair
...
10 est pair
```

Exercice 12
Écrire un programme qui redemande un mot de passe tant que celui qui est donné (par le biais d'un `input`) n'est pas le bon. Une fois le bon mot de passe donné, le programme affiche « Vous êtes connecté. » et s'arrête. Le mot de passe attendu, de type `str`, est à définir dans le programme.

Exercice 13
Écrire un code qui affiche, à l'aide d'une boucle `for`, les nombres de 0 à 99, à raison de 10 nombres par ligne :

```
0 1 2 3 4 5 6 7 8 9
10 11 12 13...
...
90 91 92 93 94 95 96 97 98 99
```

Aide : utiliser les paramètres `sep` et/ou `end` de la fonction `print` (voir la fiche sur Python ou chercher dans la librairie Python).

III. Fonctions

Exercice 14 (QCM – sans utiliser Python)
Voici ci-contre une fonction définie en Python.
que retourne la fonction f si le paramètre x vaut 15 ?

```
def f(x):
    for d in range(2,x):
        if x%d == 0:
            return d
```

1. 3
2. 5
3. 3 et 5
4. 3, 5 et 15

Exercice 15
Écrire une fonction qui prend en paramètre deux nombres et retourne le plus grand des deux.

Exercice 16
Écrire une fonction qui prend en paramètre trois nombres et retourne le plus grand des trois nombres.

Exercice 17

1. Écrire une fonction `SommeEntiers` qui prend en paramètre un entier `n` et qui retourne la somme des `n` premier entiers naturels non nuls $(1 + 2 + 3 + \cdots + n)$.
2. Écrire une fonction `SommeCarres` qui prend en paramètre un entier strictement positif `n` et qui retourne la somme des `n` premier carrés non nuls $(1^2 + 2^2 + 3^2 + \cdots + n^2)$.
3. Définir une fonction `SommeInverses` qui prend comme argument un entier n et qui retourne la somme des inverses des entiers entre 1 et n.
4. Écrire une fonction `SommeDiviseurs` qui prend en paramètre un entier naturel non nul et retourne la somme de ses diviseurs.

Exercice 18

1. Écrire une fonction `Prime` qui prend en paramètre un entier naturel non nul et retourne `True` si ce nombre est premier et `False` sinon.
2. Écrire une fonction `sumPrime(n)` qui retourne la somme des nombres premiers compris entre 1 et n (inclus).

Exercice 19
En utilisant la fonction `randint` du module `random`, écrire une fonction `Jeu` qui prend en paramètre un nombre entier `n` strictement positif, simule `n` fois le tirage d'un dé cubique équilibré (donc une valeur au hasard parmi les nombres 1,2,3,4,5 et 6), puis retourne le pourcentage de 6 obtenus.

IV. Fonctions – autres exercices

Exercice 20
Définir une fonction qui prend en argument un nombre entier `n` et qui :

- si `n` est pair, le divise par 2 ;
- si `n` est impair, le multiplie par 3 et lui ajoute 1.

puis qui retourne la nouvelle valeur de `n`.

Exercice 21
