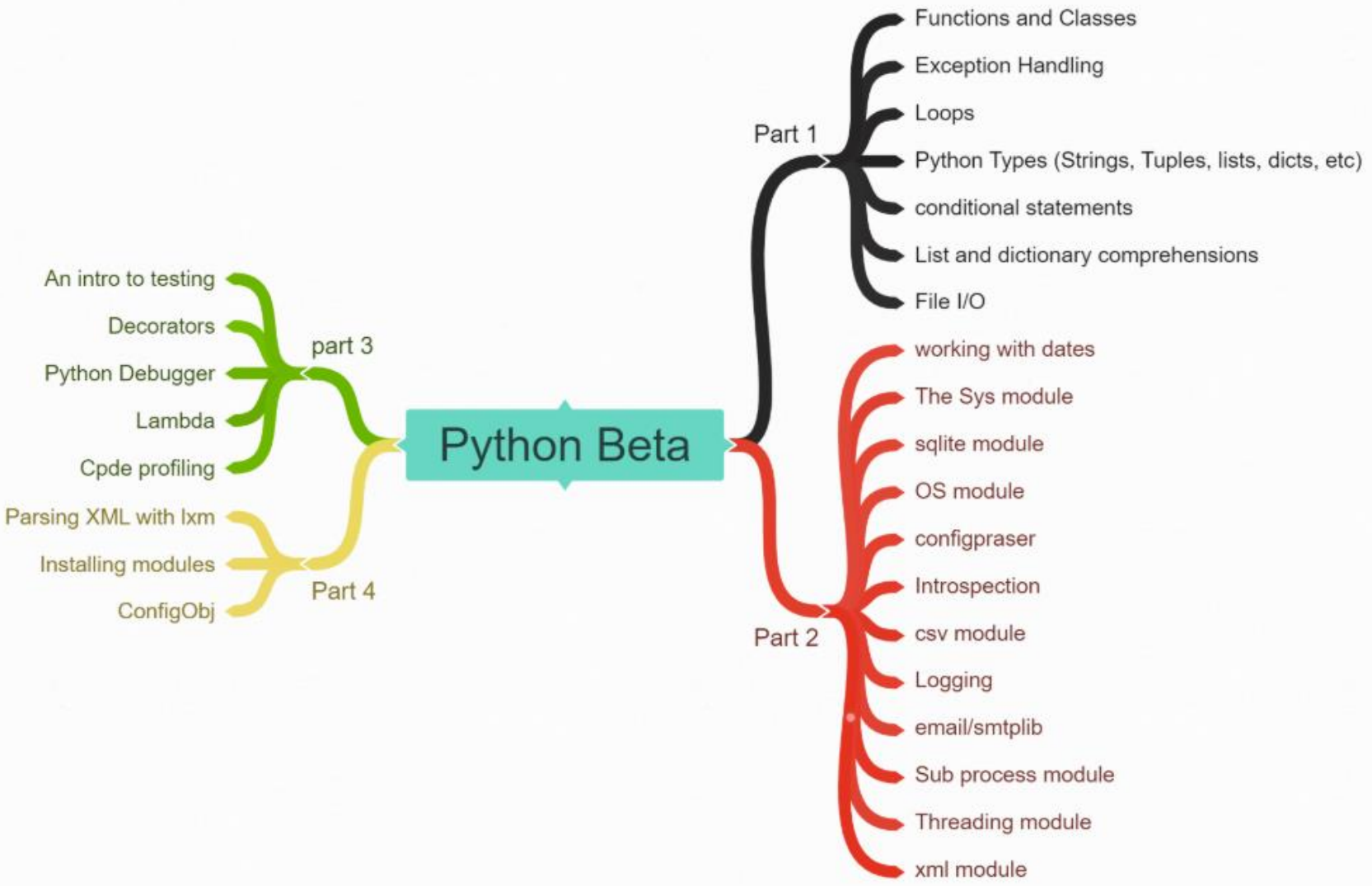
Introduction to Python

Sana Rasheed

AL NAFI,
A company with a focus on education, wellbeing and renewable energy.

Python Beta Track

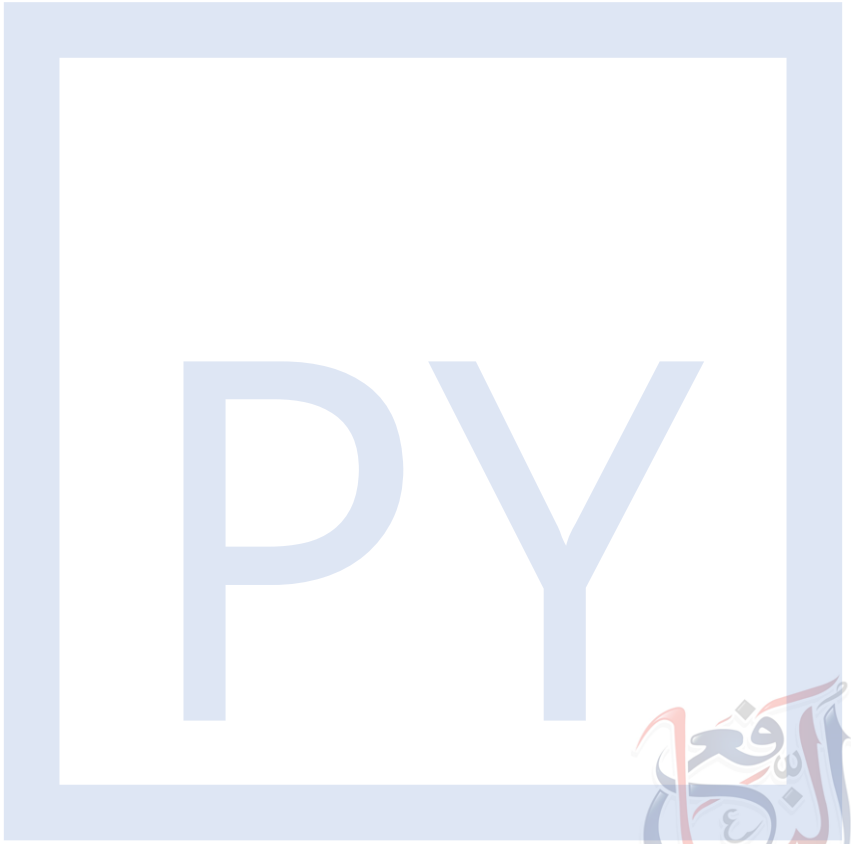


- Basic Introduction
- Built-in Modules
- Testing and Debugging
- Module Installation and Configuration


