Assignment-3

1) File IO using python
2) Read data from CSV file into python list
3) Processing python lists
4) Lambda functions in python
5) Usage of lambda functions
6) Filter data in python lists using map, filter and reduce

1) File IO using python

Python allows users to handle files i.e., to read and write files along with other operations.

Creating a file in Python:

1) Create a file using open()
2) Two parameters: filename, access mode to create a file

Mode	Description
r	Open a file for reading.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
r+	opens for reading and writing (cannot truncate a file)
w+	For writing and reading (can truncate a file)
a	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
t	Open in text mode.
b	Open in binary mode.
x	Open for exclusive creation, failing if the file already

seek() :change or move cursor to place where data must be read or written in the file

tell():return current position of file pointer from beginning of file

read():returns file content in the form of string

readLine(): Reads single line

readLines():read file into list

writelines():write list of strings to file

close(): closes the file

write():write specified string to file

append(): It adds the data at the end of file.

a) Reading a file:

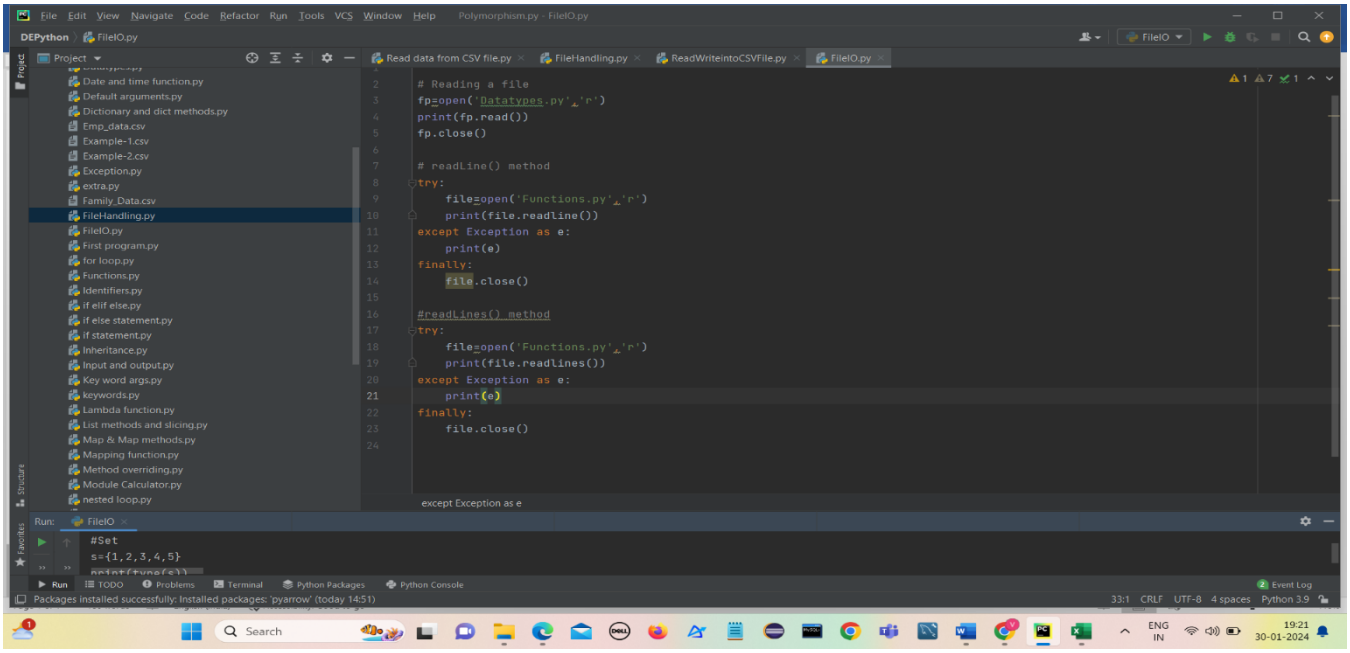
A file is read using read() function.

b) readLine():

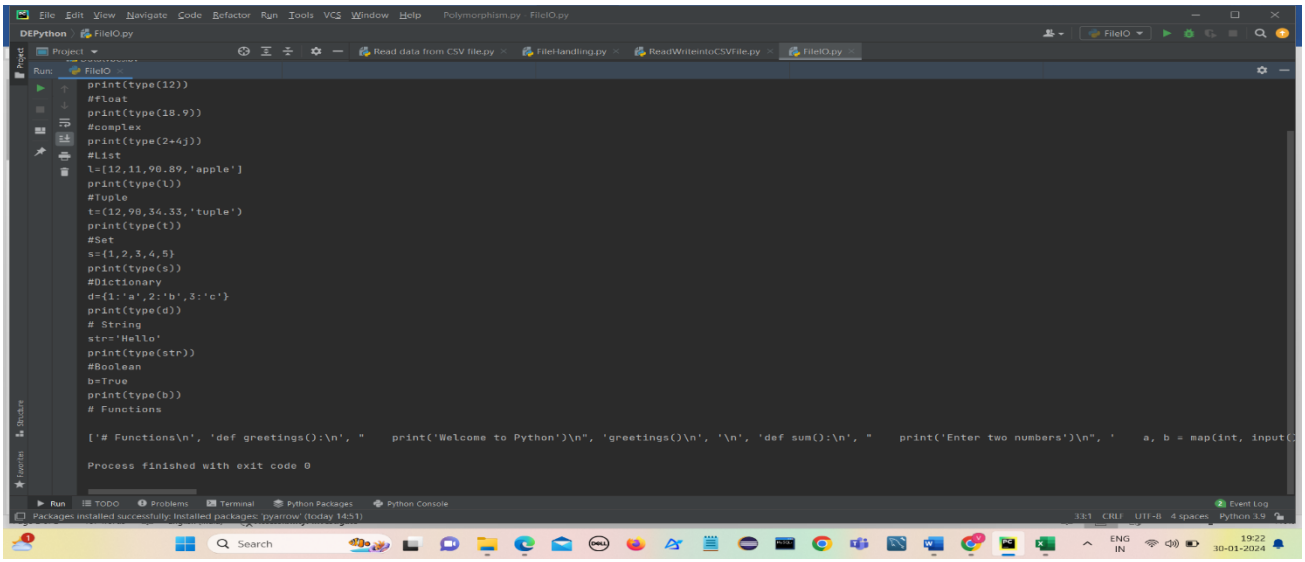
readLine() method reads a single line from file.

c) readLines():

readLines() method reads the whole fist into list



Output:

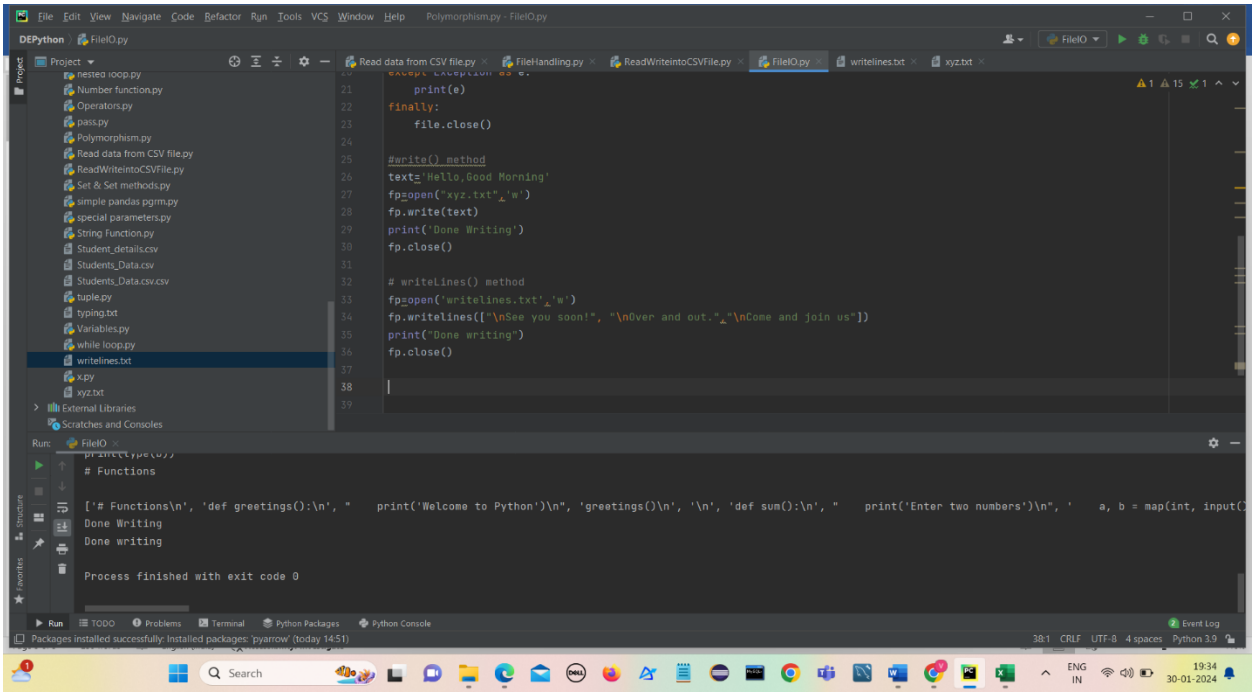


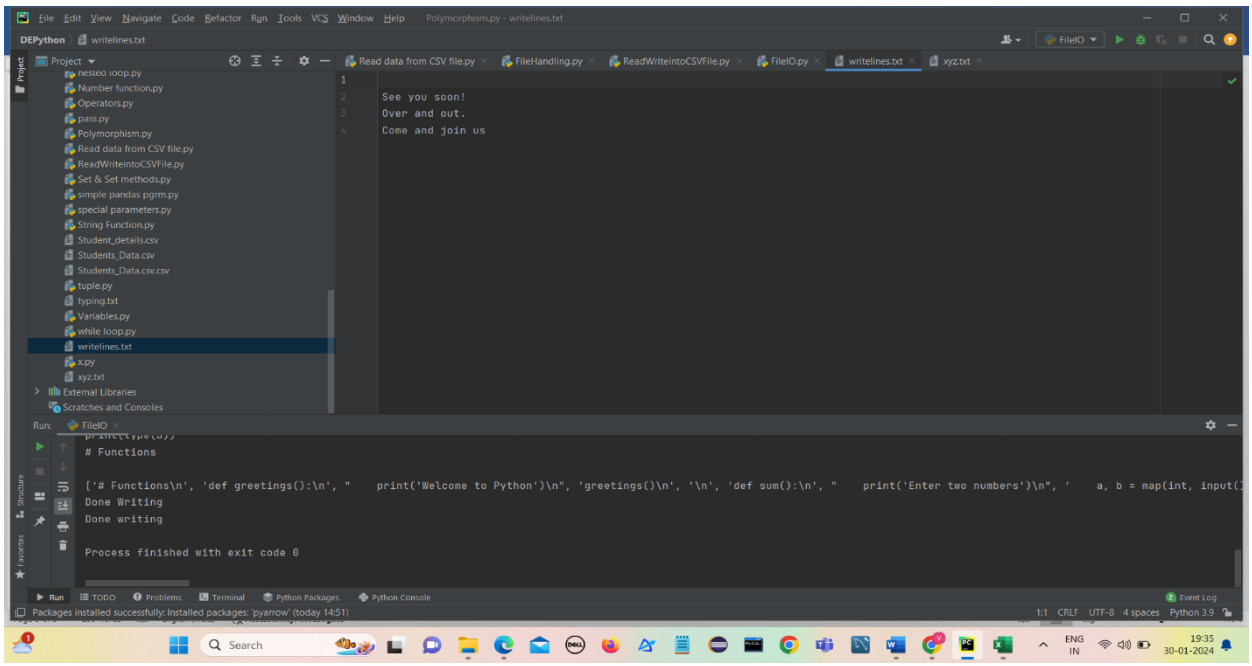
d) write():

To use write method open the file in ‘w’ mode. It is used to write specified string to file.

e) writeLines():

It is used to write list of strings to file.

