10

Exercices python

Les programmes seront rendus sous le nom *(votre nom)-programme.py*.

► Exercice 10. 11 : Premiers nombres entiers : python ⇒ Corrigé

Écrivez un programme qui demande à l'utilisateur de saisir un nombre n puis affiche les n premiers nombres entiers.

► Exercice 10. 12 : Premiers entiers impairs : python ⇒ Corrigé

Écrivez un programme qui demande à l'utilisateur de saisir un nombre n puis affiche les n premiers entiers impairs.

► Exercice 10. 13 : Somme des premiers entiers pairs : python ⇒ Corrigé

Écrivez un programme qui demande à l'utilisateur de saisir un nombre n puis calcule la somme des n premiers entiers pairs en commençant par 2.

► Exercice 10. 14 : Intérêts : python ⇒ Corrigé

Écrivez un programme qui, à partir d'un montant à épargner et un taux d'intérêt annuel, calcule et affiche le montant augmenté des intérêts pour les n années à venir (vous afficherez le résultat avec 2 décimales).

► Exercice 10. 15 : Impôt sur les bénéfices : python ⇒ Corrigé

Écrivez un programme qui calcule l'impôt sur le bénéfice d'une société, le montant du bénéfice étant demandé à l'utilisateur, le montant de l'impôt étant de 20 % si le bénéfice est inférieur à 10000 €, de $2000 + 25$ % si le bénéfice est compris entre 10000 et 15000 € et de $3000 + 30$ % si le bénéfice est supérieur à 15000 €.

► Exercice 10. 16 : Fonction racine carrée ⇒ Corrigé

Écrivez un programme qui demande un flottant et qui calcule sa racine carrée avec 3 chiffres après la virgule s'il est positif ou nul.
Sinon affichez un message d'erreur.
On pourra utiliser :

```
from math import sqrt
```

▶ Exercice 10. 17 : Fonctions diverses ⇒ Corrigé

Écrivez une fonction `vols1` qui calcule directement le volume d'une sphère de rayon r fourni en argument.
Écrivez une fonction `cube` qui retourne le cube de son argument.
Écrivez une fonction `vols2` qui calcule le volume d'une sphère de rayon r fourni en argument et qui utilise la fonction cube.
Vous pourrez utiliser :

```
from math import pi
```

▶ Exercice 10. 18 : Suite 1 ⇒ Corrigé

Considérons la suite (u_n) définie par :

$$\begin{cases} u_0 &= 1 \\ \forall\, n \in \mathbb{N}, \quad u_{n+1} &= \dfrac{u_n\,\left(6-u_n^2\right)}{4} \end{cases}$$

1. Calculez u_1 sans ordinateur.
2. Écrivez une fonction $u1\,(n)$ permettant de calculer u_n en fonction de n à l'aide d'une boucle `for`.
3. Écrivez une fonction $u2\,(n)$ permettant de calculer u_n en fonction de n à l'aide d'une boucle `while`.
4. Calculez $u1\,(1)$, $u1\,(1000)$, $u2\,(1)$ et $u2\,(1000)$.
5. Comparez le résultat obtenu avec :

```
>>> import math
>>> math.sqrt(2)
```

▶ Exercice 10. 19 : Suite 2 ⇒ Corrigé

Considérons la suite (v_n) définie par :

$$\begin{cases} v_0 &= 1 \\ \forall\, n \in \mathbb{N}, \quad v_{n+1} &= -\dfrac{v_n}{(2\,n+1)\,(2\,n+2)} \end{cases}$$

1. (a) Calculez v_1 sans ordinateur.
   (b) Écrivez une fonction $v\,(n)$ permettant de calculer v_n en fonction de n.
2. Soit $s\,(n)=\sum_{i=0}^{i=n} v_i$.
   (a) Calculez s_1.
   (b) Écrivez une fonction $s(n)$ permettant de calculer s_n en fonction de n.
3. Calculez $s\,(1000)$.
4. Comparez le résultat obtenu avec :

```
>>> import math
>>> math.cos(1)
```

▶ Exercice 10. 20 : Suite 3 ⇒ Corrigé

1. Écrivez une fonction qui calcule la somme des carrés de 1 à n : $somme1\,(n)=\sum_{k=1}^{n} k^2$.
2. Écrivez une fonction $somme2\,(n)$ qui effectue le même calcul en utilisant une (ou des) liste(s) ainsi que la fonction `sum` (cf. page 307).

