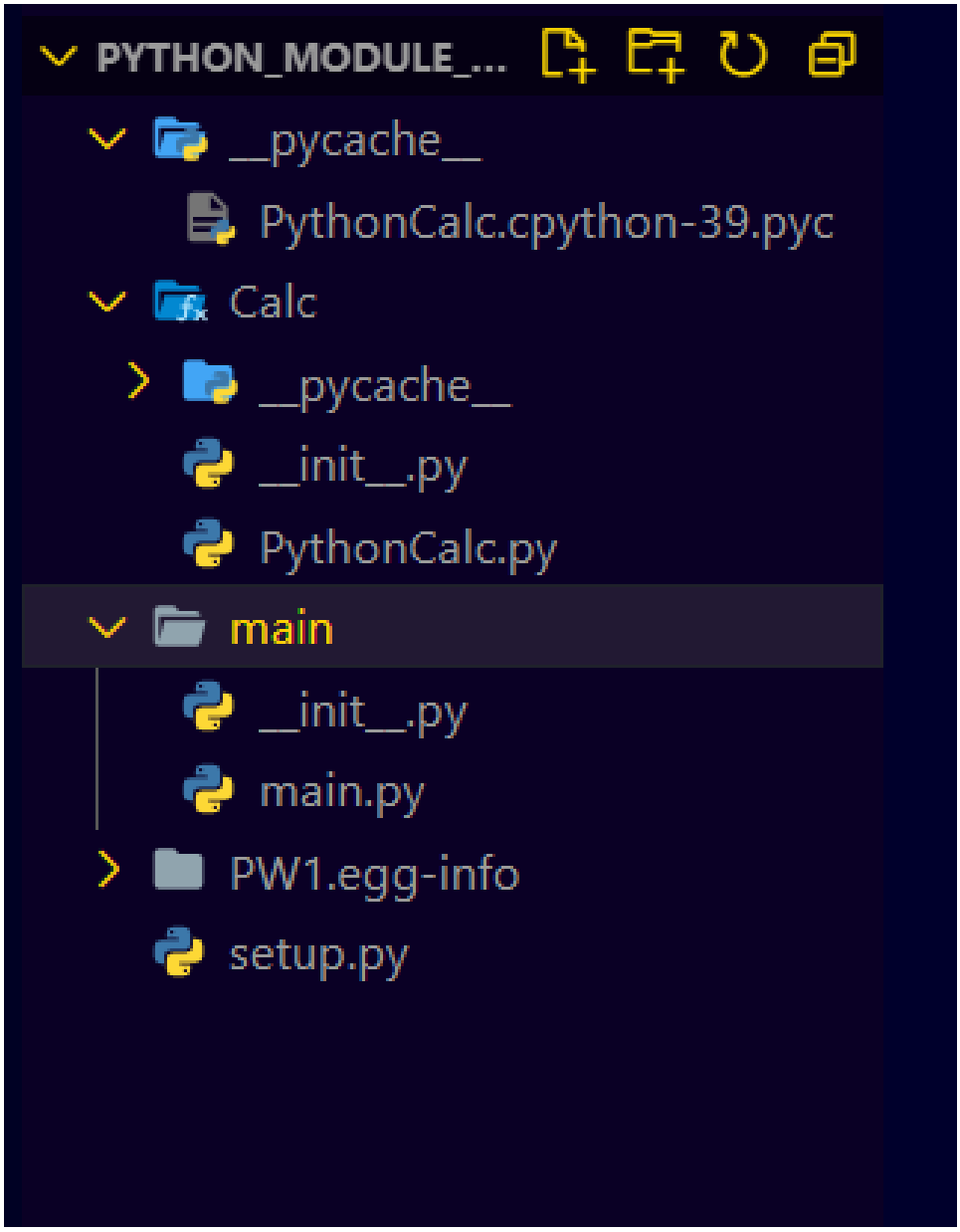
Python Modules and CSV

Pratik Wani

➢ Modules:-

- Module is a file containing Python code, typically with a .py extension.
- Module can have functions, classes, and variables which we can reuse in other Python codes
- With the help of modules we can manage our code in effective manner
- Python modules have two types:
    - <u>Pre-defined module:</u>
        - This modules are already exists in python we can use the functionalities of these modules by importing them in our code
        - To import module: from module_name import *
        - Ex: OS, Math
    - <u>User-defined module:</u>
        - This are the module created by us
        - We can import the functionalities of modules in our other codes
- To make any folder as module we have to create __init__.py file in that folder
- Here I have created PythonCalc module where I created a basic calculator and used predefined module math
- And then I import these PythonCalc module in main module to run the functionalities of PythonCalc

- Module Structure:



- PythonCalc Module:

PythonCalc.py

Calc > PythonCalc.py > ...

```python
import math
class Calculator:
    def __init__(self, a, b):
        self.a=a
        self.b=b

    def addition(self):
        return self.a+self.b

    def subtraction(self):
        return self.a-self.b

    def multiplication(self):
        return self.a*self.b

    def division(self):
        if self.b==0:
            return "B cannot be zero"
        else:
            return self.a / self.b

    def square_root(self, c):
        return math.sqrt(c)

    def power(self):
        return math.pow(self.a, self.b)
```

