Function and variable scope

Functions and variables

It is essential to understand the levels of scope in Python and how things can be accessed from the four different scope levels. Below are the four scope levels and a brief explanation of where and how they are used.

1. Local scope

Local scope refers to a variable declared inside a function. For example, in the code below, the variable **`total`** is only available to the code within the **`get_total`** function. Anything outside of this function will not have access to it.

```
def get_total(a, b):
    #local variable declared inside a function
    total = a + b;
    return total

print(get_total(5, 2))
7

# Accessing variable outside of the function:
print(total)
NameError: name 'total' is not defined
```

Run

Reset

2. Enclosing scope

Enclosing scope refers to a function inside another function or what is commonly called a **nested function**.

In the code below, I added a nested function called **`double_it`** to the **`get_total`** function.

As **`double_it`** is inside the scope for the **`get_total`** function it can then access the variable. However, the enclosed variable inside the **`double_it`** function cannot be accessed from inside the **`get_total`** function.

```
def get_total(a, b):
    #enclosed variable declared inside a function
    total = a + b

    def double_it():
        #local variable
        double = total * 2
        print(double)

    double_it()
    #double variable will not be accessible
    print(double)

    return total
```

Run

Reset

3. Global scope

Global scope is when a variable is declared outside of a function. This means it can be accessed from anywhere.

In the code below, I added a global variable called **`special`**. This can then be accessed from both functions **`get_total`** and **`double_it`**:

```

special = 5

def get_total(a, b):
    #enclosed scope variable declared inside a function
    total = a + b
    print(special)

    def double_it():
        #local variable
        double = total * 2
        print(special)

    double_it()
```

```

        return total
```

Run

Reset

4. Built-in scope

Built-in scope refers to the reserved keywords that Python uses for its built-in functions, such as **`print, def, for, in`**, and so forth. Functions with built-in scope can be accessed at any level.