Lab 2: Strings, Lists, and Formatted Output

Name: Shaheer Ziya University Number: [redacted]

Exercise 1: Operations on a Line of Text via String Manipulation

AIM:

Write a Python program that prompts the user for one line of text without punctuation marks and then asks the user to choose one of the following operations to be done on the line of text:

(a) printing the line of text with all its blank spaces removed,

(b) printing the line of text with each word in reversed order,

(c) printing the number of words in the line of text,

by using string manipulation. Your program should check if the user input of the choice of operation is valid. You can assume that the entered line of text has no punctuation marks. Here are the sample input and output of this program:

```
Enter a line of text without punctuation mark:
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 1
The line of text with all its blank spaces removed:
Iamveryveryhappy
Enter a line of text without punctuation mark:
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 2
The line of text with each word in reversed order:
I ma yrev yrev   yppah
Enter a line of text without punctuation mark:
```

```
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 3
The number of words in the line of text: 5
Enter a line of text without punctuation mark:
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 5
Invalid input. You must type 1 or 2 or 3!
```

ALGORITHM:

- Ask for the line to perform string manipulation on from the user
- Ask the user for their choice of string manip.
- Validify the choice, end program if invalid choice
- Start Control Flow depending on choice
    1. Remove whitespaces from the string & return it
    2. Separate all the words and reverse them
    3. Separate the words, count them and return count

PROGRAM:

```python
# StrManip.py
# Perform String Manipulation based on user choices
# Created by Shaheer Ziya on 23-03-2022

def main():

  # Ask for the line to perform string manip on
  line = input("Enter a line of text without punctuation mark: \n")
```

```
# Ask for type of string manipulation (Provide 3 choices)
print("Type a number 1 to 3 to do one of the following operations")
print(" 1 - print the line of text with all its blank spaces removed")
print(" 2 - print the line of text with each word in reversed order")
print(" 3 - print the number of words in the line of text")

choice = input("Enter your choice (1-3): ")

# Validfy input
if choice not in ('1', '2', '3'):
  print("Invalid input. You must type 1 or 2 or 3!")

elif choice == '1':
  print("The line of text with all its blank spaces removed:")
  print(line.replace(" ", ""))    # Replace whitespace with nothing

elif choice == '2':
  word = ""
  # Iterate over all the character in line
  for chr in line:
    # Identify the words in the line
    if chr != " ":
      word += chr
    # Reverse the identified word (Words are separated by whitespaces)
    else:
      line = line.replace(word, word[::-1])
      word = ""

  print("The line of text with each word in reversed order:")
  print(line)

elif choice == '3':
  wordCount = 0
  word = ""
  # Separate words in the line & count them
  for chr in line.replace("  ", " "):
```

```
    if chr != " ":
      word += chr
    else:
      wordCount += 1
      word = ""


  print(f"The number of words in the line of text: {wordCount}")




main()
```

OUTPUT:

```
 Type a number 1 to 3 to do one of the following operations
  1 - print the line of text with all its blank spaces removed
  2 - print the line of text with each word in reversed order
  3 - print the number of words in the line of text
 Enter your choice (1-3): 1
 The line of text with all its blank spaces removed:
 Iamveryveryhappy
```

```
Enter a line of text without punctuation mark:
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 2
The line of text with each word in reversed order:
I ma yrev yrev   happy
```

```
Enter a line of text without punctuation mark:
I am very very   happy
Type a number 1 to 3 to do one of the following operations
 1 - print the line of text with all its blank spaces removed
 2 - print the line of text with each word in reversed order
 3 - print the number of words in the line of text
Enter your choice (1-3): 3
The number of words in the line of text: 5
 Enter a line of text without punctuation mark:
 I am very very   happy
 Type a number 1 to 3 to do one of the following operations
  1 - print the line of text with all its blank spaces removed
  2 - print the line of text with each word in reversed order
  3 - print the number of words in the line of text
 Enter your choice (1-3): 5
 Invalid input. You must type 1 or 2 or 3!
```

