Python-ECG1-04-boucles-for-cor

September 29, 2021

1 TP4 - Boucles for

Nous allons voir comment écrire un programme python comportant une instruction répétitive **for**.

Algorithme en français

- **Pour** k variant de n à m,
  - Réaliser les instructions I_k
- **Fin du Pour**

Syntaxe python

```
for k in range(n,m+1) :
    instructions Ik
```

et plus généralement :

```
for element in sequence :
    bloc_instructions
```

Remarques : - La variable de boucle (ici k) peut avoir un nom différemment. On veillera à rester cohérent. - **range(n,m+1)** est la liste des entiers $[n, n+1, \ldots, m]$. - **range(n)** est la liste des entiers $[0, 1, \ldots, n-1]$.

Attention, les élèments d'un range s'arrêtent un cran avant la fin indiquée.

Exemple : Afficher 1^2, 2^2, 3^2, … , 100^2.

[1]:
```
for i in range(1,101) :
    print(i**2)
```

```
1
4
9
16
25
36
49
64
81
100
```

121
144
169
196
225
256
289
324
361
400
441
484
529
576
625
676
729
784
841
900
961
1024
1089
1156
1225
1296
1369
1444
1521
1600
1681
1764
1849
1936
2025
2116
2209
2304
2401
2500
2601
2704
2809
2916
3025
3136
3249
3364

```
3481
3600
3721
3844
3969
4096
4225
4356
4489
4624
4761
4900
5041
5184
5329
5476
5625
5776
5929
6084
6241
6400
6561
6724
6889
7056
7225
7396
7569
7744
7921
8100
8281
8464
8649
8836
9025
9216
9409
9604
9801
10000
```

1.1 Exercice 1

Pour tout entier naturel $k \leqslant 20$, calculer $y = 3k + 2$ puis afficher la valeur de y.

[2]:
```
for k in range(0,21):
    y = 3*k+2
    print(y)
```

```
2
5
8
11
14
17
20
23
26
29
32
35
38
41
44
47
50
53
56
59
62
```

1.2 Exercice 2 - Calculer le terme de rang n d'une suite récurrente

Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par

$$\begin{cases} u_0 = 0 \\ \forall n \in \mathbb{N}, \quad u_{n+1} = \sqrt{u_n + 4n} \end{cases}$$

1) Sur feuille

2) Écrire un programme python calculant et affichant u_n avec une boucle for. Ce programme et le tableau précédent doivent être cohérents.

[3]:
```
import numpy as np
n = 10  # tester avec différentes valeurs de n
u = 0
for k in range(n) :
    u = np.sqrt(u+4*k)

print(u) # non indenté, fait une fois, après la boucle for
```

```
6.492021620534664
```

3) Que faut-il changer pour que le programme affiche tous les termes de la suite, de u_0 à u_n ?

(On pourra faire du copier-coller à partir du programme précédent).

[4]:
```
import numpy as np
n = 10  # tester avec différentes valeurs de n
u = 0
print(u) # affichage de u0
for k in range(n) :
    u = np.sqrt(u+4*k)
    print(u) # indenté, fait à chaque tour de boucle
```

```
0
0.0
2.0
3.1622776601683795
3.893876944661757
4.460255255550041
4.945731013262857
5.380123698695305
5.7775534353820825
6.14634472148952
6.492021620534664
```

1.3 Exercice 3

Soit $(v_n)_{n\in\mathbb{N}}$ la suite définie par

$$\begin{cases} v_0 = 5 \\ \forall n \in \mathbb{N}, \quad v_{n+1} = v_n + \dfrac{2}{v_n} \end{cases}$$

Écrire une **fonction** python **Suite_v(n)** qui, étant donné un entier naturel n, renvoie la valeur de v_n.

[5]:
```
def Suite_v(n):
    v = 5
    for k in range(n):
        v = v + 2/v
    return v  # on indenté, fait une fois, à la fin
```

[6]:
```
# On n'oublie pas de tester la fonction.
print(Suite_v(0))
print(Suite_v(1))
print(Suite_v(2))
print(Suite_v(10))
```

```
5
5.4
5.770370370370371
8.123839381592699
```

1.4 Exercice 4 - Les compteurs

1) Comment simuler le lancer d'une pièce qui tombe sur pile avec une probabilité de $\frac{1}{3}$ en utilisant **rd.randint(1,4)** ?

Le programme doit renvoyer 'P' (chaîne de caractères) avec probabilité $\frac{1}{3}$ et 'F' avec probabilité $\frac{2}{3}$.

[7]:
```
import numpy.random as rd

def piece():  # cette fonction n'a pas de paramètre d'entrée
    r = rd.randint(1,4)
    if r == 1 :
        return 'P'
    else :
        return 'F'
```

2) Tester plusieurs fois en appelant piece() dans la console, jusqu'à obtenir au moins une fois chaque côté. Combien de lancers ont été nécessaires ?

[8]:
```
piece()
```

[8]:
```
'F'
```

3) Simuler 900 lancers de cette pièce et déterminer le nombre de 'P' obtenus. Ce nombre doit être proche de 300.

[9]:
```
compteur = 0    #initialisation du compteur

for k in range(900):
    if piece() == 'P' :
        compteur = compteur + 1  # si on tombe sur 'P', on ajoute 1 au compteur
    # pas de else ici, rien à faire sinon.

print(compteur) # non indenté, fait à la fin.
```

```
326
```

1.5 Exercice 5 - Suite récurrente d'ordre 2

Soit $(w_n)_{n\in\mathbb{N}}$ la suite définie par

$$\begin{cases} w_0 = 1,\ w_1 = 1 \\ \forall n \in \mathbb{N},\quad w_{n+2} = w_n + w_{n+1} \end{cases}$$

Écrire une **fonction** python **Suite_w(n)** qui, étant donné un entier naturel n, renvoie la valeur de w_n.

[10]:
```
def Suite_w(n):
    w = 0 # w0
```

```
    w1 = 1 # w1
    for k in range(n):
        w2 = w + w1  # calcul de w(k+2)
        w = w1
        w1 = w2
    return w
```

[11]:
```
# tester ici la fonction
print(Suite_w(0))
print(Suite_w(1))
print(Suite_w(2))
print(Suite_w(3))
print(Suite_w(4))
print(Suite_w(5))
```

```
0
1
1
2
3
5
```