Python programming

- Python is
  - powerful high-level programming language
  - object oriented: encapsulate code in objects.
  - Suitable language for beginners
    - few keywords, simple structure, clear syntax
  - interpreted: processed at runtime
  - interactive: use Python prompt

Python: installation

- Anaconda: an easy way
  - bundled with IDE and python IDE (jupyter)
  - D[illegible] m for V[illegible]
  - F[illegible] lt c[illegible]
  - In advanced options step, keep the default option (*not selected*): Add Anaconda to the system PATH
  - Once done, search ‘jupyter’ in start menu and start it
  - Alternatively, search ‘anaconda prompt’ in start menu and start it. In that command line, type ‘jupyter notebook’ and ENTER

Pleas follow the instructions provided in FutureLearn

Python: Hello, World!

- Run the following cell in your Python IDE (i.e. Jupyter Notebook)

sit720.ipynb

Code

In [2]:
```
print("Hello, World!")
print("Hello, World!")
```

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

In [3]:
```
# This is a one-line comment

'''
This is a
multi-line comment
'''

"""
This is also a
multi-line comment
"""
print("I don't see any comments")
```

```
I don't see any comments
```

Python: Types of variables

- Variable names contain letters and numbers
- You do not need to explicitly define the type of variable
- Inspect the variable type using the type() command

sit720.ipynb

```
In [5]: x = 10
        type(x)

Out[5]: int

In [6]: x = 10.5
        type(x)

Out[6]: float
```

Python data type: Lists

- Most versatile and common data type
- Contain values of different types
- Indexing starts from 0 and ends with −1 which represent the last element.

720.ipynb

Code

In [8]:
```
mylist = [1, 5, 2.57, 'abc', 4.09]
print(type(mylist))   # prints the type
print(mylist)   # prints the entire list
print("Length of the list is: ", len(mylist))
```

```
<class 'list'>
[1, 5, 2.57, 'abc', 4.09]
Length of the list is:  5
```

In [9]:
```
# indexing
numbers = [1, 2, 3, 4, 5, 6, 7, 8]
print(numbers[0])
print(numbers[3])
print(numbers[-1])    # last element
print(numbers[-2])    # can you guess the result?
```

```
1
4
8
7
```

Python data type: Tuple

- Like a list, but the items cannot be changed once initiated
- Indexing rules are similar to lists
- Syntax: tuple_name = (item_1, item_2, …., item_n)

```
In [11]: states = ('VIC', 'NSW', 'QLD', 'WA', 'ACT', 'NT', 'TAS', 'SA')
         print(type(states))
         print(states)

         <class 'tuple'>
         ('VIC', 'NSW', 'QLD', 'WA', 'ACT', 'NT', 'TAS', 'SA')

In [12]: numbers = [1, 2, 3, 4, 5, 6, 7, 8]
         numbers[2] = 32.56
         print(numbers)

         [1, 2, 32.56, 4, 5, 6, 7, 8]

In [13]: states[2] = 'Idontknow'

         ---------------------------------------------------------------------------
         TypeError                                 Traceback (most recent call last)
         <ipython-input-13-19cfe3629fa7> in <module>()
         ----> 1 states[2] = 'Idontknow'

         TypeError: 'tuple' object does not support item assignment
```

Python data type: Dictionaries

- Similar to lists, except that each element is a key-value pair.
- The syntax for dictionaries is {key1 : value1, …}, values can be of any type, even another dictionary.

```
In [14]: dict = {}
         dict['one'] = "This is one"
         dict[2] = "This is two"

         print(dict['one'])
         print(dict)
         dict['one'] = "One has changed"
         print(dict['one'])

         This is one
         {'one': 'This is one', 2: 'This is two'}
         One has changed
```

Python data type: List/Tuple/Dictionary

- Comparison among: List, Tuple, Dictionary

	List	Tuple	Dictionary
Syntax	X = [1,2,3]	X = (4,5,9)	X = {5:'python‘}
Index	Integer and starts from 0. X[0] contains value 1	Same as List. Integer and starts from 0. X[0] contains value 4	User defined index (key). There is not fixed type. X[5] contains python
Modify element	You can modify values of list. X[0]=10 will change the first element of the list from 1 to 10	Once initalised you cannot change the value. So X[0]=10 will generate error.	Same as List. You can modify value assigned at any key. So X[5] = 'jython' will change the value from ‘python’ to ‘jython’