Section Two

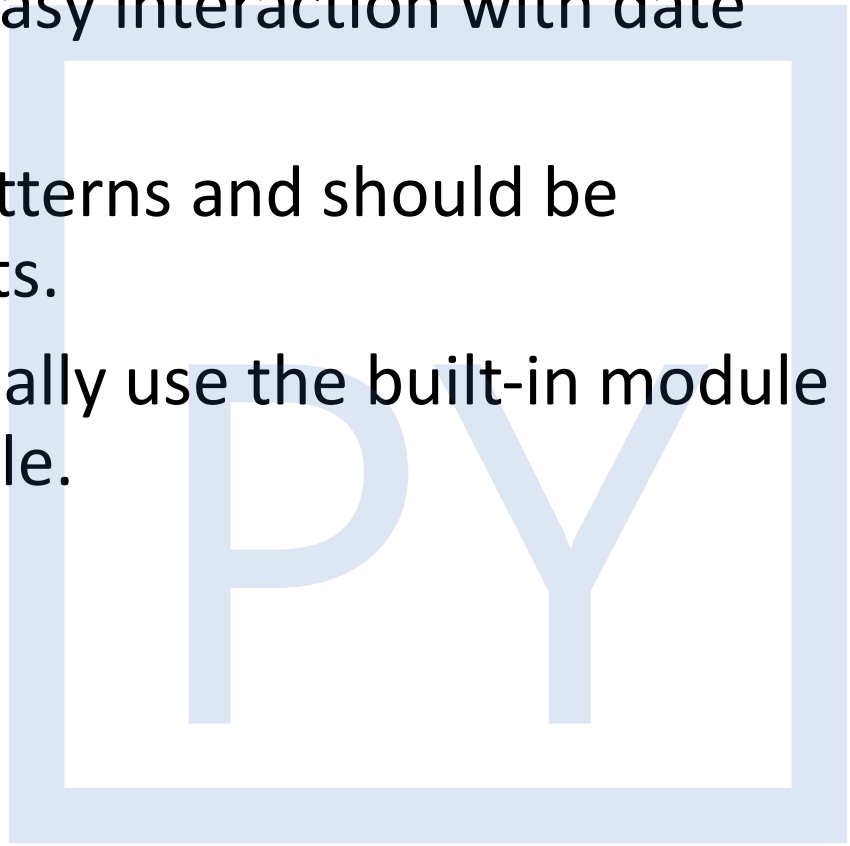
Built-in Modules

- Dates
- OS module
- Introspection
- Sqlite module
- Csv module
- Sub process module
- Requests module



Dates

- Python has a built-in module that allows easy interaction with date and time.
- Since date and time are unique in their patterns and should be handled differently from integers and floats.
- If your data contains dates, you should ideally use the built-in module to handle the dates while reading a data file.
- The module is called datetime.



Dates

- Using the import statement, you can use the functionality of datetime.

```
from datetime import datetime
# prints the current date and time
datetime.now()
```

- We imported datetime class from the datetime module. It's because the object of datetime class can access now() method.
- You can individually access the day, year, month, etc. from the datetime object.

```
my_dt = datetime.now()
my_dt.year # prints current year, 2020
my_dt.month # prints current month, 3
```

Dates

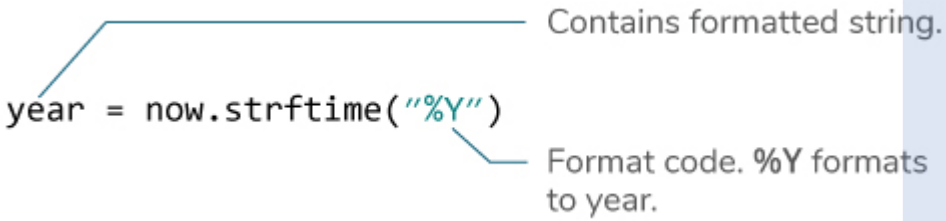
You can use a special method called *strftime* to access particular parts of the datetime object by passing special format specifiers.

```
now = datetime.now() # current date and time

year = now.strftime("%Y")
month = now.strftime("%m")
day = now.strftime("%d")
print("day: ", day)

time = now.strftime("%H:%M:%S")
print("time: ", time)

date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
print("date and time:", date_time).
```

How strftime() works?

- The strftime() method can be used to create formatted strings

- In the above program, %Y, %m, %d etc. are format codes.



- The strftime() method takes one or more format codes as an argument and returns a formatted string based on it.

```
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
```

12/24/2018, 04:59:31

Dates

DIRECTIVE	DESCRIPTION	EXAMPLE
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018

Exercise

- We recommend you to check Python strptime().
- The strptime() method creates a datetime object from a string.
- Example:

```
date_string = "21 June, 2018“
type(date_string)

date_object = datetime.strptime(date_string, "%d %B, %Y")
type(date_object)
```

- Exercise: Create datetime object for “30/01/2021”

OS Module

- The built-in os module helps interact with the operating system through python.
- You can create, access, verify and manipulate file paths using the os module.
- You can also check and set your working directory using this module.
- You can also interact with the command prompt using os module.

```
import os
os.getcwd()
os.listdir() #specify path - (“c:\tmp”)
os.chdir(“c:\\”)
os.mkdir(“c:\\temp\ostempdir”)
```

OS Module

Method	Use
os.path.exists(path)	Check if provided path exists
os.getcwd()	Returns the path for current directory
os.chdir(path)	Changes current directory to “path”
os.path.join(args*)	Joins folder names together to form a valid path
os.path.split(path)	Splits directory into folder names. Path must be in a valid directory format.
os.path.isfile(file_name)	Checks whether the file name points to an actual file in the directory. You can pass path+file_name to test whether the specific file exists in the specified directory.
os.path.isdir(path)	Checks whether path is an actual directory
os.mkdir(path)	Creates a folder in the specified directory. For this method to work, all intermediate folders must exist.

OS Module

Method	Use
os.makedirs(path)	Creates a folder in the specified directory. It also creates intermediate folders if they do not exist.
os.listdir(path)	Lists all contents in the specified path. If path is not passed as an argument or is None, it returns contents of the working directory.
os.remove(file_name)	Deletes the file from the specified directory. If the file does not exist, it raises a FileNotFoundError.
os.removedirs(file)	Deletes the file along with any intermediary folders.