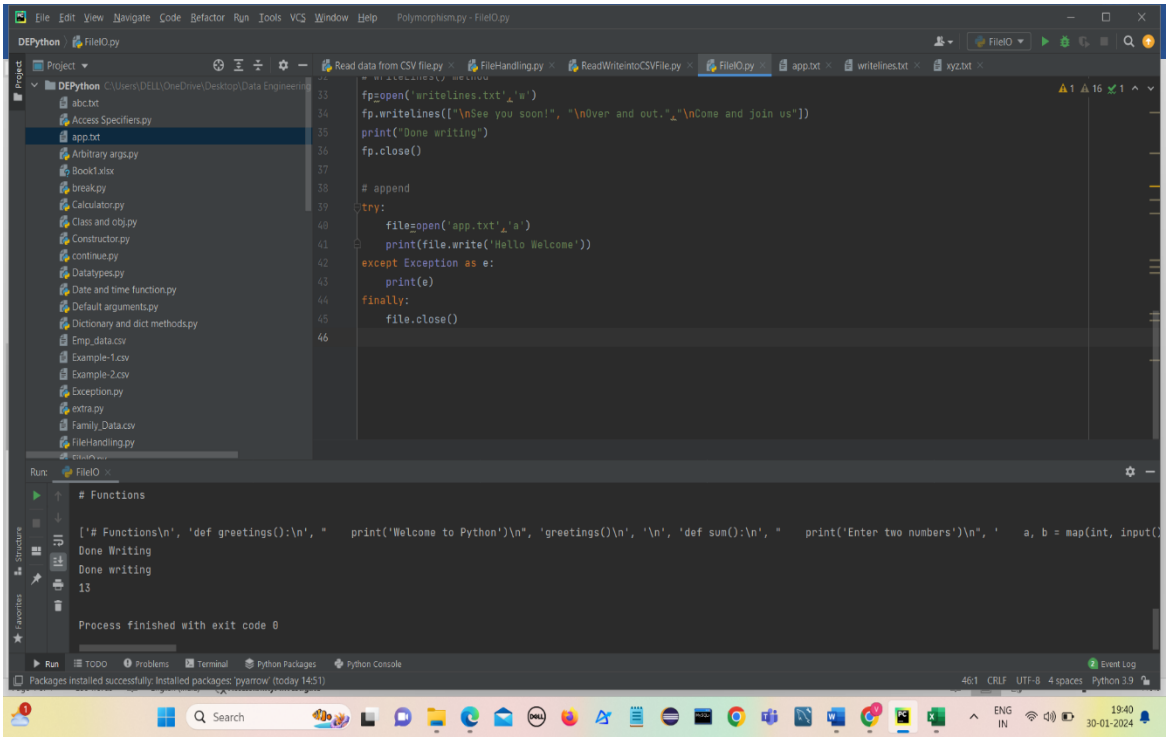


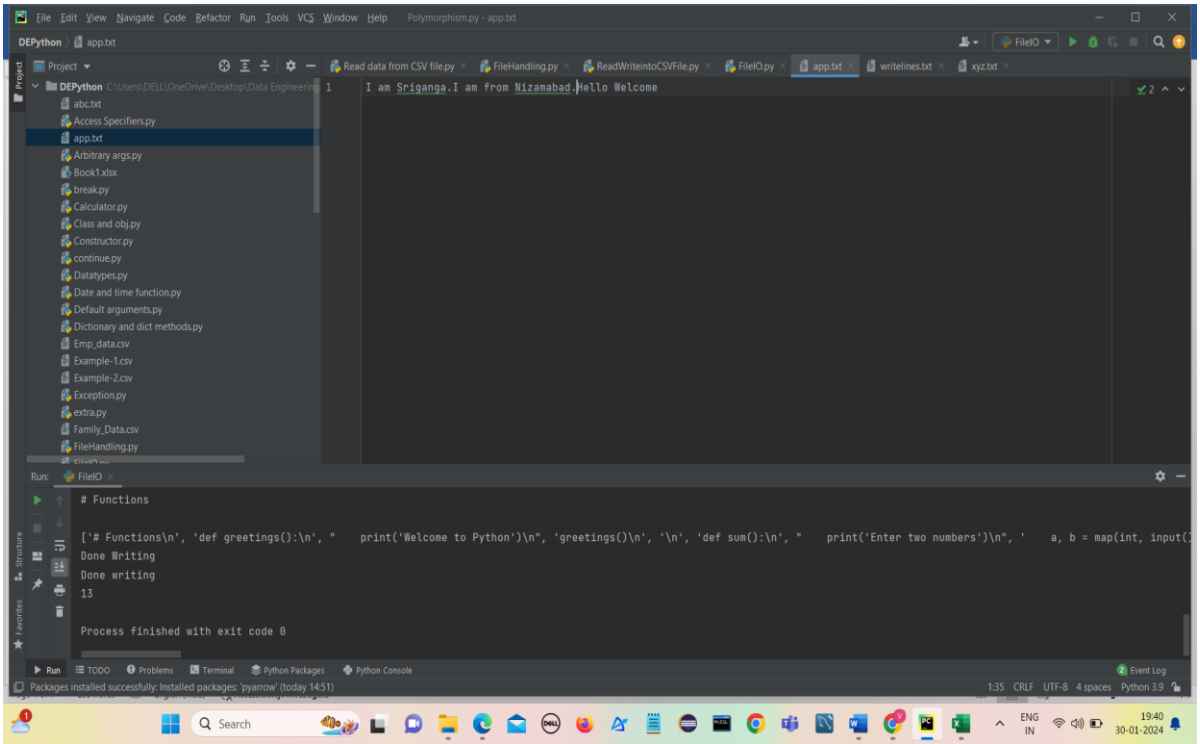


f) append():

To append data, use the append mode in the open() function that allows writing the data to the existing file.

The access mode must be ‘a’.



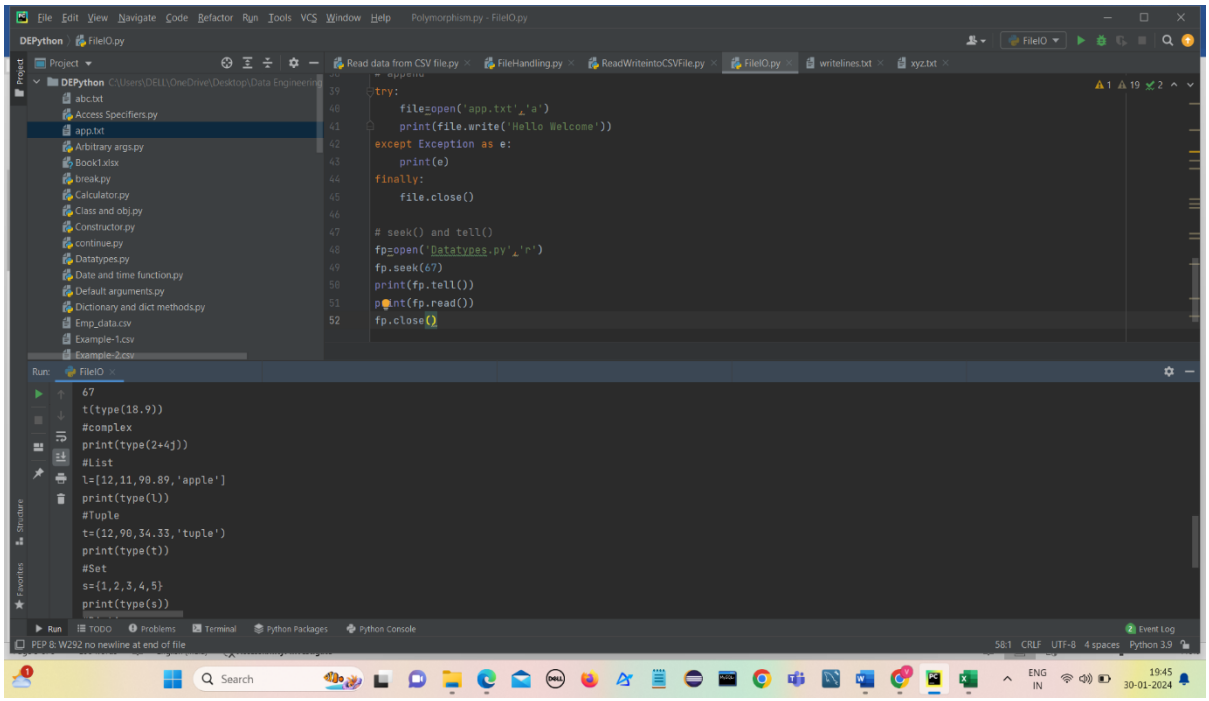


g) seek():

It is used to change or move cursor to place where data must be read or written in the file.

h) tell():

It is used to return current position of file pointer from beginning of file.

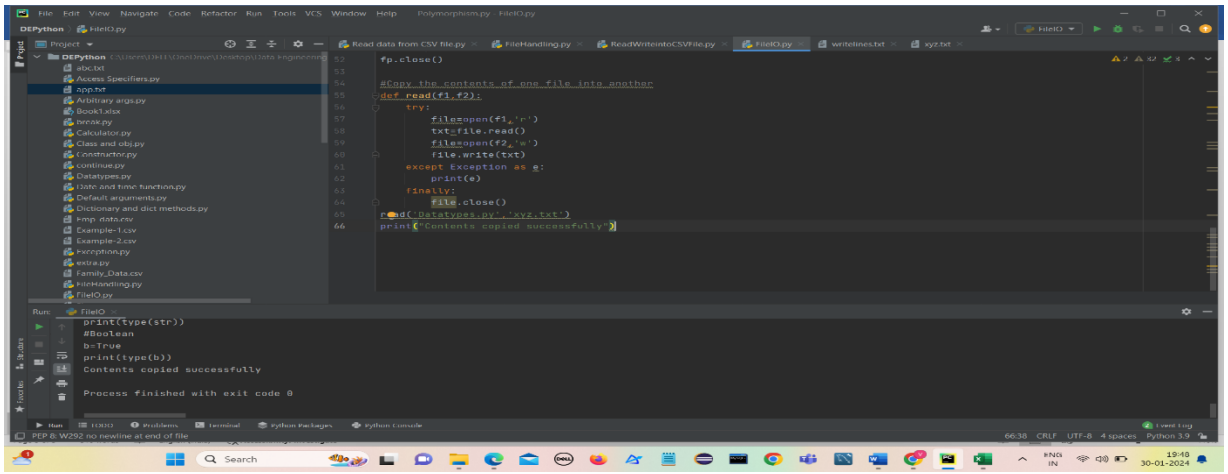


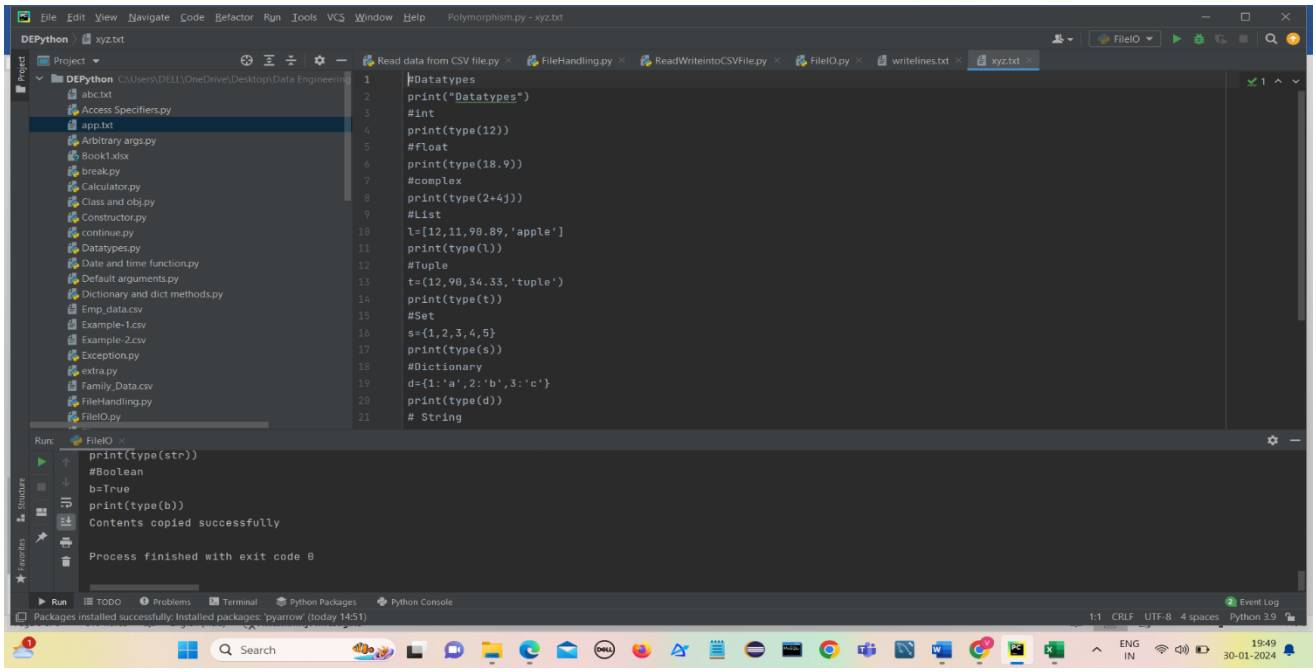
i) close()

close() in python is used to close the file

fp.close()

