Functions in Python

- *Functions are named blocks of code that are designed to do one specific job.*

- *When you call the function, python execute the code inside the function.*

Functions in Python

greeting.py > ...

```
1   # function definition
2   def greet_user():
3       """ Display a simple greeting """
4       print("Hello!")
5
6   greet_user()
7
```

Call the function

```
Hello!
```

- `def` *(=define)*
- *Function name :* `greet_user`
- *Parentheses:* `()` *(empty = no parameter)*
- *Ends in a colon* `:`
- *Docstring:* `“““ Display a simple greeting ”””`
- *Display information (No return statement)*

function.py > ...

```
1   # Define function name (parameters):
2   def reverse (word):
3
4       """Reverse a string"""
5
6       # Do some operations...
7       new_word = word[::-1]
8
9       # Return something
10      return new_word
11
12  # calling the function (argument):
13  test=reverse("12345")
14
15  print(test)
```

```
54321
```

Functions in Python

- `def` *(=define)*
- *Function name :* `reverse`
- *Parentheses:* **`(word)`** *(= parameter)*
- *Ends in a colon* **`:`**
- *Docstring:* **`""" Reverse a string """`**
- *Return a value:* **`new_word`**

pets.py > ...

```
1   def describe_pet(animal_type, pet_name):
2
3       """Display information about a pet"""
4
5       print("I have a {}.".format(animal_type.lower()))
6
7       print("My {}'s name is {}.".
8       format(animal_type.lower(),pet_name.title()))
9
10  describe_pet("dog","Teka")
11  describe_pet("cat","Mia")
```

```
I have a dog.
My dog's name is Teka.
I have a cat.
My cat's name is Mia.
```

Functions in Python

- `def` *(=define)*
- *Function name :* `describe_pet`
- *Parentheses: (= parameters)*

(animal_type, pet_name)

- *Ends in a colon* **:**
- *Docstring:*

“““Display information about a pet ”””

- *Display information (No return statement)*

Functions in Python

Returning a simple value

formatted_name.py > musician

```
1   def get_full_name(first_name, last_name):
2
3       """ Return a formatted full name"""
4
5       full_name="{} {}".format(first_name, last_name)
6
7       return full_name.title()
8
9   musician = get_full_name("jimi","hendrix")
10
11  print(musician)
12
```

```
Jimi Hendrix
```

Functions in Python

pets.py > ...

```
1   def describe_pet(animal_type, pet_name):
2
3       """Display information about a pet"""
4
5       print("I have a {}.".format(animal_type.lower()))
6
7       print("My {}'s name is {}.".
8       format(animal_type.lower(),pet_name.title()))
9
10  describe_pet("dog","Teka")
11  describe_pet("cat","Mia")
12
13  describe_pet("harry","hamster")
```

If the function has multiple parameters, you can pass arguments as

- ***Positional arguments**: they need to be in the same order the parameters were written.*

```
I have a dog.
My dog's name is Teka.
I have a cat.
My cat's name is Mia.
I have a harry.
My harry's name is Hamster.
```

Order matters in positional arguments

Animal_type = "harry" ✘

Pet_name="hamster" ✘

Functions in Python

pets.py > ...

```
1   def describe_pet(animal_type, pet_name):
2
3       """Display information about a pet"""
4
5       print("I have a {}.".format(animal_type.lower()))
6
7       print("My {}'s name is {}.".
8       format(animal_type.lower(),pet_name.title()))
9
10  describe_pet("dog","Teka")
11  describe_pet("cat","Mia")
12
13  describe_pet(animal_type="hamster",pet_name="harry")
14  describe_pet(pet_name="harry",animal_type="hamster")
```

If the function has multiple parameters, you can pass arguments as

- ***Keyword arguments**: you directly associate the name and the value within the argument.*

```
I have a dog.
My dog's name is Teka.
I have a cat.
My cat's name is Mia.
I have a hamster.
My hamster's name is Harry.
I have a hamster.
My hamster's name is Harry.
```

There is no confusion.

Use the exact names of the parameters in the function's definition.

petsD.py > ...

```
1   def describe_pet(pet_name, animal_type="dog"):
2
3       """Display information about a pet"""
4
5       print("I have a {}.".format(animal_type.lower()))
6
7       print("My {}'s name is {}.".
8       format(animal_type.lower(),pet_name.title()))
9
10  describe_pet("willie")
```

```
I have a dog.
My dog's name is Willie.
```

Default values need to be listed after all parameters that don't have default values.

Functions in Python

If the function has multiple parameters, you can pass arguments as

- ***Default values**: you can define a default value for each parameter. If an argument for a parameter is provided in the function call, Python uses the argument value.*

Functions in Python

petsD.py > ...

```
1   def describe_pet(pet_name, animal_type="dog"):
2
3       """Display information about a pet"""
4
5       print("I have a {}.".format(animal_type.lower()))
6
7       print("My {}'s name is {}.".
8       format(animal_type.lower(),pet_name.title()))
9
10  describe_pet("willie")
11
12  describe_pet(pet_name="willie",animal_type="dog")
13
14  describe_pet(animal_type="dog",pet_name="willie")
15
16  describe_pet("willie","dog")
17
18  describe_pet(pet_name="willie")
19
```

Equivalent function calls:

```
I have a dog.
My dog's name is Willie.
I have a dog.
My dog's name is Willie.
I have a dog.
My dog's name is Willie.
I have a dog.
My dog's name is Willie.
I have a dog.
My dog's name is Willie.
```

Functions in Python

Making an argument optional

formatted_name_FLM.py > ...

```
1   def get_full_name(first_name, last_name, middle_name=''):
2
3       """ Return a formatted full name"""
4
5       if middle_name:
6           full_name="{} {} {}".format(first_name, middle_name, last_name)
7       else:
8           full_name="{} {}".format(first_name, last_name)
9
10      return full_name.title()
11
12  musician = get_full_name("jimi","hendrix")
13  print(musician)
14
15  musician = get_full_name("john","hooker","lee")
16  print(musician)
```

```
Jimi Hendrix
John Lee Hooker
```

Functions in Module

Storing your functions in modules (files .py)

```
import module_name
module_name.function_name()
```

```
import petsD
petsD.describe_pet()
```

```
import module_name as mn
```

```
import pandas as pd
```

```
from module_name import function_name
function_name()
```

```
from petsD import describe_pet
describe_pet(…)
```

Functions in Module

grades.py > get_grade

```python
def get_grade(final_grade):
    if final_grade > 91:
        return "A"
    elif final_grade > 76:
        return "B"
    elif final_grade > 65:
        return "C"
    elif final_grade > 50:
        return "D"
    else:
        return "F"
```

course.py > ...

```python
from grades import get_grade

quizzes=[100,87,93,74,69,45,70]

for quiz in quizzes:
    print(get_grade(quiz), quiz)

```

```
A 100
B 87
A 93
C 74
C 69
F 45
C 70
```