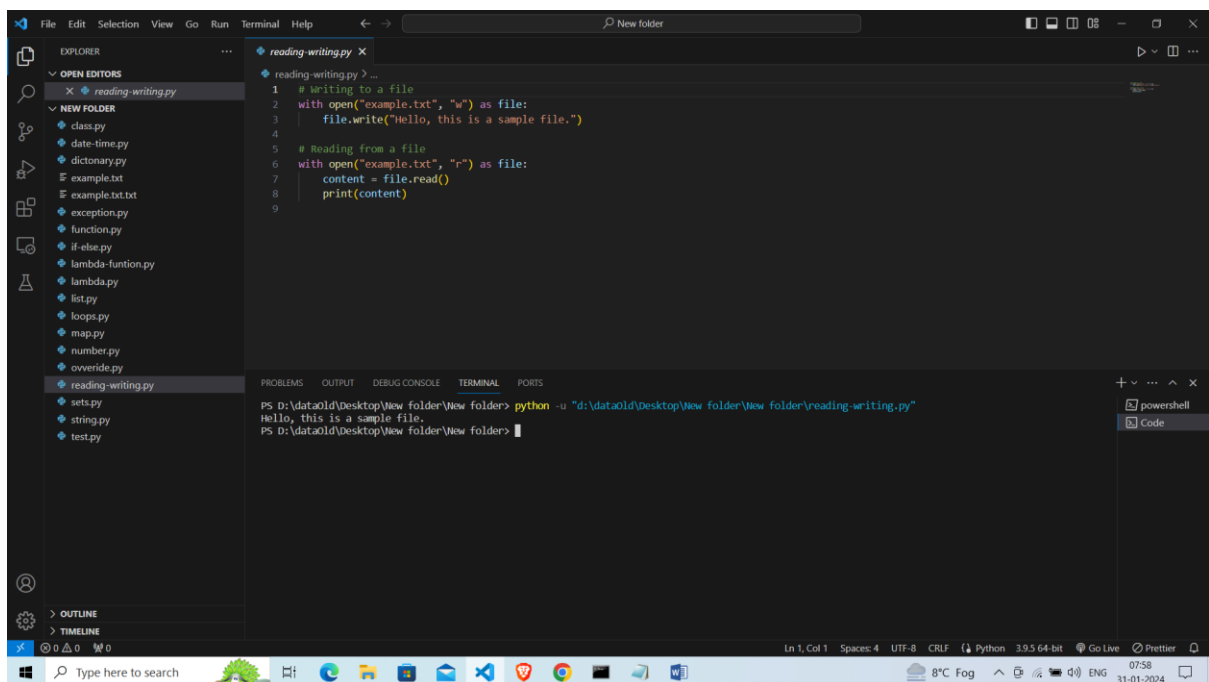


# Assignment - 8

## File I/O using Python:

File Input/Output (I/O) in Python allows you to interact with external files. The `open()` function is typically used to open a file. It takes two arguments - the name of the file and the mode in which the file is opened ('r' for reading, 'w' for writing, 'a' for appending, etc.). Once you're done with the file, it's good practice to close it using the `close()` method.

Example of reading and writing a file:



The screenshot shows a Python IDE with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays a Python script named `reading-writing.py` with the following content:

