Batch A - 2025

Week#3 (Files Handling and Exception Handling)

Important Instructions:

- Read the manual carefully and understand every topics before starting the assignment.
- Follow the deadlines properly; marks will be deducted for late submissions.
- Avoid to use AI Tools and make your own logic.
- Completing all tasks in the assignment is compulsory to receive full marks.

- **Manual Posted Date:**

  17th March, 2025 (Monday)

- **Assignment-3 Deadline:**

  24th March, 2025 (Monday at 11:59pm)

File Read/Write

1. Reading from a Text File

```
with open(r'text.txt') as fileObj:
    content = fileObj.read()
print(content)
```

i. **with open(r'text.txt') as fileObj:**
This line opens the file named "text.txt" in read mode ('r'). The with statement is used for context management, which ensures that the file is properly opened and closed, even if an exception occurs during the operation.

ii. **r'text.txt':**
Specifies the path to the file. The r before the string denotes a raw string literal, which is used to interpret backslashes as literal characters and is commonly used for specifying file paths on Windows.

iii. **content = fileObj.read():**
This line reads the contents of the opened file (fileObj) and assigns the content to the variable content. The read() method reads the entire content of the file as a single string.

```
with open(r'text.txt') as fileObj:
    content = fileObj.readlines()
print(content)
```

```
mylist = ["hello","test"]
with open(r'C:\Users\sohail\Desktop\data\demo.txt') as fileObj:
    lines = fileObj.readlines()
    mylist  = mylist + lines
    print(mylist)
```

*readlines() – This method is used to read all the lines from a text file and return them as a list of strings.

2. Writing to a Text File

```
with open(r'C:\Users\sohail\Desktop\data\demo100.txt','w') as fileObj:
    fileObj.write("I love programming!")
```

i. **with open(r'C:\Users\sohail\Desktop\data\demo100.txt', 'w') as fileObj:**
This line uses the open() function to open the file located at the specified path in write mode ('w').

ii. **fileObj.write("I love programming!")**:
This line writes the string "I love programming!" into the opened file (fileObj) in write mode.
Since the file is opened in write mode ('w'), if the file already exists, its previous content will be overwritten. If the file does not exist, a new file will be created.

iii. If file is not present with that name, it will create file.

```
with open(r'C:\Users\sohail\Desktop\data\demo100.txt','a') as fileObj:

    fileObj.write("Hello, I love programming!\n")

with open(r'C:\Users\sohail\Desktop\data\demo100.txt') as fileObj:
    content = fileObj.readlines()
    print(content)
```

*a – appends to file.

3. Reading & Writing to a Text File

```
# r+ for both reading and writing
with open('demo1.txt','r+') as fileObj:
    fileObj.write("I love programming!\n")

    fileObj.seek(0)
    content = fileObj.readlines()
    print(content)
```

4. Checking if file/directory (folder) is present

```
import  os;

p = os.path.isfile("demo.txt") #to check if file is present at particular path
p1 = os.path.isdir("C:\\Windows")#to check if dir. is present at particular path

dirList = os.listdir("C:\\Windows") # to print all directories in windows
print(p)
print(p1)
print(dirList)
```