Exercise 2: Caesar Cipher

AIM:

A Caesar cipher is a simple substitution cipher that produces a cipher text by shifting each plain-text letter in a message a fixed number of places down the alphabet in a circular fashion so that the next character after "z" and “Z” are "a" and ”A”, respectively. The plain text is your original message while the cipher text is the encrypted message. For example, if the key value (i.e. the shift length) is 2, then the word "University Physics” would be encoded as "Wpkxgtukva Rjaukeu". The original message can be recovered by decoding the encrypted message using the reverse process.

Write a Python program that prompts for a key value in the closed interval [1, 25] and the plain text to encode from the user and then outputs the cipher text as well as the decrypted cipher text. You can assume that the input plain text consists only of letters and spaces. And your program should ask the user to input the key value again if the input value is invalid. If your program is correct, then the decrypted cipher text should match the original plain text.

(Hint: Construct two strings where one contains all the plain-text letters and one contains all the corresponding cipher-text letters. Notice that the letter to substitute is at the same index in both strings.)

ALGORITHM:

- Obtain the key and plain text from the user (validifying the key)
- Iterate over the characters in the plain text
- If the character is not a letter, do nothing
- Else if it is an uppercase letter shift its position using the ord(), chr() function by key units (‘A’ = 65)
- Else if it is a lowercase letter do the shift it by key units in the alphabet (‘a’ = 97)

PROGRAM:

```
# CeaserCipher.py
# Implement A Ceaser Cipher
# Created by Shaheer Ziya

# Obtain the key and plain text

key = -1
while (key < 1) or (key > 25):
```

```
  key = int(input("Please provide a key in the interval [1,25]: "))
text = input("Enter the message you want encrypted: ")


encoded = ""


for char in text:
  if char.lower() < 'a' or char.lower() > 'z':
    encoded += char
  elif char.isupper():
    encoded += chr((ord(char) + key - 65) % 26 + 65)
  elif char.islower():
    encoded += chr((ord(char) + key - 97) % 26 + 97)



print(f"The encoded message is {encoded}")
print(f"The decoded message in {text}")
```

OUTPUT:

```
  → ~ /usr/local/bin/python3 "/Users/matthewsummons/Documents/GitHub/PHYS2160/Lab 2/CeaserCipher.py"
  Please provide a key in the interval [1,25]: 2
  Enter the message you want encrypted: University Physics
  The encoded message is Wpkxgtukva Rjaukeu
  The decoded message in University Physics
```

Exercise 3: Vector Manipulation

AIM:

Write a Python program that prompts the user for two vectors of the same length, computes their sum, difference, and dot product by manipulating the lists storing these vectors, and finally outputs the results. You can assume that each vector is input as a comma-space-separated list of integers enclosed by square brackets. And your program should ask the user to input the vectors again if they have different length. Here are the sample input and output of this program:

```
Enter the vector A: [2, 4, 6, 8]
Enter the vector B: [1, 2, 3]
Invalid input! Vectors A and B must have the same length.
Enter the vector A: [2, 4, 6, 8]
Enter the vector B: [1, 2, 3, 4]
A+B = [3, 6, 9, 12]
A-B = [1, 2, 3, 4]
A.B = 60
```

ALGORITHM:

- Obtain the text for the two vectors from the user
- Enforce same length of the vectors
- Define a class to represent vectors and their methods
- Print out the sum, difference and dot product of the two vectors