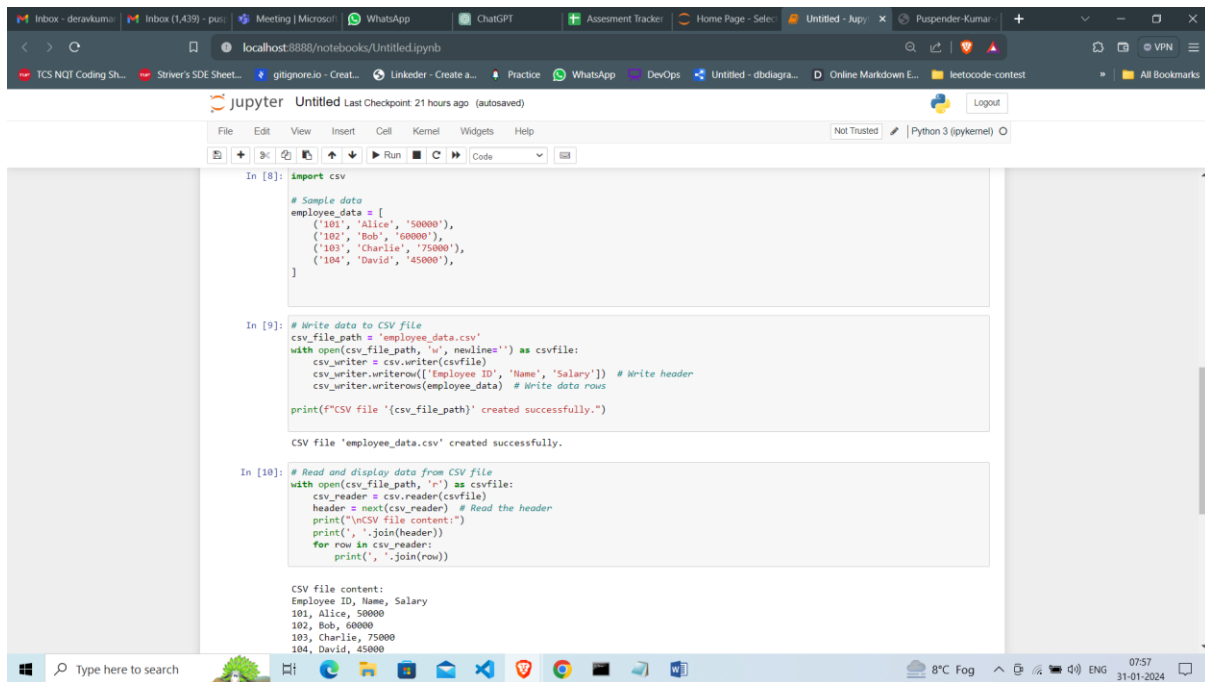
```
1 # Writing to a file
2 with open("example.txt", "w") as file:
3     file.write("Hello, this is a sample file.")
4
5 # Reading from a file
6 with open("example.txt", "r") as file:
7     content = file.read()
8     print(content)
9
```

The terminal at the bottom shows the command `python -u "d:\dataold\Desktop\New folder\New folder\reading-writing.py"` being executed, resulting in the output: `Hello, this is a sample file.`

## Read Data from CSV File into Python List:

CSV (Comma-Separated Values) files are a common format for storing tabular data. Python provides the `csv` module to handle CSV files.

Example of reading data from a CSV file into a list:



The screenshot shows a Jupyter Notebook titled 'Untitled' running on a local host. The notebook contains three code cells. The first cell imports the 'csv' module and defines a sample data list. The second cell writes this data to a CSV file named 'employee\_data.csv'. The third cell reads the data back from the CSV file and prints it. The output of the third cell shows the CSV file content as a table with columns: Employee ID, Name, Salary.

```
In [8]: import csv

# Sample data
employee_data = [
    ('101', 'Alice', '50000'),
    ('102', 'Bob', '60000'),
    ('103', 'Charlie', '75000'),
    ('104', 'David', '45000'),
]

In [9]: # Write data to CSV file
csv_file_path = 'employee_data.csv'
with open(csv_file_path, 'w', newline='') as csvfile:
    csv_writer = csv.writer(csvfile)
    csv_writer.writerow(['Employee ID', 'Name', 'Salary']) # Write header
    csv_writer.writerows(employee_data) # Write data rows

print(f"CSV file '{csv_file_path}' created successfully.")

CSV file 'employee_data.csv' created successfully.

In [10]: # Read and display data from CSV file
with open(csv_file_path, 'r') as csvfile:
    csv_reader = csv.reader(csvfile)
    header = next(csv_reader) # Read the header
    print("CSV file content:")
    print(','.join(header))
    for row in csv_reader:
        print(','.join(row))

CSV file content:
Employee ID, Name, Salary
101, Alice, 50000
102, Bob, 60000
103, Charlie, 75000
104, David, 45000
```

## Processing Python Lists:

In Python, a list is a versatile and commonly used data structure that allows you to store and manipulate a collection of items. Lists in Python are ordered and mutable, meaning you can change the contents of a list after it is created. You can perform various operations on lists, such as iterating through them, adding or removing elements, and applying functions to each element.

