What is a CSV?

CSV stands for “Comma Separated Values.” It is the simplest form of storing data in tabular form as plain text.The first line of a CSV file is the header. It contains the names of the fields/features

Methods to Read a CSV File in Python

1. Using csv.reader function by importing csv module : The next() method returns the current row and moves to the next row.

csv_reader.py

```python
import csv
rows = []
with open("Employees.csv", 'r') as file:
    csvreader = csv.reader(file)
    header = next(csvreader)
    for row in csvreader:
        rows.append(row)
    print(header)
    print(rows)
```

```
PS D:\Python> python -u "d:\Python\DE\csv_reader.py"
['EmpID', 'Name', 'Salary']
[['e101', 'Pramod', '1200000'], ['e120', 'Dinesh', '2200000'], ['e205', 'Sabesta', '1500000'], ['e331', 'Harry', '1700000'], ['e421', 'Avinash', '1300000'], ['e231', 'Joy', '230
0000'], ['e222', 'Smith', '2100000'], ['e339', 'Khan', '1800000'], ['e150', 'Dilip', '1900000'], ['e131', 'Kiran', '800000']]
PS D:\Python>
```

2. Reading CSV file using .readlines() function
   Using readlines() which returns a list of data in the file then slicing the list to display header and rows.

csv_readlines.py

```python
with open('Employees.csv') as file:
    content = file.readlines()
header = content[:1]
rows = content[1:]
print(header)
print(rows)
```

```
PS D:\Python> python -u "d:\Python\DE\csv_readlines.py"
['EmpID,Name,Salary\n']
['e101,Pramod,1200000\n', 'e120,Dinesh,2200000\n', 'e205,Sabesta,150000
0\n', 'e331,Harry,1700000\n', 'e421,Avinash,1300000\n', 'e231,Joy,23000
00\n', 'e222,Smith,2100000\n', 'e339,Khan,1800000\n', 'e150,Dilip,19000
00\n', 'e131,Kiran,800000']
PS D:\Python>
```

3. Reading CSV file using pandas library:

Using read_csv(file_name) and storing in the variable and then retrieving columns and rows using .columns and .column_name.

csv_pandas.py

```python
import pandas as pd
data = pd.read_csv("Employees.csv")
print(data)
print(data.columns) # retreives header names
print(data.Salary) # retrieves rows for Salary column
```

```
PS D:\Python> python -u "d:\Python\DE\csv_pandas.py"
  EmpID     Name   Salary
0  e101   Pramod  1200000
1  e120   Dinesh  2200000
2  e205  Sabesta  1500000
3  e331    Harry  1700000
4  e421  Avinash  1300000
5  e231      Joy  2300000
6  e222    Smith  2100000
7  e339     Khan  1800000
8  e150    Dilip  1900000
9  e131    Kiran   800000
Index(['EmpID', 'Name', 'Salary'], dtype='object')
0    1200000
1    2200000
2    1500000
3    1700000
4    1300000
5    2300000
6    2100000
7    1800000
8    1900000
9     800000
Name: Salary, dtype: int64
PS D:\Python>
```

4. Using .dictReader() to read csv file : Dict is a hash table of keys and values structured in Python. 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.

csv_dictreader.py

```python
# reading csv using DictReader()
import csv
with open('Employees.csv','r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        print(row)
```

```
PS D:\Python> python -u "d:\Python\DE\csv_dictreader.py"
{'EmpID': 'e101', 'Name': 'Pramod', 'Salary': '1200000'}
{'EmpID': 'e120', 'Name': 'Dinesh', 'Salary': '2200000'}
{'EmpID': 'e205', 'Name': 'Sabesta', 'Salary': '1500000'}
{'EmpID': 'e331', 'Name': 'Harry', 'Salary': '1700000'}
{'EmpID': 'e421', 'Name': 'Avinash', 'Salary': '1300000'}
{'EmpID': 'e231', 'Name': 'Joy', 'Salary': '2300000'}
{'EmpID': 'e222', 'Name': 'Smith', 'Salary': '2100000'}
{'EmpID': 'e339', 'Name': 'Khan', 'Salary': '1800000'}
{'EmpID': 'e150', 'Name': 'Dilip', 'Salary': '1900000'}
{'EmpID': 'e131', 'Name': 'Kiran', 'Salary': '800000'}
PS D:\Python>
```

Methods to Write a CSV file in python

1. **Write CSV file using csv.writer :** Creating a new csv file and opening it in write mode. Then creating a csv.writer object and adding the header and data rows using writerow() and writerows() from csv module.

csv_writer.py

```python
import csv
header = ['Name', 'M1 Score', 'M2 Score']
data = [['Mitushi', 62, 80], ['Vaishali', 45, 56], ['Rohit', 85, 98]]
filename = 'Students_Data.csv'
with open(filename, 'w', newline="") as file:
    csvwriter = csv.writer(file) # create a csvwriter object
    csvwriter.writerow(header) #  write the header
    csvwriter.writerows(data) # write the rest of the data
```

