AUCSC 111 LAB | Worksheet

Goal:

Text Files in - Input/Output

Objectives:

- Open and close files.
- Read from and write to text files.
- Apply different patterns for reading a text file.

References:

- W3Schools - File

Part I: Basic File Operations (30 min)

1. Write a statement to open a text file **"data.txt"** for writing new content.
2. Write a statement to open a text file **"data.txt"** for appending new content.
3. What does the following statement do?

```
f = open("myfile.txt", "x")
```

4. Write a statement to open a text file **"data.txt"** for reading existing content.
5. A file **"mydata.txt"** is opened as:

```
infile = open("mydata.txt")
```

6. Write a statement to close this file.
7. What is the difference between `read()`, `read(100)`, `readline()`, and `readlines()`? What does each one return?
8. What is the purpose of the `flush()` method?
9. Assume that **"myfile.txt"** contains the following text:

```
This is the first line.
This is the second line.
This is the third line.
This is the fourth line.
```

   Complete the following code to read the lines:

```
infile = open("myfile.txt")
line1 = ____________________________  # to read the first line
line2 = ____________________________  # to read the second line
line3 = ____________________________  # to read the remaining content
```

10. Complete the missing statements in the following code to print the lines of **"myfile.txt"**:

```
f = open("myfile.txt", "r")
for ____________________________________:
    print(line)
```

11. What is the difference between `write()` and `writelines()`?

Part II: Common Patterns for Reading a File

1. Read the File Content into a String

Write a function `vowelCount(fileName)` that takes one input argument: the name of a text file. The function should count and print the number of occurrences of each vowel in the file.

Hints:

- Use `vowels = 'aeiou'`
- `s.count('a')` returns the number of occurrences of 'a'.

Example: If **"myfile.txt"** contains the text 'Le Tour de France':

```
>>> vowelCount("myfile.txt")
a, e, i, o, and u appear, respectively, 1, 3, 0, 1, 1 times.
```

Function Template:

```
def vowelCount(fileName):
    '''Counts and prints the number of occurrences of each vowel in a
file.'''
    print('a, e, i, o, and u appear, respectively', end='')
    s = open(fileName).read()
    vowels = 'aeiou'
    for vowel in vowels:
        print(', {}'.format(s.count(vowel)), end='')
    print(' times.')
```

---

2. Read the File Content into a List of Words

Implement a function `duplicate(filename)` that checks whether a file contains duplicate words.

Example Files:

- **Duplicates.txt**:

```
This is a file with a duplicate. Just one.
You may try to find another but you'll never see it.
```

- **noDuplicates.txt**:

```
The is a file with no duplicate. None whatsoever.
You may try to find duplicates but you'll never see them.
```

Test Cases:

```
>>> duplicate('Duplicates.txt')
True
>>> duplicate('noDuplicates.txt')
False
```

Function Template:

```
def duplicate(filename):
    '''Checks whether the file contains duplicate words.'''
    # Get file content
    infile = open(filename)
    content = infile.read()
    infile.close()

    # Replace punctuation with blank spaces and obtain a list of words
    table = str.maketrans('.,;!?:\n', 7 * ' ')
    words = content.translate(table).split()

    # Check for duplicate words
    for word in words:
        if words.count(word) > 1:
            return True

    # No duplicates found
    return False
```

---

3. Read the File Content into a List of Lines

Write a function `stats(filename)` that prints the number of lines, words, and characters in a file. Your function should open the file only once.

Example:

```
>>> stats('example.txt')
line count: 3
word count: 20
character count: 98
```

Function Template:

```
def stats(filename):
    '''Prints the number of lines, words, and characters in a file.'''
    # Get file content
    infile = open(filename)
    content = infile.read()
    infile.close()

    # Replace punctuation with blank spaces and obtain a list of words
    table = str.maketrans('.,;!?:\n', 7 * ' ')
    words = content.translate(table).split()

    print('line count: {}'.format(content.count('\n')))
    print('word count: {}'.format(len(words)))
    print('character count: {}'.format(len(content)))
```

Part III: Nested For Loops

Exercise: Create a Multiplication Table

Write a program that generates and prints a multiplication table for numbers from 1 to **n** (where **n** is a user-defined positive integer). The table should be formatted nicely in rows and columns.

Example Output

If the user inputs **3**, the output should be:

```
1  2  3
2  4  6
3  6  9
```

Instructions

1. Prompt the user to enter a positive integer **n**.
2. Use a nested for loop to generate the multiplication table.
3. Format the output so that each number occupies the same amount of space.

Sample Code Structure

```
def multiplication_table(n):
    '''Generates and prints a multiplication table from 1 to n.'''
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            # Print the product of i and j
            print(f"{i * j:4}", end='')  # Adjust the spacing as needed
        print()  # Move to the next line after each row

# Main program
n = int(input("Enter a positive integer: "))
multiplication_table(n)
```

Additional Notes:

- Ensure to test your functions thoroughly.
- Follow best practices in coding style and documentation.