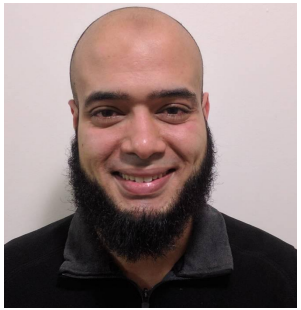
Python Programming

Local and Global Scope

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: Variable scope

- The part of a program where a variable is **accessible** is called its **scope**
  - We will learn more in Functions
- Python has **no Block scope**
  - If line 4 is activated, it is visible in line 8

```
3   if int(input()) < 1000:
4       lucky_number = 13
5
6   # lucky_number will exist
7   # ONLY if line 4 is executed
8   print(lucky_number)
```

```
/home
200
13
```

```
/home/moustafa/system-installs/anaconda3/envs/
3000
Traceback (most recent call last):
  File "/home/moustafa/00Udemy/CPP/private_git
    print(lucky_number)
NameError: name 'lucky_number' is not defined
```

Local vs Global

```
# global variable's scope is anywhere within the program
# glob1 are global var
glob1 = 20

def f():
    # Scope of Variables defined in a function is within the function ONLY
    # loc1 is a local variable: visible in f(), but NOT outside
    loc1 = 30
    print(loc1)
    print(glob1)    # is it in local? No. Is it in global? Yes. Use it
    return loc1


f()
print(glob1)    # ok: as global

print(loc1)    # NameError: name 'loc1' is not defined

```

Conflicts

```
b = 20
c = 6


def f():
    # b is local NOT global. It doesn't affect global one
    # c not in local, but in global, so used
    b = c + 1
    print(b)    # 7
    return b


f()
print(b)    # 20 NOT 7

b = f()
print(b)    # 7
```

global keyword

```
3   b = 20
4   c = 6
5
6
7   def f():
8       global b
9       # now the assigned b is actually the global one
10      b = c + 1
11      print(b)    # 7
12      return b
13
14
15  f()
16  print(b)    # 7
17
```

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: Later
  - **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
- Python has methods to show what in same spaces:
  - print(dir(__builtins__)) print(globals()) print(locals())

Constants

- Global variables are generally **bad practice** and should be avoided
  - The more we go, we better avoid them
  - Sometimes I might use for educational purposes / simplicity
- One good use case might be constants
  - Values that no one will change (e.g. PI = 3.14 / MAX_VALUE)
  - Define as all Capital letters with _
  - Note: we can still change! It is just a var
    - Using as capital letter: Our intention on one changes it!

```
PI = 3.14

def compute_area(radius):
    return PI * radius * radius
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."