Python Functions

Functions

- Starts with keyword `def`
- Function is given a name
- Can include zero or more parameters for passing data in
- To invoke, use name + arguments in parenthese

In [1]:

```
def myfunc(x):
    print('do something', x)
    if x == 1:
        return True
    else:
        return 'abc'

print(myfunc(2))
```

```
do something 2
abc
```

Documentation strings

- Often functions will include documentation strings as first statement
- Provides info about the function's intent
- Can be used to feed the `help()` command

In [2]:

```
def calc_subtotal(quantity, unit_cost):
    '''
    Computes subtotal for an order
    '''
    return quantity * unit_cost

help(calc_subtotal)
```

```
Help on function calc_subtotal in module __main__:
```

```
calc_subtotal(quantity, unit_cost)
    Computes subtotal for an order
```

...functions return `None` if return not invoked

In [3]:
```python
def myfunc(x):
    print('do something', x)

print(myfunc(35))
```

```
do something 35
None
```

What is `None`?

- It acts like `False`, but it's a different object

In [4]:
```python
def myfunc(x):
    print('do something', x)

retval = myfunc(2)
if retval:
    print('True branch of if')
else:
    print('False branch of if')
```

```
do something 2
False branch of if
```

In [5]:
```python
def myfunc(x):
    print('do something', x)

retval = myfunc(2)

if retval is None:
    print('preferred over retval == None')
if None is False:
    print('no!')
id(None), id(False)
```

```
do something 2
preferred over retval == None
(9484816, 9474016)
```

Out[5]:

Scope

- Python is *NOT* block scoped

In [6]:

```
if True:
    x = 'global x' # x will persist outside this block

print("outside the block, x =", x)

def func():
    print("---> in func")
    x = 'func x' # declare var inside function
    print("x =", x)
    d = locals()
    print("local x =", d['x'])
    d = globals()
    print("global x =", d['x'])
    print("---> leaving func")

func()

print("in main, after func call, x =", x)

def func():
    print("---> inside second func")
    # can access global variables here
    # print("x =", x)
    # ...but to change them, we need to bind
    # the name 'x' to the global var instead
    # of a new local var...
    global x
    x = 'new global x'
    print("x =", x)
    print("---> leaving second func, x =", x)

func()
print("in main, after second func call, x =", x)
```

```
outside the block, x = global x
```

```
---> in func
x = func x
local x = func x
global x = global x
---> leaving func
in main, after func call, x = global x
---> inside second func
x = new global x
---> leaving second func, x = new global x
in main, after second func call, x = new global x
```

Returning values from a function

- In Python, you can return multiple values from a function
- Wrap the return values in `()` and separate each with a comma
- Returns what's known as a `tuple` in Python

In [7]:

```
def addmul(op1, op2):
    return (op1 + op2, op1 * op2)

sum, product = addmul(2.75, 13.2)
print(sum)
print(product)
```

```
15.95
36.3
```

Parameter default values

- To give a parameter a default value, use assignment
- Parameters given defaults can be omitted from calls to function
- Omitted arguments will take on default value
- When calling, arguments can be named - can help with readability

In [8]:

```
def connect(hostname, port, timeout = 300):
    '''
    Simulates connectivity to a host on a port
    '''
    print('Hitting...', end = '')
    print(f'{hostname}:{port}...', end = '')
    print(f'Finish before {timeout} milliseconds!!')
    return

connect('www.python.org', 80)
connect('www.python.org', 80, 500)
connect(timeout = 1000, hostname = 'www.python.org', port = 443)  # when named, order doesn't matter
```

```
Hitting...www.python.org:80...Finish before 300 milliseconds!!
Hitting...www.python.org:80...Finish before 500 milliseconds!!
Hitting...www.python.org:443...Finish before 1000 milliseconds!!
```

Exercise One

- Update your Python program for order processing
- Create a function to handle the calculations
- Call the function, passing the inputs from the user
- Fully encapsulate the discount algorithm and calculations within the function
- Return subtotal, total including tax, and final total after discount from the function

Exercise Two

- Create a function called circleinfo for calculating area and circumference of a circle
- The function should accept a parameter for radius
- Return area and circumference from the function
- Area is calculated as Pi *radius* radius
- Circumference is calculated as 2 *Pi* radius
- **Hint**: Use math.pi() to get the value for Pi in the formulae (https://www.delftstack.com/howto/python/pi-in-python/#:~:text=Use%20Pi%20in%20Python.%201%20Use%20the%20math.pi,to%20Get%20the%20Pi%20Value%20in%20Python.%20)