

Mise à Niveau

Sébastien Gagné, Université d'Orléans

Cours 05

Introduction aux fonctions

Plan du cours 5

- Intérêt des fonctions
- Paramètres, arguments
- Type de retour
- Exemples

Intérêt des fonctions

Python propose, comme tous les langages modernes, l'écriture de **fonctions**.

Elles rappellent le concept mathématique des fonctions, mais en élargissent le contexte d'utilisation.

Exemple mathématique : modéliser par une fonction comment on peut calculer la surface d'un rectangle de périmètre constant égal à 100.

Intérêt des fonctions



→ La contrainte du périmètre impose la relation $2x + 2y = 100$ et donc $x + y = 50$.

→ y s'exprime en fonction de x : $y = 50 - x$.

→ L'aire est $x * y$, donc $aire = x * (50 - x)$.

Intérêt des fonctions

Cette situation met en évidence :

- Une variable, note ici x , qui représente la longueur du côté horizontal du rectangle.
- Deux fonctions, qui permettent de calculer, à partir de x , la longueur y et l'aire du rectangle.

On utilisera alors les écritures

- $y(x) = 50 - x$
- $aire(x) = x * (50 - x)$

Traduction en python

```
1 def y(x) :  
2     return 50 - x  
3  
4 def aire(x) :  
5     return x * (50 - x)  
6  
7 def afficher_resultat(x) :  
8     print("pour x =", x)  
9     print("- y vaut", y(x))  
10    print("- l'aire vaut", aire(x))  
11  
12 # programme principal  
13 afficher_resultat(7)  
14 afficher_resultat(19)  
15 c = int(input("entrez votre valeur de x : "))  
16 afficher_resultat(c)
```

- **def** → définition de la fonction : **Obligatoire**.
- **return** → renvoie un résultat à l'endroit de l'appel de la fonction : **Facultatif**. Toute instruction après **return** est ignorée !

Traduction en python

```
1 def y(x) :  
2     return 50 - x  
3  
4 def aire(x) :  
5     return x * (50 - x)  
6  
7 def afficher_resultat(x) :  
8     print("pour x =", x)  
9     print("- y vaut", y(x))  
10    print("- l'aire vaut", aire(x))  
11  
12 # programme principal  
13 afficher_resultat(7)  
14 afficher_resultat(19)  
15 c = int(input("entrez votre valeur de x : "))  
16 afficher_resultat(c)
```

```
pour x = 7  
- y vaut 43  
- l'aire vaut 301  
pour x = 19  
- y vaut 31  
- l'aire vaut 589  
entrez votre valeur de x : 15  
pour x = 15  
- y vaut 35  
- l'aire vaut 525
```

Paramètre(s) d'une fonction

En maths, une fonction a un (ou des) paramètre(s).

Par exemple, la fonction **aire** a un paramètre, de type numérique, noté x dans la définition de la fonction.

Ce nom peut être quelconque. Il signifie juste que si on fournit une valeur notée x , la fonction calcule et retourne la valeur $x * (50 - x)$.

Avec python, on peut définir des fonctions utilisant un paramètre, ou aucun paramètre, ou plusieurs !

Paramètre(s) d'une fonction

```
1 def disBonjour() :  
2     print("bonjour !")  
3  
4 def disMerci() :  
5     print("merci !")  
6  
7 # programme principal  
8 disBonjour()  
9 disMerci()
```

exemples de fonctions sans paramètre

Paramètre(s) d'une fonction

```
1 def dis(message) :  
2     print(message, "!")  
3  
4 # programme principal  
5 dis("bonjour")  
6 dis("merci")
```

exemple de fonction avec un paramètre

Paramètre(s) d'une fonction

```
1 def dis(message, personne) :  
2     print(message, personne, "!")  
3  
4 # programme principal  
5 dis("bonjour", "Alice")  
6 dis("merci", "Bob")
```

exemple de fonction avec deux paramètres

Type de retour d'une fonction

Certaines fonctions précédentes ne retournent aucune valeur. Par exemple, les fonctions précédentes ne font que de l'affichage.

D'autres fonctions (*y* ou *aire*) retournent un résultat qui peut être de type divers :

- une valeur numérique (int, float)
- une valeur booléenne
- une chaîne de caractère
- une liste, un tuple, ...

Exemples

```
1 def divide(a, b) :  
2     # retourne vrai si a divise b, faux sinon  
3     return b % a == 0  
4  
5 def pgcd(a, b) :  
6     # retourne le pgcd de a et b  
7     m = min(a, b)  
8     d = 1  
9     for i in range(1, m+1) :  
10         if divide(i, a) and divide(i, b) :  
11             d = i  
12     return d  
13  
14 # programme principal  
15 a = int(input("a : "))  
16 b = int(input("b : "))  
17 print("le pgcd de", a, "et", b, "est", pgcd(a, b))  
--
```

```
a : 15  
b : 12  
le pgcd de 15 et 12 est 3
```

Examples

```
1 def email(prenom, nom) :
2     # retourne une chaîne de caractères assimilable
3     # à une adresse email construite sur prenom et nom
4     return prenom + "." + nom + "@etu.univ-orleans.fr"
5
6 # programme principal
7 liste = [("antoine", "dupont"), ("gael", "fickou")]
8 for t in liste :
9     print("prénom :", t[0])
10    print("nom      :", t[1])
11    print("email   :", email(t[0], t[1]))
12    print("-----")
```

```
prénom : antoine
nom     : dupont
email   : antoine.dupont@etu.univ-orleans.fr
-----
prénom : gael
nom     : fickou
email   : gael.fickou@etu.univ-orleans.fr
-----
```

Exemples

```
1 def diviseurs(n) :
2     # retourne la liste des diviseurs de n
3     listeDiv = []
4     for d in range(1, n+1) :
5         if n % d == 0 :
6             listeDiv = listeDiv + [d,]
7     return listeDiv
8
9 def est_premier(n) :
10     # calcule la liste des diviseurs de n
11     liste_div = diviseurs(n)
12     # retourne True si liste_div a pour longueur 2
13     return len(liste_div) == 2
14
15 # programme principal
16 print("diviseurs de 60  :", diviseurs(60))
17 print("diviseurs de 17  :", diviseurs(17))
18 print("60 est-il premier ?", est_premier(60))
19 print("17 est-il premier ?", est_premier(17))
```

```
diviseurs de 60  : [1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 60]
diviseurs de 17  : [1, 17]
60 est-il premier ? False
17 est-il premier ? True
```