Introspection (Inspect) Module

- Introspection in Python helps determine type of object, its attributes and methods at runtime.
- It is useful for understanding the structure of the data and at times retrieving specific attributes or methods tied to an object.
- The inspect module allows a user to perform these actions on any python object or variable.
- You can use inspection methods on modules, classes, functions and objects.

Basic Methods vs Inspect Module

- You can perform introspection with basic methods or use the module introspection.
- Let's first see examples of internal methods to assess types and available methods of objects.
- In Python everything is an object so we can use *type()* and *dir()* on any variable to determine its type and associated methods respectively.
- We will also use *id()* method which returns a unique ID that Python uses to reference each variable created.

Basic Methods of Inspect

```
my_var = 'This is a variable'
my_num = 453.324
class Greeter:
    def __init__(self, _name):
        self.name = _name

    def say_hello(self):
        print('Hello {}!'.format(self.name))

    def say_goodbye(self):
        print('Goodbye {}!'.format(self.name))

my_greeter = Greeter('John')

# check available methods
print('Get id, type and available methods and attributes for Greeter class:')
print(id(my_greeter))
print(type(my_greeter))
print(dir(my_greeter))

print('\nCheck id, type and methods for a variable containing string value')
print(id(my_var))
print(type(my_var))
print(dir(my_var))

print('\nCheck id, type and methods for a variable containing numeric value')
print(id(my_num))
print(type(my_num))
print(dir(my_num))
```

Basic Methods

```
E:\Projects\Sana\Course - Python>C:/Users/USER/AppData/Local/Programs/Python/Python37/python.exe "e:/Projects/Sana/Course - Python/section_two.py"
Get id, type and available methods and attributes for Greeter class:
2248996128456
<class '__main__.Greeter'>
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__m
odule__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'name', 'say_goodbye', 'say_hello']

Check id, type and methods for a variable containing string value
2248995988384
<class 'str'>
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__
', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__s
izeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii',
'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'r
index', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

Check id, type and methods for a variable containing numeric value
2248996074032
<class 'float'>
['__abs__', '__add__', '__bool__', '__class__', '__delattr__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floordiv__', '__format__', '__ge__', '__getattribute__', '__getformat__
', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__int__', '__le__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__pos__', '__pow__', '__radd__', '
__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rmod__', '__rmul__', '__round__', '__rpow__', '__rsub__', '__rtruediv__', '__set_format__', '__setattr__', '__sizeof__'
, '__str__', '__sub__', '__subclasshook__', '__truediv__', '__trunc__', 'as_integer_ratio', 'conjugate', 'fromhex', 'hex', 'imag', 'is_integer', 'real']

E:\Projects\Sana\Course - Python>
```

Basic Methods

- You can see, how all user-defined methods and internal methods associated with classes, strings and float values are listed down using the *dir()* method.
- Similarly, *id()* returns the unique ID value that Python creates for each variable and *type()* returns the respective types.

- Let's have a look at inspect module for more thorough introspection.

Inspect Module

Type	Attribute	Description
Module	__doc__	documentation string
	__file__	filename (missing for built-in modules)
Class	__doc__	documentation string
	__name__	name with which this class was defined
	__qualname__	qualified name
	__module__	name of module in which this class was defined

Inspect Module

Type	Attribute	Description
Method	__doc__	documentation string
	__name__	name with which this method was defined
	__qualname__	qualified name
	__func__	function object containing implementation of method
	__self__	instance to which this method is bound, or None
	__module__	name of module in which this method was defined

Inspect Module

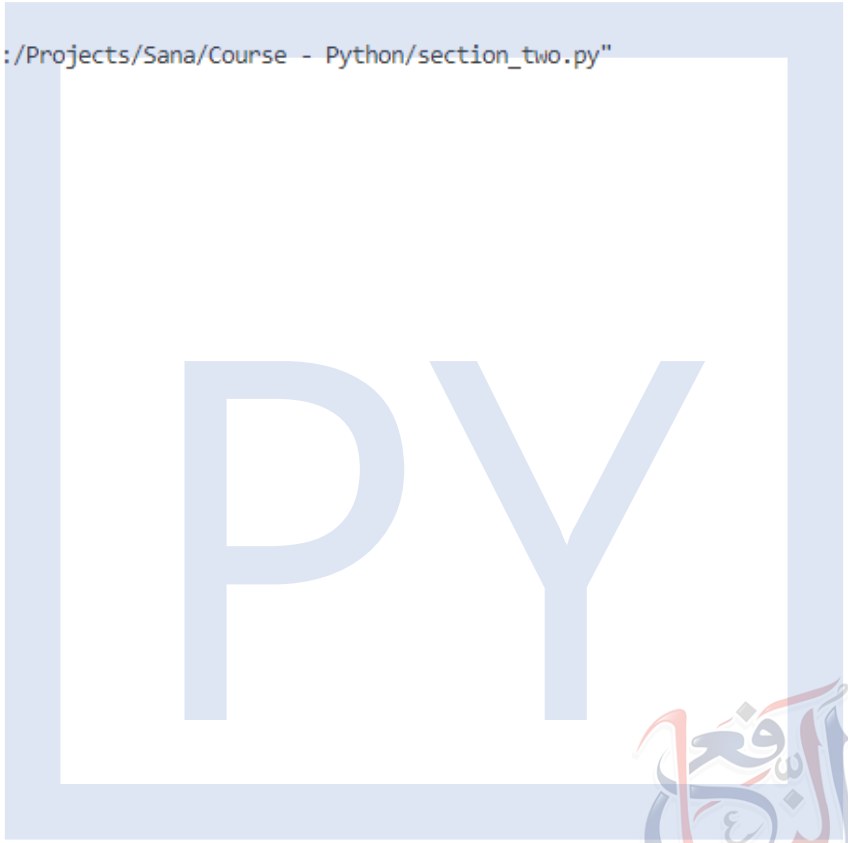
Type	Attribute	Description
Function	__doc__	documentation string
	__name__	name with which this function was defined
	__qualname__	qualified name
	__code__	code object containing compiled function bytecode
	__defaults__	tuple of any default values for positional or keyword parameters
	__kwdefaults__	mapping of any default values for keyword-only parameters
	__globals__	global namespace in which this function was defined
	__annotations__	mapping of parameters names to annotations; "return" key is reserved for return annotations.
	__module__	name of module in which this function was defined