On considère une fonction f dont l'expression est $f(x) = -2x$ si $x < 0$ et $f(x) = x^2$ sinon.
Définir une fonction qui prend en argument un nombre flottant x et qui retourne l'image $f(x)$.

Exercice 22
Même chose que l'exercice précédent, mais avec une fonction g définie en trois morceaux :

$$g(x) = \begin{cases} x^3 + 2 \text{ si } x < 0 \\ x^2 + 2 \text{ si } 0 \leqslant x < 3 \\ 14 - x \text{ si } x \geqslant 3 \end{cases}$$

Exercice 23
Définir une fonction `factorielle` qui prend comme argument un entier `n` et qui retourne le produit des entiers de `1` à `n`.

Exercice 24
Définir une fonction qui prend comme un argument un entier `n` et qui retourne le nombre d'entiers i entre -`n` et `n` (compris) pour lesquels $f(i) \leqslant g(i)$ (fonctions définies dans les exercices précédents).

Exercice 25
Définir une fonction qui prend comme argument un entier n, qui demande n nombres (flottants) et retourne la moyenne de ces nombres.

Exercice 26
Définir une fonction qui prend pour argument un nombre flottant $A > 0$ et qui retourne le plus petit entier n tel que $n^3 + 2n + 1 \geqslant A$.

Exercice 27
Définir une fonction `SommeDuree` qui prend pour argument quatre nombres entiers $m1$,$s1$,$m2$,$s2$ représentant deux durées en minutes et secondes, et qui retourne la somme des deux durées en minutes et secondes sous forme de chaîne de caractères.

Exercice 28
Définir une fonction sans argument qui demande deux nombres (flottants) et un symbole d'opération (`"+"`, `"-"`, `"/"` ou `"*"`) puis qui effectue et retourne le résultat du calcul à effectuer.

Exercice 29
Un nombre parfait est un nombre n dont la somme des diviseurs propres (c'est à dire strictement inférieurs à n) est égal à n. Définir une fonction qui affiche les entiers n parfaits situés entre 2 et 1000.
Pour cela on pourra réutiliser la fonction `SommeDiviseurs` vue dans un exercice précédent.

Exercice 30
Deux nombres entiers M et N sont amicaux si la somme des diviseurs propres de M est égal à N et si la somme des diviseurs propres de N est égale à M.
Définir une fonction qui prend en argument un nombre entier `maxi` et qui affiche les couples de nombres amicaux inférieurs à `maxi`.
On pourra tester la fonction avec `maxi=3000`.

Exercice 31
Écrire une fonction qui convertit un nombre binaire en nombre décimal.
On pourra considérer deux manière différentes de donner le nombre binaire :

1. Sous forme de chaîne de caractère
2. Sous forme d'un entier écrit seulement avec des 0 et des 1.

Exercice 32
Écrire une fonction qui prend en paramètre un entier n et affiche, pour i allant de 1 à n, i étoiles sur la ligne numéro i.
Par exemple, pour n = 5, afficher :

