Fundamentals of Computer Programming

Weekly Programming Portfolio

Submitted by,

Subash Shah

Level 4: Semester 1

Assignment Title: Weekly Program Submission

Student ID: 77576160

16th January 2025

Week - 2

1. Last week you wrote a program that printed out a cheery greeting including your name. Take a copy of it and modify it so that the user enters their name at the keyboard and then receives a greeting. For example: Hello, what is your name? Mr Apricot Hello, Mr Apricot. Good to meet you!

```
""" This program prompts user for name as an input.
Prints out a greeting message with the user's name. """

# Input name from the user.
name = input("Hello, what is your name? ")

# Displays a greeting message with the user's name.
print(f"Hello, {name}. Good to meet you !")
```

2. Write a program that prompts a user to enter a temperature in Celsius, and then displays the corresponding temperature in Fahrenheit, like so: Enter a temperature in Celsius: 32.5 32.5C is equivalent to 90.5F.

```
""" This program prompts a user to enter a temperature in celcius.
Calculates the temperature in Fahrenheit
and displays corresponding temperature in Fahrenheit to the user. """

# Takes user input for temperature in Celsius.
temperature_celcius = float(input("Enter a temperature in Celsius: "))

# Calculates temperature in Fahrenheit using the formula (°C × 9/5) + 32 = °
temperature_fahrenheit = float((temperature_celcius * 9/5) + 32)

# Displays the calculated temperature in Fahrenheit to the user.
print(f"{temperature_celcius}°C is equivalent to {temperature_fahrenheit}°F.")
```

3. The Head of Computing at the University of Poppleton is tasked with dividing a group of students into lab groups. A lab group is usually 24 students, but this is sometimes varied to create groups of similar size. Write a program that prompts for the number of students and group size, and then displays how many groups will be needed and how many will be left over in a smaller grou How many students? 113 Required group size? 22 There will be 5 groups with 3 students left over. For bonus credit, see if you can fix the grammar in the output. So if there were 101 students in groups of 20 the output would be:

```
"""This program prompts user for the number of students and group size.
Displays how many groups will be needed and how many will be left over."""

# Input: Number of students from user.
total_students = int(input("How many students? "))

# Input: Get the group size from the user.
total_groups = int(input("Required group size? "))

# Calculate the number of groups.
group_size = total_students // total_groups

# Calculate the number of students left over.
leftover_students = total_students % total_groups

# Output with correct grammer.
if group_size == 1:
    group_text = "group"
else:
    group_text = "groups"

if leftover_students <= 1:
    leftover_text = "student"
else:
    leftover_text = "students"

print(f"There will be {group_size} {group_text} with {leftover_students}
{leftover_text} left over.")
```

4. A kindly teacher wishes to distribute a tub of sweets between her pupils. She will first count the sweets and then divide them according to how many pupils attend that day. Write a program that will tell the teacher how many sweets to give to each pupil, and how many she will have left over.

```
""" This program prompts the user for total count of sweets and pupil.
It then calculates the number of sweets each pupil will get and leftover if any.
Displays how many sweets to give to each pupil, and how many she will have left
over. """

# Input: Total count of sweets.
total_sweets = int(input("How many sweets are in the tub? "))
```

```
# Input: Total count of pupils.
total_pupils = int(input("How many pupils are attending today? "))

# Calculate the number of sweets each pupil will get.
sweets_per_pupil = total_sweets // total_pupils

# Calculate the number of sweets left over.
leftover_sweets = total_sweets % total_pupils

# Output with correct grammer.
if sweets_per_pupil <=1:
    sweet_text = "sweet"
else:
    sweet_text = "sweets"

if leftover_sweets <=1:
    leftover_text = "sweet"
else:
    leftover_text = "sweets"

#Check if the total number of sweets is less than the total number of pupils.
if total_sweets < total_pupils:
    print("There will not be enough sweets for everyone attending today.")
else:
    print(f"Each pupil will get {sweets_per_pupil} {sweet_text} and there will be
{leftover_sweets} {leftover_text} left over.")
```

