Unit 6 :File Handling and Dictionaries

Syllabus :

Files: Introduction, File path, Types of files, Opening and Closing files, Reading and Writing files. Dictionary method.

Dictionaries- creating, assessing, adding and updating values.

Introduction to File Handling :

- files are nothing but the collection of data.
- Files can be stored on a secondary memory device like hard disks, pen-drives or CDs.
- Files are loaded in main memory(hard disk) or RAM
- Once the data is stored in computer file,one can retrieve it and use it as per the requirement

Advantage of storing data in file

- File is stored In computer hard disk permanently. Even though computer switched off ,the file and it's data is not removed.
- It is possible to update, modify the data whenever required.
- The stored data can be used by anybody whenever required

  ex- A file containing student data can be used by department,

  office, staff
- File can be used to store huge amount of data

  example- voters list, data of all the students in a college

File Path

- Files are stored in secondary memory.
- Each file has a path which helps system to locate the file.
- In Windows path starts with drive name. (Ex c:\, d:\).

Example : C:\Users\Gamers>d:

D:\>cd python code

D:\python code>cd Unit 1

D:\python code\Unit 1>addition.py

- In Linux or allied operating systems, path starts from root (/) directory. (Ex. /home,/user).

Types of Files

- Python supports two types of files.

The first one is a text file that store data in the form of text and readable by humans and computers.

The second one is binary file that store binary data and readable by computer only

1. **Text Contents (Text Files) :**

- Here files have all data which is text.
- Text can be stored in various ways.
- Simplest files are .txt files
- Also famous document (.docx) is also text.

Some example formats are :

a. Text Files (.txt)
b. Document Files (.docx)
c. Pdf Files (.pdf)
d. Database Files (.csv)

2. Binary Contents (Binary Files) :

- Here files have binary data.
- Various image, video, and audio data can be stored here in binary form.
- Reading these files require special handling.
- Here data stored is processed byte by byte.
- Binary files have various encodings.
- Encoding enables compressed or un-compressed storage.

- Additionally binary files include compressed files.
- Compressed files store data in minimal space.

a. Image Files (.jpeg, .png)
b. Video Files (.mp4)
c. Audio Files (.mp3)
d. Compressed Files (.zip, .rar)

Opening and Closing a File (in Python) :

- File contents need to be in RAM for accessing.
- So, file opening operation gets contents in RAM.
- Also opening the file returns a reference to file.
- File reference is used in different operations on file.
- File reference stores current position to read or write.
- There are two ways to open and close the files in python.

File Handling :

1. **Open and Close :**

Open file

Here Open() function is used to open the file. This function accepts the ‘filename’ and open mode to open the fileing”

Example:

File handler=open(“filename”, ”open mode”, ”buffering”)

F=open(“myfile.txt”,”w”)

F is file handler or file object

Myfile.text is the name and type of file

w is the write-mode in which file is open .

Buffering is the optional temporary integer which sets the size of memory for the file.

File open Modes & Description

r

Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.

r+

Opens a file for both reading and writing. The file pointer placed at the beginning of the file.

w

Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.

w+

Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.

a

Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.

a+

Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

x

To open the file in exclusive creation mode. The file cration failed if file already exists.

Close file

The file which is open must be closed using close() function/method. If the file is not closed then data in it may be corrupted or deleted I some cases. Also the memory occupied by the file is not freed.

File_ref.close()

f.close()

File Handling :

- In below example, data.csv file is opened using OPEN function with “r” access mode.
- File is closed using CLOSE function using “f” as file reference.

```
f=open("data.csv","r")
f.read()
f.close()
```

File handling methods

Read()

Write()

Writelines()

Readlines()

List()

Seek()

Tell()

```
#python program to read and display it’s content
f=open('d:\\test.txt',"r")  #open file in read mode
print(f.name)
print(f.mode)
print(f.read()) #read file and display it’s contents
 f.close()

# Python code to illustrate read() mode character wise
file = open(" d:\\test.txt ", "r")
print(file.read(6))
f.close()
```

```
file = open(‘d:\\test1.txt ','w')
file.write("This is the write command")
file.write("It allows us to write in a particular file")
file.close()

f = open('d:\\test1.txt ','w+')
f.write("This is the write command")
f.write("\nIt allows us to write in a particular file")
f.write("\n\n\nPython programming")
f.close()
```

```
#python program to write multiple lines in a text file
fo= open("d:\\test3.txt","w")
data=["welcome to python programming","\n It is fun",
        "\nPython is to learn","\nAlso it is powerful programming language"]
fo.writelines(data)
fo.close()
```

```
# Python code to illustrate append() mode
file = open('d:\\test1.txt','a')
file.write("\nThis will add this line")
file.close()

# Python code to illustrate with()
with open("d:\\test.txt") as file:
    data = file.read()

# Python code to illustrate with() alongwith write()
with open("d:\\test.txt", "w") as f:
    f.write("Hello World!!!")
print(f.read())
```