j) Copy contents of one file into another





2) Read data from CSV file into python list

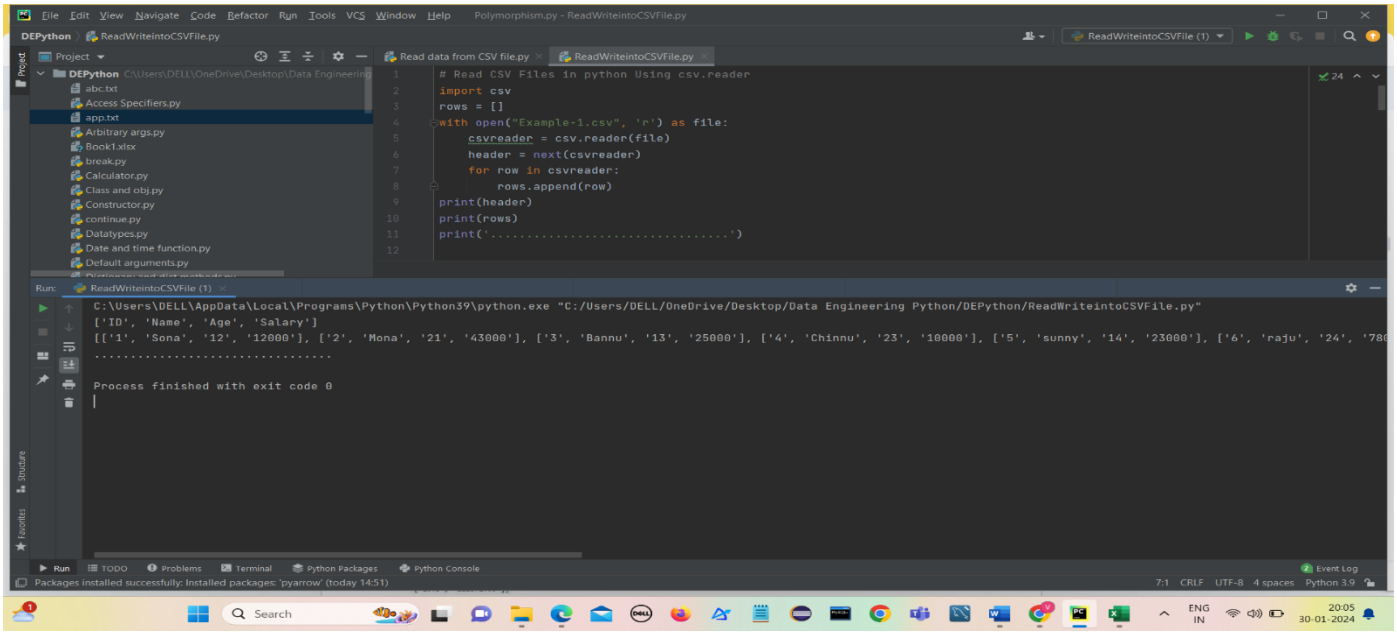
- CSV file is a type of text file that stores tabular data for better readability, easier understanding, and faster processing.
- CSV stands for “Comma Separated Values.”
- It is the simplest form of storing data in tabular form as plain text.

List of Methods to Read a CSV File in Python

1. Read CSV file using csv.reader
2. Read CSV file using .readlines() function
3. Read CSV file using Pandas
4. Read CSV file using csv.DictReader

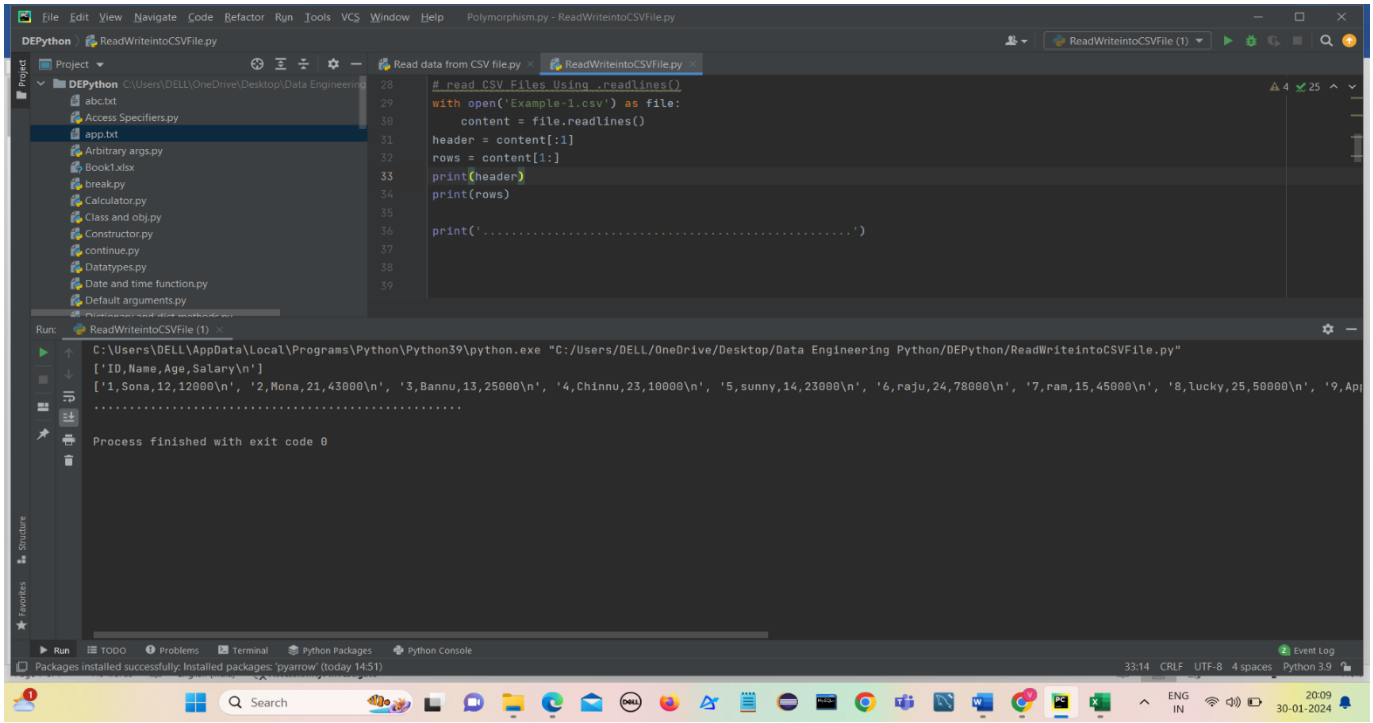
1) **Read CSV file using csv.reader**

It is used to read CSV files using the **csv.reader** object from Python's **csv** module.



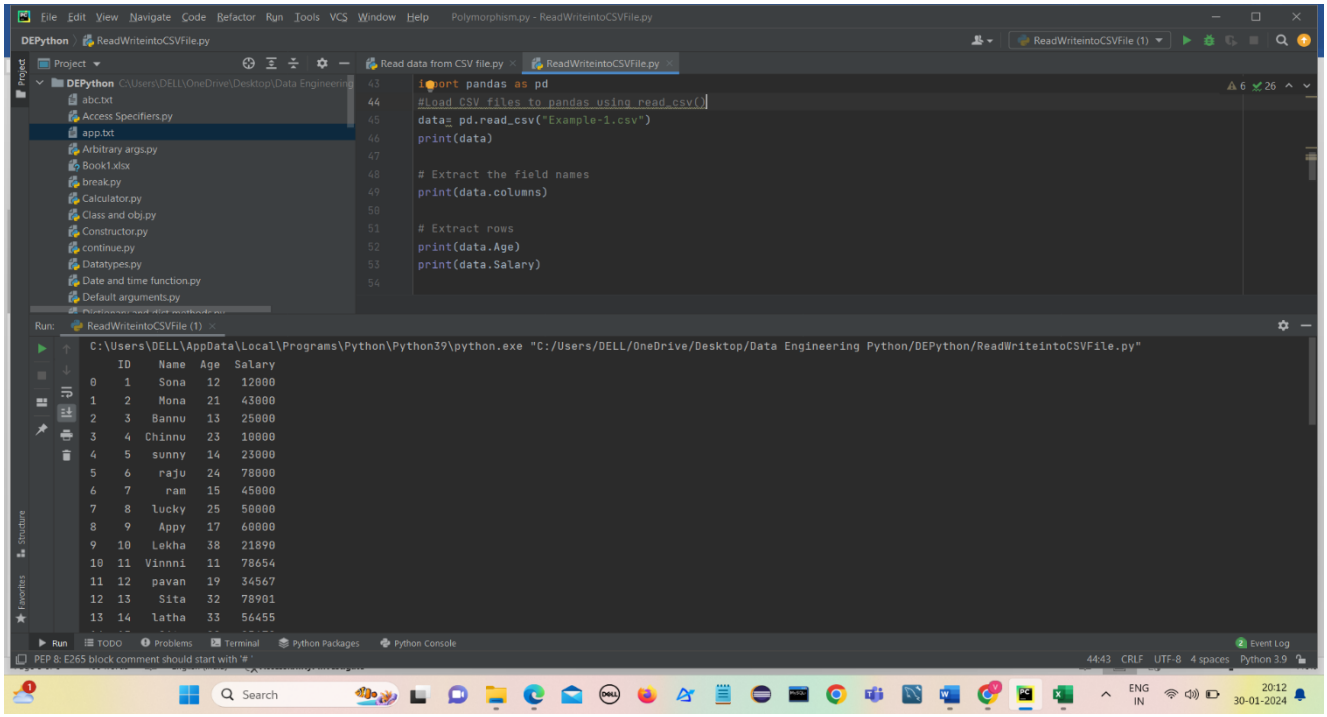
2) **Read CSV file using .readlines() function**

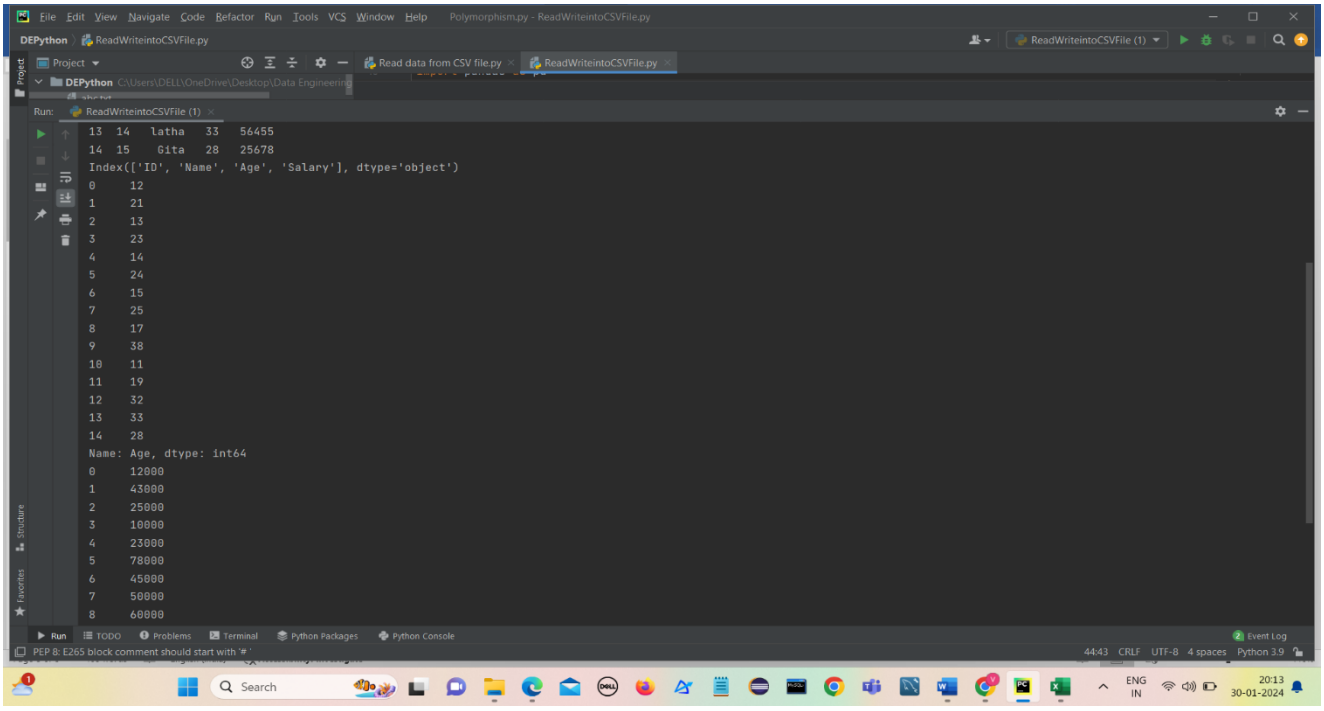
.readlines() method is to fetch the header and rows using only open() and with() statements and without the csv library.