Inspect Module

```
import inspect
import os
testvar = 'Hello'
class Greeter:
    def __init__(self, _name):
        self.name = _name

    def say_hello(self):
        print('Hello {}!'.format(self.name))

    def say_goodbye(self):
        print('Goodbye {}!'.format(self.name))

my_greeter = Greeter('John')
# lambda function
exp = lambda x: x*x

# normal python user-defined function/method
def show_name_age(first_name:str, last_name:str, age:int):
    print('{} {} is {} years old'.format(first_name, last_name, age))


inspect.getmembers(my_greeter) # returns the members of the class my_greeter
print('\nChecking if os is a module:', inspect.ismodule(os))
print('\nChecking if testvar is a module:', inspect.ismodule(testvar))
print('\nChecking if my_greeter is a class: ', inspect.isclass(my_greeter))
print('\nInspect ismethod vs. isfunction comparison'.upper())
print('ISMETHOD: \nshow_name_age:', inspect.ismethod(show_name_age), ' exp:', inspect.ismethod(exp), ' Greeter.say_hello:', inspect.ismethod(my_greeter.say_hello))
print('ISFUNCTION: \nshow_name_age:', inspect.isfunction(show_name_age), ' exp:', inspect.isfunction(exp), ' Greeter.say_hello:', inspect.isfunction(my_greeter.say_hello))
```

Inspect Module

```
E:\Projects\Sana\Course - Python>C:/Users/USER/AppData/Local/Programs/Python/Python37/python.exe "e:/Projects/Sana/Course - Python/section_two.py"

Checking if os is a module: True

Checking if testvar is a module: False

Checking if my_greeter is a class:  False

INSPECT ISMETHOD VS. ISFUNCTION COMPARISON
ISMETHOD:
show_name_age: False  exp: False  Greeter.say_hello: True
ISFUNCTION:
show_name_age: True  exp: True  Greeter.say_hello: False

E:\Projects\Sana\Course - Python>
```



Inspect Module

- Let’s look at another interesting aspect of inspect. *Signature()* method returns a callable object that allows us to inspect the parameters of a method, its type, etc.
- It takes a function as an argument and returns an object.
- The object can be accessed to retrieve parameters of the function.

Inspect Module

```
186   import inspect
187
188   # normal python user-defined function/method
189   def show_name_age(first_name:str, last_name:str, age:int):
190       print('{} {} is {} years old'.format(first_name, last_name, age))
191
192
193   # signature of a method: accesses parameters, their inferred or fixed data types.
194   sig = inspect.signature(show_name_age)
195   print(sig.parameters) # returns an dictionary with parameter names as keys and description as values
196   print(sig.parameters['first_name'].annotation) # prints the type of this parameter
197
198
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
E:\Projects\Sana\Course - Python>C:/Users/USER/AppData/Local/Programs/Python/Python37/python.exe "e:/Projects/Sana/Course - Python/section_two.py"
OrderedDict([('first_name', <Parameter "first_name: str">), ('last_name', <Parameter "last_name: str">), ('age', <Parameter "age: int">)])
<class 'str'>

E:\Projects\Sana\Course - Python>
```

Sqlite Module

- SQL is a querying language used to interact with databases.
- Using SQL, you can create simple yet powerful queries to access, manipulate and retrieve data from databases.
- Python has an internal library called sqlite3 that let's you create databases, add data to it, manipulate and access it all through python code.

- To install:
  - conda install sqlite3 or
  - conda install -c blaze sqlite3 or
  - pip install sqlite3 or
  - sudo install sqlite3

Anaconda Prompt - conda install -c blaze sqlite3

```
(base) C:\Users\sana.rasheed>conda install -c blaze sqlite3
Collecting package metadata: done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\sana.rasheed\AppData\Local\Continuum\anaconda3

  added / updated specs:
    - sqlite3


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    conda-4.8.3                |          py37_0         3.0 MB
    conda-package-handling-1.3.11|          py37_0         280 KB
    sqlite3-3.8.6              |               0         280 KB  blaze
    ------------------------------------------------------------
                                           Total:         3.6 MB

The following NEW packages will be INSTALLED:

  conda-package-han~ pkgs/main/win-64::conda-package-handling-1.3.11-py37_0
  sqlite3            blaze/win-64::sqlite3-3.8.6-0

The following packages will be UPDATED:

  conda                                  4.6.7-py37_0 --> 4.8.3-py37_0


Proceed ([y]/n)? y


Downloading and Extracting Packages
sqlite3-3.8.6        | 280 KB    | ############################################################################ | 100%
conda-4.8.3          | 3.0 MB    | ############################################################################ | 100%
conda-package-handli | 280 KB    | ############################################################################ | 100%
Preparing transaction: done
Verifying transaction: |
```

Sqlite Module – Required Methods

- sqlite3.connect(): established a connection to SQLite Database from Python
  - connection.cursor(): to get a cursor object from the connection object.
    - cursor.execute(): to execute command using a execute() method of a Cursor object.
    - cursor.fetchall(): to fetch all the records, on successful execution of a SELECT query, when read Table.
  - cursor.close(): to close SQLite Cursor object.
- sqlite3.close(): to close SQLite object connection with database.

Sqlite Module

```
import sqlite3
from sqlite3 import Error

def create_connection(db_file):
    """ create a database connection to the SQLite database specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        print("The SQLite connection is connected")
        return conn
    except Error as e:
        print(e)

    return conn
```

Sqlite Module

```
def close_connection(conn):
    """ close database connection to the SQLite database specified by db_file
    :param db_file: database file
    """
    if (conn):
        conn.close()
        print("The SQLite connection is closed")
```

Sqlite Module

```
def create_table_in_db(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)
```

Sqlite Module

```
def create_table():
    database =  r"C:\\tmp\ostempdir\database.db"
    sql_create_projects_table = """ CREATE TABLE IF NOT EXISTS STUDENTS (
                                        id integer PRIMARY KEY,
                                        name text NOT NULL,
                                        gpa integer,
                                        admission_date text
                                    ); """

    #  create a database connection
    conn = create_connection(database)

    #  create tables
    if conn is not None:
    #      create projects table
        create_table_in_db(conn, sql_create_projects_table)
    else:
        print("Error! cannot create the database connection.")

    close_connection(conn)
```

Sqlite Module

- The code creates a database by the name of database.db
- It then passes on sql queries that instantiate table called students with ID, name, gpa and admission date columns.
- So far we have created an empty database table called students.
- Now, we will look into how we can add an entry into the table.

