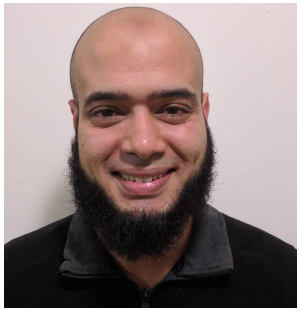
Python Programming

Define Function

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Recall Functions

- Something is written and ready-to-use by others
- We write once, and then start using
- It saves everyone time!
- Python provides us with several built-in functionalities
- Today, we learn how to write our own functions!

Calling Built-in Functions

```
print()      # print: takes nothing and print newline

print(1, 5, 'hi')      # print: takes 3 arguments and return nothing

# min: function takes 2 aruments and return result
answer = min(3, 6)

type(15)     # takes 1 thing and return its type

len('mostafa')     # takes a string and return its length

int('200')         # takes something and convert to integer

s1 = input()              # takes nothing: return read line
s2 = input('Enter: ')     # takes string: return read line

# Overall
# we have a name:, e.g. input
# it takes arguments: e.g. 'Enter ' in input('Enter ')
# it may return something: string from input('Enter ')

```

Defining our functions

```
2  def our_print(first, second):
3      print('Sum =', first + second)
4      print('Multiplication =', first * second)
5
```

- Take a minute to observe and guess what is that!
- Goal: a function that takes 2 numbers and sum them for us
- Syntax structure
  - Keyword def
  - Followed by function name (similar rules to variable names)
  - Followed by (): inside it a list of variable names, then :
    - Indented block with some logic

Code **Flow**

- Line 6 executes nothing!
  - It just defines the function for later usage

```
2   print('Hello')
3   # PEP: Leave 2 lines
4
5
6   def our_print(first, second):
7       print('Sum =', first + second)
8       print('Multiplication =', first * second)
9
10
11  our_print(1, 2)
12  print('Oh')
13  our_print(3, 4)
14
15  """
16  Hello
17  Sum = 3
18  Multiplication = 2
19  Oh
20  Sum = 7
21  Multiplication = 12
22  """
```

Parameter vs Argument

```
2
3   # parameter: the variable used in defining the function
4   # first and second are called: parameters
5   def our_print(first, second):
6       print('Sum =', first + second)
7       print('Multiplication =', first * second)
8
9
10  # argument is an expression PASSED to the function
11  # first, 2 * 3 + 1 are arguments
12  first = 1
13  our_print(first, 2 * 3 + 1)
14
```

Type Error

```
3   def our_print(first, second):
4       print('Sum =', first + second)
5       print('Subtract =', first - second)
6
7   # TypeError: When we pass incorrect no of arguments to a function
8
9   # TypeError: our_print() missing 2 required positional arguments: 'first' and 'second'
10  #our_print()
11
12  # TypeError: our_print() missing 1 required positional argument: 'second'
13  #our_print(1)
14
15  # TypeError: our_print() takes 2 positional arguments but 3 were given
16  #our_print(1, 2, 3)
17
18  # later: meaning of: "positional" argument
19
20  # TypeError: a function is called on a value of an inappropriate type
21
22  # TypeError: unsupported operand type(s) for -: 'str' and 'str'
23  #our_print('1', '2')
```

Indentation Error

```
2
3   # IndentationError: expected an indented block
4
5   def our_print(first, second):
6   print('Sum =', first + second)
7   print('Subtract =', first - second)
8
9
10  def our_print(first, second):
11      print('Sum =', first + second)
12      print('Subtract =', first - second)
```

Function name

- Same rules as variables
- Convention: snake casing
  - All lower case letters with _
  - E.g.
  - compute_sum
  - find_smallest_position

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."