CITY UNIVERSITY LONDON

Introduction to programming using Python

Session -2

Matthieu Choplin

matthieu.choplin1@city.ac.uk

http://moodle.city.ac.uk/

Objectives

- To open a file, read/write data from/to a file
- To use file dialogs for opening and saving data
- To read data from a Web resource
- To handle exceptions using the try/except/finally clauses
- To raise exceptions using the raise statements
- To become familiar with Python's built-in exception classes
- To access exception object in the handler

Open a File

We create a **file object** with the following syntax:

```
file = open(filename, mode)
```

Mode	Description
r	Open a file for reading only
r+	Open a file for both reading and writing
w	Open a file for writing only
a	Open a file for appending data. Data are written to the end of the file
rb	Open a file for reading binary data
wb	Open a file for writing binary data

Write to a File: Example

Program that creates a file if it does not exist (an existing file with the same name will be erased) and write in it:

```
def main():
    # Open file for output
    outfile = open("Python_projects.txt", "w")
    # Write data to the file
    outfile.write("Django\n")
    outfile.write("Flask\n")
    outfile.write("Ansible")
    outfile.close() # Close the output file

main() # Call the main function
```

Testing File Existence

```
import os.path

if os.path.isfile("Python_projects.txt"):
    print("Python_projects.txt exists")
```

Read from a File: Example

After a file is opened for reading data, you can use:

- the **read(size)** method to read a specified number of characters or all characters,
- the **readline()** method to read the next line
- the **readlines()** method to read all lines into a list.

Read from a File: Example with read()

```
def main():
    # Open file for input
    infile = open("Python_projects.txt", "r")
    print("Using read(): ")
    print(infile.read())
    infile.close() # Close the input file
main() # Call the main function
```

Read from a File: Example with read(size)

```
def main():
    infile = open("Python_projects.txt", "r")
    print("\nUsing read(number): ")
    s1 = infile.read(4)  # read till the 4th character
    print(s1)
    s2 = infile.read(10)  # read from 4th till 4th+10th
    print(repr(s2))  # a new line is also a character \n
    infile.close()  # Close the input file

main()  # Call the main function
```

Read from a File: Example with readline()

```
def main():
    infile = open("Python_projects.txt", "r")
    print("\nUsing readline(): ")
    line1 = infile.readline()
    line2 = infile.readline()
    line3 = infile.readline()
    line4 = infile.readline()
    print(repr(line1))
    print(repr(line2))
    print(repr(line3))
    print(repr(line4))
    infile.close() # Close the input file

main() # Call the main function
```

Read from a File: Example with readlines()

```
def main():
    # Open file for input
    infile = open("Python_projects.txt", "r")
    print("\n(4) Using readlines(): ")
    print(infile.readlines())  # a list of lines
    infile.close()  # Close the input file

main() # Call the main function
```

Append Data to a File

You can use the 'a' mode to open a file for appending data to an existing file.

```
def main():
    # Open file for appending data
    outfile = open("Info.txt", "a")
    outfile.write("\nPython is interpreted\n")
    outfile.close() # Close the input file

main() # Call the main function
```

Writing/Reading Numeric Data

To write numbers, convert them into strings, and then use the write method to write them to a file. In order to read the numbers back correctly, you should separate the numbers with a whitespace character such as " " (empty string) or '\n' (new line).

Writing/Reading Numeric Data: Example

```
from random import randint

def main():
    # Open file for writing data
    outfile = open("Numbers.txt", "w")
    for i in range(10):
        outfile.write(str(randint(0, 9)) + " ")
    outfile.close() # Close the file
    # Open file for reading data
    infile = open("Numbers.txt", "r")
    s = infile.read()
    numbers = [int(x) for x in s.split()]
    for number in numbers:
        print(number, end = " ")
    infile.close() # Close the file

main() # Call the main function
```

Exercise

Write a program that prompts the user to enter a file and counts the number of occurrences of each letter in the file regardless of case.

Only take the characters of the alphabet, you can get them with the following

```
from string import ascii_lowercase
print(ascii_lowercase)  # abcdefghijklmnopqrstuvwxyz
```

CITY UNIVERSITY LONDON

- Solution

Retrieving Data from the Web

Using Python, you can write simple code to read data from a Website. All you need to do is to open a URL link using the urlopen function as follows:

```
import urllib.request
infile = urllib.request.urlopen('http://example.org/')
html_page = infile.read().decode()
print(html_page)
```

It represents the full HTML of the page just as a web browser would see it

Exercise

- Count each letter from a web page (from the source code of the page)
- You can reuse the code of the previous and try to refactor so that both programs use the same count_letter function

