Local Variables

Local Variables

Variables defined in the main body of a script are called **global** variables. These variables are accessible inside of functions:

[4]:
```
x = 5

def get_x():
    return x
```

[5]:
```
get_x()
```

[5]: 5

The arguments parsed into and the variables defined inside the function are **local variables**. They only exist in a particular instance of a function.

In other words, these variables are not accessable from outside the function. For example:

[1]:
```
def make_var():
    func_var = 4
    return func_var
```

[2]:
```
make_var()
```

[2]: 4

[3]:
```
func_var
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-3-96e608aeebf3> in <module>
----> 1 func_var
```

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

If we were to define `func_var` as a global variable, `make_var` will instance a local variable instead of reassigning the global variable:

[6]:
```
func_var = 6

print('Before function', func_var)
print('Function return', make_var())
print('After function', func_var)
```

```
Before function 6
Function return 4
After function 6
```

Note that when referencing a variable, Python will check the local namespace **before** the global namespace (i.e. local variables are given preference).

As stated above, function arguments can also be treated as local variables.

[7]:
```
def arg_var(x):
    return x
```

[8]:
```
x = 5

arg_var(2)
```

[8]:
```
2
```