- Main Module:

main.py

main > main.py > ...

```python
# main.py

import Calc.PythonCalc as C

calc=C.Calculator(84,4)

print("Addition:", calc.addition())
print("Subtraction:", calc.subtraction())
print("Multiplication:", calc.multiplication())
print("Division:", calc.division())
print("Square Root:", calc.square_root(9))
print("Power:", calc.power())
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1> & C:/User
thon/Python39/python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python/Python_Module_
Addition: 88
Subtraction: 80
Multiplication: 336
Division: 21.0
Square Root: 3.0
Power: 49787136.0
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1>
```

➢ File I/O, Read and Write data into CSV:-

- It is a type of text file that stores tabular data for better readability, easier understanding, and faster processing.
- CSV files can be converted from a JSON file or created using Python
- CSV stands for “Comma Separated Values.”
- List of Methods to Read a CSV File in Python:
    - Read CSV file using csv.reader
    - Read CSV file using .readlines() function
    - Read CSV file using Pandas
    - Read CSV file using csv.DictReader
- List of Methods to write a CSV File in Python:
    - Write CSV file using csv.writer() function
    - Write CSV file using csv.writelines() function
    - Write CSV file using pandas
    - Write CSV file using csv.DictWriter() function

- Sample CSV file named as practice.csv:

practice.csv × CSV.py student_salary_info.csv

practice.csv

```csv
Name,Age,Occupation
John,25,Engineer
Alice,30,Teacher
Bob,22,Student
Eva,35,Doctor
Chris,28,Software Developer
Sophia,32,Marketing Analyst
```

- Reading CSV file with all four types:

CSV.py > ...

```python
import csv
import pandas as pd
import numpy as np

# 1) reading data in csv using open and reader
rows=[]
with open("practice.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)
print(header)
print("\n*******\n")
for i in rows:
    print(i)


# 2) reading data in csv using readlines
print("\n*******\n")
with open('practice.csv') as file:
    content = file.readlines()
header = content[:2]
rows = content[3:]
print(header)
print("\n*******\n")

print(rows)
print("\n*******\n")
```

CSV.py > ...

```python

# 3) reading file using pandas
data= pd.read_csv("practice.csv")
print(data)
print("\n**********\n")

#extracting the columns
print(data.columns)
print("\n**********\n")

#extracting specific column
print(data.Name)
print("\n**********\n")

# 4) reading data using DictReader
with open('practice.csv','r') as file:
    reader=csv.DictReader(file)

    for i in file:
        print(i)


```

TERMINAL

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1>
hon39/python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python/Python_Modul
['Name', 'Age', 'Occupation']

*******

['John', '25', 'Engineer']
['Alice', '30', 'Teacher']
['Bob', '22', 'Student']
['Eva', '35', 'Doctor']
['Chris', '28', 'Software Developer']
['Sophia', '32', 'Marketing Analyst']

*******

['Name,Age,Occupation\n', 'John,25,Engineer\n']

*******

['Bob,22,Student\n', 'Eva,35,Doctor\n', 'Chris,28,Software Developer\n', 'Sophia,32,Marketing Analyst']

*******

     Name  Age          Occupation
0    John   25            Engineer
1   Alice   30             Teacher
2     Bob   22             Student
3     Eva   35              Doctor
4   Chris   28  Software Developer
5  Sophia   32   Marketing Analyst
```

TERMINAL

```
**********

Index(['Name', 'Age', 'Occupation'], dtype='object')

**********

0      John
1     Alice
2       Bob
3       Eva
4     Chris
5    Sophia
Name: Name, dtype: object

**********

Name,Age,Occupation

John,25,Engineer

Alice,30,Teacher

Bob,22,Student

Eva,35,Doctor

Chris,28,Software Developer

Sophia,32,Marketing Analyst

**********
```

- Creating Series and Dataframes:
    - DataFrames:
        - A two-dimensional table with rows and columns, similar to a spreadsheet or SQL table.
    - Series:
        - A one-dimensional labeled array, often used for representing a single column or row of data.

```python
#creating series
data=np.array(['P','R','A','T','I','K'])
series=pd.Series(data)
print("\n**********\n")
print("Pandas Series: ",series)
print("\n**********\n")

#creating dataframe using pandas
header=['Name','Age','Salary']
data=[['Pratik',21,'50000'],['Vikas',23,100000],['Rushi',22,120000]]
df=pd.DataFrame(data,columns=header)
print(df)
```

```
Pandas Series:  0    P
1    R
2    A
3    T
4    I
5    K
dtype: object

**********

     Name  Age  Salary
0  Pratik   21   50000
1   Vikas   23  100000
2   Rushi   22  120000
```

- Writing in CSV file with all four types:

```python
# 1) write using pandas:

df.to_csv("student_salary_info.csv",index=False)

print("\n**********\n")
print(pd.read_csv('student_salary_info.csv'))
print("\n**********\n")

