Premiers programmes : chaînes de caractères et fonctions

Exercice 1. Tester les instructions suivantes :

1.

```
>>> a="Hello "
>>> b='World !'
>>> a+b
```

2.

```
>>> 3*a
```

Exercice 2. Modifier la fonction welcome suivante pour qu'elle ait comme argument votre prénom et l'affiche après bonjour.

```
def welcome():
    return "Bonjour"
```

```
>>> welcome("Pierre")
"Bonjour Pierre"
```

Exercice 3. 1. Implémenter une fonction qui renvoie la distance AB de deux points A et B du plan, avec l'appel :

```
>>> distance(xA,yA,xB,yB)
```

indication : la fonction racine carrée (sqrt) nécessite le module math.

```
from math import *
```

2. Modifier la fonction précédente pour que l'appel soit :

```
>>> A=[2,3]
>>> B=[5,7]
>>> distance(A,B)
```

Instructions conditionnelles

Exercice 4. Un magasin de reprographie propose un tarif dégressif. Les 20 premières photocopies sont facturées à 10 centimes et les suivantes à 8 centimes.

Implémenter une fonction qui renvoie le montant de la facture en euros pour un nombre de photocopies donné

Exercice 5. Implémenter une fonction qui, en fonction de la moyenne obtenue au baccalauréat, renvoie la mention obtenue.

```
>>>mention(16.2)
"Très bien"
```

Remarque : on pourra utiliser l'instruction `elif`

Exercice 6. 1. Implémenter la fonction `RectangleEnPremierSommet(A,B,C)` qui renvoie True si le triangle ABC est rectangle en A.
(utiliser la fonction `distance` *implémentée précédemment)*

2. Implémenter la fonction `Rectangle(A,B,C)` qui renvoie True si le triangle ABC est rectangle.

Exercice 7. 1. Implémenter la fonction `pair(n)` qui renvoie True si n est pair, False sinon.

```
>>> pair(23)
False
```

2. Implémenter une fonction `mediane(L)` qui retourne la médiane de la liste L en utilisant la définition donnée en seconde.

Exercice 8. Un numéro de Sécurité Sociale est constitué de 13 chiffres. On lui ajoute une clé de contrôle, constituée de 2 chiffres et donnée par la formule :

$$\text{clé} = 97 - r$$

où r est le reste dans la division euclidienne du numéro de Sécurité Sociale par 97.

Implémenter une fonction de contrôle : `controle(numero,cle)` vérifiant le numéro de Sécurité Sociale.

Exercice 9. 1. Implémenter une fonction `milieu(A,B)` qui renvoie le couple de coordonnées du milieu de $[AB]$.

2. Implémenter une fonction dont l'appel serait

```
>>> test_parallelogramme(A,B,C,D)
```

qui indique si le quadrilatère $ABCD$ est un parallélogramme.

Boucle bornée

Exercice 10. Implémenter une fonction `SommeGeometrique(q,n)` qui renvoie la somme

$$1 + q + q^2 + \ldots + q^n$$

Exercice 11. Écrire un script qui calcule la moyenne de plusieurs notes, dont le nombre est saisi au préalable :

```
Nombre de notes : 2
Note 1 : 15
Note 2 : 11.5
Moyenne : 13.25
```

Exercice 12. Implémenter deux fonctions : `moyenne(L)` et `ecarttype(L)` (la seconde fonction faisant un appel à la première), qui calculent la moyenne et l'écart-type de la liste L

Exercice 13 (Méthode des rectangles)**.** On suppose que l'on a défini une fonction f, par exemple :

```
>>> from math import *
>>> def f(x):
...     return(exp(-x**2))
```

Implémenter la fonction `rectangle(a,b,n)` qui donne une valeur approchée de $\int_a^b f(x)\,\mathrm{d}x$ par la méthode des rectangles à gauche avec n subdivisions régulières.

Exercice 14. Soit s un entier compris entre 3 et 18. Implémenter une fonction `somme(s)` affichant le nombre de façons d'obtenir une somme égale à s en lançant trois dés.

Bonus : La fonction affiche toutes les façons possibles

Boucle non bornée

Exercice 15 (Recherche de seuil)**.** On considère la suite définie par

$$u_0 = 800 \quad \text{et} \quad u_{n+1} = \frac{3}{4}u_n + 330 \qquad \text{pour tout entier naturel } n$$

Implémenter une fonction `depasse(seuil)` qui renvoie la plus petite valeur de n à partir de laquelle u_n dépasse le seuil.

```
>>> depasse(1300)
```

Remarque : la suite tendant vers 1320, on pourra vérifier que l'on sait arrêter le script Python lorsqu'on saisit :

```
>>> depasse(1330)
```

Exercice 16. L'ordinateur choisit un nombre entier au hasard entre 1 et 1000, et le joueur doit le deviner le plus rapidement possible. L'affichage pourra être de la forme :

```
J'ai choisi le nombre secret .
Proposition numéro 1 : 500
```

```
C'est moins .
Proposition numéro 2 : 250
C'est plus .
```

Bonus : Le nombre de tentatives est limité à 7

Exercices supplémentaires

Exercice 17 (Arithmétique)**.**

- Un nombre premier est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même).
- Un diviseur strict d'un entier naturel n est un entier naturel diviseur de n mais distinct de n.
- Un nombre parfait est un entier naturel égal à la somme de tous ses diviseurs stricts. Par exemple, 28 est parfait car ses diviseurs stricts sont 1, 2, 4, 7 et 14, et $1+2+4+7+14=28$.
- Un nombre chanceux d'Euler est un entier naturel n tel que x^2+x+n est premier, pour tout entier x de l'intervalle $[0;n-2]$.
  Par exemple, 5 est un nombre chanceux d'Euler car :

  $$0^2+0+5=5, \quad 1^2+1+5=7, \quad 2^2+2+5=11, \quad 3^2+3+5=17$$

  et ces nombres sont tous premiers.

1. Écrire une fonction `SommeDiv(n)` qui retourne la somme des diviseurs stricts d'un entier naturel.

2. Écrire trois fonctions `EstParfait(n)`, `EstPremier(n)` et `EstChanceux(n)` , chacune retournant un booléen True ou False

3. Utiliser ces trois fonctions pour afficher tous les nombres parfaits et tous les nombres chanceux d'Euler compris entre 2 et 100.

Pour vérification, on obtient :

```
Nombres parfaits entre 2 et 100 : 6 28
Nombres chanceux entre 2 et 100 : 2 3 5 11 17 41
```

Exercice 18 (Simulation)**.** Une puce se déplace sur un axe gradué, en partant de la position 0. Elle fait 4 sauts, tantôt à droite, tantôt à gauche et ceci de façon équiprobable. L'ensemble des 4 sauts s'appelle une marche.

1. Implémenter une fonction `saut()` qui retourne la position de la puce après sa marche.

2. Implémenter une fonction `simulation(n)` qui retourne une liste avec les n positions finales de la puce après n simulations de marches.

3. Implémenter une fonction `histogramme(n)` qui affiche l'histogramme des effectifs des différentes positions finales après n simulations.

4. Implémenter une fonction `frequence(n)` qui affiche, pour chacune des positions finales possibles, sa fréquence pour la simulation de n marches effectuées.