PROGRAM:

```
# VectorManip.py
# Calculate the sum, difference and dot product of two vectors
# Created by Shaheer Ziya

# Obtain the vectors from the user
# Remove the commas, brackets and ensure each number is followed by a whitespace (even trailing at the end)
textA = input("Enter the vector A: ").replace(",", "").strip("[]") + " "
textB = input("Enter the vector B: ").replace(",", "").strip("[]") + " "

# Ensure vectors are of the same length
```

```
while len(textA) != len(textB):
  print("Invalid input! Vectors A and B must have the same length.")
  textA = input("Enter the vector A: ").replace(",", "").strip("[]") + " "
  textB = input("Enter the vector B: ").replace(",", "").strip("[]") + " "

class Vec:
  def __init__(self, text) -> None:
    # Obtain entries of the vector from the given text
    num, self.entries = "", []    # Num temporarily stores the characters of the input number
    for char in text:
      if char != " ":
        num += char
      else:
        self.entries.append(int(num))
        num = ""


  def __str__(self) -> str:
      """Method to view the entries of the vector"""
      print(str(self.entries))


  def __add__(self, other: "Vec") -> str:
    """
    Obtain the entry-wise sum of two equi-length vectors
    """
    result = []
    for i in range(len(self.entries)):
      result.append(
        self.entries[i] + other.entries[i]
      )
    return str(result)


  def __sub__(self, other: "Vec") -> str:
    """
    Obtain the entry-wsie difference of two equi-length vectors
```

```
    """
    result = []
    for i in range(len(self.entries)):
      result.append(
          self.entries[i] - other.entries[i]
      )
    return str(result)



  def dot(self, other: "Vec") -> int:
    """
    Obtain the dot product of two-equi-length vectors
    """
    result = 0
    for i in range(len(self.entries)):
      result += (self.entries[i] * other.entries[i])
    return result

VecA = Vec(textA)
VecB = Vec(textB)

print(f"A + B = {VecA + VecB}")
print(f"A - B = {VecA - VecB}")
print(f"A . B = {VecA.dot(VecB)}")
```

OUTPUT:

```
Enter the vector A: [2, 4, 6, 8]
Enter the vector B: [1, 2, 3, 4]
A + B = [3, 6, 9, 12]
A - B = [1, 2, 3, 4]
A . B = 60
```

```
Enter the vector A: [1, 0, 0]
Enter the vector B: [0, 1, 0]
A + B = [1, 1, 0]
A - B = [1, -1, 0]
A . B = 0
```

```
Enter the vector A: [1, 1, 1]
Enter the vector B: [2, 2, 2]
A + B = [3, 3, 3]
A - B = [-1, -1, -1]
A . B = 6
```

Exercise 4: Bubble Sort

AIM:

Bubble sort is a simple algorithm that sorts a collection of data by 'bubbling' values to the end of the list over successive passes like air bubbles rising in water. Here are the procedures for sorting a list of numbers in ascending order using bubble sort:

(a) Starting from the first element, compare each adjacent pair of elements of the list in turn and swap the elements if the former element has a larger value. After all the elements in the list have been compared, the largest element will be at the end of the list. This completes the first pass of the algorithm.

(b) Repeat the process in (a) from the first to the second last element of the list. At the end of the second pass, the second largest element will be at the second last position of the list.

(c) Repeat the process in (a) again in a similar manner as in (b) until no more swapping occurs in a pass.