```
*
* *
* * *
* * * *
* * * * *
```

Exercice 33

1. Écrire une fonction qui prend en paramètre un entier positif n et affiche un carré d'étoiles plein de côté n.
   Par exemple, si n vaut 4, votre fonction affichera :

   ```
   ****
   ****
   ****
   ****
   ```

2. Modifier votre fonction pour qu'elle affiche un carré creux.
   Par exemple pour n valant 4,

   ```
   ****
   *  *
   *  *
   ****
   ```

Exercice 34
Écrire une fonction qui prend en entrée un entier n et affiche une pyramide d'étoiles de hauteur n. Si n est négatif, la pyramide devra être inversée.
Par exemple, sur entrée n = 4, afficher :

```
   *
  * *
 * * *
* * * *
```

Sur entrée n = -3, afficher :

```
* * *
 * *
  *
```

Exercice 35
Écrire une fonction `power(x, n)` qui retourne la valeur de x^n, bien sûr sans utiliser l'opérateur **.

Exercice 36
définir une fonction qui prend pour paramètres trois entiers `n`, `inf` et `sup`, et qui affiche la table de multiplication de `n` entre `inf` et `sup`. Par exemple, l'exécution de la fonction avec les paramètres valant respectivement 2, 3 et 5 affichera :

```
2*3=6
2*4=8
2*5=10
```

Penser à vérifier que `inf` est bien inférieur à `sup` et, si ce n'est pas le cas, échanger les valeurs.

Exercice 37
Soit f la fonction mathématique définie par $f(x) = x^3 - 3x + 2$.
Définir une fonction `tabulation` qui prend en paramètre trois nombres `inf`, `sup` et `pas` de type `float` et qui affiche les images des valeurs de `inf` à `sup` avec un pas de `pas`, sous une forme compréhensible (comme `f(1)=0`).
Même chose que pour l'exercice précédent pour ce qui est de `inf` et `sup`.

Exercice 38
On donne ci-dessous deux manières d'obtenir le PGCD (plus grand diviseur commun) de deux nombres entiers positifs A et B. Définir, suivant chacune de ces manières, une fonction qui prend en argument les valeurs A et B et qui retourne leur PGCD.

1. L'algorithme d'Euclide utilise le reste de la division entière de A par B :
   - soit Q le quotient de la division de A par B et R le reste.
   - si R=0 alors le PGCD est B
   - sinon on remplace A par B , B par R et on recommence.
2. On a propriété suivante :
   - PGCD(A,B)=PGCD(A-B,B) si A>B
   - PGCD(A,B)=PGCD(A,B-A) si A<B
   - PGCD(A,B)=A si A=B.

V. Listes

Exercice 39
Partie 1
Exécuter les instructions suivantes ligne après ligne dans un interpréteur Python. Observer à chaque étape ce qui est éventuellement retourné. Dans les cas où rien n'est retourné, ou si ce qui est retourné n'est pas une liste, refaire afficher le contenu de la liste (en entrant simplement `s`). Indiquer alors ce que font les instructions.

```
s=['Lune',2.0,'mardi',230,12,True,'samedi','a']
s
s[0]
s[-3]
s[1:]
s[:2]
s[1:3]
s[0:4:2]
s[:]
len(s)
del(s[1])
s+['jeudi','vendredi']
s
```

```
s*2
s[2]='mercredi'
s.insert(4,None)
s[3:5]=['jeudi','vendredi']
s.pop(-3)
s.pop()
s.append('dimanche')
s.remove('Lune')
s=['lundi']+s
s.index('jeudi')
s.reverse()
s.sort()
s
```

Partie 2
Écrire des lignes de code dans un fichier Python qui effectuent les actions suivantes successivement. Pour les fonctions et méthodes spécifiques, chercher dans la librairie Python sur les listes.

1. Définir la liste `l` contenant les éléments suivants dans l'ordre :
   45,17,89,38,10 et 74.
2. Trier `l` (utilisation de la méthode `sort` : `l.sort()`) puis l'afficher.
3. Ajouter l'élément « 12 » puis afficher la liste.
4. Renverser la liste puis l'afficher.
5. Afficher l'indice de l'élément « 10 ».
6. Enlever l'élément « 38 » et afficher la liste.
7. Afficher la sous-liste du 2e au 3e élément.
8. Afficher la sous-liste du début au 2e élément.
9. Afficher la sous-liste du 3e élément au dernier.
10. Afficher le deuxième élément en partant de la fin.

Exercice 40
1. Indiquer l'affichage obtenu lors de l'exécution du code suivant :

```
voyelles=['a','e','i','o','u','y']
for voyelle in voyelles:
  print(voyelle)
```

2. Faire un autre code qui effectue la même chose mais à l'aide d'une boucle de cette forme :

```
for i in range(len(voyelles)):
  ...
```

Exercice 41

1. Écrire un script qui génère la liste des carrés des nombres de 20 à 40 sans utiliser une définition de liste par compréhension.
2. Faire la même chose en utilisant une définition de liste par compréhension.

Exercice 42

Écrire un script qui crée automatiquement la liste des sinus des angles de 0° à 90°, par pas de 5°.

 la fonction `sin()` du module `math` considère que les angles fournis en argument sont en radians (360° = 2π radians).

Noter que l'on ne veut pas afficher les valeurs, mais créer la liste contenant ces valeurs.

Exercice 43

Écrire un script qui permette d'obtenir à l'écran les 15 premiers termes des tables de multiplication par 2, 3, 5, 7, 11, 13, 17, 19, ces nombres étant placés au départ dans une liste, sous la forme d'une table similaire à la suivante :