3) Read CSV file using Pandas

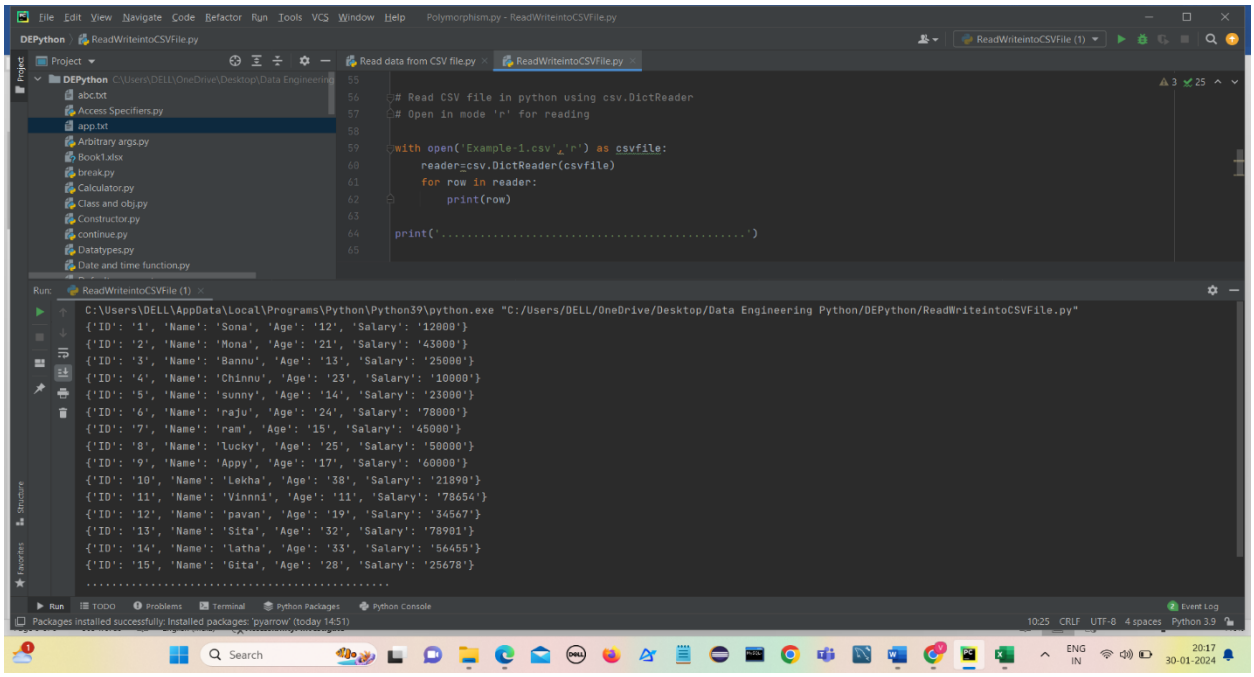
- Import pandas library
- Load CSV files to pandas using read_csv()
- Print the data





4) Read CSV file using csv.DictReader()

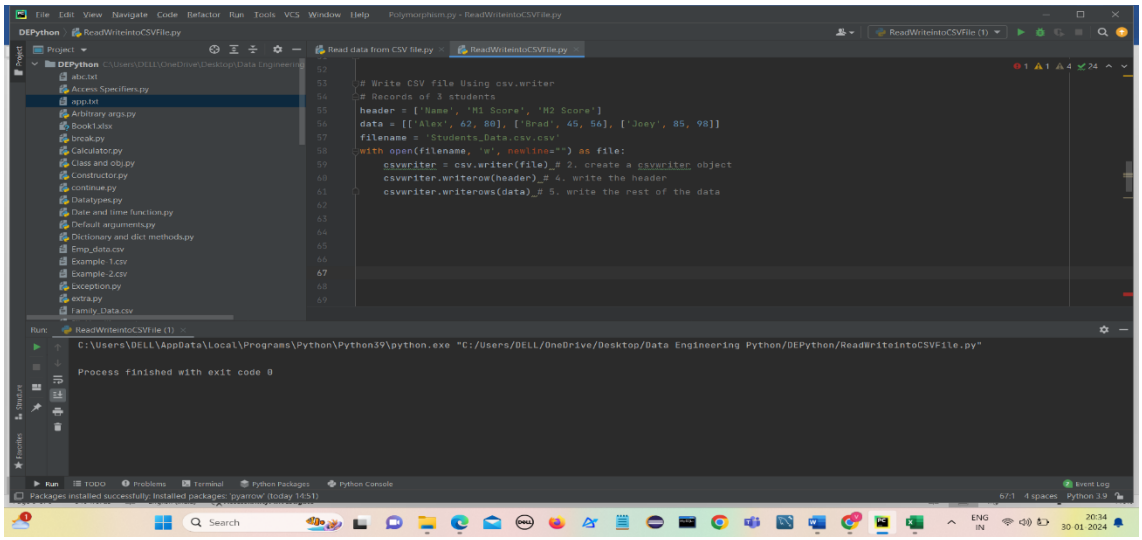
- The dict() method is used to create a dictionary object from either a specified set or iterables of keys and values.
- The csv module .DictReader is used to read CSV files.



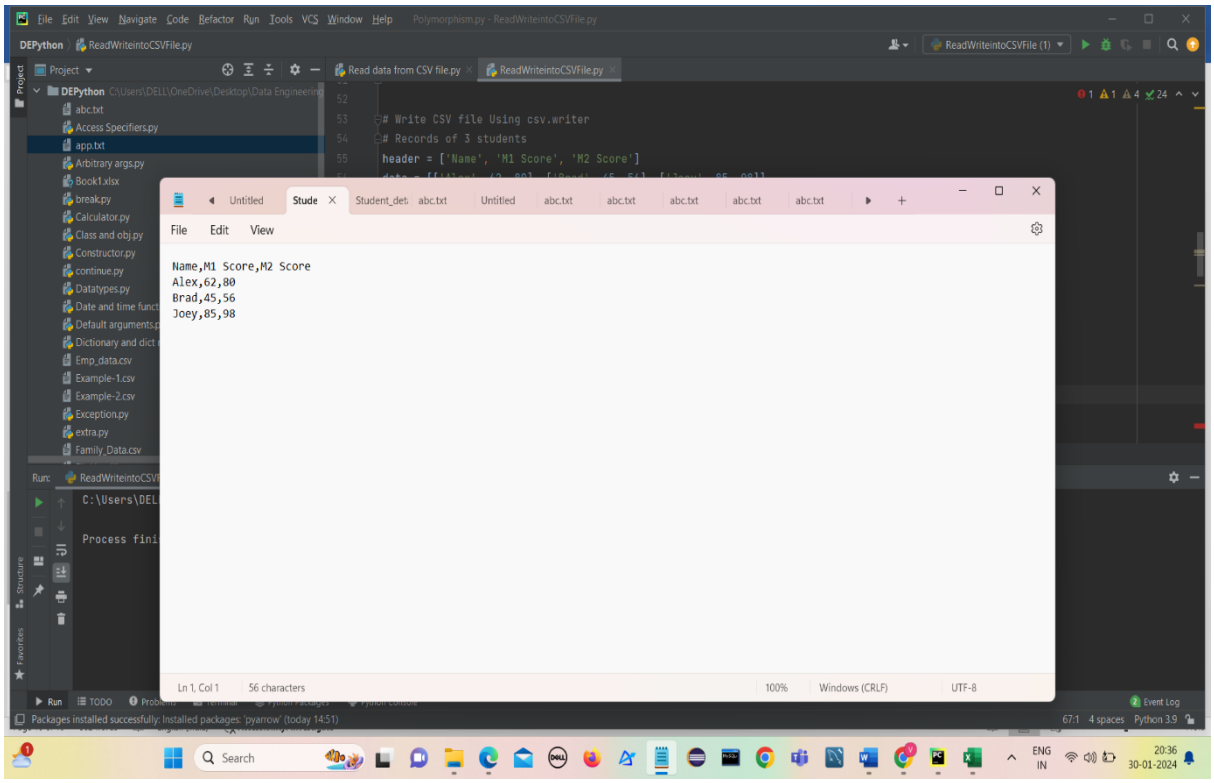
List of Methods to Write a CSV File in Python

1. Write CSV file using csv.writer
2. Write CSV file using .writelines() function
3. Write CSV file using Pandas
4. Write CSV file using csv.DictWriter

1) Write CSV file using csv.writer

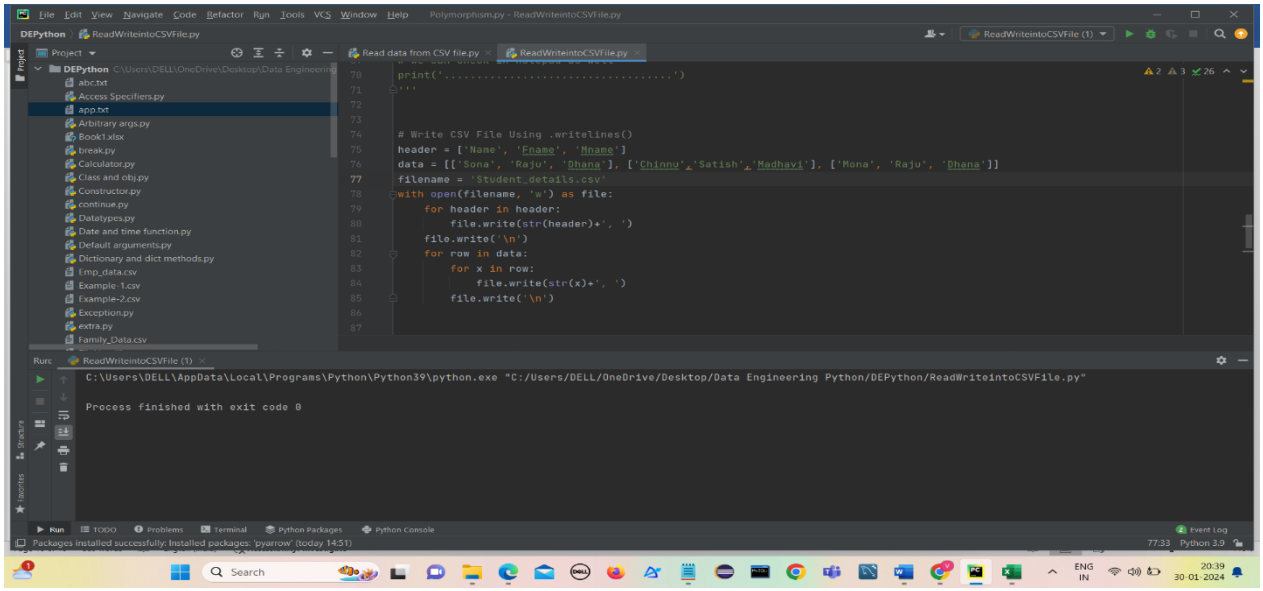


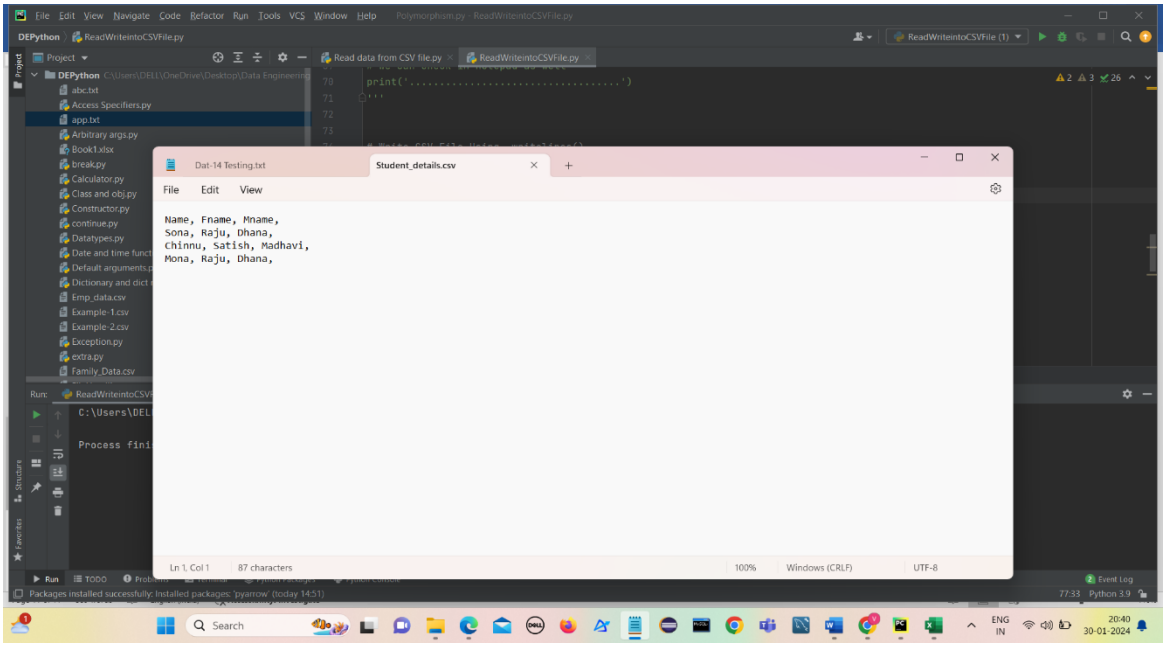
The csv.writer() function returns a writer object that converts the input data into a delimited string.



2) Write CSV file using .writelines() function

.writelines() iterates through each list, converts the list elements to a string, and then writes it to the csv file.