Week - 3

1. Modify your greeting program so that if the user does not enter a name (i.e. they just press enter), the program responds "Hello, Stranger!". Otherwise it should print a greeting with their name as before.

```
""" This program prompts user for name as an input.
If user enters a name, it will print out a greeting message with the name.
Or prints "Hello, Stranger!", instead of name. """

# Input name from the user.
name = input("Hello, what is your name? ")

# Checks for user input name.
if name:
```

```
    # Displays a greeting message with the user's name.
    print(f"Hello, {name}. Good to meet you !")

else:
    # If the name is empty, print out " Hello , Stranger!".
    print("Hello, Stranger!")
```

2. Write a program that simulates the way in which a user might choose a password. The program should prompt for a new password, and then prompt again. If the two passwords entered are the same the program should say "Password Set" or similar, otherwise it should report an error.

```
"""This program prompts user to enter new password and re-enter new password
again.
If both entered passwords matches, prints "Password Set"
Else prints error message."""

# Input for new password.
password = input("Enter new password: ")

# Input to confirm password.
confirm_password = input("Re-enter new password: ")

#Check if both password match
if password == "" and confirm_password == "":
    print("Error: Nothing was entered.")
elif password == confirm_password:
    print("Password Set.")
else:
    print("Error: Password do not match.")
```

3. Modify your previous program so that the password must be between 8 and 12 characters (inclusive) long.

```
"""This program prompts user to enter new password.
Validates the condition that the password entered is between 8 and 12 characters
long.
Prompts user again to confirm the password.
If the password matches, "Password set" is displayed otherwise an error is
displayed.
"""
```

```
# Infinite loop to keep prompting user until valid password is entered.
while True:
    # Prompt user to enter new password.
    password =  input("Enter new password (8-12 characters): ")

    # Validate password length.
    if not (8 <= len(password) <= 12):
        print("Error: Password must be between 8 and 12 characters. Try again.")
        continue

    # Prompt user to confirm password.
    confirm_password = input("Re-enter new password: ")

    # Check if passwords match.
    if password == confirm_password:
        print("Password Set.")

    else:
        print("Error: Password do not match.")
    break
```

4. Modify your program again so that the chosen password cannot be one of a list of common passwords, defined thus: BAD_PASSWORDS = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']

```
"""This program prompts user to enter new password.
Validates the condition that the password entered is between 8 and 12 characters
long.
Validates if the chosen password in not in the list of common password.
Prompts user again to confirm the password.
If the password matches, "Password set" is displayed otherwise an error is
displayed.
"""

bad_passwords = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']

# Infinite loop to keep prompting user until valid password is entered.
while True:
    # Prompt user to enter new password.
    password =  input("Enter new password (8-12 characters): ")

    # Validate password length.
    if not (8 <= len(password) <= 12):
        print("Error: Password must be between 8 and 12 characters. Try again.")
```

```
        continue
    # Check if user password is in list of bad_passwords.
    if password in bad_passwords:
        print("Error: Password is too common. Choose a more secure password.")
        continue

    # Prompt user to confirm password.
    confirm_password = input("Re-enter new password: ")

    # Validate that passwords match.
    if password == confirm_password:
        print("Password Set.")

    else:
        print("Error: Password do not match.")
    break
```

5. Modify your program a final time so that it executes until the user successfully chooses a password. That is, if the password chosen fails any of the checks, the program should return to asking for the password the first time.

```
"""This program prompts user to enter new password.
Validates that the password entered is between 8 and 12 characters long.
Ensure the chosen password in not in the list of common password.
Prompts user again to confirm the password.
Displays "Password set." if the password match.
otherwise an error is displayed and Program continues again in loop.
"""

bad_passwords = ['password', 'letmein', 'sesame', 'hello', 'justinbieber']

# Infinite loop to keep prompting user until valid password is entered.
while True:
    # Prompt user to enter new password.
    password =  input("Enter new password (8-12 characters): ")

    # Validate password length.
    if not (8 <= len(password) <= 12):
        print("Error: Password must be between 8 and 12 characters. Try again.")
        continue
    # Check if password is in list of bad_passwords.
    if password in bad_passwords:
        print("Error: Password is too common. Choose a more secure password.")
        continue
```

```
    # Prompt user to confirm password.
    confirm_password = input("Re-enter new password: ")

    # Validate that passwords match.
    if password == confirm_password:
        print("Password Set.")
        break

    else:
        print("Error: Password do not match. Try again.")
```