## Lambda Functions in Python:

A lambda function in Python is a small, anonymous function defined using the lambda keyword. Lambda functions are useful for creating short, throwaway functions without the need to formally define them using the def keyword. Lambda functions can take any number of arguments, but they can only have one expression. The syntax for a lambda function is: lambda arguments: expression.

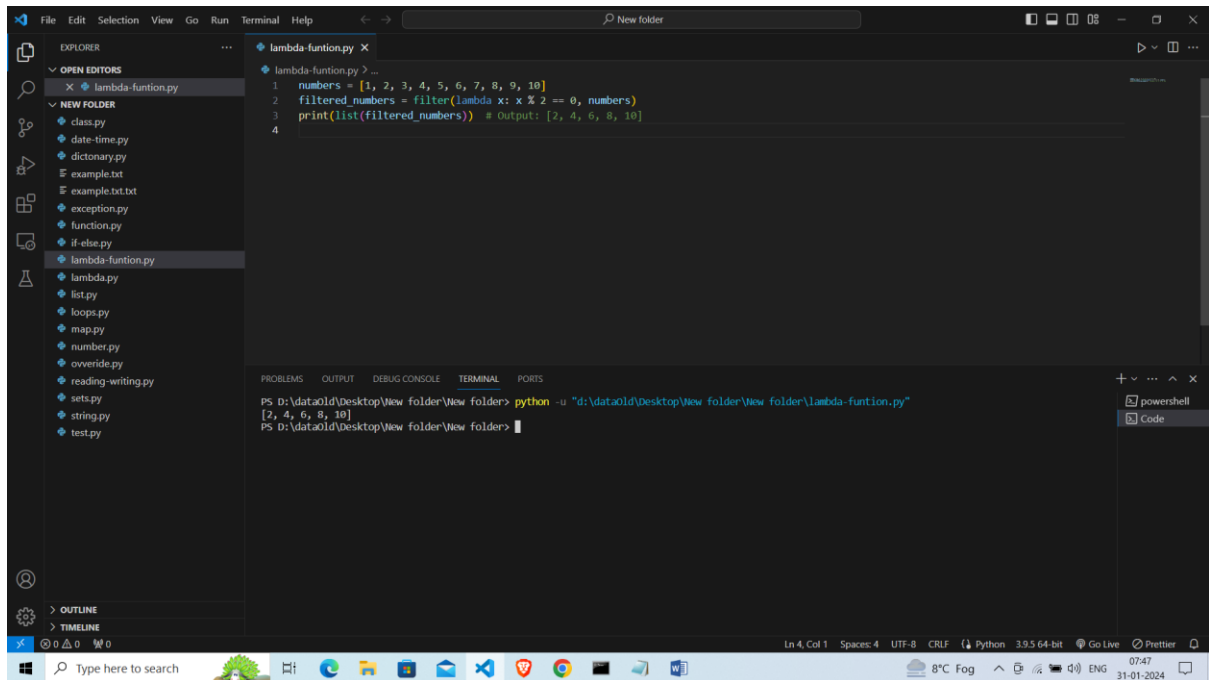
## Usage of Lambda Functions:

Lambda functions are often used in situations where a small function is required for a short period and defining a full function using def would be unnecessary. They are commonly used in functional programming constructs like map(), filter(), and reduce().

## Filter Data in Python Lists using filter and lambda:

The filter() function in Python is used to filter elements of an iterable based on a specified function. When combined with a lambda function, it becomes a concise way to filter elements from a list. The filter() function takes a function and an iterable, applying the function to each element in the iterable and returning only those for which the function evaluates to True.

## Example:



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer lists several Python files, including 'lambda-fun.py'. The main editor window displays the following Python code:

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 filtered_numbers = filter(lambda x: x % 2 == 0, numbers)
3 print(list(filtered_numbers)) # Output: [2, 4, 6, 8, 10]
4
```

The terminal at the bottom shows the command to run the script:

```
PS D:\dataold\Desktop\New folder\New folder> python -u "d:\dataold\Desktop\New folder\lambda-fun.py"
[2, 4, 6, 8, 10]
PS D:\dataold\Desktop\New folder\New folder>
```

## Use of Lambda Function in Python:

Lambda functions find application in situations where a small, anonymous function is needed for a short duration, such as in functional programming, sorting, filtering, and mapping.

## Practical Uses of Python lambda function:

- **Sorting:** Lambda functions are commonly used as the key function in sorting operations.

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left lists files in a project, with 'lambda-fun.py' selected. The main editor displays the following Python code:

