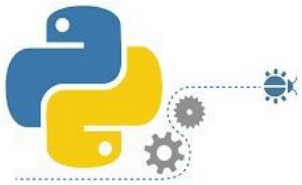
Algorithmique et Programmation en Python

Table des matières



L'objectif de la deuxième séquence 2 est d'introduire la structure itérative utilisant la boucle « pour » d'abord et la boucle « tant que » ensuite.

Algorithmique et Programmation en Python

Fiche élève n°1 : Boucles « pour» _ Moyennes

Calculs de moyennes :

Voici une série de 10 notes : 12 ; 11 ; 8 ; 15 ; 12 ; 14 ; 17 ; 2 ; 6 ; 13.
On veut écrire un algorithme qui calcule la moyenne de ces dix notes et l'affiche.

Une nouvelle structure algorithmique : « Pour i allant de 1 à 10 : »

Vous allez avoir besoin de saisir les 10 notes successivement. Pour cela, il existe une structure itérative qui permet de recommencer une ou plusieurs actions en boucle.

Voici le début de l'algorithme demandé : S ← 0

Pour i allant de 1 à 10 :
N ← Saisir une note
S ← S + N
FinPour

1°) Quel est le rôle de la variable S ? Quel est son type ?
Compléter cet algorithme pour qu'il calcule la moyenne des 10 notes. Ce calcul sera stocké dans une variable m avant d'être affiché. Quel est le type de la variable m ?
Programmer cet algorithme en langage Python.

En Python, l'instruction « pour i de 1 à 10 » se traduit par :

```
for i in range (1 , 11):        # terminer par :
    instruction1                # indenter le bloc d'instructions à
    répéter, c'est ce
                                 qui délimitera le contenu de la boucle.
    instruction2
    ……..
# la boucle se termine lorsqu'on revient écrire au niveau du « for »
```

*L'instruction **range(n)** renvoie les entiers de 0 à n-1.*
*L'instruction **for i in range(n) :** se traduit donc par « pour i allant de 0 à n-1 »*

2°) Modifier le programme précédent pour qu'il calcule la moyenne de x notes.
On saisit d'abord le nombre x de notes à traiter.

3°) Voici une série statistique :

x_i	12	25	30	44	57	65
n_i	2	5	11	18	15	6

Ecrire un algorithme qui calcule et affiche la moyenne de cette série.

4°) Traduire cet algorithme en langage Python.

Exercices de programmation avec une boucle « pour » :

Exercice 1

Ecrire un programme qui réalise :

- l'utilisateur entre deux nombres entiers a et b avec a>b.
- le programme renvoie le compte à rebours de a jusqu'à b.

Exercice 2

N étant un entier positif, on appelle **factorielle de N** le nombre $N\times(N-1)$ .... $3\times2\times1$
Exemple : 5 ! = $5\times4\times3\times2\times1$ = 120.

Ecrire un programme qui réalise :

- l'utilisateur entre un entier N.
- le programme renvoie factorielle de N.

Exercice 3

Ecrire un programme qui réalise :

- l'utilisateur entre deux entiers a et b
- le programme renvoie tous les couples d'entiers (x,y) où x est entre 0 et a et y entre 0 et b.

Exercice 4

On considère la fonction suivante : $f(x) = 2(x-1,5)^2-4$ sur I=[-10 ; 10].
Ecrire un programme qui dresse la table de valeurs de cette fonction de -10 à 10 avec un pas de 1.
Ce programme sera écrit avec une fonction qui réalise le calcul de l'image d'un nombre x donné en argument.

Algorithmique et Programmation en Python

Fiche élève n° 2 : Boucles « pour » _ Simulation

Simulation :

Le module random permet de générer des nombres aléatoires.
random() renvoie un flottant choisi dans l'intervalle [0 ; 1]
randint(a,b) renvoie un entier choisi aléatoirement dans [a ; b]

Quel jeu le programme suivant simule-t-il ?

```
from random import randint

a = randint (1,6)
b = randint (1,6)
c = randint (1,6)

if a > b :
    m = a
    a = b
    b = m
if b > c:
    m = b
    b = c
    c = m
if a > b :
    m = a
    a = b
    b = m

if a == 1 and b ==2 and c == 4:
    print "gagné!"
else :
    print "perdu"
```

1°) Ecrire un programme qui simule 1000 fois le jeu ci-dessus et qui calcule la fréquence de parties gagnées.