6. Write a program that displays the "Seven Times Table". That is, the result of multiplying 7 by every number from 0 to 12 inclusive. The output might start: 0 x 7 = 0 1 x 7 = 7 2 x 7 = 14 and so on.

```
"""This program displays "Seven Times Table" as output from number 0 to 12."""

# Initialize the counter variable to 0.
i=0

# Start a loop that will run 13 times (from 0 to 12).
while (i<=12):

    # Print the current value of i multiplied by 7 in the format "i*7=result".
    print(f"{i} * 7 = {i*7}")

    # Increment the counter variable by 1.
    i+=1
```

7. Modify your "Times Table" program so that the user enters the number of the table they require. This number should be between 0 and 12 inclusive.

```
"""This program prompts user for input.
Validates if user input is a number between 0 to 12 inclusive.
Displays "Times Table" for that number from 0 t0 12."""

# Infinite loop to keep prompting user until valid password is entered.
while True:
    try:
        # Prompt the user to enter a number.
        number = int(input("Enter a number between 0 and 12: "))
```

```
        # Check if the number is within the valid range.
        if (0 <= number <= 12):
            break

        else:
            # Inform the user if the number is out of range.
            print("Please enter a number between 0 and 12 inclusive.")

    except ValueError:
        # Handle the error if the input is not a valid integer.
        print("Error: Invalid input. Please enter a number.")

# Loop to generate and display the "Times table".
for i in range(13):
    print(f"{i} * {number} = {i*number}")
```

8. Modify the "Times Table" again so that the user still enters the number of the table, but if this number is negative the table is printed backwards. So entering "-7" would produce the Seven Times Table starting at "12 times" down to "0 times"

```
"""This program prompts user for input.
Validates if user input is a number less 12 inclusive.
If user input is negative number "Times Table" is printed for that number from 12
to 0.
Otherwise displays "Times Table" for that number from 0 t0 12."""

# Infinite loop to keep prompting user until valid password is entered.
while True:
    try:
        # Prompt the user to enter a number.
        number = int(input("Enter a number that is less than or equal to 12: "))

        # Check if the number is within the valid range.
        if (number <= 12) :
            break

        else:
            # Inform the user if the number is out of range.
            print("Please enter a number less than or equal to 12.")

    except ValueError:
        # Handle the error if the input is not a valid integer.
        print("Error: Invalid input. Please enter a number.")
```

```
# Checks if user input is positive number.
if number >= 0:
    # Loop to display "Times table" for positive number.
    for i in range(13):
        print(f"{i} * {number} = {i*number}")

else:
    # Loop to display "Times table" for negative number.
    for j in range(12,-1,-1):
        print(f"{j} * {number} = {j*number}")
```

Week - 4

1. Functions are often used to validate input. Write a function that accepts a sing integer as a parameter and returns True if the integer is in the range 0 to 100 (inclusive), or False otherwise. Write a short program to test the function.

```
"""This program has a function that accepts a single
integer as a parameter and returns True if the integer is in the range 0 to 100
(inclusive), or False otherwise."""

# Function to validate if a number is within the range of 0 to 100.
def validate_integer (number):
    # Returns True if the number ib between 0 to 100 inclusive, otherwise False.
    return 0 <= number <= 100

def test():
    # List of test numbers to validate.
    test_number = [1,0,-20,89,100,101,-9999]

    #Loop through each number in the test_number list.
    for number in test_number:
        # Calls the validate integer function to checkif the number is within the
range.
        result = validate_integer(number)

        # Prints the result (True or False) along with number.
        print(f"{number} : {result}")

if __name__ == "__main__":
    # Call the test function when the script is run directly.
    test()
```

