Assignment Questions

Assignments: Text and CSV File Handling

Assignment 1: Word Count in a Text File

Write a Python program to:

1. Open a text file named sample.txt.
2. Count and display the total number of words, lines, and characters in the file.
3. Display the frequency of each word in the file.

Assignment 2: File Copy with Line Numbers

Write a Python program to:

1. Read the contents of a text file named input.txt.
2. Copy the content into a new file named output.txt, but add line numbers at the beginning of each line in the new file.

Assignment 3: Reading and Writing CSV Files

Write a Python program to:

1. Read a CSV file named students.csv with columns: Name, Age, Grade.
2. Display the average age and the names of all students who scored an A grade.
3. Write the filtered students (with A grade) to a new CSV file named top_students.csv.

Sample **students.csv**:

```
Name,Age,Grade

John,20,A

Anna,22,B

Mike,19,A

Sophia,21,C
```

Assignment 4: Update CSV Data

Write a Python program to:

1. Load a CSV file products.csv with columns: ProductID, Name, Price.
2. Increase the price of all products by 10%.
3. Save the updated data to a new file named updated_products.csv.

Sample **products.csv**:

```
ProductID,Name,Price
101,Laptop,50000
102,Phone,20000
103,Tablet,15000
```

Assignment 5: Search in a File

Write a Python program to:

1. Search for a specific word (given by the user) in a text file data.txt.
2. Display all the lines containing the word along with their line numbers.

Assignments: Exception Handling

Assignment 6: Handling Division by Zero

Write a Python program to:

1. Take two integers as input from the user.
2. Perform division and handle the ZeroDivisionError if the denominator is 0.
3. Display an appropriate error message and ask for the input again.

Assignment 7: File Handling with Exceptions

Write a Python program to:

1. Open a file data.txt and read its content.
2. Handle the following exceptions:
   - FileNotFoundError if the file does not exist.
   - PermissionError if the program lacks permission to read the file.
   - Any other exception with a general error message.

Assignment 8: Nested Try-Except

Write a Python program to:

1. Perform the following tasks in a nested try-except block:
    - o Open a text file example.txt (may not exist).
    - o Read integers from the file and calculate their sum.
2. Handle the following exceptions:
    - o FileNotFoundError if the file is missing.
    - o ValueError if the file contains non-integer values.
    - o Any other exception.

Assignment 9: Retry on Error

Write a Python program to:

1. Take two integers as input from the user.
2. Perform subtraction, but handle any ValueError if the user enters invalid data.
3. Use a loop to keep retrying until valid inputs are provided.

Assignment: Create a psv Module

Objective:

Create a Python module named **psv** that provides functionalities similar to Python's built-in csv module but for **Pipe Separated Value (|)** files. The module should include the following functions:

1. **read_psv(file_path)**
   Reads a PSV file and returns the data as a list of dictionaries, where each dictionary represents a row with column headers as keys.
2. **write_psv(file_path, fieldnames, rows)**
   Writes data to a PSV file using the provided fieldnames and rows.
3. **append_psv(file_path, fieldnames, rows)**
   Appends new rows to an existing PSV file.

Hints and Reference Code

Step 1: Module Structure

Create a Python file named psv.py. The structure of the module can include:

```python
# psv.py
def read_psv(file_path):
    """Reads a PSV file and returns a list of dictionaries."""
    pass  # Implementation here

def write_psv(file_path, fieldnames, rows):
    """Writes data to a PSV file."""
    pass  # Implementation here

def append_psv(file_path, fieldnames, rows):
    """Appends data to an existing PSV file."""
    pass  # Implementation here
```

Step 2: Reading a PSV File

```python
def read_psv(file_path):
    """
    Reads a PSV file and returns a list of dictionaries.
    Each dictionary represents a row with column headers as keys.
    """
    data = []
    try:
        with open(file_path, 'r') as file:
            lines = file.readlines()
            headers = lines[0].strip().split('|')  # Extract column headers
            for line in lines[1:]:
                values = line.strip().split('|')
                row = dict(zip(headers, values))  # Map headers to values
                data.append(row)
    except FileNotFoundError:
        print(f"Error: The file {file_path} does not exist.")
    except Exception as e:
        print(f"An error occurred: {e}")
    return data
```

Step 3: Writing to a PSV File

```python
def write_psv(file_path, fieldnames, rows):
    """
    Writes data to a PSV file.
    :param file_path: The path to the PSV file.
    :param fieldnames: A list of column headers.
    :param rows: A list of dictionaries representing the rows.
    """
    try:
        with open(file_path, 'w') as file:
            file.write('|'.join(fieldnames) + '\n')  # Write headers
            for row in rows:
                line = '|'.join(str(row[field]) for field in fieldnames)
                file.write(line + '\n')  # Write each row
    except Exception as e:
        print(f"An error occurred while writing to {file_path}: {e}")
```

Step 4: Appending to a PSV File

```python
def append_psv(file_path, fieldnames, rows):
    """
    Appends new rows to an existing PSV file.
    :param file_path: The path to the PSV file.
    :param fieldnames: A list of column headers.
    :param rows: A list of dictionaries representing the rows.
    """
    try:
        with open(file_path, 'a') as file:
            for row in rows:
                line = '|'.join(str(row[field]) for field in fieldnames)
                file.write(line + '\n')
    except Exception as e:
        print(f"An error occurred while appending to {file_path}: {e}")
```

Step 5: Test the Module

```
import psv

# Test data
fieldnames = ['Name', 'Age', 'Grade']
rows = [
    {'Name': 'John', 'Age': '20', 'Grade': 'A'},
    {'Name': 'Anna', 'Age': '22', 'Grade': 'B'}
]

# Writing to a PSV file
psv.write_psv('students.psv', fieldnames, rows)

# Reading the PSV file
data = psv.read_psv('students.psv')
print("Data Read from PSV File:")
print(data)

# Appending to the PSV file
new_rows = [{'Name': 'Sophia', 'Age': '21', 'Grade': 'A'}]
psv.append_psv('students.psv', fieldnames, new_rows)

# Reading after append
updated_data = psv.read_psv('students.psv')
print("Updated Data Read from PSV File:")
print(updated_data)
```