3. Écrivez une fonction qui calcule le produit des carrés de 1 à n : $produit1\left(n\right)=\prod_{k=1}^{n}k^{2}$.

4. Écrivez une fonction $produit2\left(n\right)$ qui effectue le même calcul en utilisant une (ou des) liste(s) même si cela présente assez peu d'intérêt...

▶ Exercice 10. 21 : Suite 4 ⇒ Corrigé

Considérons la suite (u_n) définie par :

$$\left\{\begin{array}{rl} u_0 & = 1 \\ \forall n\in\mathbb{N},\quad u_{n+1} & = \sqrt{1+u_n} \end{array}\right.$$

1. Calculez u_1 et u_2 sans ordinateur.
2. Écrivez une fonction $u1\left(n\right)$ permettant de construire la liste u_0, u_1, ...,u_n en fonction de n. Vous pourrez utiliser :

```
from math import sqrt
```

3. Soit la fonction $u2\left(n\right)$:

```
def u2 (n):
    return [sqrt(1+cpt) for cpt in range(0,n+1)]
```

Que fait cette fonction ? Cela répond-il à la question précdente ?

4. Modifiez la fonction $u1\left(n\right)$ afin qu'elle prenne en paramètre la valeur de u_0 de façon à pouvoir être modifiée par l'utilisateur.
5. Soit la fonction $u3\left(n\right)$:

```
def u3 (n,u_0):
    ma_liste = [u_0]
    for cpt in range(1,n+1):
        ma_liste.append(sqrt(1+ma_liste[cpt-1]))
    return ma_liste
```

Que fait cette fonction ?

▶ Exercice 10. 22 : Année bissextile : python ⇒ Corrigé

Écrivez un programme qui détermine si une année n est bissextile.
On rappelle que si n n'est pas divisible par 4, l'année n'est pas bissextile.
Si n est divisible par 4, l'année est bissextile sauf si n est divisible par 100 et pas par 400.

▶ Exercice 10. 23 : Devinette 1 : python ⇒ Corrigé

Écrivez un programme dans lequel l'utilisateur doit deviner un nombre pair compris entre 10 et 100 généré par l'ordinateur.
Vous pourrez utiliser :

```
import random
# nombre aléatoire compris entre 10 et 100
n = random.randint(10, 100)
```

▶ Exercice 10. 24 : Devinette 2 : python ⇒ Corrigé

Écrivez un programme dans lequel l'ordinateur devine un nombre pair entre 0 et 100 choisi par l'utilisateur (version dichotomique).

► Exercice 10. 25 : pgcd ⇒ Corrigé

Le pgcd de deux entiers a et b peut être trouvé grâce à l'algorithme suivant :

```
1:  VARIABLES
2:  a, b, r : int
3:  DEBUT_ALGORITHME
4:      LIRE a et b
5:      TANT_QUE b ≠ 0 FAIRE
6:          DEBUT_TANT_QUE
7:          r ← reste(a,b) # ou a%b
8:          a ← b
9:          b ← r
10:         FIN_TANT_QUE
11:     AFFICHER a
12: FIN_ALGORITHME
```

Algorithme 38 : Euclide

Écrivez une fonction qui calcule le pgcd de deux entiers a et b.

► Exercice 10. 26 : ppcm ⇒ Corrigé

Écrivez une fonction qui calcule le ppcm de deux entiers a et b : le ppcm de a et b est donné par le quotient du produit de a et b et du pgcd de a et b.

► Exercice 10. 27 : Méthode des trapèzes ⇒ Corrigé

Écrivez les algorithme et fonction en python correspondant à la méthode des trapèzes abordée page 88.

