LAB CYCLE-7

Experiment No : 1

Date : 19/12/2024

Aim:

Write a Python program to read a file line by line and store it into a list

Pseudocode:

```
1. Define function read_file_to_list(filename)
2. Set filename to "example.txt"
3. Call read_file_to_list with filename and store the result in 'lines'
4. If lines is not empty
       Print "Lines from the file:"
   End If
5. For each line in 'lines'
       Print the line
   End For
```

```
read_file_to_list(filename)

1. Try
       Open the file with the given filename in read mode
       Create an empty list to store lines
       For each line in the file:
       Remove leading and trailing whitespace from the line
       Add the stripped line to the list
       Return the list of lines
2. Catch FileNotFoundError
       Print "The file 'filename' was not found."
       Return an empty list
3. Catch Any other Exception
```

```
        Print "An error occurred"
        Return an empty list
```

Method :

Function	Description	Syntax
open	The open() function is used to open files in Python.	open(filename,mode)
strip	Remove leading and trailing whitespace from a string.	strip()

Source Code :

```
def read_file_to_list(filename):
    try:
        with open(filename, 'r') as file:
            return [line.strip() for line in file]
    except FileNotFoundError:
        print(f"The file '{filename}' was not found.")
        return []
    except Exception as e:
        print(f"An error occurred: {e}")
        return []
if __name__ == "__main__":
```

```
filename = "example.txt"

lines = read_file_to_list(filename)
if lines:

    print("Lines from the file:")

    for line in lines:

        print(line)
```

Output:

```
Lines from the file:

Hello, World!

This is a test

Python is fun
```

Result:

The program is successfully executed and the output is verified.

Experiment No : 2

Date : 19/12/2024

Aim:

Python program to copy odd lines of one file to other

Pseudocode:

```
1.  Define function copy_odd_lines(source_file, destination_file)
2.  Set source_file = "main.txt"
3.  Set destination_file = "odd.txt"
4.  Call copy_odd_lines(source_file, destination_file)
5.  Try to open and read source_file
        For each line, print the line to console
6.  Except FileNotFoundError
        Print "The source_file was not found."
7.  Except Other Errors
        Print "An error occurred while reading the file."
8.  Try to open and read destination_file
        For each line, print the line to console
9.  Except FileNotFoundError
        Print "The destination_file was not found."
10. Except Other Errors
        Print "An error occurred while reading the file."

copy_odd_lines(source_file, destination_file)
1.  Try to open source_file for reading and destination_file for writing
        For each line in source_file:
            If line number is odd (index % 2 != 0)
                Write the line to destination_file
            End If
        End For
2. Print success message "Odd lines copied from source_file to destination_file"
3. Except FileNotFoundError:
        Print "The source_file was not found."
4. Except Other Errors
        Print "An error occurred: (error message)"
```

Method :

Function	Description	Syntax
write	Write a specified text to the file	file.write()

Source Code :

```
def copy_odd_lines(source_file, destination_file):
    try:
        with open(source_file, 'r') as src, open(destination_file, 'w') as dest:
            for i, line in enumerate(src, start=1):
                if i % 2 != 0:
                    dest.write(line)
        print(f"Odd lines from '{source_file}' have been copied to
          '{destination_file}'.")
    except FileNotFoundError:
        print(f"The file '{source_file}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
if __name__ == "__main__":
    source_file = "main.txt"
    destination_file = "odd.txt"
    copy_odd_lines(source_file, destination_file)
```

```
try:
    print("\nContents of the source file:")
    with open(source_file, 'r') as source:
        for line in source:
            print(line, end='')
except FileNotFoundError:
    print(f"The file '{source_file}' was not found.")
except Exception as e:
    print(f"An error occurred while reading the file: {e}")
try:
    print("\nContents of the destination file:")
    with open(destination_file, 'r') as dest:
        for line in dest:
            print(line, end='')
except FileNotFoundError:
    print(f"The file '{destination_file}' was not found.")
except Exception as e:
    print(f"An error occurred while reading the file: {e}")
```

Output:

```
Contents of main.txt:
One
Two
Three
Four
Five
Contents of odd.txt:
One
Three
```

Five

Result:

The program is successfully executed and the output is verified.

Experiment No : 3

Date : 19/12/2024

Aim:

Write a Python program to read each row from a given csv file and print a list of strings

Pseudocode:

```
1. Define function read_csv_as_strings(file_name)
2. Set file_name = "example.csv"
3. Call read_csv_as_strings(file_name)

read_csv_as_strings(file_name)
1. Try to open file_name for reading
       Read file as CSV
           For each row in the CSV
               Print the row
           End For
2. If file is not found
       Print "The file was not found."
   End If
3. If any other error occurs
       Print "An error occurred"
```

Method :

Function	Description	Syntax
csv.reader	It is used to read data from CSV	csv.reader(file, delimiter=',', quotechar='"')

Source Code :

```
import csv
def read_csv_as_strings(file_name):
    try:
        with open(file_name, 'r') as file:
            csv_reader = csv.reader(file)
            for row in csv_reader:
                print(row)
    except FileNotFoundError:
        print(f"The file '{file_name}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
if __name__ == "__main__":
    file_name = "example.csv"
    read_csv_as_strings(file_name)
```

Output:

```
Alice,30,New York

Bob,25,Los Angeles

Charlie,35,Chicago
```

Result:

The program is successfully executed and the output is verified.

