CITY
UNIVERSITY OF LONDON
EST 1894

STEM Digital Academy

School of Science & Technology

www.city.ac.uk

Local and Global Variables

Programming and Algorithms

Lecture by

Dr Daniil Osudin

```
n = 3
for i in range(1,n+1):
    print("Hello World!")
```

```
Hello World!
Hello World!
Hello World!
```

What will we Cover?

- The difference between the local and global variables
- The scope of variables
- Design and implementation of user defined functions in programs

Local Variables

- A variable created inside the function
- Local variables do not exist outside the function in which they are created
- Local variables are erased from memory once the block of code that they were created in is no longer in use

Local Variables Example

- `total` is a local variable

```
def calculate_average(x, y, z): #function header
    # This function calculates an average of 3 numbers

    total = x + y + z
    average = total / 3
    return average

num1, num2, num3 = 13, 19, 27
result = calculate_average(num1, num2, num3)
print(result)
```

```
19.666666666666668
```

- Any other local variables?

Global Variables

- A variable accessible from anywhere within the program after the point they have been created
- Global variables remain available when the program is executing

Global Variables Example

- `num1` , `num2` and `num3` are global variables

```
def calculate_average(x, y, z): #function header
    # This function calculates an average of 3 numbers

    total = x + y + z
    average = total / 3
    return average

num1, num2, num3 = 13, 19, 27
result = calculate_average(num1, num2, num3)
print(result)
```

```
19.666666666666668
```

- Any other global variables?

Scope of Variables

- Local variables exist in a local scope of a function
- Global variables (assigned outside of functions) are said to have global scope
- A body of a function can access global variables, but it is generally considered as bad practice to do so – as this makes it harder to follow a program's logic (if unavoidable, use the keyword `global <variable name>`)

Common Errors in Python

Calling a local variable outside of the scope it was created in

```
def fruits():
    quantity_apples = 10

fruits()    # function call
print("Apples =", quantity_apples)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_408\174478521.py in <module>
      3
      4 fruits()    # function call
----> 5 print("Apples =", quantity_apples)

NameError: name 'quantity_apples' is not defined
```

`quantity_apples` has local scope only

Error location

Error type

Error description

Scope of Variables II

```
def numbers1():
    n = 10
    return n

def numbers2():
    n = 20
    return n

n = numbers1()
print("n =", n)
```

```
n = 10
```

Variables in different scopes can have the same name

It is considered bad practice to use the same name for local and global variables

Passing Arguments

- Parameter in a function is a local copy of the argument passed to the function
- If the value is changed within the function, it will not change the original value

Passing Arguments Example

```
def increment(num1):
    num1 += 1
    print("The number after the increment within the function is:", num1)

num1 = 0
print("The original value of the number before the function call is:", num1)
increment(num1)
print("The value of the number after the function call is:", num1)
```

```
The original value of the number before the function call is: 0
The number after the increment within the function is: 1
The value of the number after the function call is: 0
```

Try It Yourself

Write a program in python environment that

- Reads in a string of numbers separated by spaces and stores the numbers in a list
- List is then passed to a function `distinct_numbers` to create a list of distinct numbers
- The list is then returned to the main program for printing#
- For example: if input = 1 2 4 5 2 3 1 2 3 6, then output = 1 2 4 5 3 6