2. Write a function that has a single string as its parameter, and returns the number of uppercase letters, and the number of lowercase letters in the string. Test the function with a short program.

```
"""This program has a function that takes a single string as its parameter,
and returns the number of uppercase letters, and the number of lowercase letters
 in the string and a short program to test the function."""

# Function to count uppercase and lowercase letters in a string
def count_letters(text):
    # Count the number of uppercase letters using list comprehension and the
isupper method.
    uppercase_count = sum(1 for i in text if i.isupper())

    # Count the number of uppercase letters using list comprehension and the
isupper method.
    lowercase_count = sum(1 for i in text if i.islower())

    # Returns the count as tupple (uppercase_count, lowercase_count)
    return uppercase_count, lowercase_count

def test():
    # Short program to test the function
    text = "Hello, My name is Subash Shah."
    uppercase, lowercase = count_letters(text)
    print(f"Test string: '{text}'")
    print(f"Number of uppercase letters: {uppercase}")
    print(f"Number of lowercase letters: {lowercase}")

# Run the program.
if __name__ == "__main__":
    test()
```

3. Modify your "greetings" program so that the first letter of the name entered is always in uppercase with the rest in lowercase. This should happen even if the user entered their name differently. So if the user entered arthur, ARTHUR, or ev arTHur the name should be displayed as Arthur.

```
""" This program has a functions that prompts user for name as an input.
If user enters a name, it will print out a greeting message with the name.
Or prints "Hello, Stranger!", instead of name in formatted way. """

# Define function for user_input.
def user_input():
```

```
    # Prompt user for name as an input.
    name = input("Hello, what is your name? ")
    # Returns user input name.
    return name

def capitalize_name(user_name):
    formatted_name = user_name.capitalize()
    return formatted_name

# Define function to greet user.
def greet_user(formatted_name):

    # Checks if the user input name.
    if formatted_name:
        # Displays a greeting message with the user's name.
        print(f"Hello, {formatted_name}. Good to meet you!")

    else:
        # If the name is empty, prints out " Hello , Stranger!".
        print("Hello, Stranger!")

# Main program
def main():
    name = capitalize_name(user_input())
    greet_user(name)

#Run the program
if __name__ == "__main__":
    main()
```

4. When processing data it is often useful to remove the last character from so input (it is often a newline). Write and test a function that takes a string paramet and returns it with the last character removed. (If the string contains one or fewer characters, return it unchanged.)

```
"""This program has a function  that takes a string parameter
and returns it with the last character removed.
(If the string contains one or fewer characters, returns it unchanged.)"""

# Define function to remove last character of a string.
def remove_last_character(text):

    # Checks if the string has at least two characters.
    if len(text) <= 1:
```

```
        # Returns the string unchanged if it has one or fewer characters.
        return(text)
    else:
        # Returns the string with the last character removed.
        return text[:-1]

# Test the function with some examples.
def test():
    test_string = ["Hello", "I", "", "am", "Subash", "Shah","..."]
    for i in test_string:
        print(remove_last_character(i))

# Run program
if __name__ == "__main__":
    test()
```