Students_Data.csv

```
Name,M1 Score,M2 Score
Mitushi,62,80
Vaishali,45,56
Rohit,85,98
```

2. Write CSV File Using writelines() : writelines() iterates through each list, converts the list elements to a string, and then writes it to the csv file.

csv_writelines.py

```python
header = ['Name','Score', 'Class']
rows = [['Mitushi', 90 , 12],['Aayushi',90,12],['Himanshi',90,12]]
filename = 'Student_scores.csv'
with open(filename,'w') as file:
    for value in header:
        file.write(str(value) +', ')
    file.write('\n')
    for row in rows:
        for data in row:
            file.write(str(data)+', ')
        file.write('\n')
```

Student_scores.csv

```
Name, Score, Class,
Mitushi, 90, 12,
Aayushi, 90, 12,
Himanshi, 90, 12,
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Python> python -u "d:\Python\DE\csv_writelines.py"
PS D:\Python>
```

3. Write in CSV using pandas

Writing into csv using dataframe() and columns attribute takes the list of header names. The to_csv() function from pandas is used to write into csv files.

csv_write_pandas.py

DE > csv_write_pandas.py > ...

```python
import pandas as pd
header = ['Name','Score', 'Class']
rows = [['Mitushi', 90 , 12],['Aayushi',90,12],['Himanshi',90,12]]
data = pd.DataFrame(rows, columns=header)
data.to_csv('Stud_score.csv', index=False)
```

Stud_score.csv

```
Name,Score,Class
Mitushi,90,12
Aayushi,90,12
Himanshi,90,12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
PS D:\Python> python -u "d:\Python\DE\csv_writelines.py"
PS D:\Python> python -u "d:\Python\DE\csv_write_pandas.py"
```

4. Write CSV File Using csv.DictWriter : Writing into csv file by creating dictionary and using DictWriter() with fieldnames parameter

csv_dictwriter.py

DE > csv_dictwriter.py > ...

```python
import csv
data=[{'Name':'Rudra','Age':25,'Salary':300000},
    {'Name':'Krushna','Age':24,'Salary':400000},
    {'Name':'Ram','Age':21,'Salary':20000}]
with open('Emp_Data.csv','w',newline='') as file:
    field_names=['Name', 'Age', 'Salary']
    writer=csv.DictWriter(file,fieldnames=field_names)
    writer.writeheader()
    writer.writerows(data)
```

Emp_Data.csv

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

Read CSV file data into list :

csv_list.py

DE > csv_list.py > ...

```python
import csv
with open("Employees.csv", 'r') as file:
    csvreader = csv.reader(file)
    print(csvreader)
    list_csv = list(csvreader)
    print(list_csv)
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS D:\Python> python -u "d:\Python\DE\csv_list.py"
<_csv.reader object at 0x000001D64A3ADAE0>
[['EmpID', 'Name', 'Salary'], ['e101', 'Pramod', '1200000'], ['e120',
'Dinesh', '2200000'], ['e205', 'Sabesta', '1500000'], ['e331', 'Harry'
, '1700000'], ['e421', 'Avinash', '1300000'], ['e231', 'Joy', '2300000
'], ['e222', 'Smith', '2100000'], ['e339', 'Khan', '1800000'], ['e150'
, 'Dilip', '1900000'], ['e131', 'Kiran', '800000']]
PS D:\Python>
```

Pandas : Pandas is a library used for data manipulation in python. There are two types of data structures in pandas.

Series and DataFrame : Series being one-dimensional same as array capable of storing different types of values.

Explorer (Ctrl+Shift+E)

DE > pandas_practice.py > ...

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

# creating Series using numpy array
ser = pd.Series()
print("Pandas Series: ", ser)
data = np.array([1,2,3,4])
ser =pd.Series(data)
print("Pandas Series: ", ser)


# creating dataframe using dictionary
# Creating Dictionary
dict = {
    'series': ['Friends', 'Money Heist', 'Marvel'],
    'episodes': [200, 50, 45],
    'actors': [' David Crane', 'Alvaro', 'Stan Lee']
}

# Creating Dataframe
df = pd.DataFrame(dict)
print(df)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
  ser = pd.Series()
Pandas Series:  Series([], dtype: float64)
Pandas Series:  0    1
1    2
2    3
3    4
dtype: int32
        series  episodes        actors
0      Friends       200   David Crane
1  Money Heist        50        Alvaro
2       Marvel        45      Stan Lee
PS D:\Python>
```

Lambda Functions

Lambda functions are anonymous or nameless function written in a single expression. The syntax is like lambda arguments: expression. Lambda works for only one expression.

lambda_function.py

DE > Lambda_Functions > lambda_function.py > ...

```python
name = 'Mitushi'
upper = lambda string : string.upper()
print(upper(name))
```

TERMINAL

```
PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\lambda_function.py"
MITUSHI
PS D:\Python>
```

Python Lambda Function with if-else

if_else_lambda.py

DE > Lambda_Functions > if_else_lambda.py > ...