2°) Déterminer la probabilité d'obtenir 4 - 2 - 1 en lançant trois dés bien équilibrés.

Algorithmique et Programmation en Python

Fiche élève n° 3 : Boucles « tant que » : le lièvre et la tortue

1°) Que réalise l'algorithme ci-dessous ? :

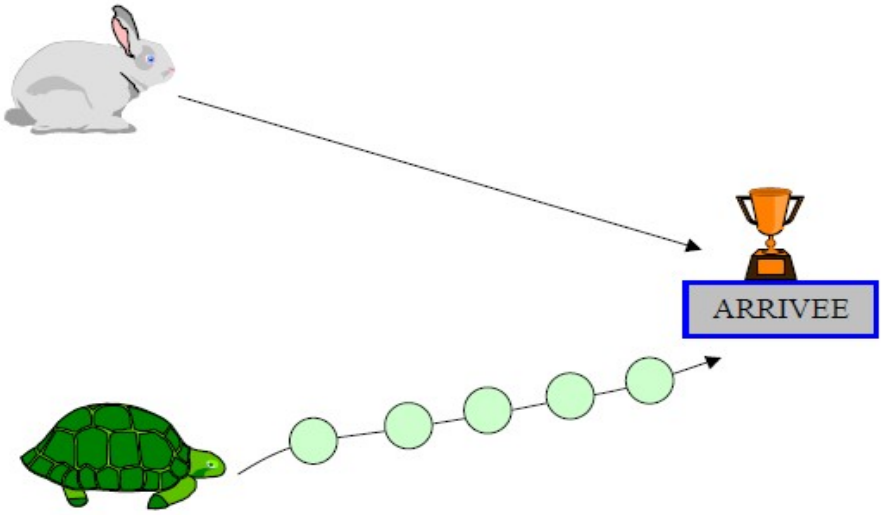
```
compteur ← 0
        Dé ← entier aléatoire
entre 1 et 6
        tant que Dé ≠ 6 faire
                compteur ←
compteur + 1
                Dé ← entier
aléatoire entre 1 et 6
        fin tant que
        Afficher compteur
```

Cet algorithme utilise une boucle conditionnelle « *tant que* » qui permet de répéter un traitement tant que la condition reste vraie. Dès que la condition devient fausse, le programme poursuit avec l'instruction qui suit le « *fin tant que* ».

En langage Python, on écrit :

```
        while (condition) :
                instruction 1
                instruction 2
                …....
                instruction n
        #Le retour au niveau du « while » marque la fin du bloc d'instructions à répéter.
```

2°) Le jeu du lièvre et de la tortue



Régle du jeu : A chaque tour, on lance un dé. Si le 6 sort , alors le lièvre gagne la partie, sinon la tortue avance d'une case. La tortue a 5 cases à franchir avant d'atteindre l'arrivée. Dans ce cas, c'est la tortue qui gagne.

L'objectif est de déterminer si le jeu est à l'avantage du lièvre ou de la tortue.

Question 1 : compléter l'algorithme ci-dessous :

```
position_tortue ← 0
    position_lievre ← 0
    tant que position_tortue < ..... et ..... faire
        Dé = entier aléatoire entre 1 et 6
        si  Dé = ..... alors
            position_lievre ← .....
        sinon
            position_tortue ← .....

        si position_lievre = 6 alors
            Afficher .....
        sinon
            Afficher .....
```

Question 2 : traduire en langage Python cet algorithme et le tester.

Question 3 : modifier le programme précédent pour simuler 1000 parties.

Question 4 : compter le nombre de parties gagnées par le lièvre.

Question 5 : exécuter ce programme plusieurs fois et compléter le tableau ci-dessous :

Echantillon numéro	Nombres de parties gagnées par le lièvre	Fréquence de gain du lièvre
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Question 6 :

