Name: SMIRTI KUMARI

PRN: 1062233392

Division: 3rd

Academic Bank of Credits(ABC id): 442526746802

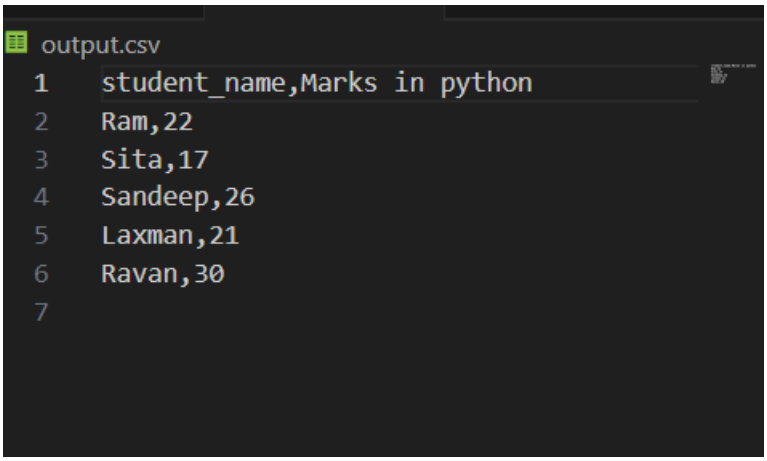
Python FAT-2 Questions

Qno.1 create a program that reads a CSV file and extracts specific columns, then writes the extracted data to a new CSV file.

Solution:

```
    import csv


def extract_columns(input_file, output_file, columns):
    with open(input_file, 'r') as f_in:
        reader = csv.DictReader(f_in)
        fieldnames = reader.fieldnames
        extracted_data = [{column: row[column] for column in columns} for row
in reader]

    with open(output_file, 'w', newline='') as f_out:
        writer = csv.DictWriter(f_out, fieldnames=columns)
        writer.writeheader()
        writer.writerows(extracted_data)

input_file = './python_assignment.csv' #specify path
output_file = 'output.csv'
columns_to_extract = ['student_name','Marks in python'] #specify column name

extract_columns(input_file, output_file, columns_to_extract)
```

OUTPUT:



Qno.2 Develop a program that reads a JSON file and Converts it to a Python dictionary.

Solution:

```
import json


def read_json_file(file_path):
    with open(file_path, 'r') as json_file:
      data = json.load(json_file)
    return data

if __name__ == "__main__":
    file_path = './csvjson.json'
    json_data = read_json_file(file_path)
    print("JSON data converted to Python dictionary:")
    print(json_data)
```

OUTPUT:

```
JSON data converted to Python dictionary:
[{'student_name': 'Ram', 'Marks in python': 22, 'Marks in IoT': 25, 'Marks in Web': 28, 'Att
endance': 60}, {'student_name': 'Sita', 'Marks in python': 17, 'Marks in IoT': 25, 'Marks in
 Web': 18, 'Attendance': 76}, {'student_name': 'Sandeep', 'Marks in python': 26, 'Marks in I
oT': 27, 'Marks in Web': 28, 'Attendance': 45}, {'student_name': 'Laxman', 'Marks in python'
: 21, 'Marks in IoT': 22, 'Marks in Web': 12, 'Attendance': 72}, {'student_name': 'Ravan', '
Marks in python': 30, 'Marks in IoT': 30, 'Marks in Web': 30, 'Attendance': 93}]
```

Qno.3 Develop a function that takes a list of integers and returns a new list containing only the unique elements, preserving their order.

Solution:

```python
def unique_elements(input_list):
    seen = set()
    unique_list = []
    for item in input_list:
        if item not in seen:
            seen.add(item)
            unique_list.append(item)
    return unique_list

# Example usage:
input_list = [1, 2, 3, 4, 2, 3, 5, 6, 1]
print(unique_elements(input_list))
```

OUTPUT:

```
[1, 2, 3, 4, 5, 6]
```

Qno.4 Develop a program to find the factorial of a given number using recursion.