Sqlite Module

```
def add_student(conn, student):
    """
    Create a new student entry into the student table
    :param conn:
    :param student:
    :return: student id
    """
    sql = ''' INSERT INTO STUDENTS(name,gpa,admission_date)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, student)

    return cur.lastrowid
```

Sqlite Module

```
def main_function():
    database = r"C:\\tmp\ostempdir\database.db"

    #create a database connection
    conn = create_connection(database)
    with conn:
    #     create a new project
        student = ('Alan', 1.9, '2019-1-30');
        student_id = add_student(conn, student)

    print('The Student ID:', student_id)
    close_connection(conn)
```

Sqlite Module - Read Table

```
### READ TABLE

database = r"C:\\tmp\ostempdir\database.db"

#create a database connection
conn = create_connection(database)

try:
    cursor = conn.cursor()

    table_name = "STUDENTS"
    sql_string = "SELECT * from " + table_name
    sqlite_select_query = sql_string

    cursor.execute(sqlite_select_query)
    records = cursor.fetchall()
    print("Total rows are:  ", len(records))

    cursor.close()

except sqlite3.Error as error:
        print("Failed to read data from sqlite table", error)
finally:
        close_connection(conn)

print(records)
```

Sqlite Module

- You can do a lot more with databases, then just create entries.
- You can access, manipulate or delete them as well.
- What you need to have is a good understanding of SQL to be able to create powerful queries to interact with your database.

- To read more about SQL statements, you can check https://www.w3schools.com/sql/sql_quickref.asp.

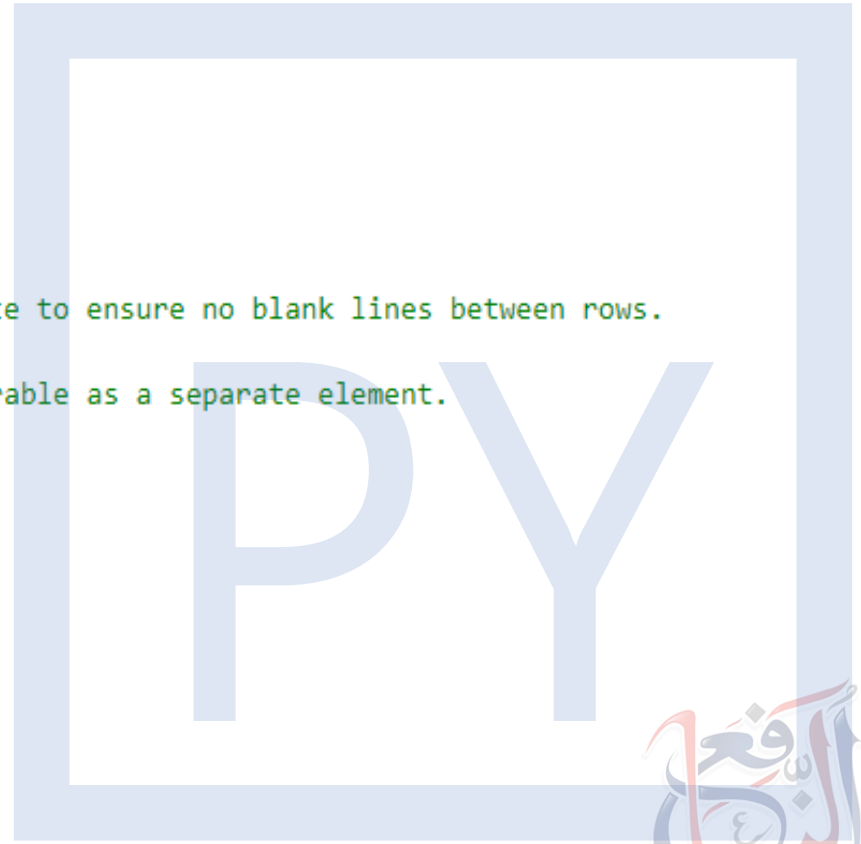
CSV Module

- CSV files follow a particular way of storing data
- Each element is separated either by a comma, space, colon, etc.
- This helps store tables of data in a convenient format
- Using the csv module, you can easily read and write tabular format of data.
- It takes the IO format obtained from open() a step further by parsing it to be readily readable, reducing coding steps from the user's end

CSV Module

- Write File

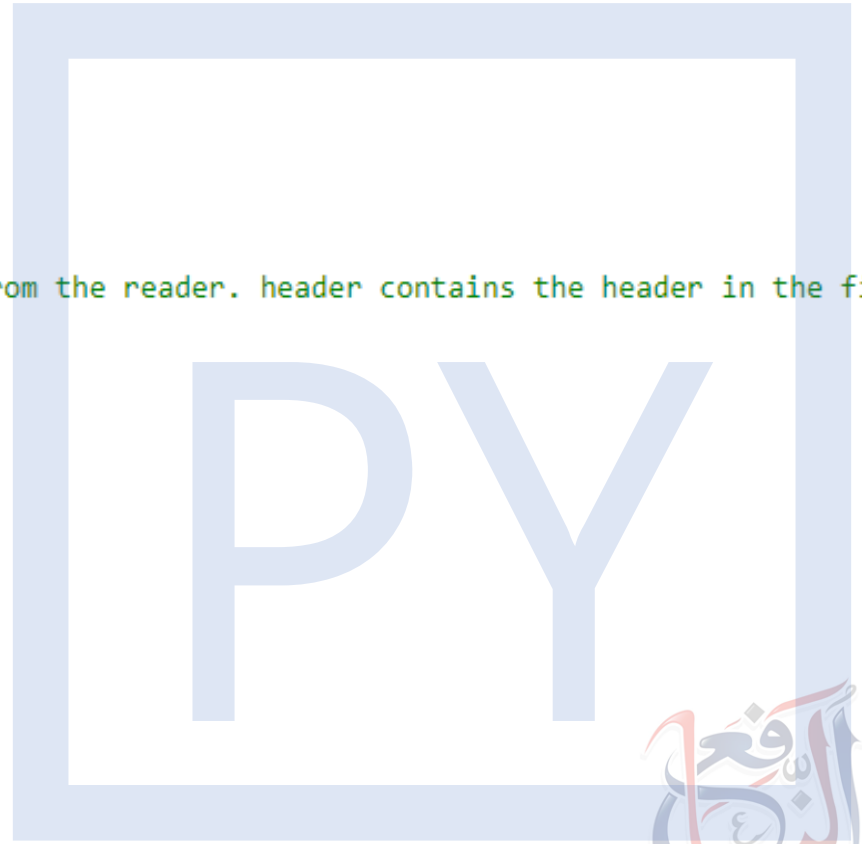
```
36
37  import csv
38  file = 'E:\\Scores.csv'
39  data = [('Mark', 200), ('Lucas', 400), ('Santa', 442), ('Nathan', 600)]
40  writer = csv.writer(open(file, 'w', newline='')) # newline set to empty space to ensure no blank lines between rows.
41  for row in data:
42      writer.writerow(row) # each row is written with each element in the iterable as a separate element.
43
44
45
```