Write a Python program that generates a list composed of 10 random integers in the range of 1 to 100, prints out the list, and finally uses bubble sort to sort the integers in the list in ascending order with the updated list printed out after each swapping. Here is the sample output of this program:

```
Sorting Process of a List of Integers by Bubble Sort:
[83, 74, 89, 26, 18, 69, 75, 98, 47, 77]
[74, 83, 89, 26, 18, 69, 75, 98, 47, 77]
[74, 83, 26, 89, 18, 69, 75, 98, 47, 77]
[74, 83, 26, 18, 89, 69, 75, 98, 47, 77]
[74, 83, 26, 18, 69, 89, 75, 98, 47, 77]
[74, 83, 26, 18, 69, 75, 89, 98, 47, 77]
[74, 83, 26, 18, 69, 75, 89, 47, 98, 77]
[74, 83, 26, 18, 69, 75, 89, 47, 77, 98]
[74, 26, 83, 18, 69, 75, 89, 47, 77, 98]
[74, 26, 18, 83, 69, 75, 89, 47, 77, 98]
[74, 26, 18, 69, 83, 75, 89, 47, 77, 98]
[74, 26, 18, 69, 75, 83, 89, 47, 77, 98]
[74, 26, 18, 69, 75, 83, 47, 89, 77, 98]
[74, 26, 18, 69, 75, 83, 47, 77, 89, 98]
[26, 74, 18, 69, 75, 83, 47, 77, 89, 98]
[26, 18, 74, 69, 75, 83, 47, 77, 89, 98]
```

```
[26, 18, 69, 74, 75, 83, 47, 77, 89, 98]
[26, 18, 69, 74, 75, 47, 83, 77, 89, 98]
[26, 18, 69, 74, 75, 47, 77, 83, 89, 98]
[18, 26, 69, 74, 75, 47, 77, 83, 89, 98]
[18, 26, 69, 74, 47, 75, 77, 83, 89, 98]
[18, 26, 69, 47, 74, 75, 77, 83, 89, 98]
[18, 26, 47, 69, 74, 75, 77, 83, 89, 98]
```

ALGORITHM:

- Define a function to perform Bubble Sort, where each step is printed out to the user
- Iterate over the length of the input array
- Then in each iteration, iterate from the first element to the (n-i-1)th element, where n is the number of elements in the array (because the last ith elements will be in order)
- Perform swapping if the “left” element is larger than the “right” element at that index

Use numpy to obtain an array of 10 random integers in the range [1, 100) and perform bubble sort on that array.

PROGRAM:

```python
# BubbleSort.py
# Implement Bubble Sort in Python
# Created by Shaheer Ziya

import numpy as np

def bubbleSort(arr):
    n = len(arr)
    print("Sorting Process of a List of Integers by Bubble Sort: ")
    print()

    # Traverse through all array elements
    for i in range(n-1):
        # Last i elements are already in place
        for j in range(0, n-i-1):
            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater than the next element
```

```
        if arr[j] > arr[j + 1]:
            arr[j], arr[j + 1] = arr[j + 1], arr[j]
        print(arr)



# 10 numbers in the range [1, 100)
rndm = np.random.randint(1, 100, 10)
# Numpy Arrays are iterable & mutable. Work like python's inbuilt lists
bubbleSort(rndm)
```

OUTPUT:

```
/usr/local/bin/python3 "/Users/matthewsummons/Documents/GitHub/PHYS2160/Lab 2/BubbleSort.py"
➜  ~ /usr/local/bin/python3 "/Users/matthewsummons/Documents/GitHub/PHYS2160/Lab 2/BubbleSort.py"
Sorting Process of a List of Integers by Bubble Sort:

[33 48 47 89 91  1 52 57 32 51]
[33 47 48 89 91  1 52 57 32 51]
[33 47 48 89 91  1 52 57 32 51]
[33 47 48 89 91  1 52 57 32 51]
[33 47 48 89  1 91 52 57 32 51]
[33 47 48 89  1 52 91 57 32 51]
[33 47 48 89  1 52 57 91 32 51]
[33 47 48 89  1 52 57 32 91 51]
[33 47 48 89  1 52 57 32 51 91]
[33 47 48 89  1 52 57 32 51 91]
[33 47 48 89  1 52 57 32 51 91]
[33 47 48 89  1 52 57 32 51 91]
[33 47 48  1 89 52 57 32 51 91]
[33 47 48  1 52 89 57 32 51 91]
[33 47 48  1 52 57 89 32 51 91]
[33 47 48  1 52 57 32 89 51 91]
[33 47 48  1 52 57 32 51 89 91]
[33 47 48  1 52 57 32 51 89 91]
[33 47 48  1 52 57 32 51 89 91]
[33 47  1 48 52 57 32 51 89 91]
[33 47  1 48 52 57 32 51 89 91]
[33 47  1 48 52 57 32 51 89 91]
[33 47  1 48 52 32 57 51 89 91]
[33 47  1 48 52 32 51 57 89 91]
[33 47  1 48 52 32 51 57 89 91]
[33  1 47 48 52 32 51 57 89 91]
[33  1 47 48 52 32 51 57 89 91]
[33  1 47 48 52 32 51 57 89 91]
[33  1 47 48 32 52 51 57 89 91]
[33  1 47 48 32 51 52 57 89 91]
[ 1 33 47 48 32 51 52 57 89 91]
[ 1 33 47 48 32 51 52 57 89 91]
[ 1 33 47 48 32 51 52 57 89 91]
[ 1 33 47 32 48 51 52 57 89 91]
[ 1 33 47 32 48 51 52 57 89 91]
```

```
[ 1 33 47 32 48 51 52 57 89 91]
[ 1 33 47 32 48 51 52 57 89 91]
[ 1 33 32 47 48 51 52 57 89 91]
[ 1 33 32 47 48 51 52 57 89 91]
[ 1 33 32 47 48 51 52 57 89 91]
[ 1 32 33 47 48 51 52 57 89 91]
[ 1 32 33 47 48 51 52 57 89 91]
[ 1 32 33 47 48 51 52 57 89 91]
[ 1 32 33 47 48 51 52 57 89 91]
[ 1 32 33 47 48 51 52 57 89 91]
```

