Semaine 6

Initiation à l'algorithmique et programmation

Revekka Kyriakoglou

Plan du cours

- Fonctions

Motivation

Pourquoi les Fonctions ?

- **Réutilisez Votre Code :** Les fonctions vous permettent d'écrire un bloc de code une fois et de l'utiliser autant de fois que vous le souhaitez.
- **Simplifiez Votre Code :** Les fonctions contribuent à maintenir votre code organisé et facile à comprendre.
- **Facilitez le Débogage :** S'il y a un bug dans une fonction, vous n'avez qu'à le corriger une fois à l'intérieur de la fonction.
- **Collaborez et Conquérez :** Les fonctions favorisent la collaboration.

Exemples

Vous connaissez déja des fonctions internes à Python comme `range()`, `print()` ou `len()`.

Exemples

Vous connaissez déja des fonctions internes à Python comme `range()`, `print()` ou `len()`.

- A ces fonctions vous passez aucune, une ou plusieurs variable(s) entre parenthèses. Ces variables sont appelées **arguments**. Il peut s'agir de n'importe quel type d'objet Python (str, int, float etc).
- Elles effectuent une action.
- Elles renvoient un objet Python ou rien du tout.

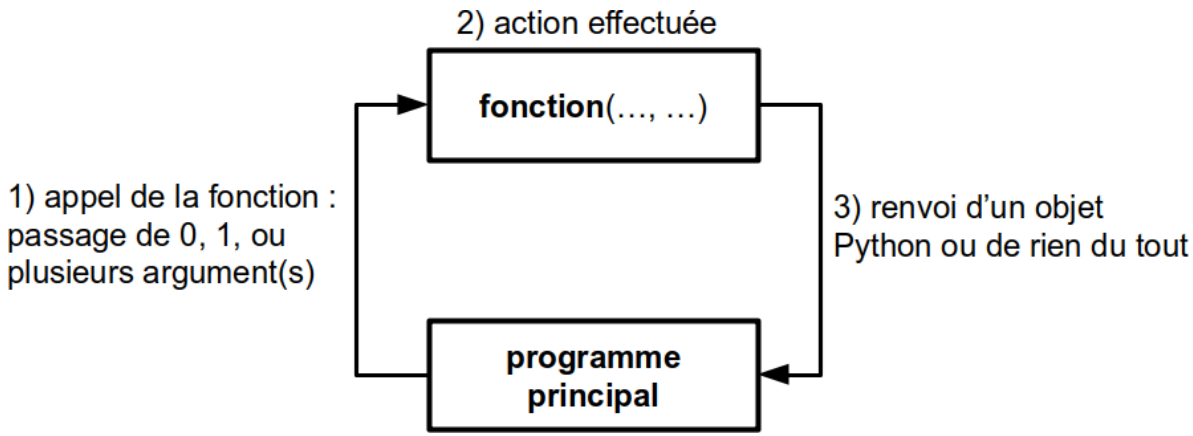

Figure – Figure : `https://python.sdv.univ-paris-diderot.fr/09_fonctions/`

Syntax

```
def function_name(parameters):
    """
    Docstring: Description of the function.
    """
    # Function body:
    # Code that defines the function's behavior.
    # ...
    # Optional:
    # Return statement to return a value.
    return result
```

Syntax

```
def function_name(parameters):
    """
    Docstring: Description of the function.
    """
    # Function body:
    # Code that defines the function's behavior.
    # ...
    # Optional:
    # Return statement to return a value.
    return result
```

- Il faut définir une fonction avant de l'appeler.
- Pour définir une fonction, Python utilise le mot-clé `def`.
- La syntaxe de def utilise les deux-points (:).
- L'indentation de ce bloc d'instructions (qu'on appelle le **corps** de la fonction) est obligatoire.

Exemple

```
def hello():
    """
    This function prints Hello word.
    """
    print("Hello word!")


# Example of calling the function
hello()
```

Passage d'arguments

Le nombre d'arguments que l'on peut passer à une fonction est variable.

Passage d'arguments

Le nombre d'arguments que l'on peut passer à une fonction est variable.

- `len()` prend un paramètre :

```
my_string = "Hello, World!"
# Call the len() function with one parameter
length = len(my_string)
# Print the result
print(f"The length of the string is: {length}")
```

- `max()` prend deux paramètres :

```
# Call the max() function with two parameters
maximum_value = max(42, 17)
# Print the result
print(f"The maximum value between {number1}
and {number2} is: {maximum_value}")
```

Exemple

 En Python, vous n'êtes pas obligé de préciser le type des arguments que vous lui passez.

```
def greet(name):
    """
    This function greets the person passed
    in as a parameter.
    """
    print ("Hello" + name+ "!")

# Example of calling the function
greet("Alice")
```

Renvoi de résultats

 Les fonctions sont capables de renvoyer un ou plusieurs objets à la fois.

Renvoi de résultats

 Les fonctions sont capables de renvoyer un ou plusieurs objets à la fois.

```
def greet(name):
    """
    This function greets the person passed
    in as a parameter.
    """
    greeting = "Hello" + name+ "!"
    return greeting

# Example of calling the function
message = greet("Alice")
print(message)
```

Renvoi de résultats

```
def greet(name):
    """
    This function greets the person passed
    in as a parameter.
    """
    greeting_en = "Hello␣" + name+ "!"
    greeting_fr = "Bonjour␣"+ name+ "␣!"
    return greeting_en, greeting_fr

# Example of calling the function
message_en, message_fr = greet("Alice")
print(message_en)
```

Exemple

```
def carre_cube(x):
    return x**2, x**3
carre_cube(2)
```