CSV Module

- Read File

```
37  import csv
38  file = 'E:\\Data.csv'
39  reader = csv.reader(open(file, 'r'))
40  header = reader.__next__() # this pops out the first line of the file from the reader. header contains the header in the file.
41  for row in reader:
42      print(row) # each row is printed as a list of items.
43
44
45
```



Subprocess Module

- Subprocess module allows interaction with external applications and software through Python.
- You can use subprocess to call, run, pass arguments and interact with any software by a few lines of code.
- You can interact with the command prompt using python.

Subprocess Module

- Call()
  - Subprocess has a method call() which can be used to start a program. The parameter is a list of which the first argument must be the program name

- save process output (stdout) use *check_output()*
  - We can get the output of the program and store it in the string directly using check_output.

- Set shell = True for creating a new process

Subprocess Module Example

The following code example works on Windows CMD.

```
45  import subprocess
46
47  # Run the command
48  subprocess.call(['dir'], shell=True) # returns the list of files in the current directory
49
50
51
52
53
54
55
56
57
58
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
File Not Found
1
>>> sp.call(['dir'], shell=True)
 Volume in drive E is Work
 Volume Serial Number is 0AEF-C3B3

 Directory of E:\Projects\Sana\Course - Python

02/23/2020  07:32 PM    <DIR>          .
02/23/2020  07:32 PM    <DIR>          ..
01/30/2020  10:44 PM            52,168 Artificial Intelligence Conference by Intel.pptx
02/19/2020  11:10 PM           605,563 Artificial Intelligence Conference.pptx
01/29/2020  11:14 PM             1,625 code_snippets.py
01/30/2020  11:57 PM            10,673 conference.xlsx
02/17/2020  07:39 PM         1,041,321 Part 1.pptx
02/23/2020  07:31 PM           998,412 Part two.pptx
02/23/2020  08:08 PM             1,662 section_two.py
02/02/2020  03:52 PM               411 Temperature.csv
02/02/2020  03:51 PM             9,405 Temperature.csv.xlsx
02/17/2020  08:14 PM           540,574 Template.potx
02/23/2020  07:01 PM    <DIR>          __pycache__
              10 File(s)      3,261,814 bytes
               3 Dir(s)  467,235,491,840 bytes free
0
```

Subprocess Module Example

The following code example works on Windows CMD.

```python
# Import the module
import subprocess

# Run the command
output =  subprocess.check_output(['dir'], shell=True) # returns the list of files in the current directory
print(output)
```

ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   1: cmd

```
:\Projects\Sana\Course - Python>python section_two.py
' Volume in drive E is Work\r\n Volume Serial Number is 0AEF-C3B3\r\n\r\n Directory of E:\\Projects\\Sana\\Course - Python\r\n\r\n02/23/2020  07:32 PM    <DIR>          .\r\n02/23/2020  07:32 PM
  <DIR>          ..\r\n01/30/2020  10:44 PM           52,168 Artificial Intelligence Conference by Intel.pptx\r\n02/19/2020  11:10 PM           605,563 Artificial Intelligence Conference.pptx\r\
01/29/2020  11:14 PM            1,625 code_snippets.py\r\n01/30/2020  11:57 PM           10,673 conference.xlsx\r\n02/17/2020  07:39 PM        1,041,321 Part 1.pptx\r\n02/23/2020  07:31 PM
      998,412 Part two.pptx\r\n02/23/2020  09:55 PM            1,534 section_two.py\r\n02/02/2020  03:52 PM              411 Temperature.csv\r\n02/02/2020  03:51 PM            9,405 Temperatur
.csv.xlsx\r\n02/17/2020  08:14 PM          540,574 Template.potx\r\n02/23/2020  07:01 PM    <DIR>          __pycache__\r\n              10 File(s)      3,261,686 bytes\r\n               3 Dir(s)
 467,235,491,840 bytes free\r\n'

:\Projects\Sana\Course - Python>
```