Python data type: Strings

- Stores text messages and have the same indexing rules as lists and tuples

sit720.ipynb

Code

In [15]:
```
str = "Monty Python and the flying circus!"

print(str)          # Prints complete string
print(str[0])       # Prints first character of the string
print(str[2:5])     # Prints characters starting from 3rd to 5th
print(str[2:])      # Prints string starting from 3rd character
print(str * 2)      # Prints string two times
print(str + "TEST") # Prints concatenated string
```

```
Monty Python and the flying circus!
M
nty
nty Python and the flying circus!
Monty Python and the flying circus!Monty Python and the flying circus!
Monty Python and the flying circus!TEST
```

Python data type: Slicing

- Part of a list/string using the syntax [start:stop], which e ween i

```
In [16]: # slicing
         l = [1, 2, 3, 4, 5, 6, 7, 8]
         s = "This is a string."

         print(l[2:5])
         print(s[0:6])

         [3, 4, 5]
         This i
```

```
In [17]: # [start:stop:step]
         print(s[2:10:1])
         print

         # These two are equal
         print(s[0:10:2])
         print(s[:10:2])

         is is a
         Ti sa
         Ti sa
```

```
In [18]: print(s[10:2:-1])
         print

         # reversing a list or string
         print(s[::-1])

         s a si s
         .gnirts a si sihT
```

- Define a step for slicing as in [start:stop:step]
- Slicing Step can be a negative value as well (-ve direction)

Branching in Python

- Branching
  - Selection of a code block to run or not
  - This is for creating different paths of execution in a program
  - Structure:

How to create code block?
Defined based on indentation

Indentation

```
statement1 = True
statement2 = False

if statement1:
    print("First statement is true")
elif statement2:
    print("Second statement is true")
else:
    print("Both statements are false")

print("This is printed outside the if-else block")
```

Branching in Python

- Branching
  - Selection of a code block to run or not
  - This is for creating different paths of execution in a program
  - Structure:

How to create code block?
Defined based on indentation

Indentation

Boolean

```
statement1 = True
statement2 = False

if statement1:
    print("First statement is true")
elif statement2:
    print("Second statement is true")
else:
    print("Both statements are false")

print("This is printed outside the if-else block")
```

Branching in Python

- Branching
  - Selection of a code block to run or not
  - This is for creating different paths of execution in a program
  - Structure:

```
statement1 = True
statement2 = False

if statement1:
    print("First statement is true")
elif statement2:
    print("Second statement is true")
else:
    print("Both statements are false")

print("This is printed outside the if-else block")
```

Boolean

Branching in Python: Output

```python
statement1 = True
statement2 = False

if statement1:
    print("First statement is true")
elif statement2:
    print("Second statement is true")
else:
    print("Both statements are false")

print("This is printed outside the if-else block")
```

```
First statement is true
This is printed outside the if-else block
```

Change the values of statement1 and statement2 to different Boolean values and observe the output.

Branching in Python: Code example 2

Please try with these code examples and try to understand why the outputs are different.

```
statement1 = False

if statement1:
    print("printed if statement1 is True")

    print("still inside the if block")
```

```
if statement1:
    print("printed if statement1 is True")

print("now outside the if block")
```

Iterations

- The iterations is used to re-execute the same code block for a specified number of times.
- This is important to access elements of objects like lists, arrays and tuples.
- Two structures for doing iterations in Python are:

For loops

The for loop runs for a fixed amount of iterations:

Code example #1

```
l = [1, 2, 3, 4, 5]
for item in l:
    print(item)
```

Structure of loop

Code block under the loop

While Loops

The following is an example on how to use a while loop:

Code example #4

```
i = 0

while i < 5:
    print(i),
    i += 1

print
print("done") # Note that this is printed outside the loop
```

Iterations: For loops

Code example #1

```
l = [1, 2, 3, 4, 5]
for item in l:
    print(item)
```

You will get the following output:

```
1
2
3
4
5
```

Code example #2

```
# range(start, stop) creates a range of values from start to stop-1
x = list(range(2, 6))
print("Initial list: {}".format(x))

for idx, item in enumerate(x):
    x[idx] = item**2

print("The new list: {}".format(x))
```

You will get the following output:

```
Initial list: [2, 3, 4, 5]
The new list: [4, 9, 16, 25]
```

During each step of the for loop, $enumerate(x)$ iterates through the list and store the index in idx and value in $item$.

Iterations: While loops

Code example #4

```
i = 0

while i < 5:
    print(i),
    i += 1

print
print("done") # Note that this is printed outside the loop
```

You will get the following output:

```
0 1 2 3 4
done
```

Your task

- Define a string with value equal to “Python is fun!”. How can you print every third character of it?
- Create a list containing five integers. Modify each even index element by adding it with previous index element. Print the list before and after modification.
- Create a tuple with three random integers. Now try to swap the first and the last element. What is the output?
- Create a dictionary with the key/values: 0/'we', '1'/'love', 'last':'python programming'. Try to print each value specifying the keys.

Thank You.