

Exercices de programmation en Python

Exercice 1

Pour calculer en Python le reste de la division d'un entier a par un entier b , on utilise l'opérateur `%` :

```
>>> 15 % 4
3
>>> 21 % 5
1
```

1. Décrire un algorithme qui donne le reste de la division euclidienne d'un entier positif a par un entier positif b et qui utilise exclusivement l'opérateur de soustraction `-`.
2. Ecrire une fonction en Python qui prend pour paramètres deux entiers positifs a et b et qui renvoie le reste de la division de a par b et qui s'appuie sur l'algorithme précédent (on s'interdit donc d'utiliser `%`)

```
def reste(a , b :int)->int:
    ..compléter sur plusieurs lignes...
```

3. Quelles modifications apporter au corps de la fonction pour qu'elle renvoie le quotient entier de a par b ? (On s'interdira l'utilisation de l'opérateur `//` de Python!)

Exercice 2

On part d'un entier n auquel on fait subir une transformation :

si n est pair, on le divise par deux ;

si n est impair, on le multiplie par 3, et ajoute 1.

Puis, on recommence sur le résultat jusqu'à obtenir 1. Par exemple, en partant de $n=10$, on obtient la suite de nombres :

10 5 16 8 4 2 1 .

Conjecture : quel que soit l'entier n , on finit par retomber sur 1.

1. Appliquer l'algorithme précédent pour $n = 12$ puis pour $n = 7$
2. Ecrire une fonction `suivant` qui prend pour argument un entier naturel n et qui renvoie l'entier qui suit lorsqu'on lui applique les opérations 1 ou 2.
Par exemple :

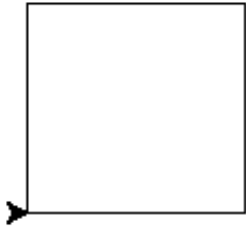
```
>>> suivant(10)
5
>>> suivant(5)
16
```

3. Ecrire un programme qui utilise la fonction `suivant` et qui affiche la suite des nombres de n à 1 lorsqu'on applique l'algorithme .

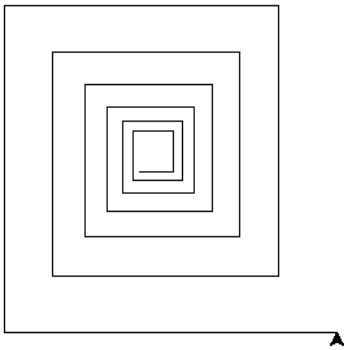
Exercice 3

(utiliser le module turtle)

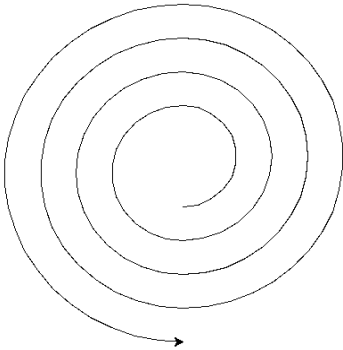
1. Écrire les instructions pour afficher un carré de côté 100.



2. Écrire les instructions pour afficher une spirale 'carrée' comme ci-dessous.



3. Écrire les instructions pour afficher une spirale 'circulaire' comme ci-dessous.



Exercice 4

Un mot est dit *palindrome* s'il se lit de manière identique à l'endroit et à l'envers.

Par exemple le mot "kayak" est un palindrome.

Un mot formé d'une seule lettre est un palindrome.

1. Décrire un algorithme qui teste si un mot est un palindrome.
2. Écrire une fonction `est_palindrome` qui teste si un mot donné en paramètre est un palindrome et qui renvoie un booléen.

Exercice 5

1. On définit la fonction `bonjour` par le code suivant :

```
1 def bonjour(nom)
2     return 'Bonjour ' + nom + ' !'
```

Donner le résultat de l'exécution de `bonjour('Alan')`.

2. On exécute le programme suivant :

```
1 une_chaine='Bonjour'
2 x = (une_chaine[2] == une_chaine[3])
3 y = (une_chaine[4] == une_chaine[1])
```

Donner le type et les valeurs des variables `x` et `y`.

3. Écrire une fonction `occurrences_lettre(une_chaine,une_lettre)` prenant en paramètres une chaîne `une_chaine` et une lettre `une_lettre` et renvoyant le nombre d'occurrences de `une_lettre` dans `une_chaine`.

