INFO-H-100 - Informatique

Séance d'exercices 3
Introduction à Python
Tests et fonctions

Université Libre de Bruxelles
Faculté des Sciences Appliquées

2011-2012

Fonctions

Une fonction est une séquence d'instructions qui a un nom. Elle peut recevoir en entrée des arguments et peut renvoyer une valeur de retour.

```
longueur = len('SPAM')
```

On peut voir une fonction comme une boite noire qui effectue un travail.

- Ses arguments sont les informations dont elle a besoin pour faire son travail.
- Sa valeur de retour est le résultat de son travail.

Composition : un argument d'une fonction peut être toute expression compatible :

```
x = math.sin(degrees / 360.0 * 2 * math.pi)
x = math.exp(math.log(x+1))
```

Définition de fonction

Une définition de fonction spécifie le nom, les paramètres (optionnels) et la séquence d'instructions de la fonction.

Chaque ligne de la séquence d'instruction est indentée, c'est à dire décalée vers la droite (par exemple de 4 espaces).

```
>>> def times(x, y):
      return x * y

>>> def pretty_print(a_string):
      print '*' * (len(a_string) + 4)
      print '* ' + a_string + ' *'
      print '*' * (len(a_string) + 4)

>>> y = times(2, 3)
>>> print y
6
>>> pretty_print('Python')
**********
* Python *
**********
```

Retour et paramètres

Le mot clé `return` interrompt la fonction et définit son résultat.

```
>>> def get_ratio(x, y):
      return float(x) / y
      print 'done.'

>>> get_ratio(3,4)
0.75
```

Ici, l'instruction `print 'done.'` n'est jamais exécutée.

L'ordre des arguments est important, pas leur nom.

```
>>> get_ratio(4,3)
1.3333333333333333
```

```
>>> x = 4
>>> y = 3
>>> get_ratio(y, x):
0.75
```

Variables locales

Les paramètres et les variables définies à l'intérieur d'une fonction sont des variables locales à leur fonction, c'est-à-dire qu'ils n'existent pas en dehors de leur fonction.

```
>>> def pretty_print(a_string):
      size = len(a_string) + 4
      print '*' * size
      print '* ' + a_string + ' *'
      print '*' * size

>>> pretty_print('Python')
**********
* Python *
**********
>>> a_string
NameError: name 'a_string' is not defined
>>> size
NameError: name 'size' is not defined
```

On parle de portée d'une variable : la zone dans laquelle elle est visible.

Documenter ses fonctions

Un doctring est un commentaire éventuellement multiligne (encadré par des """) placé au début du corps d'une fonction.

```
>>> def get_sum(x, y):
      """ returns the sum of x and y """
      return x + y

>>> help(get_sum)
'Help on function get_sum in module __main__:
get_sum(x, y)
  returns the sum of x and y
```

Bonne habitude : documenter clairement ses fonctions.

Dire ce que la fonction fait et pas comment elle le fait.

Expressions booléenes

Une expression booléenne est une expression dont la valeur est soit vrai (`True`) , soit faux (`False`). Ces expressions sont de type `bool`.

```
>>> 5 == 5
True
>>> 5 != 5
False
```

Elle peut se composer d'opérandes et de comparateurs :
`<` `<=` `>` `>=` `!=` `==`

Ne pas confondre = (assignation) et == (comparaison d'égalité).

Tests simples

L'instruction `if` permet de tester une condition et d'exécuter du code si cette condition est vérifiée.

```
x = 2
if x >= 0 :
  print 'x est positif'
```

Le code exécuté est constitué du code indenté qui suit l'instruction `if`.

