PYTHON découverte du langage

Les fonctions

Une fonction peut retourner une valeur

return

```
#simulation du lancer d'un dé : obtenir 6 pour gagner
#pour importer la bilbiothèque random qui contient
#la fonction randint()
#elle renvoie un nombre entier aléatoire compris
#entre deux valeurs
from random import*
def tirage ():
    dé=randint(1,6)
    if dé==6:
        a=("gagné !")
    else:
        a=("perdu !")
    return(dé,a)
print(tirage())
```

```
# pour définir la fonction f(x)=2x-5
def f(x):
    return 2*x-5
#pour executer la fonction
print(f(1))
#pour executer la fonction avec plusieurs valeurs
print(f(-3), f(5), f(10.5))
```

Pour créer des fonctions

def

Pour appliquer (ou appelée) une fonction, il suffit d'utiliser son nom dans la suite du programme.

Les boucles

Boucle bornée
Quand on connait le nombre de répétition

for

```
s=0
#avec la fonction range(50) on parcourt les entiers
#de 0 jusqu'à 49
for i in range(50):
    s = s + i
print("s=", s)
```

Boucle non bornée
Si on ne connait pas à l'avance le nombre de répétitions

while

```
n=0
while 2**n <40000:
    n = n + 1
print(n)
```

Instruction conditionnelle

On réalise un test pour vérifier si la condition voulue est vraie (True) ou fausse (False).

a == b : a est-il égal à b ?
a > b : a est-il supérieur à b ?
a >= b : a est-il supérieur ou égale à b ?
a != b : est-il différent de b ?

```
a=2
#test d'égalité
print("est-il vrai que a=4 ?", a==4)
print("est-il vrai que a=2 ?", a==2)
```

```
>>> # script executed
est-il vrai que a=4 ? False
est-il vrai que a=2 ? True
>>>
```

On utilise des instructions conditionnelles lorsque certaines instructions ne doivent être exécutées que lorsqu'une condition est réalisée.

if ("si")
if...else ("si...sinon")
if...elif...else ("si...sinon... si...sinon")

```
a=65
if a<0:
    print("a contient une valeur négative")
elif a<=10:
    print("a contient une valeur entre 0 et 10")
elif a<=100:
    print("a contient une valeur entre 10 et 100")
else:
    print("a contient une valeur supérieure à 100")
```

Remarque : le symbole : marque le début d'un bloc. C'est l'indentation (=décalage) qui délimite le bloc d'instructions (par exemple avec la syntaxe def, les boucles for et while ainsi que les tests if.)

Le langage Python : découverte du langage (2/2)

1. Instruction conditionnelle

On utilise des instructions conditionnelles lorsque certaines instructions ne doivent être exécutées que lorsqu'une condition est réalisée.
On réalise alors un **test** pour vérifier si la condition voulue est **vraie** (True) ou **fausse** (False).

Quelques opérateurs de test en python :
a == b : a est-il égal à b ?
a > b : a est-il supérieur à b ?
a >= b : a est-il supérieur ou égal à b ?
a != b : a est-il différent de b ?

```
1  # affectation
2  a=2
3  # test d'égalité avec ==
4  print("Est-il vrai que a=4 ?", a==3)
5  print("Est-il vrai que a=2 ?", a==2)
```

```
>>> # script executed
Est-il vrai que a=4 ? False
Est-il vrai que a=2 ? True
>>>
```

Python nous fournit les structures conditionnelles suivantes :
La condition if (“si”) ;
La condition if…else (“si…sinon”) ;
La condition if…elif…else (“si…sinon si… sinon”)

```
>>> a=35
if a<0:
    print ("a contient une valeur négative")
elif a<10:
    print ("a contient une valeur entre 0 et 10 exclu")
elif a<100:
    print ("a contient une valeur entre 10 et 100 exclu")
else:
    print ("a contient une valeur supérieure à 100")

a contient une valeur entre 10 et 100 exclu
>>>
```

Remarque : Le symbole : marque le début d'un bloc. C'est l'indentation (=décalage) qui délimite le bloc d'instructions.