Exercice 6

En partant d'une ligne de signes '+' et '-' consécutifs ,on fabrique un tableau triangulaire de signes "+" et "-" en écrivant sous chaque couple de signes celui qui est le résultat d'une multiplication respectant la règle des signes.

Ci-dessous le tableau à 7 lignes

```

+ - + + - + +
- - + - - +
  + - - + -
    - + - -
      - - +
        + -
          -
```

Le but de l'exercice est d'afficher le triangle des signes pour un nombre donné de signes au départ.Pour cela , compléter les fonctions ci-dessous :

1.

```

1 def multiplier_signes(a,b):
2     """
3     a et b sont deux chaînes
4     de l'ensemble { '+' , '-' }
5
6     la fonction renvoie la chaîne '+' ou '-'
7     résultat du produit des signes a et b
8     """
9     #.....
10    #Compléter le code
11    #.....
12
13    assert multiplier_signes('-', '-') == '+'
14    assert multiplier_signes('-', '+') == '-'

```

Construction du triangle des signes :

Pour obtenir le triangle de la figure ci-dessus, on va appliquer la fonction `multiplier_signes` plusieurs fois et de manière consécutive.

L'étape de départ sera d'afficher une ligne de signes '+' et '-' de manière aléatoire.

on utilise la méthode `choices()` du module `random` pour fabriquer de manière aléatoire une liste d'éléments égaux aux signes '+' ou '-'.

comme dans l'exemple :

```

>>> from random import choices
>>> print (choices(('+', '-'), k=3))
['+', '-', '+']

```

2. Compléter alors le code de la fonction `triangle_signe`

```

from random import choices
def triangle_signe(n):
    """
    n est le nombre de lignes
    """

    #la liste des n signes '+' et '-'
    a=choices(('+', '-'), k= n)

    .....#A compléter sur plusieurs lignes.....
    .....

```

Exercice 6

Le but de l'exercice est de calculer le nombre de jours écoulés depuis le premier janvier 2020.

Pour cela on écrit trois fonctions :

- `bissextile(a)` qui teste si une année `a` est bissextile ou non.
- `nb_jours_ecoules(j,m,a)` qui renvoie le nombre de jours écoulés depuis le 1er janvier de l'année en cours .

- `nb_jours_depuis_2020(j,m,a)` renvoie le nombre de jours écoulés depuis le 1er janvier 2020 à la date `(j,m,a)` (`a` supérieur ou égal à 2020).

Complétez alors chacune des fonctions suivantes.

```

1.
1
2 def bissextile(a):
3     """
4     Une année est bissextile si le nombre a vérifie l'une
5     des deux conditions suivantes:
6
7     - a est divisible par 4 et n'est pas divisible par 100
8     ou
9     - a est divisible par 400
10
11     entree: a -int le numéro de l'année
12     sortie : -bool un booléen égal à vrai si l'années est
13     bissextile , égal à False sinon
14     """
15
16     #Ecrire le code complet ci-après (le code peut dépasser les deux
17     lignes!)
18     .....
19     .....

```

2. Pour les fonctions suivantes, Compléter les pointillés par l'instruction manquante.

```

1 def nb_jours_ecoules(j,m,a):
2     """
3     entree:
4     j : -int le numéro du jour dans le mois j>=1 et j<=31
5     m : -int le numéro du mois dans l'année , m>=1
6     a:  -int le numéro de l'année
7
8     sortie : le nombre de jours écoulées depuis le premier
9     janvier de l'année en cours (a)
10    """
11
12    # la liste du nombre de jours par mois de l'année non bissextile
13    #par ordre croissant du mois
14
15    if bissextile(a)==True:
16        Mois=[31,29,31,30,31,30,31,31,30,31,30,31]
17    else:
18        .....
19        #calcul du nombre de jours
20    nb=.....
21    for i in range(.....):
22        nb = .....
23    return nb + j

```

```
3.
1 def nb_jours_depuis_2020(j,m,a):
2     """
3     Pour j et m voir la spécification de la fonction
4     précédente.
5     a : -int numéro de l'année en cours 2020<= a <=2026
6     sortie: nombre de jours écoulés depuis le premier
7     janvier 2020.
8
9     """
10
11
12     nb=0
13     .....
14     .lignes à compléter
15     .....
16     return nb
```

```
assert nb_jours_ecoules(10,2,2026)==2233
assert nb_jours_ecoules(16,9,2026)==2451
```