3) Write CSV file using Pandas

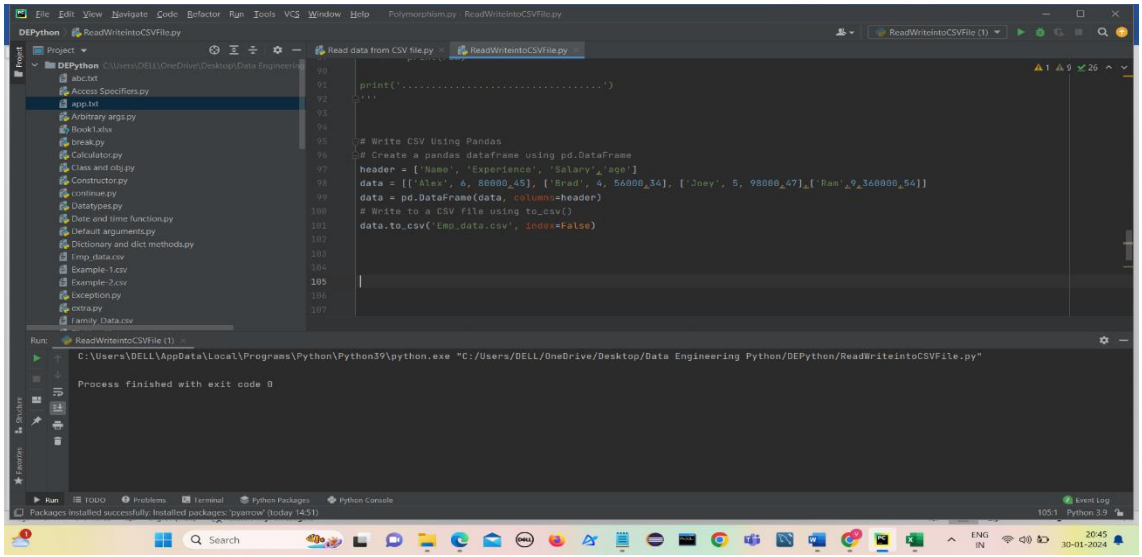
Create a pandas dataframe using pd.DataFrame

Syntax: pd.DataFrame(data, columns)

The data parameter takes the records/observations, and the columns parameter takes the columns/field names.

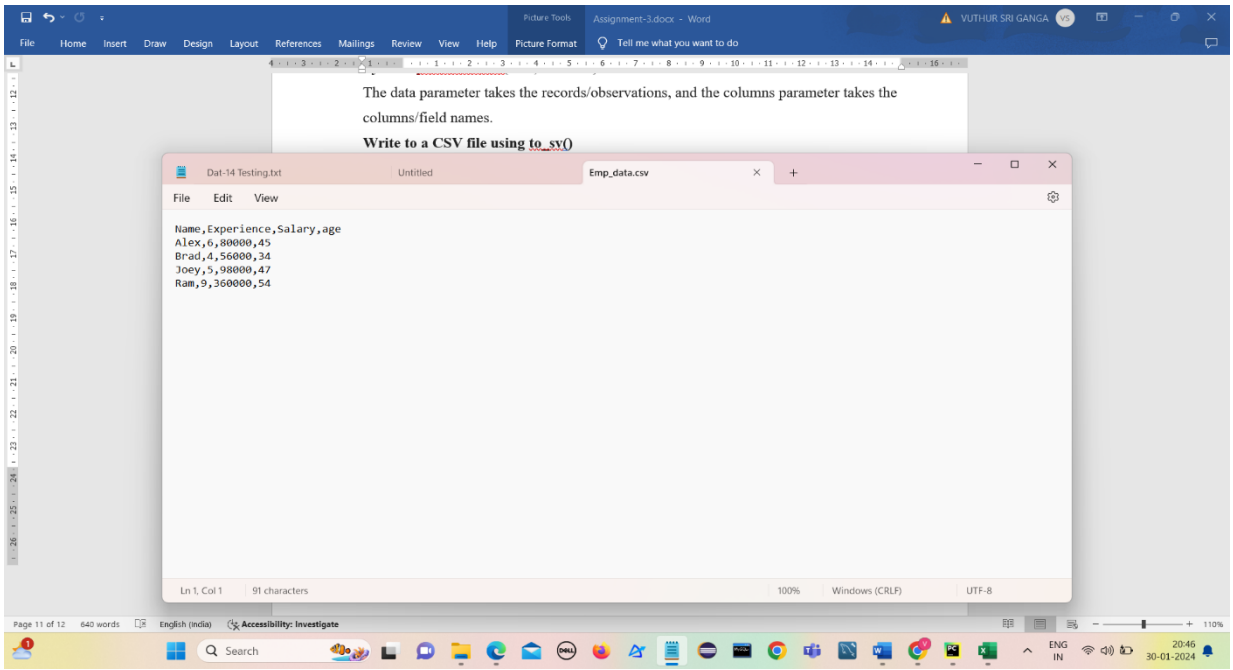
Write to a CSV file using to_sv()

Syntax: DataFrame.to_csv(filename, sep=',', index=False)



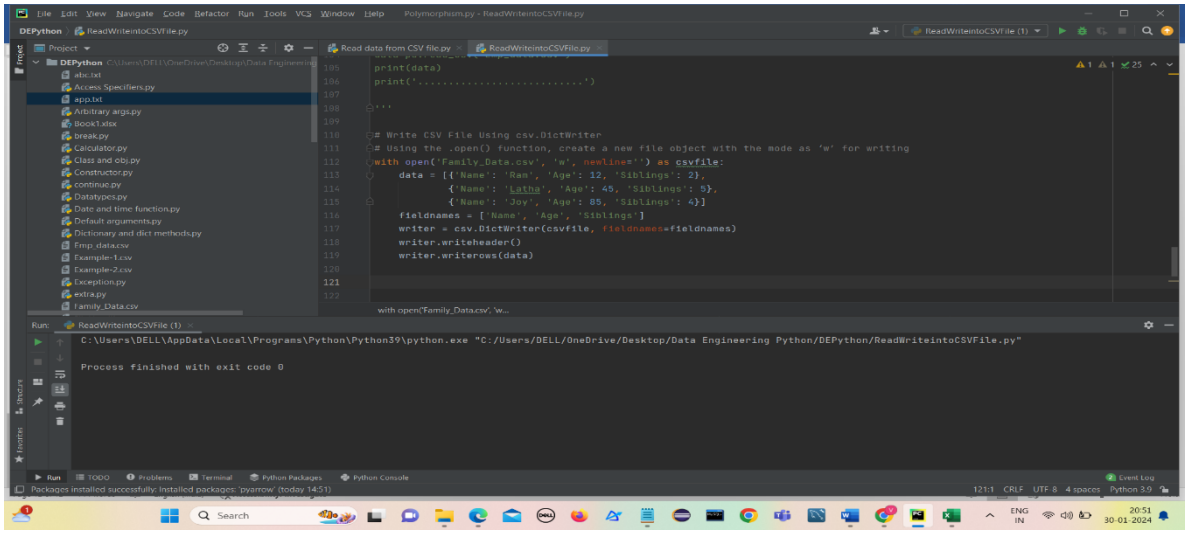
**separator is ',' by default.

index=False to remove the index numbers.

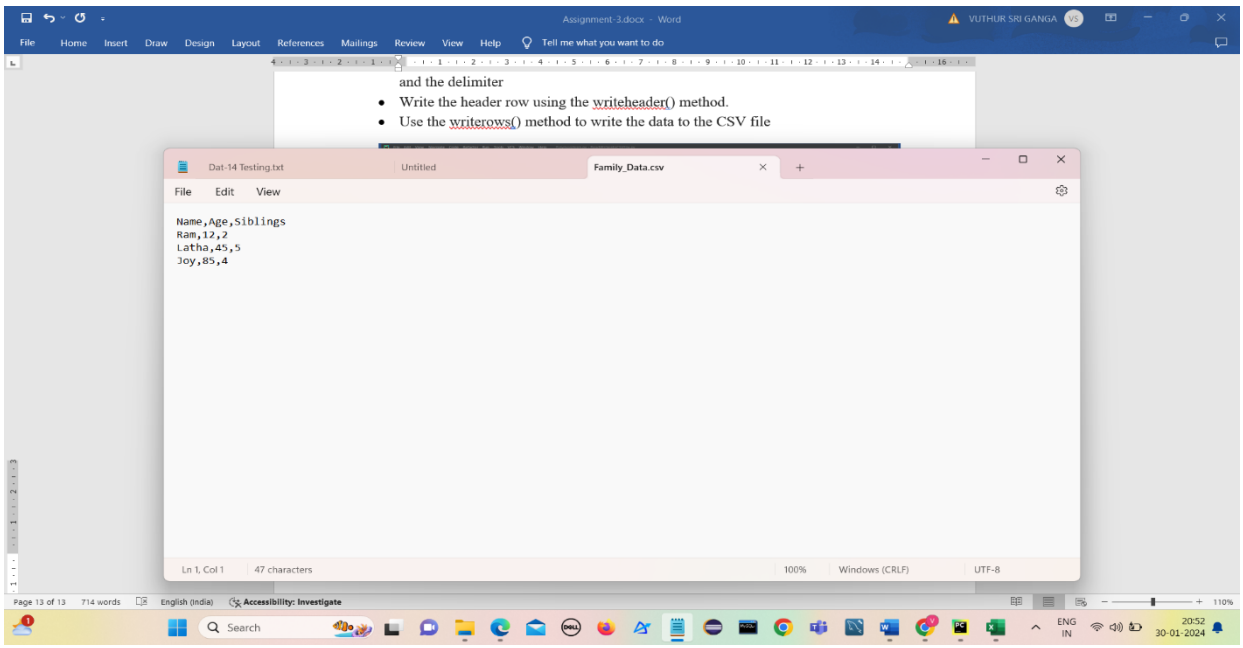


4) Write CSV file using csv.DictWriter

- Using the .open() function, create a new file object with the mode as ‘w’ for writing
- Type in the data you want to write to the CSV file as a list of dictionaries
- Create a csv.DictWriter object specifying the file object, the fieldname parameters, and the delimiter
- Write the header row using the writeheader() method.
- Use the writerows() method to write the data to the CSV file



-



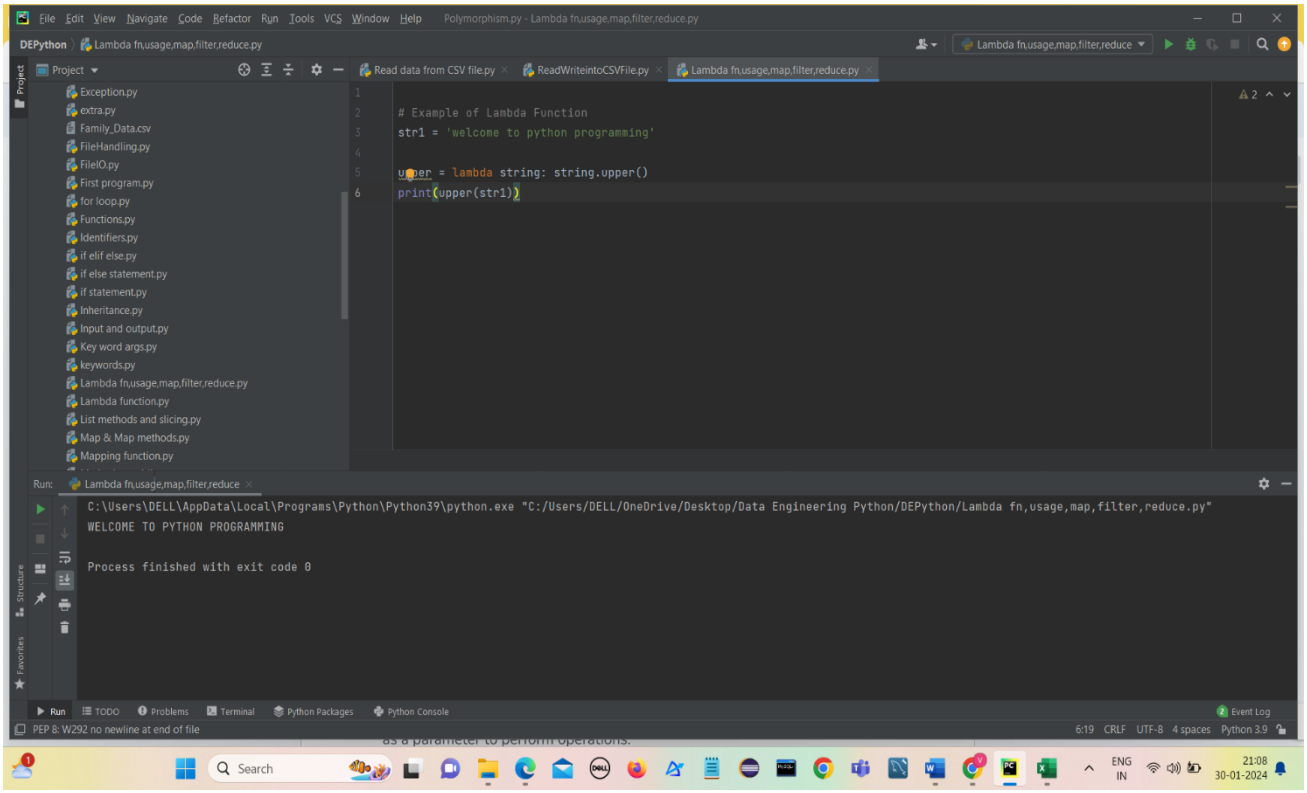
4) Lambda Functions

Python Lambda Functions are anonymous functions means that the function is without a name.