# 2) write in  csv file using DictReader

data=[{'Name':'Rudra','Age':25,'Salary':300000},
    {'Name':'Krushna','Age':24,'Salary':400000},
    {'Name':'Ram','Age':21,'Salary':20000}]

with open('test.csv', 'w', newline='') as file:
    field_names=['Name', 'Age', 'Salary']

    writer=csv.DictWriter(file,fieldnames=field_names)

    writer.writeheader()

    writer.writerows(data)
```

```python
# 3) write in  csv file using csv.writer

header=['Name','Score','Class']
data=[['Pratik',92,'A'], ['Rushi',80,'B'], ['Joey',93,'A']]

filename='Students_Data_write.csv'
with open(filename, 'w', newline="") as file:
    csvwriter= csv.writer(file)
    csvwriter.writerow(header)
    csvwriter.writerows(data)

# 4) write in  csv file using csv.writerlines

header=['Name','Score','Class']
data=[['Pratik',92,'A'], ['Rushi',80,'B'], ['Joey',93,'A']]
filename = 'Student_Data_writelines.csv'
with open(filename, 'w') as file:
    for header in header:
        file.write(str(header)+', ')
    file.write('n')
    for row in data:
        for x in row:
            file.write(str(x)+', ')
        file.write('n')
```

student_salary_info.csv × test.csv

student_salary_info.csv

```csv
Name,Age,Salary
Pratik,21,50000
Vikas,23,100000
Rushi,22,120000
```

student_salary_info.csv test.csv × Student

test.csv

```csv
Name,Age,Salary
Rudra,25,300000
Krushna,24,400000
Ram,21,20000
```

student_salary_info.csv | test.csv

Students_Data_write.csv

```
Name,Score,Class
Pratik,92,A
Rushi,80,B
Joey,93,A
```

student_salary_info.csv | test.csv | Students_Data_write.csv | Student_Data_writelines.csv | practice.csv | CSV.py

Student_Data_writelines.csv

```
Name, Score, Class, nPratik, 92, A, nRushi, 80, B, nJoey, 93, A, n
```

- Processing Python List:
    - An ordered collection of elements
    - Lists are Mutable, meaning we can modify their contents by adding, removing, or changing elements.
    - Lists are heterogeneous.

student_salary_info.csv | test.csv | Students_Data_write.csv | Student_Data_

List_processing.py > ...

```python
my_list=[1,2,3,4,5]
#append
my_list.append(4)
print(my_list)

#pop
popped_element=my_list.pop()
print(my_list)

#insert
my_list.insert(0,0)
print(my_list)

#length
print(len(my_list))

#sort
my_list.sort()
print(my_list)

#slicing
print(my_list[0:3])
print(my_list[-2:-1])
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
TERMINAL
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1> & C:
Python/Python39/python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python/Python_
.py"
[1, 2, 3, 4, 5, 4]
[1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5]
6
[0, 1, 2, 3, 4, 5]
[0, 1, 2]
[4]
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1>
```

- Lambda Function in python:
  - Lambda functions are also known as anonymous functions or lambda expressions.
  - They are often used in situations where a small, one-time function is needed for a short duration.

```
student_salary_info.csv    test.csv    Students_Data_write.csv
Lambda.py > ...
1  numbers=[1,2,3,4]
2
3  squared_numbers=map(lambda x: x**2, numbers)
4
5  print(list(squared_numbers))
6
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
TERMINAL
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data
hon39/python.exe "c:/Users/dell/OneDrive/Documents/Desktop/P
[1, 4, 9, 16]
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data
```

- Filter Data in Python Lists using filter and lambda:

a) Use of Lambda Function in Python:

- We can pass small function as an argument
- Lambda functions are Useful for defining custom sorting criteria when sorting lists of elements.
- They can be used to create a filter version of list

b) Practical Uses of Python lambda function:

- Lambda functions are used with map(), filter(), and reduce()
- For applying different sorting conditions we can use Lambda
- Lambda functions are also used to handle a simple events in GUI programming
- Simple callback functions can also be created with the help of lambda functions

c) Using Lambda Function with map(), filter(), and reduce():

- Map(): Applies a given function to all items in an iterable and returns an iterator that produces the results.
- Filter(): The filter() function is to filter the iterable based on some condition
- Reduce(): reduce function is part of the functools module in Python 3, It is useful to apply a binary function to the items of an iterable

Lambda.py > ...

```python

#map
L=['1','2','3']
new_L=list(map(int,L))

print(new_L[0]+new_L[1])
print("\n*********")

#filter
odd_elements=list(filter(lambda x: x%2≠0,L))

print(odd_elements)
print("\n*********")

#reduce
from functools import reduce
Sum=reduce(lambda x,y: x+y,L)

print(Sum)
print("\n*********")

```

```
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1> & C:/
hon39/python.exe "c:/Users/dell/OneDrive/Documents/Desktop/Pratik Wani/Data Engineering/Python/Python_Module_Assi
[1, 4, 9, 16]
3

*********
[1, 3]

*********
6

*********
PS C:\Users\dell\OneDrive\Documents\Desktop\Pratik Wani\Data Engineering\Python\Python_Module_Assignment_1>
```