# Tutorial_Session4

March 17, 2018

## 1 FILE HANDLING AND ERROR HANDLING

### 1.1 File Handling

- Files provide a means of communication between the program and the outside world.
- A file is a stream of bytes, comprising data of interest.
- **File Operations:** > * Reading from a file > * Writing to a file > * Append to a file > * Rename a file > * Delete a file

#### 1.1.1 Opening a File

- Built-in function used: **open()**
- This function takes the name of the file as the first argument. The second argument indicates the mode for accessing the file.
- Modes for opening a file:
  - read(r): to read the file
  - write(w): to write to the file
  - append(a): to write at the end of the file.

Syntax:

```
f = open(file_name, access_mode)
```

- Opening a non-existent file in w or a mode creates a new file with the given name
- However, opening a non-existent file in r mode leads to an error.

** Accessing attribute of an instance of class** * To specify an attribute of a class (or class instance), we write the name of the class (or class instance) followed by a dot, followed by the name of that attribute. The method **lower** defined in class **str** has been invoked for the object **name**.

```
In [8]: f = open('PYTHON','w')
```

### 1.1.2 Writing to a File

- Built-in function used: **write()**
- To use write function, specify name of the file object, followed by the dot operator (.), followed by the name of the function.
- Note that apart from writing the given data into a file, the function write also returns the number of characters written into the file.

```
In [9]: f.write('''Python:
        Python is Simple.
        Simple syntax.''')

Out[9]: 40
```

### 1.1.3 Reading a File

- Built-in function used: **read()**
- To use read function, specify name of the file object, followed by the dot operator (.), followed by the name of the function
- The read() function retrieves the contents of the entire file.

```
In [11]: f.read() #attempt to read file without read mode on.


         ---------------------------------------------------------------------------

         UnsupportedOperation                      Traceback (most recent call last)

         <ipython-input-11-99d52ada1abc> in <module>()
        ----> 1 f.read() #attempt to read file without read mode on.


         UnsupportedOperation: not readable


In [16]: f = open('PYTHON','r')
         f.read()

Out[16]: 'Python:\nPython is Simple.\nSimple syntax.'

In [17]: f.read() # reading a file reached EOF yields empty string

Out[17]: ''
```

We can read a fixed number of bytes from the file by specifying the number of bytes as the argument to read function.

```
In [18]: f = open('PYTHON','r')
         f.read(6)

Out[18]: 'Python'
```

**Displaying the multi-line string using the print function**

```
In [19]: f = open('PYTHON','r')
         print(f.read())
```

```
Python:
Python is Simple.
Simple syntax.
```

### 1.1.4 Function close

- When a file is no longer required for reading or writing, it should be closed by invoking the function close

```
In [20]: f.close()
```

The function close also saves a file, which was opened for writing. Once a file is closed, it cannot be read or written any further unless it is opened again and an attempt to access the file results in an I/O (input/output) error:

```
In [21]: f.write('Python was developed by Guido Van Rossum in 1991.')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-21-12657766d5ec> in <module>()
----> 1 f.write('Python was developed by Guido Van Rossum in 1991.')


ValueError: I/O operation on closed file.
```

### 1.1.5 Functions readline and readlines

**readline** function: reads a stream of bytes beginning the current position until a newline character is encountered

```
In [22]: f = open('PYTHON','r')
         line = f.readline()
         print('line1:', line)
         line = f.readline()
         print('line2:', line)
         line = f.readline()
         print('line3:', line)
         line = f.readline()
         print('line4:', line)
         f.close()
```

```
line1: Python:

line2: Python is Simple.

line3: Simple syntax.
line4:
```

**readlines** function: returns all the remaining lines of the file in the form of a list

```
In [23]: f = open('PYTHON','r')
         lines = f.readlines()
         print(lines)
         f.close()
```

```
['Python:\n', 'Python is Simple.\n', 'Simple syntax.']
```

### 1.1.6 Function writelines

**writelines** function: takes a list of lines to be written in the file as an argument

```
In [24]: f = open('PYTHON', 'w')
         description = ['Python:\n', 'Python is simple.\n']
         f.writelines(description)
         f.close()

         f = open('PYTHON', 'r')
         print(f.read())
         f.close()
```

```
Python:
Python is simple.
```