Syntax: lambda arguments : expression

This function can have any number of arguments but only one expression, which is evaluated and returned.

Example:



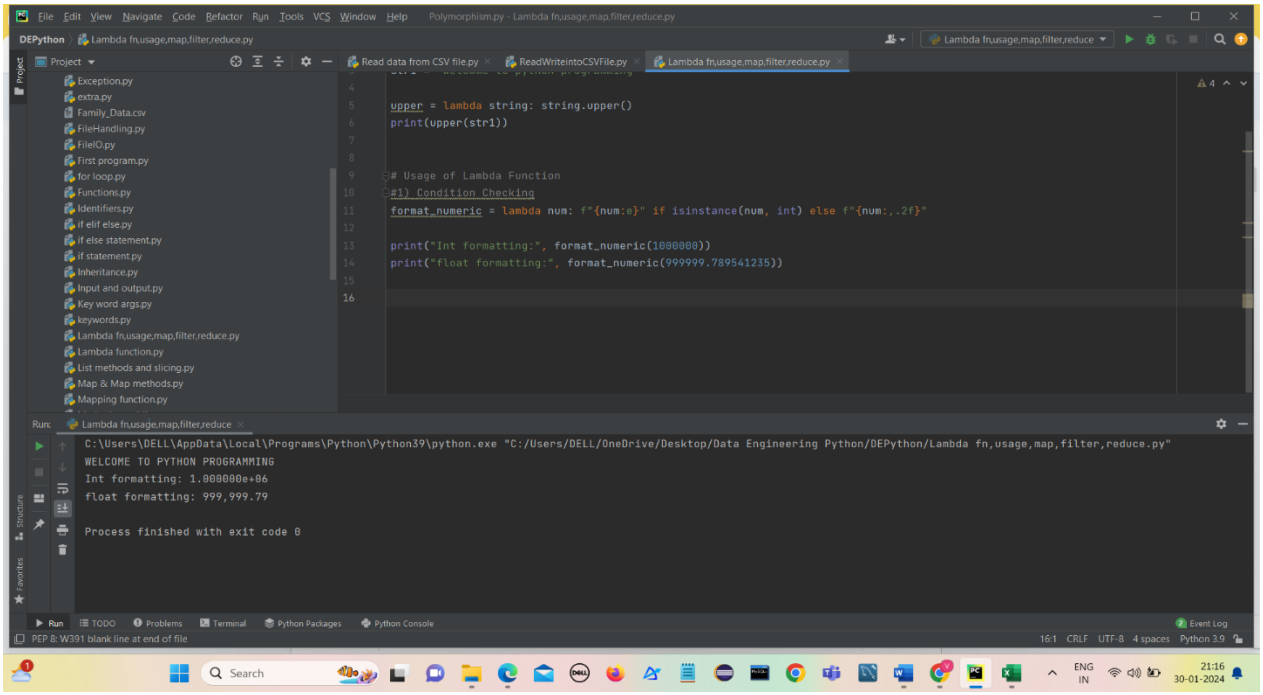
In the example, we defined a lambda function(**upper**) to convert a string to its upper case using upper().

This code defines a lambda function named **upper** that takes a string as its argument and converts it to uppercase using the **upper()** method

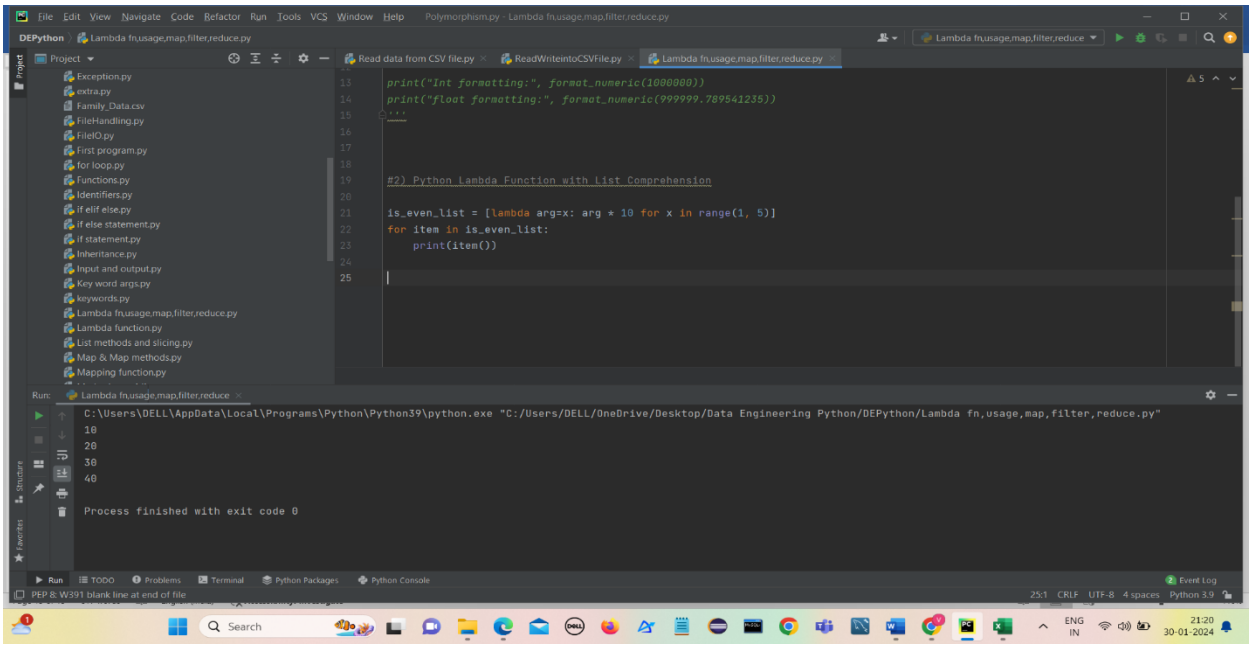
5) Usage of Lambda Functions:

a) Condition Checking Using Python lambda function

'format_numric' calls the lambda function, and the num is passed as a parameter to perform operations.

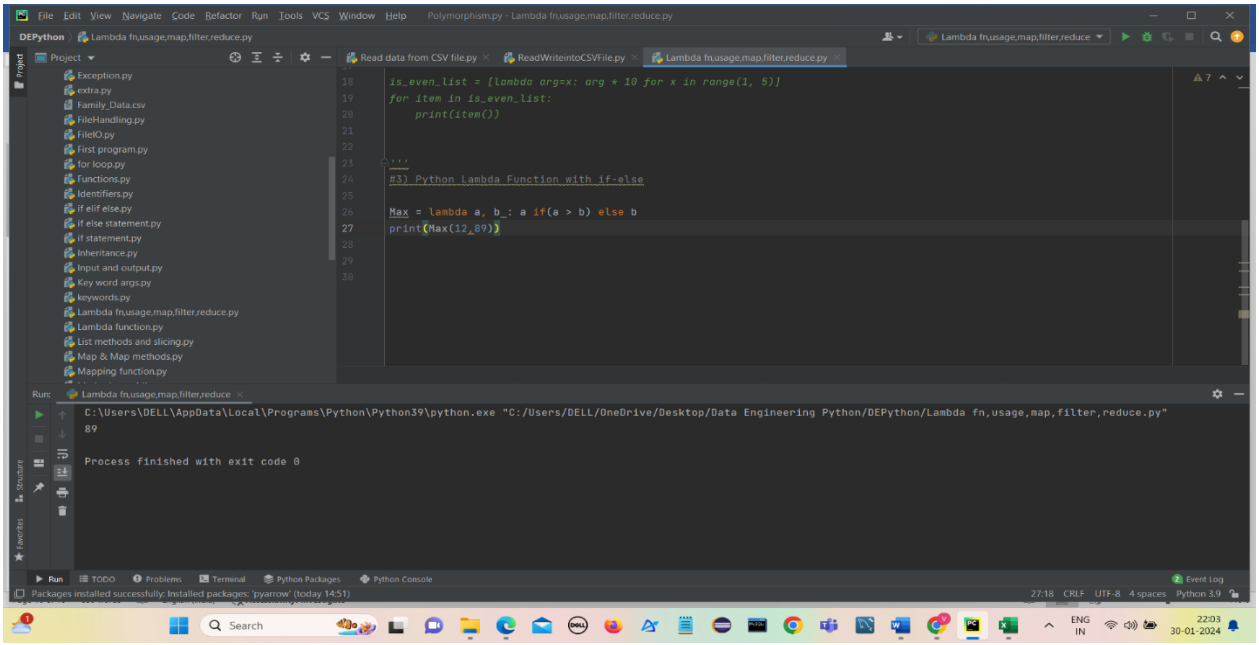


b) Python Lambda Function with List Comprehension



Lambda function is used in list comprehension.

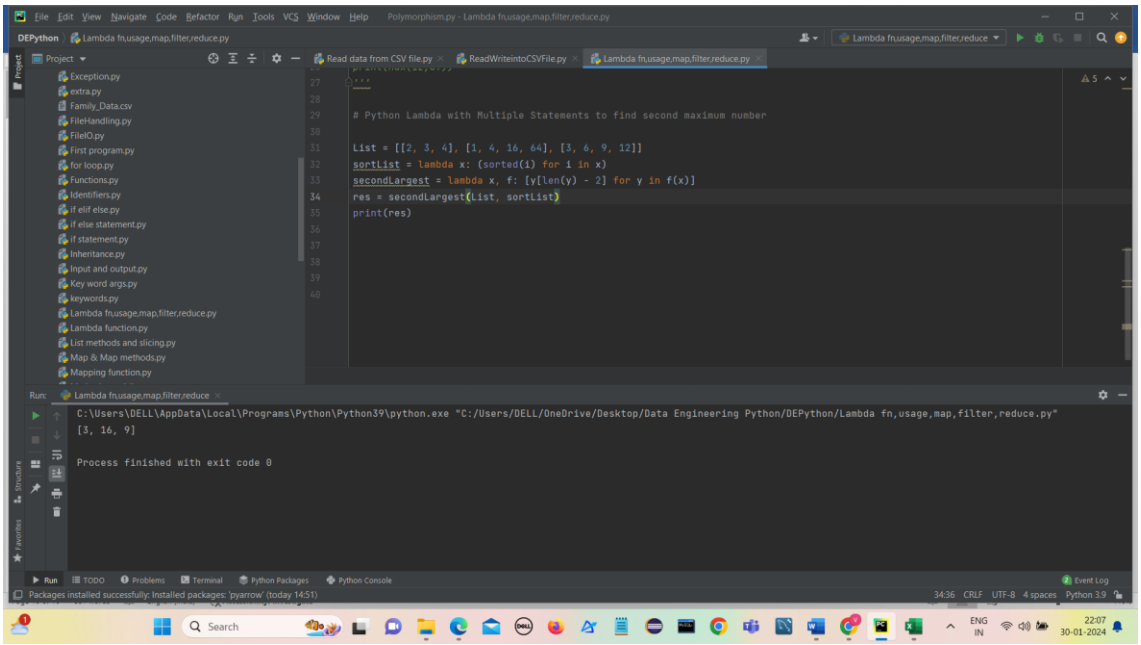
c) Python Lambda Function with if-else



Max lambda function is used to find the maximum of two integers.

d) Python Lambda with Multiple Statements to find second maximum number

Lambda functions do not allow multiple statements, however, we can create two lambda functions and then call the other lambda function as a parameter to the first function to find second maximum number.