5. Write and test a function that converts a temperature measured in degrees centigrade into the equivalent in fahrenheit, and another that does the reverse conversion. Test both functions. (Google will find you the formulae).

```
"""This program has two functions that converts temperature measured in degrees
centigrade into the equivalent in fahrenheit and vice-versa."""

# Define a function to convert celcius to fahrenheit
def celcius_to_fahrenheit(celcius):
    # Returns the equivalent temperature in fahrenheit
    return (celcius * 9/5) + 32

# Define a function to convert fahrenheit to centigrade
def fahrenheit_to_celcius(fahrenheit):
    # Returns the equivalent temperature in celcius
    return (fahrenheit - 32) * 5/9

#Program to check both function.
print("Converts celcius to fahrenheit.")
temperature_in_celcius = [0,38.5,-2,]
for i in temperature_in_celcius:
    print(f"{i}°C is equal to {celcius_to_fahrenheit(i)}°F")

print("Converts fahrenheit to celcius.")
temperature_in_fahrenheit = [32,101.3,28.4]
for i in temperature_in_fahrenheit:
    print(f"{i}°F is equal to {fahrenheit_to_celcius(i):.2f}°C")
```

6. Write a program that takes a centigrade temperature and displays the equivalent in fahrenheit. The input should be a number followed by a letter C. The output should be in the same format.

```
"""This program prompts user to enter temperature in celcius,
It has a function that converts the temperature in celcius to farenheit.
Displays the farenheit temperature as a result."""

#  Define function to prompt user to enter temperature in celcius.
def user_input():
    while True:
        # Prompt user to enter temperature in celcius
        temperature = input("Enter temperature in celcius example(32C):
").upper()
        try:
            # Check if the input is a number and ends with letter 'C'
            if temperature.endswith('C'):
                # Convert the input to float and return it
                float(temperature[:-1])
                # Returns user input temperature.
                return temperature
            else:
                # If input is not a number or does not end with 'C', prompt user
to enter again
                print("Ivalid input. Please enter the temperature in the correct
format (e.g., 25C)")
        except ValueError:
            # If input is not a number, prompt user to enter again
            print("Invalid input. Please enter a valid number")

# Define function to convert celcius to farenheit
def celcius_to_fahrenheit(temperature):
    # Remove 'C' from the input and convert to float
    temperature = float(temperature[:-1])
    # Convert celcius to farenheit using formula.
    fahrenheit = (temperature*9/5)+32
    # Returns the equivalent temperature in fahrenheit
    return fahrenheit

# Define function to display result
def display_result(temperature,fahrenheit):
    # Display the farenheit temperature as a result
    print(f"{temperature} celcius is equal to {fahrenheit:.2f}F")

# Main program.
```

```
def main():
    temperature = user_input()
    fahrenheit = celcius_to_fahrenheit(temperature)
    display_result(temperature,fahrenheit)

# Run program
if __name__ == "__main__":
    main()
```

7. Write a program that reads 6 temperatures (in the same format as before), and displays the maximum, minimum, and mean of the values. Hint: You should know there are built-in functions for max and min. If you hunt, you might also find one for the mean.

```
# """This program prompts user for six different temperature in celcius.
# It has a function that converts celcius to fahrenheit.
# Displays max, min and mean from 6 different fahrenheit as a result."""

from statistics import mean

# Define function to prompt user to enter temperature in Celsius
def user_input():
    temperatures = []
    for i in range(6):
        while True:
            # Prompt user to enter temperature in Celsius
            temperature = input(f"Enter temperature {i + 1} in Celsius (e.g.,
32C): ").upper()
            if temperature.endswith('C'):
                try:
                    # Validates the numeric part.
                    float(temperature[:-1])
                    # Append the temperature to the list including 'C'.
                    temperatures.append(temperature)
                    break
                except ValueError:
                    # If the number part is not valid, prompt the user to enter
again
                    print("Invalid input. Please enter a valid number followed by
'C'.")
            else:
                # If input does not end with 'C', prompt user to enter again
                print("Invalid input. Please enter the temperature in the correct
format (e.g., 25C).")
```

```
    return temperatures

# Define function to convert celcius to farenheit
def celcius_to_fahrenheit(temperature):
    # Remove 'C' from the input and convert to float
    temperature = float(temperature[:-1])
    # Convert celcius to farenheit using formula.
    fahrenheit = (temperature*9/5)+32
    # Returns the equivalent temperature in fahrenheit
    return fahrenheit

# Main program
def main():
    temperatures = user_input()
    fahrenheit_temperatures = [celcius_to_fahrenheit(i) for i in temperatures]
    print(f"Temperatures in Fahrenheit: {fahrenheit_temperatures}")
    # Calculate and display maxmium, minimum and mean of the fahrenheit
temperatures using built in-function.
    maxmimum_temperature = max(fahrenheit_temperatures)
    minimum_temperature = min(fahrenheit_temperatures)
    mean_temperature = mean(fahrenheit_temperatures)
    print(f"Maximum temperature: {maxmimum_temperature}F")
    print(f"Minimum temperature: {minimum_temperature}F")
    print(f"Mean temperature: {mean_temperature}F")

# Run program
if __name__ == "__main__":
    main()
```