Exercise 5: Printing a Histogram

AIM:

Below is the table of the accommodation in government primary schools by district in Hong Kong, 2019 (Source: Student Enrolment Statistics, 2019/20 by Education Bureau).

District	Numbers of Students (nearest hundreds)	District	Numbers of Students (nearest hundreds)
Central & Western	7	Sai Kung	7
Wan Chai	25	Sha Tin	8
Eastern	22	Tai Po	8
Southern	8	North	8
Yau Tsim Mong	14	Yuen Long	27
Sham Shui Po	22	Tuen Mun	7
Kowloon City	29	Tsuen Wan	18
Wong Tai Sin	6	Kwai Tsing	0
Kwun Tong	15	Islands	0

Write a Python program to show the statistics of the data in this table by printing a histogram in the following format:

```
Accommodation in Government Primary Schools by Districts in HK, 2019
(in nearest hundreds)
Central & Western | ******* (7)
          Wan Chai | ************************* (25)
                   ⋮
           Islands | (0)
```

Your program should define some variables to store the given data and then print the variables using `print` with the format specifier.

ALGORITHM:

- Iterate over the data in the table
- For each line, print the area right-aligned, followed by | and the number of asterisks corresponding to the number of students in the area shown in brackets after the *s

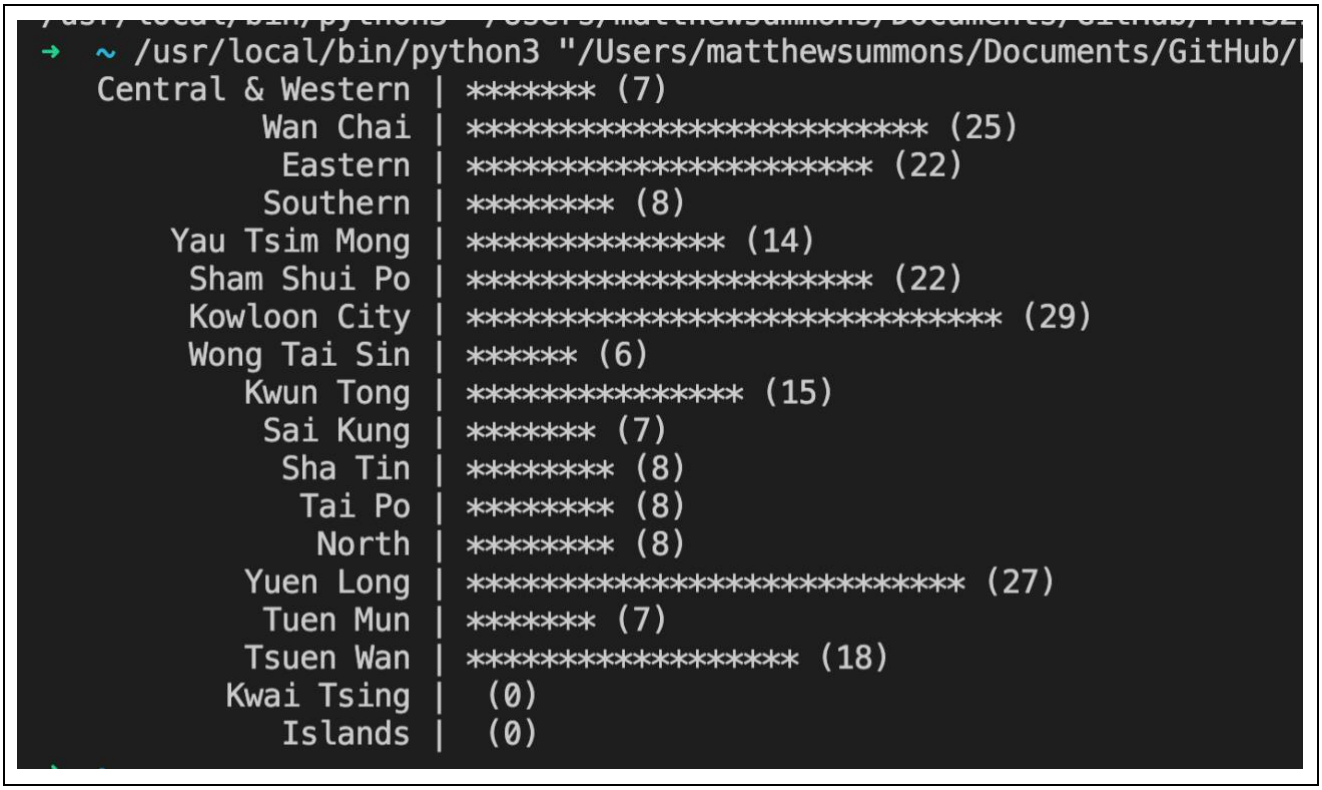