5. JSON Filling

Python has a built-in package called JSON, which can be used to work with JSON data. JavaScript Object Notation (JSON) is a standardized format commonly used to transfer data as text that can be sent over a network. It's used by lots of APIs and Databases, and it's easy for both humans and machines to read.

```
import json

numbers = [2, 3, 5, 7, 11, 13,14]

filename = r'C:\Users\sohail\Desktop\data\numbers.json'
with open(filename, 'w') as f_obj:
    json.dump(numbers, f_obj)
```

```
import json

stu = {
    "name":"Qasim",
    "age": 21, "height":5
}

filename = r'C:\Users\sohail\Desktop\data\numbers2.json'
with open(filename, 'w') as f_obj:
    json.dump(stu, f_obj)
```

```
#if file is not there, it will create that file
import json

numbers = [2, 2, 5, 7, 11, 13]

filename = r'C:\Users\sohail\Desktop\data\numbers10.json'
with open(filename, 'w') as f_obj:
    f_obj.write(str(numbers))
```

```python
# how to read json using json.load
# this will give error because file numbers.json contains dictionary and you
# are loading it in list
import json

data = []
filename = r'C:\Users\sohail\Desktop\data\numbers.json'
with open(filename) as f_obj:
    data = json.load(f_obj)

print(data)
print(data[1])
```

```
---------------------------------------------------------------------------
JSONDecodeError                           Traceback (most recent call last)
<ipython-input-3-0ef5916ad4b0> in <module>
      5 filename = r'C:\Users\sohail\Desktop\data\numbers.json'
      6 with open(filename) as f_obj:
----> 7     data = json.load(f_obj)
      8
      9 print(data)

~\anaconda3\lib\json\__init__.py in load(fp, cls, object_hook, parse_float, parse_int, p
arse_constant, object_pairs_hook, **kw)

    291     kwarg; otherwise ``JSONDecoder`` is used.
    292     """
--> 293     return loads(fp.read(),
    294         cls=cls, object_hook=object_hook,
    295         parse_float=parse_float, parse_int=parse_int,

~\anaconda3\lib\json\__init__.py in loads(s, cls, object_hook, parse_float, parse_int, p
arse_constant, object_pairs_hook, **kw)
    355             parse_int is None and parse_float is None and
    356             parse_constant is None and object_pairs_hook is None and not kw):
--> 357         return _default_decoder.decode(s)
    358     if cls is None:
    359         cls = JSONDecoder

~\anaconda3\lib\json\decoder.py in decode(self, s, _w)
    335
    336         """
--> 337         obj, end = self.raw_decode(s, idx=_w(s, 0).end())
    338         end = _w(s, end).end()
    339         if end != len(s):

~\anaconda3\lib\json\decoder.py in raw_decode(self, s, idx)
    353             obj, end = self.scan_once(s, idx)
    354         except StopIteration as err:
--> 355             raise JSONDecodeError("Expecting value", s, err.value) from None
    356         return obj, end

JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

```python
#this will work correct as you are loading dictionary in dictionary
import json

stu = {}
filename = r'C:\Users\sohail\Desktop\data\numbers2.json'
with open(filename) as f_obj:
    stu = json.load(f_obj)

print(stu)
print(stu["name"])
```

Exception Handling

Python has many built-in exceptions that are raised when your program encounters an error (Something in the program goes wrong).
When these exceptions occur, the Python interpreter stops the current process and passes it to the Calling process until it is handled. If not handled, the program will crash.

```
print("First")
a = 5
b = 0
try:
    c = a/b
    print(c)
except ZeroDivisionError:
    print("Handle ZeroDivisionError")
```

```
First
Handle ZeroDivisionError
```

```
# here there is no error so it will not go in "except"
print("First")
a = 5
b = 10
try:
    c = a/b
    print(c)
except ZeroDivisionError:
    print("Handle ZeroDivisionError")


print("last")
```

```
First
0.5
last
```

```
#except is required with try
print("First")
a = 5
b = 5
try:
    a/b
```

```
  File "<ipython-input-20-d18666dcb13e>", line 9

    ^
SyntaxError: unexpected EOF while parsing
```

```
# in this code if zero division error occurs then it will simply pass and
# do nothing else if error does not occur, it will move to "else" part
print("First")
a = 5
b = 0
try:
    c = a/b
except ZeroDivisionError:
    pass
else:
    print(c)

print("last")
```

```
First
last
```

1. Handling multiple errors/exceptions

```
# here is list index out of range error
print("First")
a = 5
b = 4
d = []
try:
    c = a/b
    e = d[5]
except ZeroDivisionError:
    print("Handle ZeroDivisionError")
else:
    print(c)

print("last")
```

```
First
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-30-a807a8d40bab> in <module>
      5 try:
      6     c = a/b
----> 7     e = d[5]
      8 except ZeroDivisionError:
      9     print("Handle ZeroDivisionError")

IndexError: list index out of range
```

```
# here list index out of range error has been handled
print("First")
a = 5
b = 5
d = []
try:
    c = a/b
    e = d[5]
except ZeroDivisionError:
    print("Handle ZeroDivisionError")
except IndexError:
    print("Handle IndexError")
else:
    print(c)

print("last")
```

```
First
Handle IndexError
last
```

2. Printing error message when error is initially unknown

```
print("First")
a = 5
b = 1
d = []
try:
    c = a/b
    e = d[5]
except Exception as e:
    print("Handle Error == " +str(e))
else:
    print(c)

print("last")
```

```
First
Handle Error == list index out of range
last
```

3. Handling errors when some error are known some are unknown

```
a=5
b="xyz"
try:
    c=a/b
    print(c)
except ZeroDivisionError:
    print("ZDE")

except:
    print("There is some error")
```

```
There is some error
```