8. Modify the previous program so that it can process any number of values. The input terminates when the user just pressed "Enter" at the prompt rather than entering a value.

```
"""This program prompts user for different temperature in celcius infinitely.
if user enters 0, it will break the loop and run the other functions.
It has a function that converts celcius to fahrenheit.
Displays max, min and mean from user input temperatures in fahrenheit as a
result."""

from statistics import mean

# Define function to prompt user to enter temperature in Celsius
def user_input():
```

```
    temperatures = []
    while True:
        # Prompt user to enter temperature in Celsius
        temperature = input(f"Enter temperature in Celsius (e.g., 32C):
").upper()
        # Check if user wants to exit the loop
        if temperature == "":
            break
        if temperature.endswith('C'):
            try:
                # Validates the numeric part.
                float(temperature[:-1])
                # Append the temperature to the list including 'C'.
                temperatures.append(temperature)
            except ValueError:
                # If the number part is not valid, prompt the user to enter again
                print("Invalid input. Please enter a valid number followed by
'C'.")
        else:
            # If input does not end with 'C', prompt user to enter again
            print("Invalid input. Please enter the temperature in the correct
format (e.g., 25C).")
    return temperatures

# Define function to convert celcius to farenheit
def celcius_to_fahrenheit(temperature):
    # Remove 'C' from the input and convert to float
    temperature = float(temperature[:-1])
    # Convert celcius to farenheit using formula.
    fahrenheit = (temperature*9/5)+32
    # Returns the equivalent temperature in fahrenheit
    return fahrenheit

# Main program
def main():
    temperatures = user_input()
    fahrenheit_temperatures = [celcius_to_fahrenheit(i) for i in temperatures]
    print(f"Temperatures in Fahrenheit: {fahrenheit_temperatures}")
    maxmimum_temperature = max(fahrenheit_temperatures)
    minimum_temperature = min(fahrenheit_temperatures)
    mean_temperature = mean(fahrenheit_temperatures)
    print(f"Maximum temperature: {maxmimum_temperature}F")
    print(f"Minimum temperature: {minimum_temperature}F")
    print(f"Mean temperature: {mean_temperature}F")
```

```
# Run program
if __name__ == "__main__":
    main()
```

Week - 5

1. Using command-line arguments involves the sys module. Review the docs for this module and using the information in there write a short program that when run from the command-line reports what operating system platform is being used.

```
"""This program uses the sys module to retrieve the platform information and when
run
from the command-line prints what operating system platform is being used."""

import sys

def main():
    # Retrieve the platform information using sys module
    platform = sys.platform
    print(f"The operating system platform being used is: {platform}")

# Call the main function to start the program
if __name__ == "__main__":
    main()
```

2. Write a program that, when run from the command line, reports how many arguments were provided. (Remember that the program name itself is not an argument).

```
"""This program when run from the command line, reports how many number of
arguments were provided"""

import sys

def main():
    # Get the number of arguments provided to the program by deducting the
program name.
    number_arguments = len(sys.argv) -1
    # Return the number of arguments
    return number_arguments

print(f"Number of arguments provided: {main()}")

# Call the main function to start the program
```

```
if __name__ == "__main__":
    main()
```