Solution:

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

```
number = 5
print("Factorial of", number, "is", factorial(number))
```

OUTPUT:

```
Factorial of 5 is 120
```

Qno.5 Write a Python program to count the number of lines, words, and characters in a text file.

Solution:

```
def count_lines_words_characters(file_path):
    line_count = 0
    word_count = 0
    character_count = 0

    with open(file_path, 'r') as file:
        for line in file:
            line_count += 1
            words = line.split()
            word_count += len(words)
            character_count += len(line)

    print("Number of lines:", line_count)
    print("Number of words:", word_count)
    print("Number of characters:", character_count)
file_path = './txt.txt'
print(count_lines_words_characters(file_path))
```

OUTPUT:

```
Number of lines: 4
Number of words: 69
Number of characters: 448
```

Qno.6 Implement a stack and use it to check if a given string of parentheses is balanced.

Solution:

```
class Stack:

    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if not self.is_empty():
            return self.items.pop()
        else:
            return None

    def is_empty(self):
        return len(self.items) == 0

def is_balanced_parentheses(string):
    stack = Stack()
    for char in string:
        if char in '({[':
            stack.push(char)
        elif char in ')}]':
            if stack.is_empty():
                return False
            top_char = stack.pop()
            if (char == ')' and top_char != '(') or \
               (char == '}' and top_char != '{') or \
               (char == ']' and top_char != '['):
                return False
    return stack.is_empty()

# Example usage:
input_string = "{[()]"
print("Balanced" if is_balanced_parentheses(input_string) else "Not balanced")
```

OUTPUT:

```
Not balanced
```

Qno.7 Write a program to find the longest consecutive sequence of numbers in a list.

Solution:

```
def longest_consecutive_sequence(nums):
    num_set = set(nums)
    longest_sequence = 0

    for num in num_set:
        if num - 1 not in num_set:
            current_num = num
            current_sequence = 1

            while current_num + 1 in num_set:
                current_num += 1
                current_sequence += 1

            longest_sequence = max(longest_sequence, current_sequence)

    return longest_sequence
nums = [100, 4, 200, 1, 3, 2]
print("Longest consecutive sequence length:",
longest_consecutive_sequence(nums))
```

OUTPUT:

```
Longest consecutive sequence length: 4
```

Qno.8 Create a program that reads an integer from the user and handles the ValueError if the input is not an integer.

Solution:

```
def read_integer():
```

```
    while True:
        try:
            user_input = input("Enter an integer: ")
            num = int(user_input)
            return num
        except ValueError:
            print("Error: Please enter a valid integer.")

# Example usage:
integer = read_integer()
print("You entered:", integer)
```

OUTPUT:

```
Enter an integer: ram
Error: Please enter a valid integer.
Enter an integer: 20
You entered: 20
```

Qno.9 Write a function that divides two numbers and handles the ZeroDivisionError.

Solution:

```
    def safe_divide(dividend, divisor):

    try:
        result = dividend / divisor
        return result
    except ZeroDivisionError:
        print("Error: Division by zero.")
        return None

a = 10
b = 2
result = safe_divide(a, b)
if result is not None:
    print("Result of division:", result)
else:
```

```
    print("Division operation failed.")
```

OUTPUT:

```
Error: Division by zero.
Division operation failed.
PS C:\Users\athar\Desktop\Python> & C:/
ython.exe c:/Users/athar/Desktop/Python
Result of division: 5.0
```

Qno.10 Develop a program that opens a file and handles FileNotFoundError if the file does not exist.

Solution:

```
def open_file(file_path):

    try:
        with open(file_path, 'r') as file:
            content = file.read()
            print("File content:")
            print(content)
    except FileNotFoundError:
        print("Error: File not found.")

file_path = './txt.txt'
open_file(file_path)
```

OUTPUT:

```
Error: File not found.
PS C:\Users\athar\Desktop\Python> & C:/Users/athar/AppData/Local/Programs/Python/Python312/p
ython.exe c:/Users/athar/Desktop/Python/10.py
File content:
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut
 labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea com
modo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
 pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit an
im id est laborum.
```