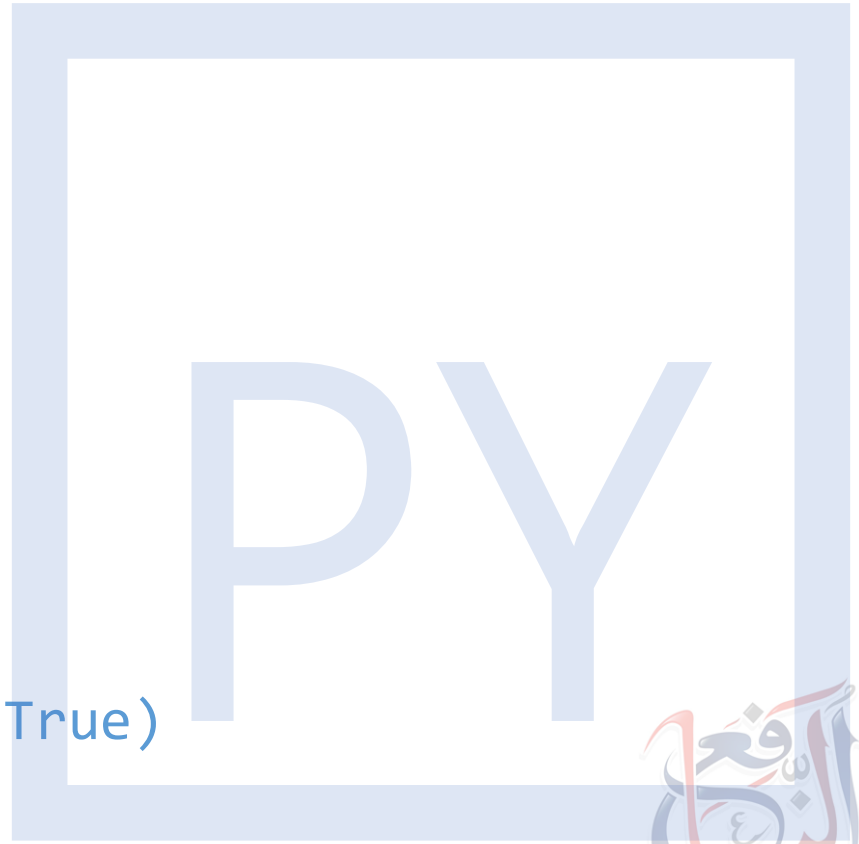
Subprocess Module Example

- Create file “data_read.py

```
Import csv
file = "c:\\tmp\ostempdir\Score.csv”
reader = csv.reader(open(file, 'r'))
for row in reader:
    print(row)
```

- Call this file as subprocess

```
import subprocess
theproc = subprocess.check_output("python
c:\\tmp\ostempdir\data_read.py", shell = True)
theproc
```



Subprocess Module Exercise

- Every computer has an internet browser. For this part of the exercise, you are required to use Python to open a window of your browser.

1. Find the path where your browser.exe file is stored. [e.g. Chrome.exe]. You can use any other application as well. As long as you mention the name.
2. Use a pythonic way to access that path and file and run that application.

Requests module

- We learned to read into Python information that was stored locally in your PC. What about when the information is elsewhere? Say, from a website?
- Certain websites provide an API to access information that they stream on their website. Developers can simply write code to access that information, play with it and store insights derived from it locally on their system.
- You can read information online from a library called requests.

Requests module

- The requests module allows you to send HTTP requests using Python.

- The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

Request Methods

Method	Description
delete(*url*, *args*)	Sends a DELETE request to the specified url. DELETE requests are made for deleting the specified resource (file, record etc).
get(*url*, *params, args*)	Sends a GET request to the specified url
head(*url*, *args*)	Sends a HEAD request to the specified url. HEAD requests are done when you do not need the content of the file, but only the status_code or HTTP headers.
post(*url*, *data, json, args*)	Sends a POST request to the specified url. The post() method is used when you want to send some data to the server.

Requests module

- Remember, do not overcrowd the network by making repeated requests since its clogs up the service and makes it harder and longer for the network to respond.

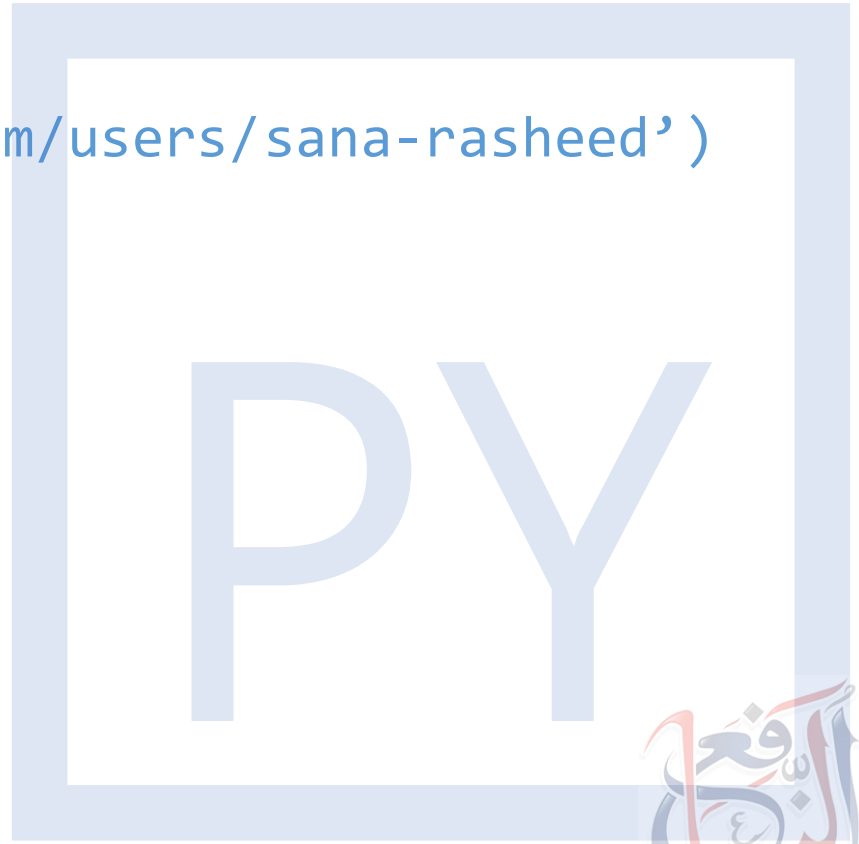
- Use APIs responsibly.

Request Methods – Get, Head

```
import requests

x = requests.get("https://github.com/")
print(x)
<Response [200]>

print(x.status_code)
200
404 NOT FOUND status means that the resource you were looking for was not found

print(x.text)
print(x.headers)


x = requests.head("https://github.com/")
print(x.hearders)
print(x.headers['Content-Type'])
```

Request Methods – Get

```
import requests

res = requests.get('https://api.github.com/users/sana-rasheed’)
print(res)
<Response [200]>

print(res.status_code)
200

data = res.json()
print("Followering: ", data['following'])
print("Followers: ", data['followers’])
```