CITY UNIVERSITY LONDON

- Solution

Exception Handling

What happens if the user enters a file or an URL that does not exist? The program would be aborted and raises an error. For example, if you run count_letter.py with an incorrect input:

```
c:\session6\python count_letter.py
Enter a filename: non_existant_file.txt
Traceback (most recent call last):
  File "path_to/count_letter.py", line 18, in  <module>
    main()
  File "path_to/count_letter.py", line 7, in main
    f = open(filename)
FileNotFoundError: [Errno 2]
No such file or directory: 'non_existant_file.txt'

Process finished with exit code 1
```

The try ... except clause

Catching one exception type

```
try:
 <body>
except <ExceptionType>:
 <handler>
```

The try ... except clause

Catching several exception types

```
try:
 <body>
except <ExceptionType>:
 <handler1>
 <handler1>
...
except <ExceptionTypeN>:
 <handlerN>
except:
 <handlerExcept>
else:
<process_else> # will be executed if not exception
finally:
<process_finally> # executed with or without exception
```

Example

```
def main():
    try:
        number1, number2 = int(
          input("Enter two integers,"
                "separated by a comma: "))
        result = number1 / number2
        print("Result is " + str(result))
    except ZeroDivisionError:
        print("Division by zero!")
    except SyntaxError:
        print("A comma may be missing in the input")
    except:
        print("Something wrong in the input")
    else:
        print("No exceptions")
    finally:
        print("The finally clause is executed")
main()
```

Raising Exceptions

You learned how to write the code to handle exceptions in the preceding section. Where does an exception come from? How is an exception created? Exceptions are objects and objects are created from classes. An exception is raised from a function. When a function detects an error, it can create an object of an appropriate exception class and raise the object, using the following syntax:

```
raise ExceptionClass("Something is wrong")
```

Processing Exceptions Using Exception Objects

You can access the exception object in the except clause with the **as** keyword.

```
try:
    number = int(input("Enter a number: "))
    print("The number entered is", number)
except NameError as ex:
    print("Exception:", ex)
```

Using the with statement

It is good practice to use the **with** keyword when dealing with file objects. This has the advantage that the file is properly closed after its suite finishes, even if an exception is raised on the way. It is also much shorter than writing equivalent try-finally blocks:

```
with open('Python_projects.txt', 'r') as f:
    read_data = f.read()

assert f.closed
```

The json file format

- json (Javascript Object Notation) is a lightweight **data interchange format** with which you:
  - dump data ("serialize")
  - load data ("deserialize")

```
import json
serialized_data = json.dumps(
    ['foo', {'bar': ('baz', None, 1.0, 2)}])
print(serialized_data)
deserialized_data = json.loads(a)
print(deserialized_data)
```

Example with a simple rest API

The website https://restcountries.eu/ has a json API for getting information about countries (capital, population...)

Using **urllib** and **json** libraries, let us retrieve the capitals of each country

```
import json
from urllib import request

infile = request.urlopen(
    'https://restcountries.eu/rest/v1/all')
content_as_python_obj = json.loads(infile.read().decode(
for country in content_as_python_obj:
    print(country['capital'])
```

API

- In the previous case, an API (Application Programming Interface) is simply a specification of remote calls exposed to the API consumers
- We are using the API as a service by just calling (doing a GET) its available urls

CITY UNIVERSITY LONDON

Example with the Google map API

```
from urllib import parse, request
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = input('Enter location (q to quit): ')
    if len(address) < 1 or address.lower() == 'q':  # sentinel value, pr
        break
    url = serviceurl + parse.urlencode({'sensor': 'false', 'address': ad
    print('Retrieving', url)
    uh = request.urlopen(url)
    data = uh.read().decode('utf-8')
    print('Retrieved', len(data), 'characters')
    js = json.loads(data)
    if 'status' not in js or js['status'] != 'OK':
        print('==== Failure To Retrieve ====')
        print(data)
        continue
    lat = js["results"][0]["geometry"]["location"]["lat"]
    lng = js["results"][0]["geometry"]["location"]["lng"]
    print('lat', lat, 'lng', lng)
    location = js['results'][0]['formatted_address']
    print(location)
```

Example with the Twitter API using the client Tweepy

1. Navigate to https://apps.twitter.com/
2. Click the button to create a new application
3. Enter dummy data
4. Once the application is created, get the following:
   - consumer_key
   - consumer_secret
   - access_token
   - access_secret

Get tweet with #python

```
import tweepy

consumer_key = 'get_your_own'
consumer_secret = 'get_your_own'
access_token = 'get_your_own'
access_secret = 'get_your_own'

def main():
    auth = tweepy.auth.OAuthHandler(consumer_key,
                                    consumer_secret)
    auth.set_access_token(access_token, access_secret)
    api = tweepy.API(auth)

    tweets = api.search(q='#python')
    for t in tweets:
        print(t.created_at, t.text, '\n')
main()
```