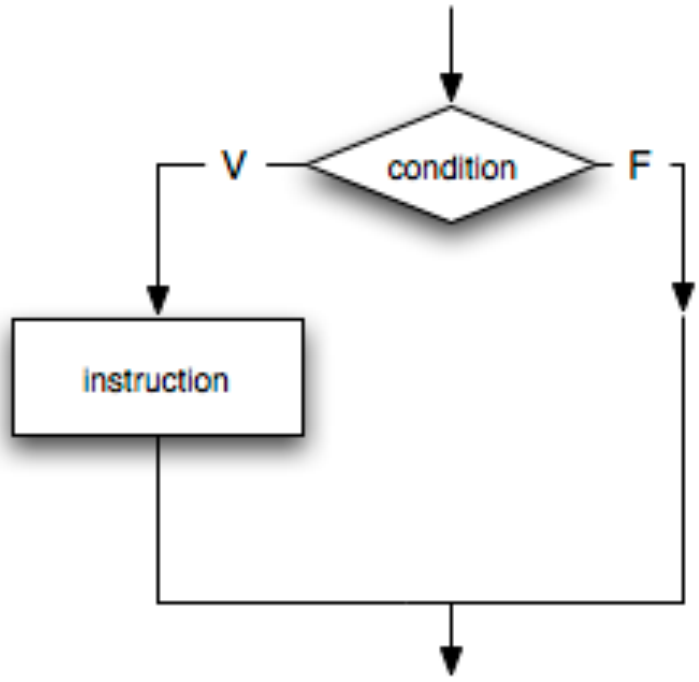
Une deuxième forme d'instruction `if` est disponible :

```
x = 2
if x % 2 == 0 :
  print 'x est pair'
else :
  print 'x est impair'
```

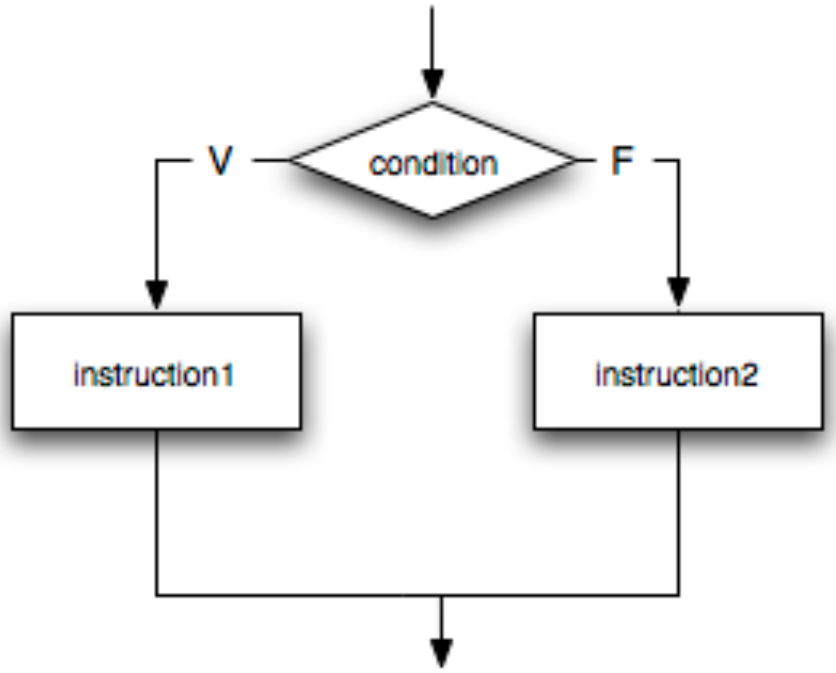
Dans ce cas, les instructions indentées après le `else` sont exécutées uniquement si la condition du `if` est fausse.

Tests simples

```
if condition:
  instruction
```




```
if condition:
  instruction1
else:
  instruction2
```



Tests et fonctions

```
def is_even(number):
  """ returns True if number is even.
      Otherwise returns False. """
  if number % 2 == 0 :
    return True
  else :
    return False

x = int(raw_input())
if is_even(x):
  print str(x) + ' est pair'
```

Autres versions :

```
def is_even(number):
  return number % 2 == 0
```

```
def is_even(number):
  test = False
  if number % 2 == 0 :
    test = True
  return test
```

Exercices

- Exercices 1 à 8.
- Pour chaque exercice, écrire la ou les fonctions nécessaires et les tester à l'aide de valeurs pertinentes.