```
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
3 6 9 12 15 18 21 24 27 30 33 36 39 42 45
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75
# etc.
```

Exercice 44

Soit la liste suivante :
`['Yun-Sung','Mark','Charline','Aissatou','Adam','Jérôme','Bouchra']`.
Écrire un script qui affiche chacun de ces prénoms avec le nombre de caractères correspondant.
Indication : la fonction `len` s'applique aussi aux chaînes de caractères.

Exercice 45

Écrire un script qui recherche le mot le plus long dans une phrase donnée.
L'utilisateur du script doit pouvoir entrer une phrase de son choix, dont on supposera qu'elle ne contient pas de caractères spéciaux, en particulier pas de point ni de virgule (seulement des mots séparés par des espaces).
Indication : l'instruction `chaine.split(" ")` transforme une chaîne de caractères `chaine` en liste en prenant comme séparateur la chaîne `" "`. Autrement dit, avec une telle instruction on obtient une liste de mots à partir d'une chaîne de caractères contenant des mots séparés par des espaces.

Exercice 46

Écrire un script capable d'afficher la liste de tous les jours d'une année imaginaire, laquelle commencerait un jeudi. Le script utilisera lui-même trois listes : une liste des noms de jours de la semaine, une liste des noms des mois, et une liste des nombres de jours que comportent chacun des mois (ne pas tenir compte des années bissextiles). Exemple de sortie :

```
jeudi 1 janvier
vendredi 2 janvier
samedi 3 janvier
dimanche 4 janvier
# etc., jusqu'au jeudi 31 décembre.
```

VI. Listes et fonctions

Exercice 47
Écrire une fonction en Python qui prend pour argument une liste et une valeur x et qui retourne le nombre d'occurrences (apparitions) de `x` dans la liste.

Exercice 48
Une série statistique est une liste de couples de valeurs `(x,n)` représentant une valeur et un effectif. Écrire une fonction en Python qui prend en argument une série statistique et qui retourne la moyenne pondérée de cette série.

Exercice 49
Écrire une fonction en Python qui prend en argument une liste de nombres et retourne le maximum de la liste ainsi que la liste des indices de tous les éléments égaux au maximum.

Exercice 50
Écrire une fonction qui prend en argument une liste de nombres entiers quelconques, certains d'entre eux étant présents en plusieurs exemplaires, qui retourne une copie triée de cette liste dans laquelle il n'y a pas de doublon (on pourra utiliser la fonction `sorted()`).

Exercice 51
Écrire une fonction `progression(a, b, n)`, où `a`, `b` et `n` sont trois entiers, et qui retourne une liste de taille `n` contenant les entiers `a`, `a + b`, `a + 2b`, `a + 3b`, ..., `a + (n-1)b`.

Exercice 52
Écrire une fonction `interlace(l1, l2)` qui prend en entrée deux listes `l1` et `l2` de même longueur et retourne une liste de longueur double qui contient les valeurs des listes de façon entrelacée, c'est-à-dire :
`[l1[0], l2[0], l1[1], l2[1], ..., l1[n], l2[n]]` (`n` est ici la longueur des deux listes).
Par exemple, appliquée à `[0, 1, 6]` et `[2, 4, 7]`, elle va retourner la liste `[0, 2, 1, 4, 6, 7]`.

Exercice 53

1. Écrire une fonction qui prend en argument un mot (de type `str`) et retourne `True` si le mot commence et se termine par la même lettre et `False` sinon.
2. Écrire une fonction qui prend en argument deux mots (de type `str`) et retourne `True` si les deux mots commencent par la même lettre et se terminent par la même lettre et `False` sinon.

Exercice 54
Écrire une fonction `Double` qui prend en argument un mot (de type `str`) et retourne le mot obtenu en doublant chaque lettre. Par exemple, `Double("bon")` retourne `"bboonn"`.

Exercice 55

1. Définir une fonction Python prenant comme argument un entier naturel n et qui renvoie une liste de la forme : `[1,1,1,2,1,3,1,4,1,5,1,6,...,1,n]`.
2. Définir une fonction Python prenant comme argument un entier naturel n et qui construit une liste de la forme : `[1,1,2,1,2,3,1,2,3,4,...,1,2,3,..., n]`.