6)Filter data in python lists using map, filter and reduce

a) Using lambda() Function with filter()

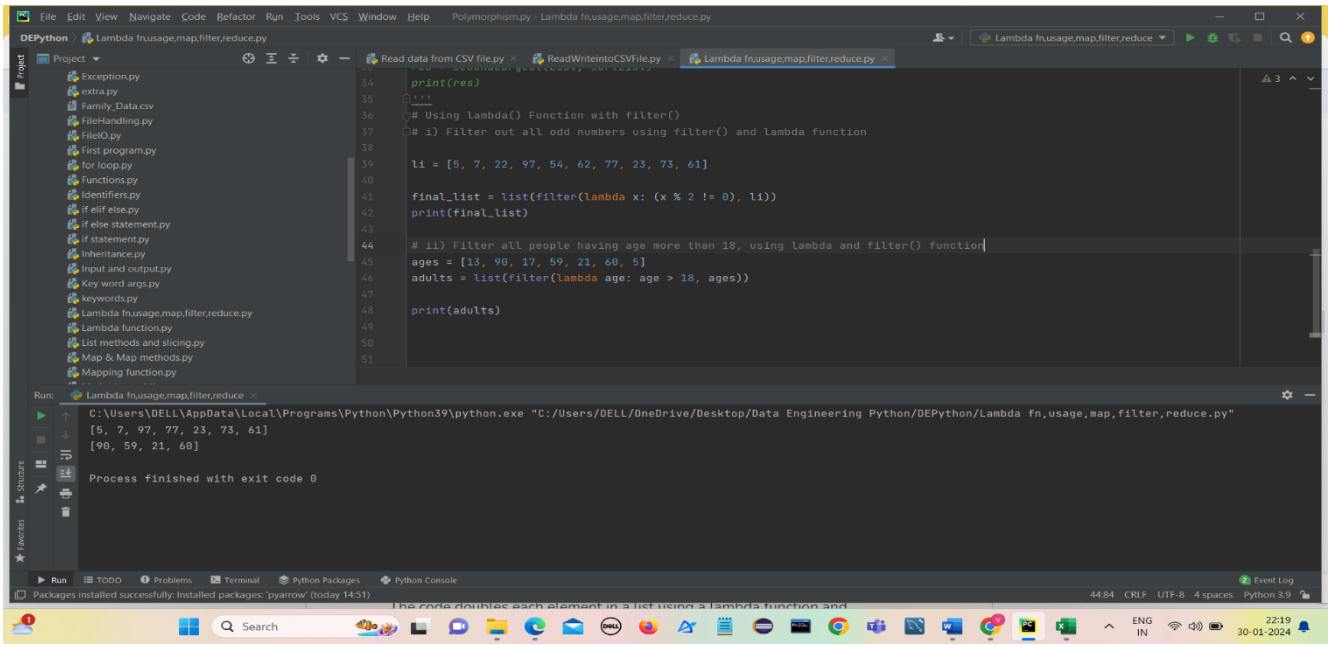
The filter() function in Python takes in a function and a list as arguments to filter out all the elements of a sequence for which result is true.

i) Filter out all odd numbers using filter() and lambda function

It filters a list and gives a list of odd numbers

ii) Filter all people having age more than 18, using lambda and filter() function.

It filters a list of ages and extracts the ages of adults (ages greater than 18) using a lambda function and the **'filter'** function. It then prints the list of adult ages.



b) Using lambda() Function with map()

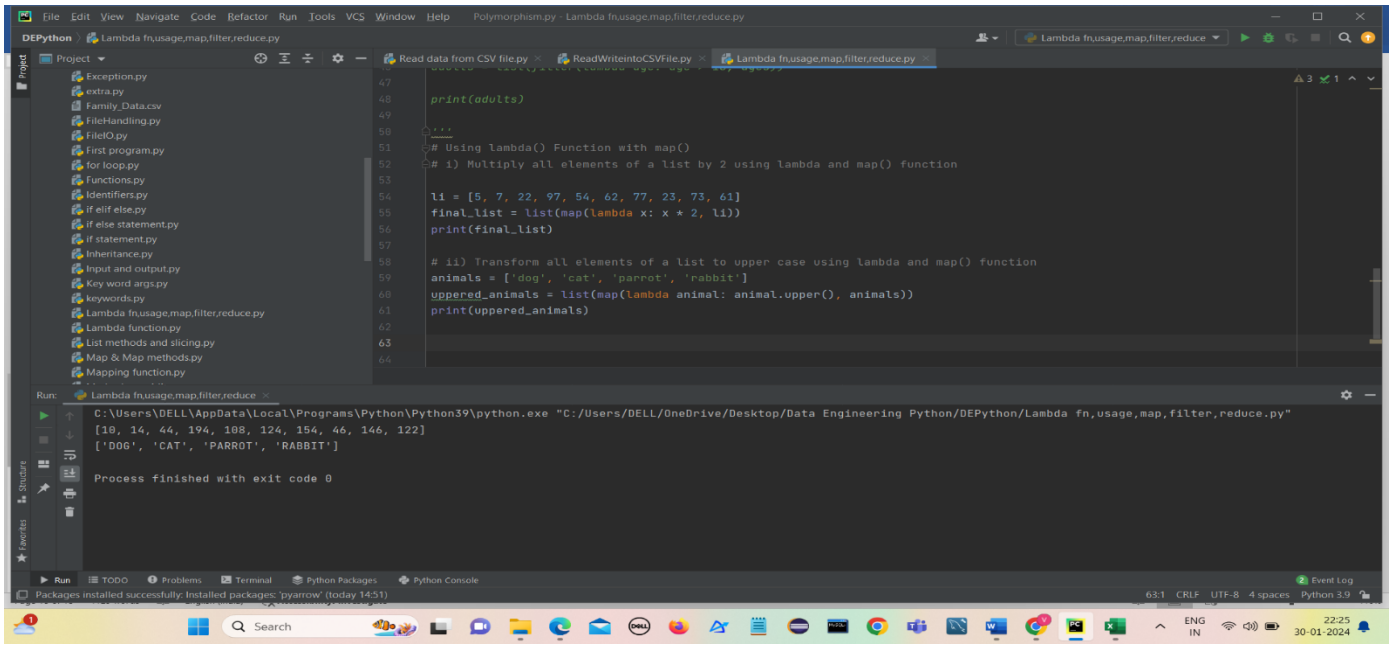
The map() function in Python takes in a function and a list as an argument.
The function is called with a lambda function and a list and a new list is returned which contains all the lambda-modified items returned by that function for each item.

i) Multiply all elements of a list by 2 using lambda and map() function

The code doubles each element in a list using a lambda function and the **'map'** function.
It then prints the new list with the doubled elements.

ii) Transform all elements of a list to upper case using lambda and map() function

The code converts a list of animal names to uppercase using a lambda function and the **'map'** function. It then prints the list with the animal names in uppercase.



c) Using lambda() Function with reduce()

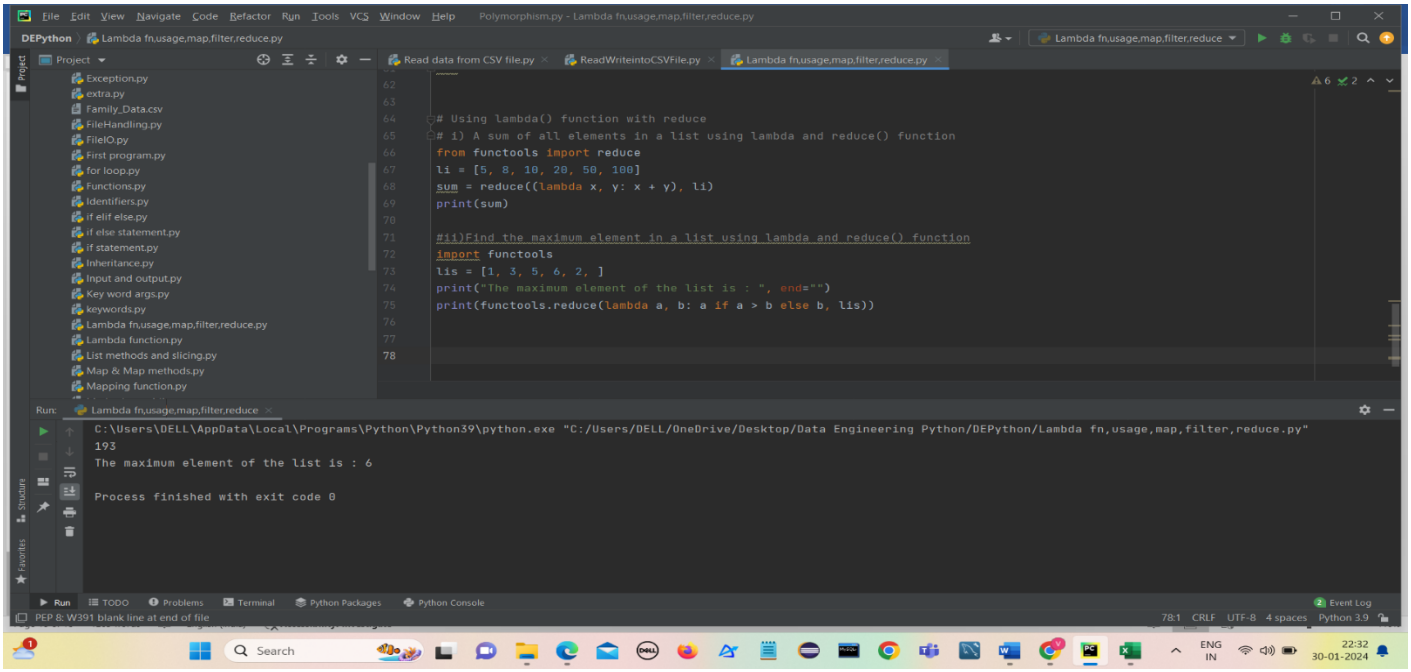
- The reduce() function in Python takes in a function and a list as an argument.
- The function is called with a lambda function and an iterable and a new reduced result is returned.
- This performs a repetitive operation over the pairs of the iterable.
- The reduce() function belongs to the ***functools*** module.

i) A sum of all elements in a list using lambda and reduce() function

It imports **'reduce'**, defines a list, applies a lambda function that adds two elements at a time, and prints the sum of all elements in the list.

ii) Find the maximum element in a list using lambda and reduce() function

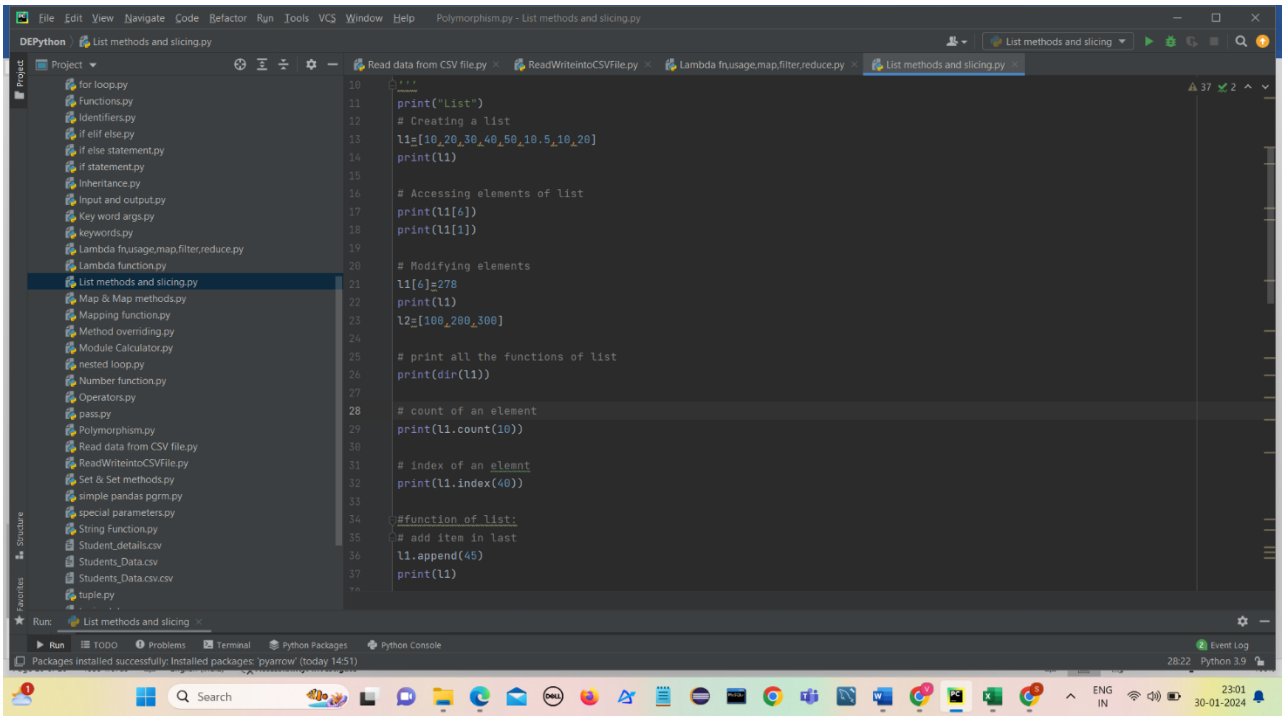
The code uses the **'functools'** module to find the maximum element in a list (**'lis'**) by employing the **'reduce'** function and a lambda function.

