Python

PII: Functions
Objective: DRY

Example 1

```
#Function Definition
def nameAndAge(name: str, age: int) ->
str:
    print("Hi "+ name + " you are " +
str(age) + " years old")


#Function Call
nameAndAge("Driss", 32)
```

What the hell are they?

- Block of statements that relate to one another whose goal is to complete a specific task
- Enable me to use them over and over again
- For example, I want to add two numbers
- Instead of defining two vars and adding them every time
- I can create a template which takes two ints and returns their sum
- Some functions MAY OR MAY NOT RETURN A VALUE
- Any logic executed on an object within the function gets reflected outside it
  - i.e. Any modifications made within your parameters inside the function gets reflected outside
  - For example, I have listA defined within my function I add to it
  - Then I make a call to my function and invoke the append method
  - All the items that were inside my function defintion join with the new members I added

<! Hold Your POSITION />

- On the right I am defining a function that takes **positional arguments**
- That means in my function call **Driss** holds the position for name and the value of **32** holds the position for age

```
#Function Definition
def nameAndAge(name: str, age: int) ->
str:
    print("Hi "+ name + " you are " +
str(age) + " years old")


#Function Call
nameAndAge("Driss", 32)
```

<! **Keyword** To Get Access To The Property />

- In Python the second type of arguments are called **Keyword Arguments**
- Python allows me to make function calls using keyword arguments
- When using keyword arguments order DOES NOT matter in contrast to positional arguments

```python
#Function Definition
def nameAndAge(name: str, age: int) -> str:
    print("My name is "+ name + " and I am " + str(age) + " years old")


'''
Invoking the function using Keyword Args
'''

nameAndAge(age=28, name="Robert Smith")
```

None

- When you want to give some flexibility to the user when invoking a function set the parameter to None
- This will make the 2nd field optional and fire the function error free

Rules To Remember

- Python reads line by line therefore if you define a function on line 8 you **CANNOT** make a call to it on line 3.
- This will generate a **Runtime Error**
- When Naming:
  - Use lowercase and underscores to be more descriptive
  - **Never** use leading special symbols
  - **Never** use reserved keywords(e.g. **is, break, class, True, del, def,** etc.)
- Do not forget about case sensitivity
  - VarOne **IS NOT** the same as varOne

If you <**default /**> to the bank in Morocco you will not sleep

- In Python functions have the ability to be defined with default values
- This will allow the default values to hold the values of the arguments that were not
  - passed in
- To use default values use the assignment operator = and give the default value you want
- In Python you must remember either you have a function with every single argument having a default or none
- In your function defintion you cannot have parameterA having a default value and parameterB not having a default value
-

Default Parameters

```
def greet(name, msg="Good morning sir/mam!"):
        print("Hello", name + ', ' + msg)
greet("Youssef")
greet("Ilham", "How are you doing today?")
```

Thank you