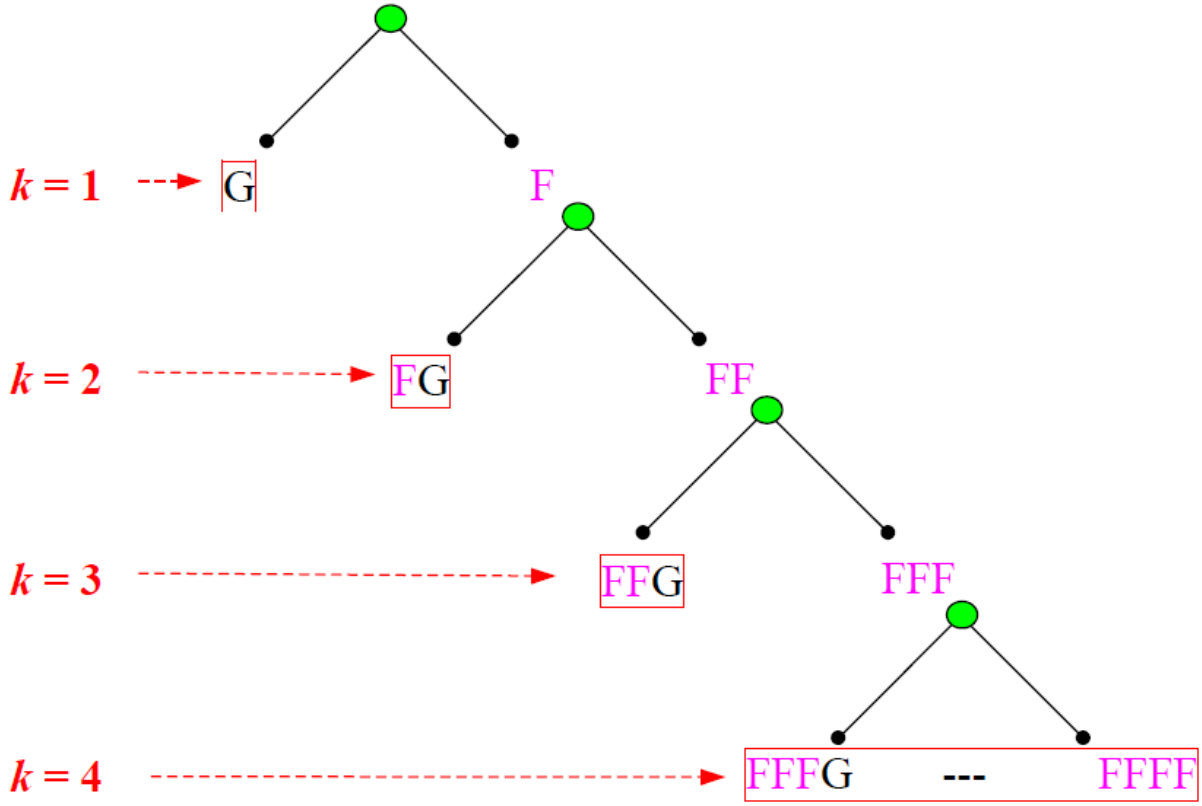
La probabilité que le lièvre gagne est $1-\left(\frac{5}{6}\right)^6$.

Déterminer un intervalle de fluctuation au seuil de 95% pour des échantillons de taille 1000.
Les fréquences observées ci-dessus sont-elles dans cet intervalle ?

Fiche élève n°4 : Boucles « *tant que* » : planification des naissances

Le gouverneur d'une planète lointaine décide d'imposer une politique de planification des naissances basée sur la règle suivante.
Les naissances au sein d'une famille s'arrêtent :

- soit à la naissance du premier garçon
- soit lorsque la famille comporte quatre enfants.



On suppose que les probabilités de donner naissance à un garçon ou à une fille sont égales.
On se propose de simuler cette politique de planification des naissances à l'aide d'un programme en langage Python et d'observer les proportions de garçons et de filles.

1°) On décide de générer un entier aléatoire entre 0 ou 1 pour simuler une naissance (0 pour une fille et 1 pour un garçon) . Compléter l'algorithme ci-dessous :

```
Variables : garçon, fille, nb_enfants, naissance de type entiers
garçon ← 0
fille ← 0
nb_enfants ← 0
tant que ..... et .....
        naissance ← entier aléatoire entre 0 et 1
        si .....
                .....
        sinon
                .....
        nb_enfants ← ..... ....

        Afficher fille, garçon
```

2°) Créer une fonction en langage Python traduisant cet algorithme qui correspond à une famille.

3°) Ecrire un programme en langage Python qui simule un échantillon de 1000 familles.

Voici les résultats pour 3 échantillons :

```
>>>
0.511578947368 0.488421052632
>>> ============================
>>>
0.503164556962 0.496835443038
>>> ============================
>>>
0.491133799033 0.508866200967
```

Il semblerait que la répartition des sexes reste équilibrée.