3. Write a program that takes a bunch of command-line arguments, and then prints out the shortest. If there is more than one of the shortest length, any will do. Hint: Don't overthink this. A good way to find the shortest is just to sort them.

```
"""This program that takes a bunch of command-line arguments, and then prints out
the shortest argument."""

import sys

def main():
    # Check if there are no arguments provided.
    if len(sys.argv) < 2:
        print("Please provide one or more command-line arguments.")
        return

    # Get all arguments except the program name.
    arguments = sys.argv[1:]

    # Sort arguments by length
    sorted_arguments = sorted(arguments, key=len)

    # Print the shortest argument
    print(f"The shortest argument is: {sorted_arguments[0]}")

if __name__ == "__main__":
    main()
```

4. Write a program that takes a URL as a command-line argument and reports whether or not there is a working website at that address. Hint: You need to get the HTTP response code. Another Hint: StackOverflow is your friend.

```
"""This program takes a URL as a command-line argument and reports
whether or not there is a working website at that address."""

import sys
import requests

# Function to check if a website is working
def check_website(url):
```

```
    try:
        # Send a HEAD request to the URL
        response = requests.head(url, timeout=5)
        # Check the status code of the response
        if response.status_code == 200:
            # If the status code is 200, prints website is working.
            print(f"The website at {url} is working.")
        else:
            # If the status code is not 200, prints the status code.
            print(f"The website at {url} returned status code:
{response.status_code}")
    # Handle exceptions
    except requests.RequestException as e:
        # If an exception occurs, prints an error message.
        print(f"Error: Could not connect to {url}. Details: {e}")

def main():
    # Check if the user provided a URL as a command-line argument
    if len(sys.argv) != 2:
        # If not, prints usage message and exits.
        print("Usage: python check_website.py <URL>")
        return
    # Get the URL from the command-line argument
    url = sys.argv[1]
    # Call the check_website function with the URL
    check_website(url)

if __name__ == "__main__":
    main()
```

5. Last week you wrote a program that processed a collection of temperature readings entered by the user and displayed the maximum, minimum, and mean. Create a version of that program that takes the values from the command-line instead. Be sure to handle the case where no arguments are provided!

```
"""This program takes values from the command line and converts temperature
readings
from Celsius to Fahrenheit
and calculates the maximum, minimum, and mean temperatures and displays the
result."""

import sys
from statistics import mean
```

```
# Define function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit(temperature):
    # Remove 'C' from the input and convert to float
    temperature = float(temperature[:-1])
    # Convert Celsius to Fahrenheit using formula
    fahrenheit = (temperature * 9 / 5) + 32
    # Returns the equivalent temperature in Fahrenheit
    return fahrenheit

# Main program
def main():
    # Check if the user provided temperature readings as command-line arguments
    if len(sys.argv) < 2:
        # If not, print usage message and exit
        print("Please provide temperature readings in Celsius as command-line
arguments (e.g., 25C 32C).")
        return
    # Get the temperature readings from the command-line arguments
    temperatures = sys.argv[1:]

    # List to store the converted temperatures in Fahrenheit
    fahrenheit_temperatures = []

    # Iterate over the temperature readings
    for temp in temperatures:
        # Check if the temperature reading ends with 'C'
        if temp.endswith('C'):
            try:
                # Validate and convert the temperature
                fahrenheit_temperatures.append(celsius_to_fahrenheit(temp))
            except ValueError:
                # Handle invalid temperature format
                print(f"Invalid temperature format: {temp}. Please provide a
valid number followed by 'C'.")
                return
        else:
            # Handle invalid temperature format
            print(f"Invalid temperature format: {temp}. Please provide a valid
number followed by 'C'.")
            return
    # Print the converted temperatures
    print(f"Temperatures in Fahrenheit: {fahrenheit_temperatures}")

    # Calculate and print the maximum, minimum, and mean temperatures
    max_temperature = max(fahrenheit_temperatures)
```

```
    min_temperature = min(fahrenheit_temperatures)
    mean_temperature = mean(fahrenheit_temperatures)

    # Print the calculated values
    print(f"Maximum temperature: {max_temperature}F")
    print(f"Minimum temperature: {min_temperature}F")
    print(f"Mean temperature: {mean_temperature}F")

# Run program
if __name__ == "__main__":
    main()
```