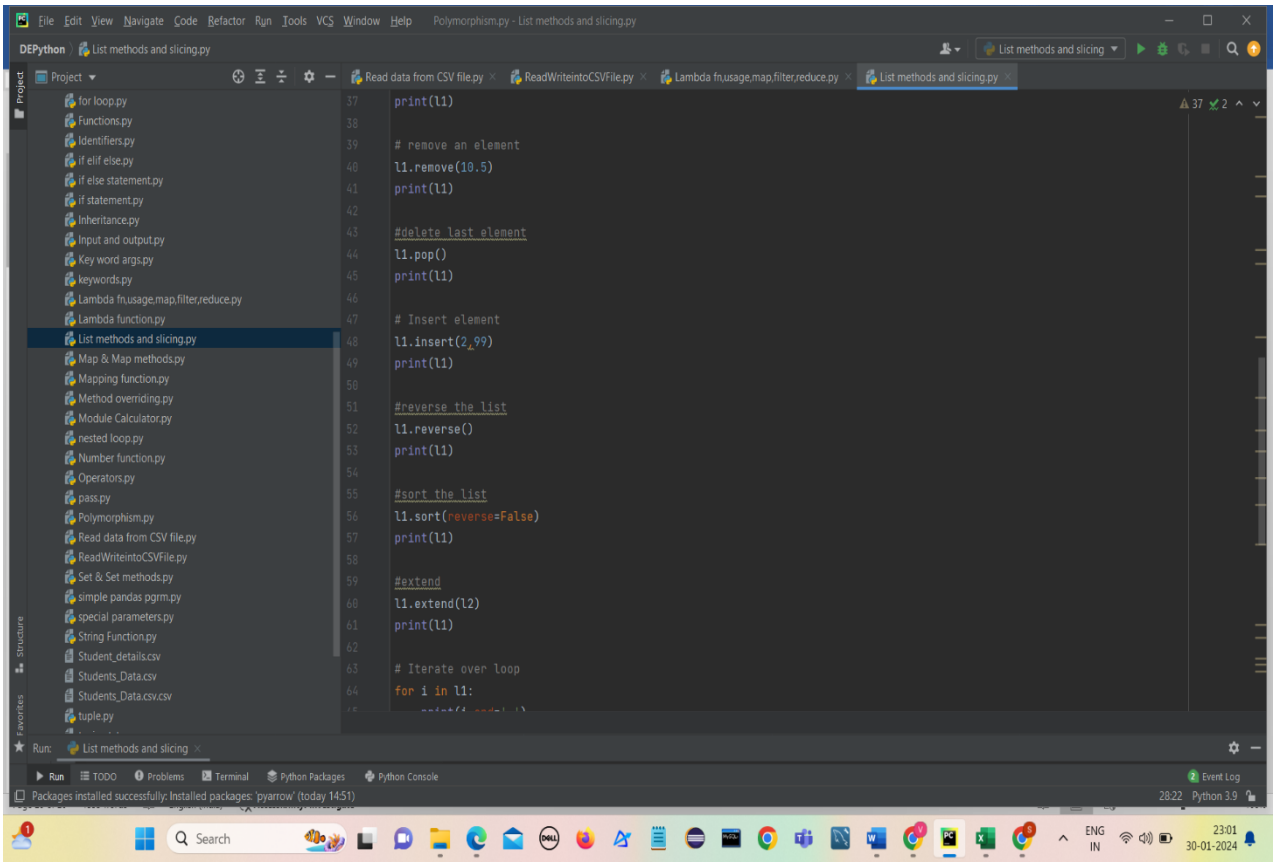


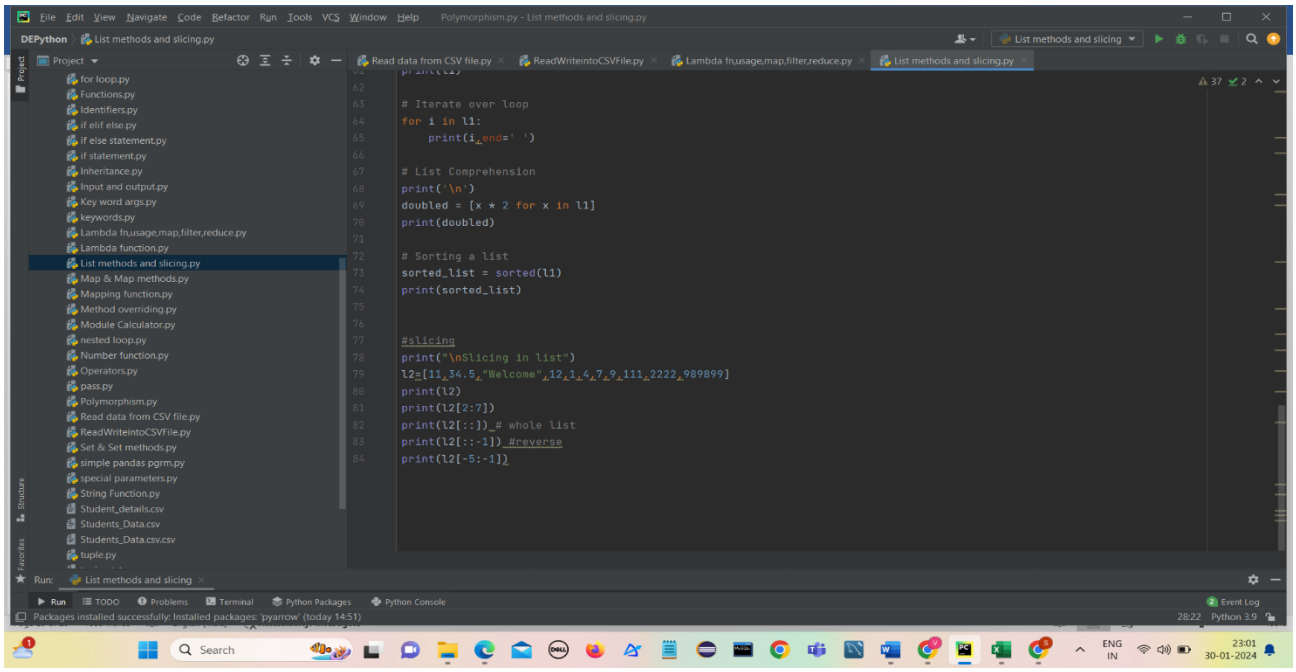
3. Processing python lists

Processing Python lists involves various operations such as creating lists, accessing elements, modifying elements, iterating through lists, and performing various transformations and manipulations.

a) Creating list
b) Accessing elements of list
c) Modifying elements
d) Count of element
e) Index of element
f) Add item in the last
g) Remove element
h) Delete last element
i) Insert element
j) Reverse list
k) Sort list
l) Extend list
m) Iterate over loop
n) List comprehension
o) Sorting list using sorted()
p) Slicing


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Polymorphism.py - List methods and slicing.py
DEPython
List methods and slicing.py
Project
Read data from CSV file.py
ReadWriteintoCSVFile.py
Lambda fn,usage,map,filter,reduce.py
List methods and slicing.py
for loop.py
Functions.py
Identifiers.py
if elif else.py
if else statement.py
if statement.py
Inheritance.py
Input and output.py
Key word args.py
keywords.py
Lambda fn,usage,map,filter,reduce.py
Lambda function.py
List methods and slicing.py
Map & Map methods.py
Mapping function.py
Method overriding.py
Module Calculator.py
nested loop.py
Number function.py
Operators.py
pass.py
Polymorphism.py
Read data from CSV file.py
ReadWriteintoCSVFile.py
Set & Set methods.py
simple pandas pgrm.py
special parameters.py
String Function.py
Student_details.csv
Students_Data.csv
Students_Data.csv.csv
tuple.py
'''
print("List")
# Creating a list
l1=[10,20,30,40,50,10.5,10,20]
print(l1)
# Accessing elements of list
print(l1[6])
print(l1[1])
# Modifying elements
l1[6]=278
print(l1)
l2=[100,200,300]
# print all the functions of list
print(dir(l1))
# count of an element
print(l1.count(10))
# index of an elemnt
print(l1.index(40))
#function of list:
# add item in last
l1.append(45)
print(l1)
Run: List methods and slicing
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
28:22 Python 3.9
Search
ENG IN
23:01 30-01-2024


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Polymorphism.py - List methods and slicing.py
DEPython
List methods and slicing.py
Project
Read data from CSV file.py
ReadWriteintoCSVFile.py
Lambda fn,usage,map,filter,reduce.py
List methods and slicing.py
print(l1)
# remove an element
l1.remove(10.5)
print(l1)
#delete last element
l1.pop()
print(l1)
# Insert element
l1.insert(2,99)
print(l1)
#reverse the list
l1.reverse()
print(l1)
#sort the list
l1.sort(reverse=False)
print(l1)
#extend
l1.extend(l2)
print(l1)
# Iterate over loop
for i in l1:
Run: List methods and slicing
Run TODO Problems Terminal Python Packages Python Console
Event Log
Packages installed successfully: Installed packages: 'pyarrow' (today 14:51)
28:22 Python 3.9
Search
ENG IN
23:01 30-01-2024



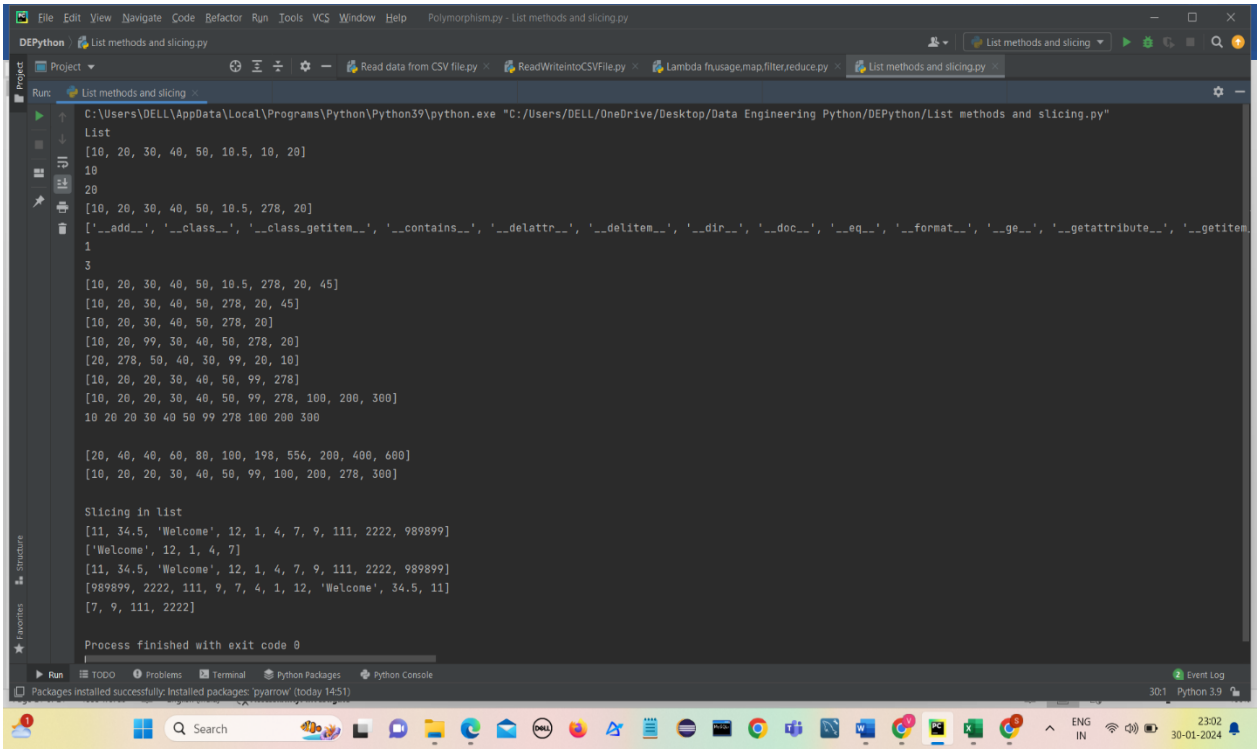
```
# Iterate over loop
for i in l1:
    print(i,end=' ')

# List Comprehension
print('\n')
doubled = [x * 2 for x in l1]
print(doubled)

# Sorting a list
sorted_list = sorted(l1)
print(sorted_list)


#slicing
print("\nSlicing in list")
l2=[11,34.5,"Welcome",12,1,4,7,9,111,2222,989899]
print(l2)
print(l2[2:7])
print(l2[::])_# whole list
print(l2[::-1]) #reverse
print(l2[-5:-1])
```

Output:



```
C:\Users\DELL\AppData\Local\Programs\Python\Python39\python.exe "C:/Users/DELL/OneDrive/Desktop/Data Engineering Python/DEPython/List methods and slicing.py"
List
[10, 20, 30, 40, 50, 10.5, 10, 20]
10
20
[10, 20, 30, 40, 50, 10.5, 278, 20]
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem
1
3
[10, 20, 30, 40, 50, 10.5, 278, 20, 45]
[10, 20, 30, 40, 50, 278, 20, 45]
[10, 20, 30, 40, 50, 278, 20]
[10, 20, 99, 30, 40, 50, 278, 20]
[20, 278, 50, 40, 30, 99, 20, 10]
[10, 20, 20, 30, 40, 50, 99, 278]
[10, 20, 20, 30, 40, 50, 99, 278, 100, 200, 300]
10 20 20 30 40 50 99 278 100 200 300

[20, 40, 40, 60, 80, 100, 198, 556, 200, 400, 600]
[10, 20, 20, 30, 40, 50, 99, 100, 200, 278, 300]

Slicing in list
[11, 34.5, 'Welcome', 12, 1, 4, 7, 9, 111, 2222, 989899]
['Welcome', 12, 1, 4, 7]
[11, 34.5, 'Welcome', 12, 1, 4, 7, 9, 111, 2222, 989899]
[989899, 2222, 111, 9, 7, 4, 1, 12, 'Welcome', 34.5, 11]
[7, 9, 111, 2222]

Process finished with exit code 0
```