Python Programming Continued...

Vivek M R

1) Tuples

- **Ordered sequence** of elements of different data types, such as integer, float, string, list or even a tuple.
- Enclosed in **parenthesis**, accessed using index values, starting from **0**.
- Immutable data type (elements cannot be changed once created)

```
>>> tuple1 = (1, 2, 3, 4, 5) #tuple of integers
>>> tuple1
(1, 2, 3, 4, 5)

>>> tuple2 =('Economics', 87, 'Accountancy', 89.6) #tuple of mixed data
>>> tuple2
('Economics', 87, 'Accountancy', 89.6)
```

Tuples - Operations

- **Accessing elements**

```
>>> tuple1 = (1,2,3,4,5)
>>> tuple1[0]  #returns the first element of tuple1
1
>>> tuple1[-1] #accessing elements from right
5
```

- **Concatenation of tuples**

```
>>> tuple1 = (1,3,5,7,9)
>>> tuple2 = (2,4,6,8,10)
>>> tuple1 + tuple2
(1, 3, 5, 7, 9, 2, 4, 6, 8, 10)
```

Tuples - Operations

- **Slicing**
  ```
  >>> tuple1 = (10,20,30,40,50,60,70,80)
  >>> tuple1[2:7] #elements from index 2 to index 6
  (30, 40, 50, 60, 70)
  >>> tuple1[:5] #slice starts from zero index
  (10, 20, 30, 40, 50)
  ```

- **Counting**
  ```
  >>> tuple1.count(10) #Returns the number of times the given element appears in the tuple
  1
  ```

- **Other Functions:** len(), index(), sorted(), min(), max(), sum()

2) Dictionaries

- mapping between a set of keys and a set of values.
- The key-value pair is called an **item** separated through colon (:).
- **Unsorted, mutable**.
- **Keys are unique** and should be of any immutable data type, i.e., number, string or tuple.

```
>>>dict3 = {'Mohan':95,'Ram':89,'Suhel':92, 'Sangeeta':85}
>>> dict3
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85}
>>>dict3['Ram']
89
```

Dictionaries - Operations

Adding a new item

```
>>> dict1 = {'Mohan':95,'Ram':89,'Suhel':92, 'Sangeeta':85}
>>>dict1['Meena'] = 78
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 92, 'Sangeeta': 85, 'Meena': 78}
```

Modifying an existing item

```
#Marks of Suhel changed to 93.5
>>> dict1['Suhel'] = 93.5
>>> dict1
{'Mohan': 95, 'Ram': 89, 'Suhel': 93.5, 'Sangeeta': 85, 'Meena': 78}
```

Dictionaries - Operations

- **Traversing**

```
for key in dict1:
        print(key,':',dict1[key])
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

```
for key,value in dict1.items():
        print(key,':',value)
Mohan: 95
Ram: 89
Suhel: 92
Sangeeta: 85
```

3) File Handling

- Files - named location on a secondary storage
- Types- text file and binary file.
- **Opening**

  ```
  >>>file_object= open(file_name, access_mode)
  #access_modes = r, w, a
  ```
- **Closing**

  ```
  >>> file_object.close()
  ```
- **Writing**
- write() - for writing a single string

  ```
  >>> myobject.write("Hey I have started using files in Python\n")
  ```
- writelines() - for writing a sequence of strings

  ```
  >>> lines = ["Hello\n", "Writing multiline strings\n", "This is the third line"]
  >>> myobject.writelines(lines)
  ```

File Handling

- **Reading**
  ```
  >>> file_object.read(10) # read first 10 characters
  'Hello ever'

  >>>myobject.readline(10) #read line upto 10 characters
  ‘Hello ever'

  >>> print(myobject.readlines()) # read all lines
  ['Hello \n', 'Writing multiline strings\n', 'This is the third line']
  ```

- tell(), seek()

File Handling - Pickle Module

- deals with binary files.
- **Dumping(Writing)**

```
import pickle
listvalues=[1,"Geetika",'F', 26]
fileobject=open("mybinary.dat", "wb") #wb-write binary
pickle.dump(listvalues,fileobject)
```

- **Loading (Reading)**

```
print("The data that were stored in file are: ")
fileobject=open("mybinary.dat","rb")
objectvar=pickle.load(fileobject)
print(objectvar) # to print loaded data
```

4) Exception Handling

- Handling unexpected events
- Exceptions are errors that get triggered automatically.
- Divide by zero, open file that does not exist etc.

```
>>> print(50/0) #ZeroDivisionError
>>> print(var+10) #NameError
```

Handling Exceptions:

try:

[program statements where exceptions might occur]

except [exception-name]:

[code for exception handling if the exception-name error is encountered]

Exception Handling

```
 x = 5
 y = "hello"
 z = x + y
```

o/p: TypeError: unsupported operand type(s) for +: 'int' and 'str'

```
x = 5
y = "hello"
try:
        z = x + y
except TypeError:
   print("Error: cannot add an int and a str")
```

o/p: Error: cannot add an int and a str

Exception Handling

- **Catching Specific Exception**

```
def fun(a):
   if a < 4:
      b = a/(a-3)
   print("Value of b = ", b)
try:
   fun(3) #comment this line and see
   fun(5)
except ZeroDivisionError:
   print("ZeroDivisionError Occurred and Handled")
except NameError:
   print("NameError Occurred and Handled")
```

Exception Handling

- **Try with Else Clause**

```
def AbyB(a , b):
   try:
      c = ((a+b) / (a-b))
   except ZeroDivisionError:
      print ("a/b result in 0")
   else:
      print (c)
AbyB(2.0, 3.0)
AbyB(3.0, 3.0)
```

o/p: -5.0

```
        a/b result in 0
```