### 1.1.7 Functions seek and tell

- **seek()**: to reach desired position in a file Syntax:

  seek(offset) # offset indicates the number of bytes to be moved # Returns the new absolute position

- **tell()**: to find current position of the file object

```
In [25]: f = open('PYTHON','r')
         print(f.readline())
         f.seek(0) # Changes the current file position to the begining of the file
         print('After seeking file pointer in the beginning\n')
         print(f.readline())
```

```
Python:

After seeking file pointer in the beginning

Python:
```

```
In [26]: f.seek(0)
         f.read(4)

Out[26]: 'Pyth'

In [27]: f.tell()

Out[27]: 4
```

### 1.2 Error Handling

- Error occurs when something goes wrong.
- The errors in Python programming may be categorized as:
  - Syntax Errors
  - Exceptions

### 1.3 Syntax Error

A syntax error occurs when a rule of Python grammar is violated.

```
In [53]: print('Hello)   # missing quote mark at the end of the string


           File "<ipython-input-53-5741a0378b6f>", line 1
         print('Hello)   # missing quote mark at the end of the string
                                                                      ^
     SyntaxError: EOL while scanning string literal


In [54]: for i in range(0, 10)   # missing colon in the for statement


           File "<ipython-input-54-a1c3f00d6a60>", line 1
         for i in range(0, 10)   # missing colon in the for statement
                                                                     ^
     SyntaxError: invalid syntax
```

## 1.4 Exceptions

- Errors that occur at execution time.

- These errors disrupt the flow of the program at a run-time by terminating the execution at the point of occurrence of the error.
- We have noticed that whenever an exception occurs, a Traceback object is displayed which includes error name, its description, and the point of occurrence of the error such as line number.

### 1.4.1 NameError

This exception occurs whenever a name that appears in a statement is not found globally.

```
In [28]: marks = Input('Enter your marks')
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-28-e2d0c49b2ae3> in <module>()
----> 1 marks = Input('Enter your marks')


NameError: name 'Input' is not defined
```

```
In [30]: price=10
         print(Price)
```

```
---------------------------------------------------------------------------

NameError                                 Traceback (most recent call last)

<ipython-input-30-f01eed036878> in <module>()
  1 price=10
----> 2 print(Price)


NameError: name 'Price' is not defined
```

### 1.4.2 TypeError

This exception occurs when an operation or function is applied to an object of inappropriate type.

```
In [31]: 'sum of 2 and 3 is ' + 5
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-31-135c7253899e> in <module>()
----> 1 'sum of 2 and 3 is ' + 5


TypeError: must be str, not int
```

### 1.4.3 ValueError

This exception occurs whenever an inappropriate argument value, even though of correct type, is used in a function call.

```
In [32]: int('Hello')
```

```
---------------------------------------------------------------------------

ValueError                                Traceback (most recent call last)

<ipython-input-32-6765ce49acfe> in <module>()
----> 1 int('Hello')


ValueError: invalid literal for int() with base 10: 'Hello'
```

### 1.4.4 ZeroDivisionError

This exception occurs when we try to perform numeric division in which the denominator happens to be zero.

```
In [33]: 78/(2+3-5)
```

```
---------------------------------------------------------------------------

ZeroDivisionError                         Traceback (most recent call last)

<ipython-input-33-5c961f5673b5> in <module>()
----> 1 78/(2+3-5)


ZeroDivisionError: division by zero
```

### 1.4.5 OSError

This exception occurs whenever there is an error related to input/output.

```
In [34]: # opening a non-existent file for reading or reading a file opened in write mode
         f = open('passwordFile.txt')
```

```
---------------------------------------------------------------------------

FileNotFoundError                         Traceback (most recent call last)

<ipython-input-34-880578fe47b9> in <module>()
  1 # opening a non-existent file for reading or reading a file opened in write mode
----> 2 f = open('passwordFile.txt')


FileNotFoundError: [Errno 2] No such file or directory: 'passwordFile.txt'
```