PROGRAM:

```
# Histogram.py
# Print a histogram for some data in a give format
# Created by Shaheer Ziya

# Hard Code the tabular data
stdPopData = {
    "Central & Western" : 7,
    "Wan Chai": 25,
    "Eastern" : 22,
    "Southern" : 8,
    "Yau Tsim Mong" : 14,
    "Sham Shui Po" : 22,
    "Kowloon City" : 29,
    "Wong Tai Sin" : 6,
    "Kwun Tong" : 15,
    "Sai Kung" : 7,
    "Sha Tin" : 8,
    "Tai Po": 8,
    "North" : 8,
    "Yuen Long" : 27,
    "Tuen Mun" : 7,
    "Tsuen Wan" : 18,
    "Kwai Tsing" : 0,
    "Islands" : 0
}

# Print the histogram using f-strings
for area, pop in stdPopData.items():
  print(f"{area:>20} | {'*'*pop} ({pop})")
```

OUTPUT:


~ /usr/local/bin/python3 "/Users/matthewsummons/Documents/GitHub/
Central & Western | ******* (7)
Wan Chai | ************************* (25)
Eastern | ********************** (22)
Southern | ******** (8)
Yau Tsim Mong | ************** (14)
Sham Shui Po | ********************** (22)
Kowloon City | ****************************** (29)
Wong Tai Sin | ****** (6)
Kwun Tong | *************** (15)
Sai Kung | ******* (7)
Sha Tin | ******** (8)
Tai Po | ******** (8)
North | ******** (8)
Yuen Long | ***************************** (27)
Tuen Mun | ******* (7)
Tsuen Wan | ****************** (18)
Kwai Tsing | (0)
Islands | (0)