```
1 # Using lambda function for sorting
2
3 names = ["Alice", "Bob", "Charlie", "David"]
4 sorted_names = sorted(names, key=lambda x: len(x))
5 print(sorted_names) # Output: ['Bob', 'Alice', 'David', 'Charlie']
6
```

The bottom panel shows the Terminal with the command `python -u "d:\dataold\Desktop\New folder\New folder\lambda-fun.py"` and its output: `['Bob', 'Alice', 'David', 'Charlie']`.

- GUI Applications: Lambda functions are sometimes used in GUI applications as quick event handlers.

### Using `lambda()` Function with `map()`, `filter()`, `reduce()`:

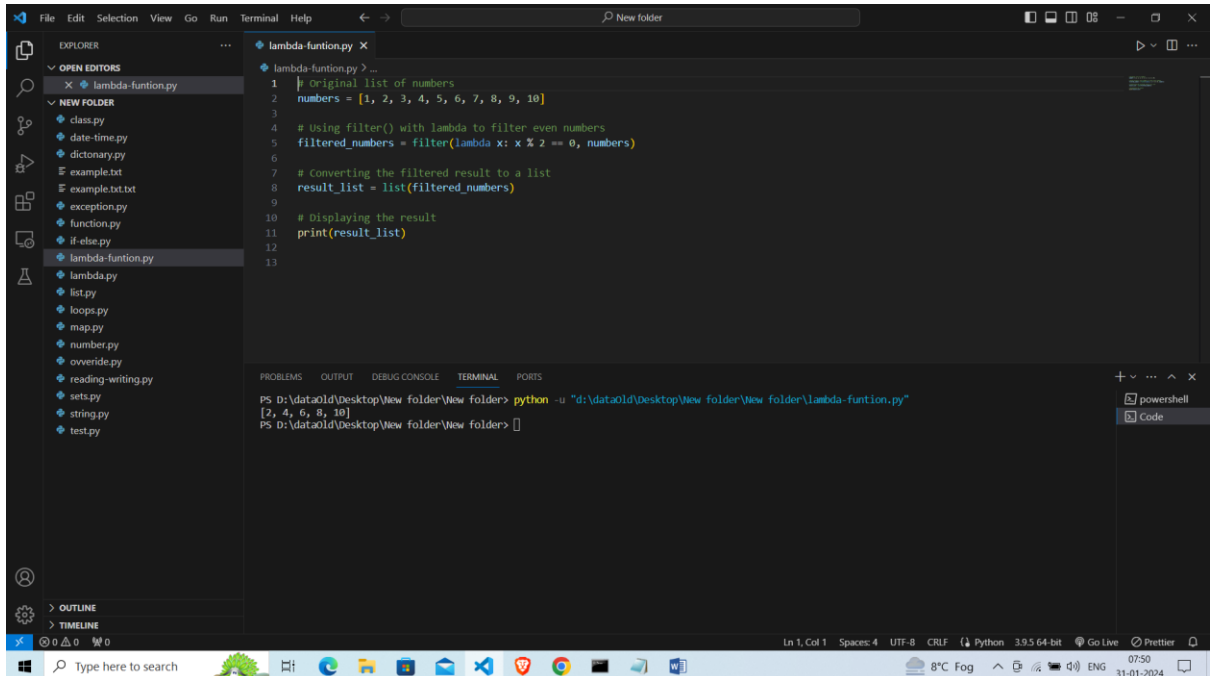
- **`map()`**: The `map()` function applies a given function to all the items in an iterable and returns an iterator that produces the results.

The screenshot shows the Visual Studio Code interface. The Explorer panel on the left lists files in a project, with 'lambda-fun.py' selected. The main editor displays the following Python code:

```
1 # Using map with lambda function
2
3 numbers = [1, 2, 3, 4, 5]
4 squared_numbers = map(lambda x: x**2, numbers)
5 print(list(squared_numbers)) # Output: [1, 4, 9, 16, 25]
6
7
```

The bottom panel shows the Terminal with the command `python -u "d:\dataold\Desktop\New folder\New folder\lambda-fun.py"` and its output: `[1, 4, 9, 16, 25]`.

- **filter():** As discussed earlier, filter() can be used in conjunction with lambda functions to filter elements based on a condition.

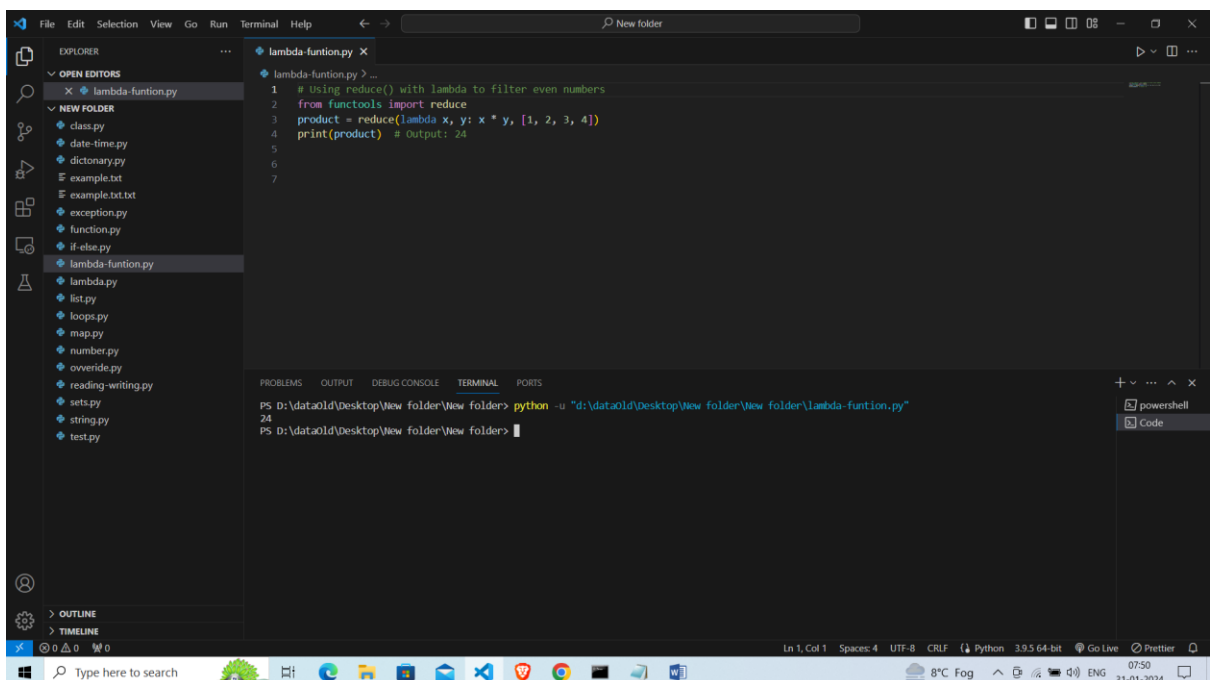


The screenshot shows the Visual Studio Code editor with a file named `lambda-fun.py` open. The code in the editor is as follows:

```
1 # Original list of numbers
2 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3
4 # Using filter() with lambda to filter even numbers
5 filtered_numbers = filter(lambda x: x % 2 == 0, numbers)
6
7 # Converting the filtered result to a list
8 result_list = list(filtered_numbers)
9
10 # Displaying the result
11 print(result_list)
12
13
```

The terminal at the bottom shows the command `python -u "d:\dataold\Desktop\New folder\New folder\lambda-fun.py"` being executed, resulting in the output `[2, 4, 6, 8, 10]`.

- **reduce():** The reduce() function is part of the functools module and is used to successively apply a binary function to the items of an iterable, reducing them to a single accumulated result.



The screenshot shows the Visual Studio Code editor with a file named `lambda-fun.py` open. The code in the editor is as follows:

```
1 # Using reduce() with lambda to filter even numbers
2 from functools import reduce
3 product = reduce(lambda x, y: x * y, [1, 2, 3, 4])
4 print(product) # Output: 24
5
6
7
```

The terminal at the bottom shows the command `python -u "d:\dataold\Desktop\New folder\New folder\lambda-fun.py"` being executed, resulting in the output `24`.