### 1.4.6 IndexError

This exception occurs whenever we try to access an index that is out of a valid range.

```
In [35]: colors = ['red', 'green', 'blue']

In [36]: colors[4]
```

```
---------------------------------------------------------------------------

IndexError                                Traceback (most recent call last)

<ipython-input-36-1f89a8f1c888> in <module>()
----> 1 colors[4]


IndexError: list index out of range
```

## 1.5 Problem: Copying contents of a file to another file

```
In [41]: import os

         def fileCopy(file1,file2):
             '''
             Objective: This method copies the contents in a file to another file
             Input Parameter: file1 - input file, file2 - output file
             Return Value: None
```

```
        '''
        '''
        Approach:
        Read input from file1, line by line and copy to file2 until
        null string is returned on reading
        '''
        f1 = open(file1, 'r')
        f2 = open(file2, 'w')
        line = f1.readline()
        while line != '':
            f2.write(line) #write the line from f1 with additional newline
            line = f1.readline()
        f1.close()
        f2.close()

    def main():
        '''
        Objective: To call function fileCopy to copy contents in a file to another file.
        Input Parameter: None
        Return Value: None
        '''
        fileName1=input('Enter the source file name: ')
        fileName2=input('Enter the destination file name : ')
        fileCopy(fileName1, fileName2)

    if __name__ == '__main__':
        main()
```

```
Enter the source file name: studentMarks
Enter the destination file name : test
```

### 1.6 Computing moderated marks

- The file studentMarks contains the student data that includes roll number (rollNo), name (name), and marks (marks) for each student.
- The data about each student is stored in a separate line. Sample data in the file is shown below:

```
4001,Nitin Negi,75
4002,Kishalaya Sen,98
4003,Kunal Dua,80
4004,Prashant Sharma,60
4005,Saurav Sharma,88
```

- We define addPerCent as the percentage of maxMarks that should be added to the marks obtained to get the moderated marks, subject to the upper limit of maxMarks.

- The output file moderatedMarks containing moderated marks of the students

    1. Open file studentMarks in read mode.
    2. Open file moderatedMarks in write mode.
    3. Read one line of input (line1) from studentMarks.
    4. while (line1 != ''):

```
> Compute moderated marks and write one line of output in the file
```

moderatedMarks. > Read one line of input (line1) from studentMarks.

### 1.7 Problem: Compute moderated marks based on user input

```
In [1]: import sys
        def computeModeratedMarks(file1, file2, addPercent):
            '''
            Objective: To compute moderated marks of students
            Input Parameters: file1, file2: file names - string values
                              addPercent  numeric value
            Return Value: None
            Side effect: A new file  file2 of moderated marks is produced
            '''
            try:
                fIn = open(file1, 'r')
                fOut = open(file2,'w')
            except IOError:
                print('Problem in opening the file'); sys.exit()
            line1 = fIn.readline()
            while(line1 != ''):
                sList = line1.split(',')
                try:
                    rollNo = int(sList[0])
                    name = sList[1]
                    marks = int(sList[2])
                except IndexError:
                    print('Undefined Index'); sys.exit()
                except (ValueError):
                    print('Unsuccessful conversion to int'); sys.exit()
                maxMarks= 100
                moderatedMarks = marks+((addPercent*maxMarks) /100)
                if moderatedMarks > 100:
                    moderatedMarks = 100
                fOut.write(str(rollNo) + ',' + name + ',' +\
                str(moderatedMarks) + '\n')
                line1 = fIn.readline()
            fIn.close()
            fOut.close()
        def main():
            '''
```

```
        Objective: To compute moderated marks based on user input
        Input Parameter: None
        Return Value: None
        '''
        import sys
        sys.path.append('F:\PythonCode\Ch09')
        # To compute moderated marks of students
        file1 = input('Enter name of file containing marks:')
        file2 = input('Enter output file for moderated marks:')
        addPercent = int(input('Enter moderation percentage:'))
        computeModeratedMarks(file1, file2, addPercent)
    if __name__=='__main__':
        main()
```

```
Enter name of file containing marks:studentMarks
Enter output file for moderated marks:moderatedMarks
Enter moderation percentage:10
```