► Exercice 10. 28 : Table de multiplication ⇒ Corrigé

1. Écrivez un programme qui interroge l'utilisateur sur une multiplication de deux nombres compris et choisis aléatoirement entre 1 et 10.
2. Modifiez le programme précédent de façon à ce que l'ordinateur affiche "Bravo" ou "Dommage" en fonction de la réponse de l'utilisateur.
3. Créez ensuite une boucle `for` afin que l'ordinateur fasse une série de 10 multiplications.
4. Comptez ensuite les bonnes réponses de façon à afficher en fin de programme :
   (a) "Félicitations : tant de bonnes réponses, tant de mauvaises sur tant."
   (b) "C'est moyen : tant de bonnes réponses, tant de mauvaises sur tant."
   (c) "Retournez en CE2 : tant de bonnes réponses, tant de mauvaises sur tant."

   Les seuils pourront par exemple être mis à 0,8 et 0,5.
5. Modifiez le programme précédent pour qu'il demande à l'utilisateur s'il veut refaire une autre série de multiplications. Si oui, le programme devra revenir au début par l'intermédiaire d'une boucle `while`. Si non, le programme s'arrêtera.

   ⚠ À chaque série, les compteurs intermédiaires de réponses devront être remis à 0.
6. Complétez le programme pour qu'il affiche en sortant le nombre total de bonnes réponses sur le total des questions (forcément un multiple de 10 pour ce dernier nombre).
7. Utilisez un chronomètre de façon à ce que la réponse soit comptée bonne si elle est donnée dans un laps de temps limité, 5 secondes, par exemple. Si la réponse est trop tardive, l'ordinateur affichera le temps de réponse.
   Complétez ensuite le programme de façon à ce que le nombre de réponses tardives soit affiché et qu'il intervienne dans l'appréciation.
8. Modifiez le programme précédent de façon à ce qu'il gère les erreurs liées à une faute de frappe (chaîne à la place d'un nombre, par exemple) avec les instructions `try` et `except` (voir le poly page 76).

— Remarques —

Pour générer un entier compris entre 1 et 10, on peut utiliser la fonction `randint` de la bibliothèque *random*. Quelques fonctions du module *random* sont évoquées page 230.
Pour déclencher un chronomètre, on peut utiliser la fonction `time` de la bibliothèque `time`. Pour l'arrêter, c'est la même fonction.
On peut également utiliser la fonction `clock` de la bibliothèque *time*. Le module *time* est détaillé dans le chapitre page 236.

L'affichage sera par exemple le suivant :

```
Appuyez sur Entrée pour démarrer
test 1 : 1 x 5 = 2
Dommage
test 2 : 2 x 7 = 14
bon mais trop tard : 5.61 secondes
test 3 : 4 x 8 = 32
Bravo !!!
test 4 : 3 x 6 = 18
Bravo !!!
test 5 : 5 x 8 = 40
Bravo !!!
test 6 : 5 x 6 = 30
Bravo !!!
test 7 : 2 x 9 = 18
Bravo !!!
test 8 : 10 x 9 = 90
Bravo !!!
test 9 : 5 x 6 = 30
Bravo !!!
test 10 : 4 x 10 = 40
Bravo !!!

Félicitations ! 8 bonne(s) réponse(s), 1 mauvaise(s) et 1 trop lente(s) sur 10

Une autre série ? (O/N) O

Appuyez sur Entrée pour démarrer
test 1 : 9 x 7 = 63
Bravo !!!
test 2 : 8 x 5 = 40
Bravo !!!
test 3 : 2 x 8 = 15
Dommage
test 4 : 4 x 7 = 28.5
Dommage
test 5 : 6 x 5 = hhj
could not convert string to float: 'hhj'
Dommage
test 6 : 2 x 7 = 13.75
Dommage
test 7 : 10 x 5 = 50
Bravo !!!
test 8 : 4 x 9 = 36
Bravo !!!
test 9 : 5 x 6 = 30
Bravo !!!
test 10 : 4 x 5 = 20
Bravo !!!

C'est moyen : 6 bonne(s) réponse(s), 4 mauvaise(s) et 0 trop lente(s) sur 10

Une autre série ? (O/N) N

14 de bonnes réponses sur 20, à bientôt
```