Readline() method

Readline() method is used to print all the lines in program.

```
f= open(‘text4.txt’,’r‘)
print(f.readlie())
print(f.readlie())
f.close()
```

List() method

The list() method used to disply contents of file as a list

```
#python program to write multiple lines in a text file
fo= open(“d:\\test3.txt”,”w”)
data=[“welcome to python programming”,”\n It is fun”,
        “\nPython is easy to learn”,”\nAlso it is powerful programming language”]
fo.writelines(data)
fo.close()
```

```
#program to display the contents of the file using for loop
f=open('D:\\test3.txt','r')
For line in f:
    print(line)
f.close()

#output
```

Seek() and tell() method

Seek method() used to change file pointer position and tell() method returns the current position of file pointer

file_object.seek(offset[,whence])

- **offset** – This is the position of the read/write pointer within the file.
- **whence** – This is optional and defaults to 0 which means absolute file positioning, other values are 1 which means seek relative to the current position and 2 means seek relative to the file's end.

```
#Python program using seek() and tell() methods
F=open('d:\\test3.txt','r')
print("\tthe contents of the file are…")
print(F.read())

Print('the current position of file pointer is: ')
Print(F.tell())

F.seek(0)
Print('Now the position of file cursor is:')
Print(F.tell())
```

Unit 6 :File Handling and Dictionaries

Syllabus :

Files: Introduction, File path, Types of files, Opening and Closing files, Reading and Writing files. Dictionary method.

Dictionaries- creating, assessing, adding and updating values.

Python Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key

Python Dictionary Methods

popitem()

Removes the last inserted key-value pair

setdefault()

Returns the value of the specified key. If the key does not exist: insert the key, with the specified value

update()

Updates the dictionary with the specified key-value pairs

values()

Returns a list of all the values in the dictionary

Clear Method

Definition and Usage

The clear() method removes all the

elements from a dictionary.

Syntax

dictionary.clear()

Parameter Values

No parameters

print(car)

o/p:{ }

Example

Remove all elements from the car list:

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
car.clear()
print(car)
```

o/p:{ }

Copy Method

Definition and Usage

The copy() method returns a copy of the specified dictionary.

Syntax

dictionary.copy()

Parameter Values

No parameters

Copy the car dictionary:

```
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

x = car.copy()
print(x)
```

o/p:{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}

Definition and Usage

The fromkeys() method returns a dictionary with the specified keys and the specified value.

Syntax

dict.fromkeys(*keys*, *value*)

Parameter Values

Parameter Description *keys* Required. Default value of a key is None
Create a dictionary with 3 keys, all with the value 0:

```
x = ('key1', 'key2', 'key3')
y = 0

thisdict = dict.fromkeys(x, y)
print(thisdict)
```

o/p:['key1': 0, 'key2': 0, 'key3': 0]

Same example as above, but without specifying the value:

```
x = ('key1', 'key2', 'key3')

thisdict = dict.fromkeys(x)

print(thisdict)
```

O/P:['key1': None, 'key2': None, 'key3': None]

Get Method

- Definition and Usage
- The get() method returns the value of the item with the specified key.
- Syntax
- *dictionary*.get(*keyname*, *value*)
- Parameter Values
- ParameterDescription*keyname*Required. The keyname of the item you want to return the value from*value*Optional. A value to return if the specified key does not exist.
  Default value None
  Get the value of the "model" item:
- ```
  car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
  }

  x = car.get("model")

  print(x)
  ```
- O/P:Mustang
  ```

Item Method

- Definition and Usage
- The items() method returns a view object. The view object contains the key-value pairs of the dictionary, as tuples in a list.
- The view object will reflect any changes done to the dictionary, see example below.
- Syntax
- *dictionary*.items()
- Parameter Values
- No parameters
- More Examples
- Example
- When an item in the dictionary changes value, the view object also gets updated:
- ```
  car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
  }

  x = car.items()

  car["year"] = 2018

  print(x)
  ```

- O/p:dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 2018)])
  ```

- Definition and Usage
- The keys() method returns a view object. The view object contains the keys of the dictionary, as a list.
- The view object will reflect any changes done to the dictionary, see example below.
- Syntax
- *dictionary*.keys()
- Parameter Values
- No parameters
- When an item is added in the dictionary, the view object also gets updated:
- ```
  car = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
  }

  x = car.keys()

  car["color"] = "white"

  print(x)
  ```
- O/p:dict_keys(['brand', 'model', 'year', 'color'])
-
  ```

POP Method

- **Definition and Usage**
- The pop() method removes the specified item from the dictionary.
- The value of the removed item is the return value of the pop() method, see example below.
- **Syntax**
- *dictionary*.pop(*keyname*, *defaultvalue*)
- Parameter Values

Parameter	Description
keyname	Required. The keyname of the item you want to remove
defaultvalue	Optional. A value to return if the specified key do not exist. If this parameter is not specified, and the no item with the specified key is found, an error is raised

- **Remove "model" from the dictionary:**
- car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
  }

  car.pop("model")

  print(car)
- O/p:{'brand': 'Ford', 'year': 1964}
-