2. Les fonctions

Une fonction en informatique est un peu comme une fonction en maths. Elle prend un ensemble d'argument et renvoie quelque chose.
En Python, on peut créer des fonctions à l'aide de l'instruction def.
Une fonction peut renvoyer une valeur : on utilise pour cela l'instruction return à la fin de la définition de la fonction.
Pour appliquer (ou **appelée**) une fonction, il suffit d'utiliser son nom dans la suite du programme.

```
# on définit la fonction f(x)=2x-5
def f(x):
    return 2*x-5

# on exécute la fonction
print(f(1))

# ou alors on exécute la fonction
print(f(-3), f(5), f(10.5))
```

```
>>> # script executed
-3
-11 5 16.0
```

Une fonction intéressante est la fonction randint(a, b). Cette fonction renvoie un nombre entier aléatoire compris au sens large entre a et b. Pour pouvoir l'utiliser, il faut **importer** la bibliothèque random qui la contient.

```
# Simulation du lancer d'un dé : on gagne si on obtient 6
from random import*
def tirage ():
    dé = randint(1,6)
    if dé == 6:
        a="gagné !"
    else:
        a="Perdu !"
    return(dé,a)
print(tirage())

```

```
>>> # script executed
(6, 'gagné !')
>>> tirage()
(4, 'Perdu !')
>>>
```

3. Les boucles

Une boucle permet d'exécuter un même bloc d'instruction plusieurs fois

Il existe deux types de boucle en programmation :
- boucle bornée : For (pour) : lorsque l'on connait le nombre de répétition à effectuer.

La fonction range() permet de générer une séquence de nombres est souvent utilisé dans les boules for simples.

```
s = 0
for i in range(50):
    s = s + i
print("S =",s)
```

```
>>> # script executed
S = 1225
>>>
```

- boucle non bornée : While (tant que) : Si on ne connait pas à l'avance le nombre de répétitions

```
n = 0
while 2**n < 40000 :
    n = n + 1
print(n)
```

```
>>> # script executed
16
>>>
```

Exercices

Exercice 1

La fonction randint() de Python est une fonction qui renvoie un nombre entier aléatoire compris entre deux valeurs. Pour pouvoir l'utiliser, il faut importer la bibliothèque random qui la contient. Le script suivant présente une fonction tirage qui simule le lancer de deux dés et renvoie le numéro obtenu sur chaque dé et leur somme.

Compléter le script :

```
from random import *
def tirage():
    a = ...(1, 6)
    b = randint(...)
    return a, b, ...
print(...)
```

Exercice 2

Écrire un programme qui affiche 10 fois la phrase « encore une fois ! » et une fois la phrase « c'est terminé ! »

Exercice 3

Une somme de 3000 € est placée à la banque au taux de 8 %. Je souhaite savoir à l'aide d'un programme Python au bout de combien d'année je disposerai d'une somme supérieure à 4000€. Indications : Le montant à la fin de la première année peut être calculé à l'aide du coefficient multiplicateur : 3000×(1+8/100) = 3000x1,08 = 3240 €. De la même manière, Le montant à la fin de la deuxième année est égal à 3240×1.08 = 3499,2 €.

Exercice 4

On cherche à automatiser le calcul d'une remise lors de soldes à l'aide d'un script Python. Définir une fonction prix_solde qui tient compte des conditions suivantes : si le prix de l'article dépasse 150 euros, alors une remise de 30% est appliquée, sinon une remise de 20% est appliquée.

Correction

Exercice 1

```
from random import *
def tirage():
    a = randint(1, 6)
    b = randint(1, 6)
    return a, b, a+b
print(tirage())

```

Exercice 2

```
for i in range(10):
    print("encore une fois!")
print("C'est terminé !")
```

Exercice 3

```
somme = 3000
annee = 0
while somme < 5000:
    somme = somme*1.08
    annee = annee + 1
print(annee)
```

```
>>> # script executed
7
>>>
```

Exercice 4

```
def prix_solde(prix):
    if prix >150:
        remise = prix*30/100
    else :
        remise = prix*20/100
    return(prix-remise)
```

```
>>> # script executed
>>> prix_solde (300)
210.0
>>> prix_solde (90)
72.0
>>>
```