Introduction to Python

Dr. Padraig Corcoran

Executing a Python file

- Python is an interpreted language; you write Python (.py) files with a text editor and the python interpreter directly runs/executes these files.
- Consider the python file “helloworld.py” which contains the single line:

```
print("Hello, World!")
```

- To run this type the following on the command line:

```
C:\Users\Your Name>python helloworld.py
```

- The output should read:

```
Hello, World!
```

Python Indentations

- Python uses indentation to indicate a block of code.
- Consider the following code:

```
if 5 > 2:
    print("Five is greater than two!")
```

- Python will give you an error if you skip the indentation:

```
if 5 > 2:
print("Five is greater than two!")
```

Comments

- Python has commenting capability for the purpose of in-code documentation.
- Comments start with a #, and Python will render the rest of the line as a comment:

```
#This is a comment.
print("Hello, World!")
```

Python Variables

- Python has no command for declaring a variable; a variable is created the moment you first assign a value to it.

```
x = 5
y = "John"
print(x)
print(y)
```

- Variables do not need to be declared with any particular type and can even change type after they have been set.

```
x = 4 # x is of type int
x = "Sally" # x is now of type str
print(x)
print(type(x)) # Will be a string in this case
```

Python Lists

- List is a collection which is ordered and changeable.
- Create a List:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

- You access the list items by referring to the index number:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

- To change the value of a specific item, refer to the index number:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

- You can loop through the list items using a for loop:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
    print(x)
```

- To determine how many items a list have, use the len() method:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

- To add an item to the end of the list, use the append() method:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

- The del keyword removes the specified index:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Python If ... Else

- If statement:

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

- The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

```
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
```

- The else keyword catches anything which isn't caught by the preceding conditions.

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

Python For Loops

- A for loop is used for iterating over a sequence.

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Python Functions

- A function is a block of code which only runs when it is called.
- A function may take parameters and return a result.
- Consists of a header and body.

```
def my_function(x):
    return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

Python Classes and Objects

- Objects provide a way of encapsulating data and operations that can be performed on that data.
- Create a class named MyClass, with a property named x:

```
class MyClass:
    x = 5
```

- Create an object named p1, and print the value of x:

```
p1 = MyClass()
print(p1.x)
```

Object constructor

- All classes have a function called __init__(), which is always executed when the class is being initiated.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)
print(p1.name)
print(p1.age)
```

- The self parameter is a reference to the class itself.

Object Methods

- Objects can also contain methods; functions that belongs to the object.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Python Modules

- Consider a module to be the same as a code library; a set of functions you want to include in your application.
- Save this code in a file named mymodule.py

```
def greeting(name):
    print("Hello, " + name)
```

- Import the module named mymodule, and call the greeting function:

```
import mymodule
mymodule.greeting("Jonathan")
```

References

Python Tutorial, W3schools.com, https://www.w3schools.com/python/