Experiment No : 4

Date : 19/12/2024

Aim:

Write a Python program to read specific columns of a given CSV file and print the content of the columns.

Pseudocode:

```
1. Define function read_specific_columns(file_name, column_indices)
2. Set file_name = "example.csv"
3. Set column_indices = [0, 2]
4. Call read_specific_columns(file_name, column_indices)


read_specific_columns(file_name, column_indices)

1. Try to open file_name for reading
       For each row in the CSV
           Select columns from row using column_indices
           Print the selected columns
       End For
2. If file is not found:
       Print "The file was not found."
   End If
3. If column index is out of range
       Print "One of the column indices is out of range."
   End If
4. If any other error occurs
       Print "An error occurred"
```

Source Code :

```
import csv
def read_specific_columns(file_name, column_indices):
    try:
        with open(file_name, 'r') as file:
            csv_reader = csv.reader(file)
            for row in csv_reader:
                selected_columns = [row[index] for index in column_indices]
                print(selected_columns)
    except FileNotFoundError:
        print(f"The file '{file_name}' was not found.")
    except IndexError:
        print(f"One of the column indices is out of range.")
    except Exception as e:
        print(f"An error occurred: {e}")
if __name__ == "__main__":
    file_name = "example.csv"
    column_indices = [0, 2]
    read_specific_columns(file_name, column_indices)
```

Output:

Contents of example.csv:

```
name,age,city

Alice,30,New York

Bob,25,Los Angeles

Charlie,35,Chicago
```

Odd output:

```
['name', 'city']

['Alice', 'New York']

['Bob', 'Los Angeles']

['Charlie', 'Chicago']
```

Result:

The program is successfully executed and the output is verified.

Experiment No : 5

Date : 19/12/2024

Aim:

Write a Python program to write a Python dictionary to a csv file. After writing the CSV file, read the CSV file and display the content.

Pseudocode:

```
1. Ask user for the number of entries (num_entries)
2. Initialize empty lists: names, ages, cities
3. For each entry (1 to num_entries)
       Ask for name, age, city
       Append name to names, age to ages, city to cities
   End For
4. Create a dictionary data
       data['Name'] = names
       data['Age'] = ages
       data['City'] = cities
5. Open file 'data.csv' for writing:
       Write header (Name, Age, City)
       For each index in range(len(names)):
               Write data for the current row
       End For
6. Open 'data.csv' for reading
       For each row in the CSV
               Print the row
       End For
```

Method :

Function	Description	Syntax
DictWriter	Used for writing data to a CSV file in dictionary form.	csv.DictWriter(file, fieldnames)
writeheader	To write the header to a CSV file.	writer.writeheader()
writerow	It is used to write a single row of data to a CSV file.	writer.writerow(row)

Source Code :

```
import csv
num_entries = int(input("Enter the number of entries you want to add: "))
names = []
ages = []
cities = []
for i in range(num_entries):
    name = input(f"Enter name{i+1}: ")
    age = input(f"Enter age for {name}: ")
    city = input(f"Enter city for {name}: ")
    names.append(name)
    ages.append(age)
    cities.append(city)
data = {
    'Name': names,
    'Age': ages,
    'City': cities
}
```

```
with open('data.csv', mode='w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=data.keys())
    writer.writeheader()
    for i in range(len(data['Name'])):
        row = {key: data[key][i] for key in data}
        writer.writerow(row)
with open('data.csv', mode='r') as file:
    reader = csv.DictReader(file)
    print("\nCSV file contents:")
    for row in reader:
        print(row)
```

Output:

```
Enter the number of entries you want to add: 3

Enter name1: Swarna

Enter age for Alice: 22

Enter city for Alice: Kalady

Enter name2: Shahma

Enter age for Bob: 21

Enter city for Bob: Malappuram

Enter name2: Rinu

Enter age for Bob: 21

Enter city for Bob: Kozhikode

data.csv Content:
Name,Age,City
```

```
Swarna,22,Kalady

Shahma,21,Malappuram

Rinu,21,Kozhikode


CSV file contents:

{'Name': 'Swarna', 'Age': '22', 'City': 'Kalady'}

{'Name': 'Shahma', 'Age': '21', 'City': 'Malappuram'}

{'Name': 'Rinu', 'Age': '21', 'City': 'Kozhikode'}
```

Result:

The program is successfully executed and the output is verified.