```python
format = lambda num: f"{num:e}" if isinstance(num,int) else f"{num:.2f}"
print("Int formatting",format(10000))
print("Float Formatting",format(20.2434444444))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\if_else_lambda.py"
Int formatting 1.000000e+04
Float Formatting 20.24
PS D:\Python>
```

Lambda inside list comprehension : Here a list of lambda functions is created using list comprehension and each list item is called outside the for loop.

```
list_lambda ×
DE > Lambda_Functions > list_lambda > ...
1   even_list = [lambda arg=x: arg * 10 for x in range(1, 5)]
2   for item in even_list:
3       print(item())

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Code

PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\list_lambda"
10
20
30
40
PS D:\Python>
```

Multiple arguments in lambda function with if-else

```
multiple_Arguments.py ×
DE > Lambda_Functions > multiple_Arguments.py > ...
1   Max = lambda a, b : a if(a > b) else b
2   print(Max(1, 2))

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORT

PS D:\Python> python -u "d:\Python\DE\Lambda_Func
ts.py"
2
PS D:\Python>
```

Filter with lambda : The filter() function in Python is used to filter elements from an iterable (e.g., a list) based on a given function.

lambda_filter.py

DE > Lambda_Functions > lambda_filter.py > ...

```python
#----------Example 1
words = ["apple", "banana", "cherry", "date", "elderberry"]

# Using filter with lambda
filtered_words = filter(lambda x: len(x) <= 5, words)

# Converting the filter object to a list for display
result_list = list(filtered_words)

print("Original words:", words)
print("Filtered words with length <= 5:", result_list)

#----------Example 2
# Filter out all odd numbers
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
print('Original List',li)
final_list = list(filter(lambda x: (x % 2 != 0), li))
print("list with odd numbers",final_list)

#----------Example 3
# Filter all people having age more than 18
ages = [13, 90, 17, 59, 21, 60, 5]
adults = list(filter(lambda age: age > 18, ages))

print(adults)
```

PROBLEMS OUTPUT TERMINAL ... Code

```
"
Original words: ['apple', 'banana', 'cherry', 'date', 'elderberry']
Filtered words with length <= 5: ['apple', 'date']
Original List [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
list with odd numbers [5, 7, 97, 77, 23, 73, 61]
[90, 59, 21, 60]
PS D:\Python>
```

Using lambda() Function with map()

The map() function in Python is used to apply a specific function to all items in an iterable (e.g., a list).The map() function in Python takes in a function and a list as an argument.

result = map(lambda x: expression(x), iterable)

lambda_map.py

DE > Lambda_Functions > lambda_map.py > ...

```python
# Multiply all elements of a list by 2

li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

final_list = list(map(lambda x: x*2, li))
print(final_list)

# Squaring numbers
numbers = [1, 2, 3, 4, 5]

# Using map with lambda to square each number
squared_numbers = map(lambda x: x ** 2, numbers)

# Converting the map object to a list for display
result_list = list(squared_numbers)

print("Original numbers:", numbers)
print("Squared numbers:", result_list)

# Transform all elements of a list to upper case
animals = ['dog', 'cat', 'parrot', 'rabbit']
uppered_animals = list(map(lambda animal: animal.upper(), animals))

print(uppered_animals)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code

```
Filtered words with length <= 5: ['apple', 'date']
Original List [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
list with odd numbers [5, 7, 97, 77, 23, 73, 61]
[90, 59, 21, 60]
PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\lambda_map.py"
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]
Original numbers: [1, 2, 3, 4, 5]
Squared numbers: [1, 4, 9, 16, 25]
['DOG', 'CAT', 'PARROT', 'RABBIT']
PS D:\Python>
```

Using lambda() Function with reduce()

The reduce() function is part of the functools module in Python and is used to apply a specified binary function to the items of an iterable successively to reduce the iterable to a single cumulative value.

lambda_reduce.py

DE > Lambda_Functions > lambda_reduce.py > ...

```python
# A sum of all elements in a list
import functools
li = [5, 8, 10, 20, 50, 100]
sum = functools.reduce((lambda x, y: x + y), li)
print(sum)

# Find the maximum element in a list
lis = [1, 3, 5, 6, 2, ]
print("The maximum element of the list is : ", end="")
print(functools.reduce(lambda a, b: a if a > b else b, lis))
```

PROBLEMS OUTPUT TERMINAL ... Code

```
Filtered words with length <= 5: ['apple', 'date']
Original List [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
list with odd numbers [5, 7, 97, 77, 23, 73, 61]
[90, 59, 21, 60]
PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\lambda_map.py"
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]
Original numbers: [1, 2, 3, 4, 5]
Squared numbers: [1, 4, 9, 16, 25]
['DOG', 'CAT', 'PARROT', 'RABBIT']
PS D:\Python> python -u "d:\Python\DE\Lambda_Functions\lambda_reduce.py
"
193
The maximum element of the list is : 6
PS D:\Python>
```