Python Programming

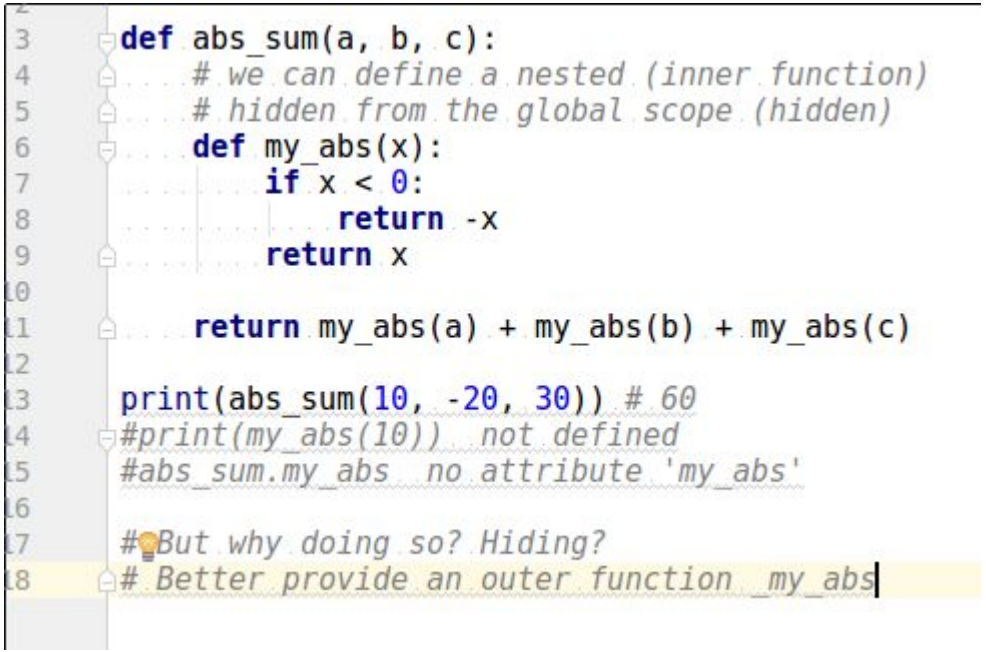
Nested function

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)

Nested Functions

- We can create a function inside a function inside a function!
- We call them nested or inner!



```
def abs_sum(a, b, c):
    # we can define a nested (inner function)
    # hidden from the global scope (hidden)
    def my_abs(x):
        if x < 0:
            return -x
        return x

    return my_abs(a) + my_abs(b) + my_abs(c)

print(abs_sum(10, -20, 30)) # 60
#print(my_abs(10))  not defined
#abs_sum.my_abs  no attribute 'my_abs'

#But why doing so? Hiding?
# Better provide an outer function _my_abs
```

Scope

- Back then, we mentioned about scopes and referred to enclosing scope!

```
3   def outer():
4       outer_loc1 = 30 # for inner func: this is an enclosing scope
5
6       def inner():
7           print(outer_loc1)   # 30: local? No. Enclosing? Yes, use it
8       inner()
9
10  outer()
11
12  """
13  But how python searches for variable?
14  We learned before about local, global and built-in
15  """
```

Namespaces

- In a Python program, there are four types of namespaces:
  - **Built-In (**e.g. len, int, max, sum, TypeError, etc)
  - **Global:** contains any names defined at the level of the main program
  - **Enclosing:** for nested functions: the scope of the enclosing function
  - **Local:** local to the function and remains in existence until the function terminates.
- Using a variable in a function: Python search order?
  - Is it local? Then it is a local variable in a local namespace
  - Is it enclosing? Then it enclosing namespace
  - Is it global? Then it global namespace
  - Is it in Built-In? Then it Built-In namespace
  - None? Error

LEGB Rule

```
4   glob1 = 20   # global
5
6   def outer():
7       outer_loc1 = 30
8       x = 15          # another outer local
9
10      def inner():
11          inner_loc = -5
12          x = 7       # another inner local
13          print(inner_loc)    # -5
14          print(x)            # 7: is it in my local scope? Yes, use it
15          print(outer_loc1)   # 30: local? No. Enclosing? Yes, use it
16          print(outer_loc2)   # 40: local? No. Enclosing? Yes, use it
17          print(glob1)        # 20: local? no, enc? no, global? Yes, use
18
19      outer_loc2 = 40
20      inner()
21      print(x)    # 15: local? yes, use it. inner x has no effect
22
23  outer()
```

nonlocal

- nonlocal keyword helps us modify variables in enclosing scope

```
4   glob1 = 20   # global
5
6   def outer():
7       outer_loc1 = 30
8
9       def inner():
10          #glob1 += 1          # UnboundLocalError
11          #outer_loc1 += 1     # UnboundLocalError
12
13          global glob1
14          glob1 += 1
15          nonlocal outer_loc1
16          outer_loc1 += 1
17
18      inner()
19      print(outer_loc1)   # 31
20
21  outer()
22  print(glob1)    # 21
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."