Requests module

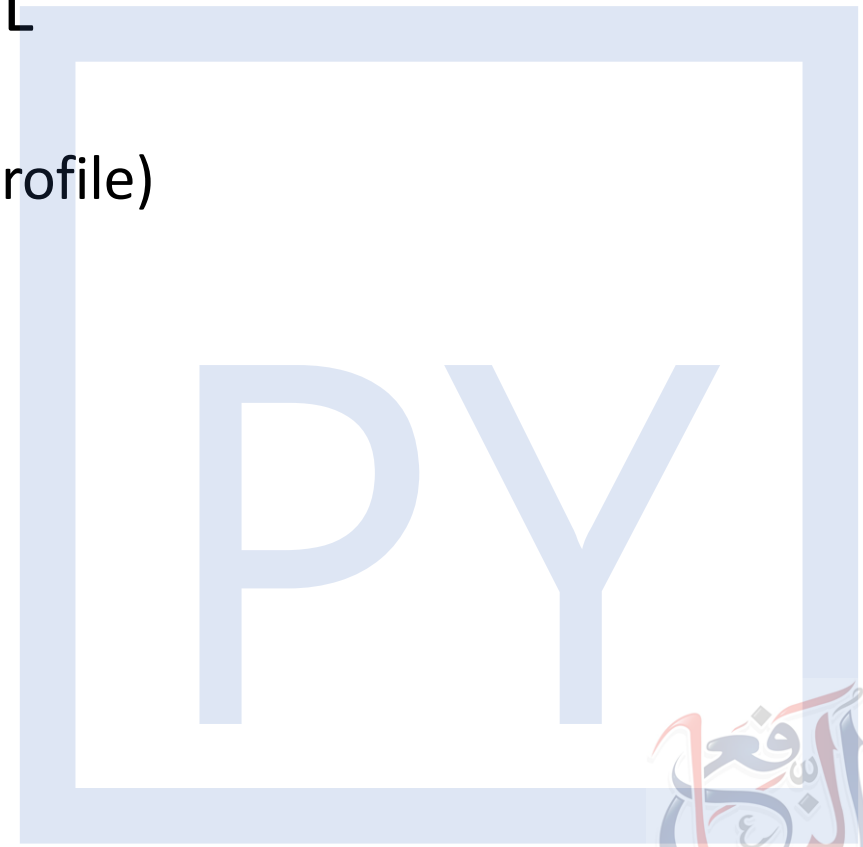
```python
import requests

# api-endpoint
url = 'https://official-joke-api.appspot.com/jokes/programming/random'
response = requests.get(url)
data = response.json()[0]
for key in data:
    print(key, ":", data[key])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
E:\Projects\Sana\Course - Python>python section_two.py
id : 18
type : programming
setup : Why did the programmer quit his job?
punchline : Because he didn't get arrays.

E:\Projects\Sana\Course - Python>
```

Quick Exercise

- Get and inspect the headers of a given URL
  - http://www.google.com
  - https://api.github.com/users/(create_your_profile)



Request Method - Post

```
import requests
myobj = [('Ali', 56),('Ahmed', 87),('Amna', 76)]

r = requests.post("http://httpbin.org/post", data=myobj)
r
<Response [200]>

r.text

data = r.json()
data[‘form’]
```

```
{'Ahmed': '87', 'Ali': '56', 'Amna': '76'}
```

Request Module Exercise 1

- 'http://dummy.restapiexample.com/api/v1/employees'
- The above website stores information about employees such as their names, salaries, ages, etc.
- Your task is to read the data from the API and store it in Python and then extract the following aspects of each employee:

1. ID
2. Name
3. Salary
4. Employee_age

- How many employees are under the age of 25?
- How many earn more than 40,000?
- Do the older employees earn more?

Request Module Exercise 2

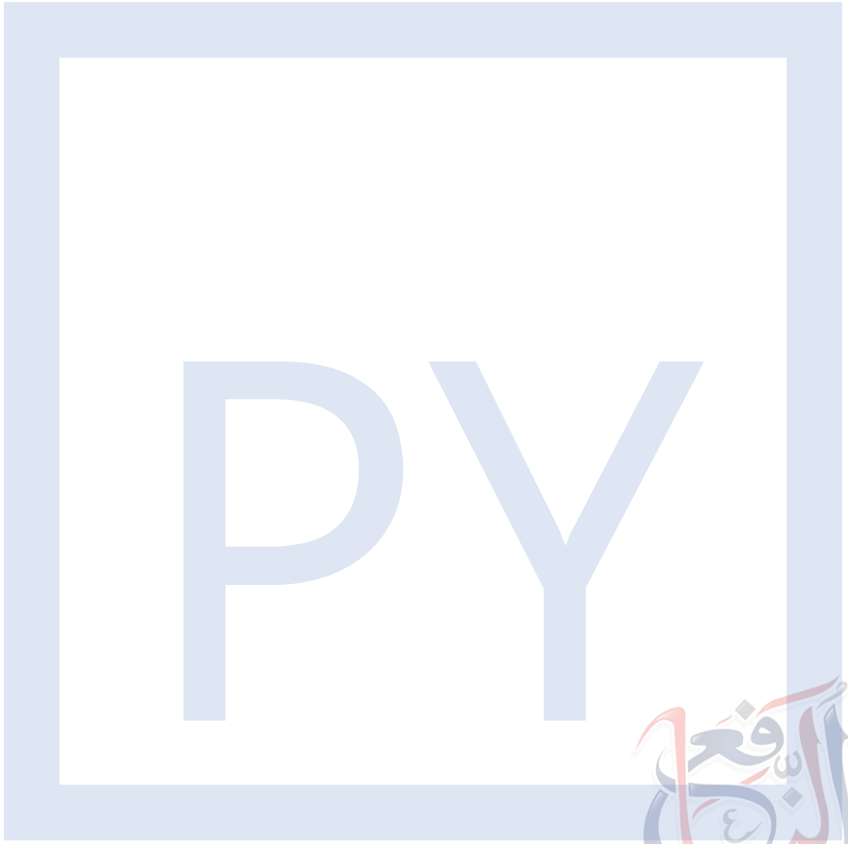
- The employee data you have extracted from previous exercise, post this data on `http://httpbin.org/post`. Apply condition on the response, if response status is 200 then convert response data into json format and answer the following questions:
- What are the values in ‘headers’.
- What is the value of ‘Content-Length’.
  - HINT: data['headers']['Content-Length']
- Display the values in ‘form’.

Requests module

- Remember, do not overcrowd the network by making repeated requests since its clogs up the service and makes it harder and longer for the network to respond.
- Use APIs responsibly. Give a couple of seconds gap between API calls to ensure fair usage.

Recap

- Dates
- OS module
- Introspection
- Sqlite module
- Csv module
- Sub process module
- Requests module



جزاك اللهُ

To ask questions, Join the Al Nafi Official Group

https://www.facebook.com/groups/alnafi/

(This group is only for members to ask questions)