6. Write a program that takes the name of a file as a command-line argument, and creates a backup copy of that file. The backup should contain an exact copy of the contents of the original and should, obviously, have a different nam Hint: By now, you should be getting the idea that there is a built-in way to do the heavy lifting here! Take a look at the "Brief Tour of the Standard Library" in the doc

```
"""This program takes the name of a file as a command-line argument, and
creates a backup copy of that file renaming it."""

import sys
import shutil

def main():
    # checks if the user has provided the required arguments.
    if len(sys.argv) != 2:
        # if not, prints the usage and exits.
        print("Usage: python backup_file.py <file_path>")
        return
    # gets the file name from the user and store on variable.
    original_file_path = sys.argv[1]

    # Stores file name in backup variable adding .backup
    backup_file_path = + "backup-" + original_file_path

    try:
        # Try to copy the file to the backup location
        shutil.copyfile(original_file_path, backup_file_path)

        # If the copy is successful, prints a success message.
        print(f"Backup created successfully: {backup_file_path}")
    except FileNotFoundError:
        # If the file does not exist, prints an error message.
```

```
        print(f"Error: The file '{original_file_path}' was not found.")
    except Exception as e:
        # If any other error occurs, prints the error message.
        print(f"An error occurred: {e}")

# Call the main function
if __name__ == "__main__":
    main()
```

Week - 6

1. Write a function that accepts a positive integer as a parameter and then returns a representation of that number in binary (base 2). Hint: This is in many ways a trick question. Think!

```
"""This program has a function that accepts a positive integer as a parameter and
then
returns a representation of that number in binary (base 2)"""

# Define a function to convert decimal to binary
def decimal_to_binary(decimal):
    # Check if the input is a positive integer
    if decimal == 0:
        # If the number is 0, return 0
        return "0"

    binary = ""
    # Loop until the number is 0
    while decimal > 0:
        # Append the remainder of the number divided by 2 to the binary string
        binary = str(decimal % 2) + binary
        # Update the number by dividing it by 2
        decimal = decimal // 2
    # Return the binary representation of the number
    return binary

# Define function to get user input
def get_user_input():
    while True:
        try:
            # Prompt user to enter a non-negative decimal number
            decimal = int(input("Enter a non-negative decimal number: "))
            if decimal >= 0:
                return decimal
```

```
            else:
                print("Please enter a non-negative number.")
        except ValueError:
            # Handle invalid input
            print("Invalid input. Please enter a valid number.")

# Function to display result
def display_result(decimal):
    # Convert decimal to binary and display the result
    binary_representation = decimal_to_binary(decimal)
    print("The binary representation of the decimal number is:",
binary_representation)

# Main program
def main():
    decimal_number = get_user_input()
    display_result(decimal_number)

# Run the program
if __name__ == "__main__":
    main()
```

2. Write and test a function that takes an integer as its parameter and returns the factors of that integer. (A factor is an integer which can be multiplied by another to yield the original).

```
"""This program has a function that takes an integer as its parameter and returns
the
factors of that integer. There is also a test function with list of numbers."""

# Define a function to find factors of an integer.
def find_factors(number):

    # Initilize an empty list to store factors of an integer.
    factors = []

    # Iterate over all numbers from 1 to the given number (inclusive).
    for i in range(1, number + 1):
        # Check if i is a factor of the number
        if number % i == 0:
            # Add i to the list of factors
            factors.append(i)
    # Return the list of factors
```

```
    return factors

# Short program to test the function
def test_find_factors():
    test_numbers = [15, 28, 50, 1]
    for num in test_numbers:
        factors = find_factors(num)
        print(f"The factors of {num} are: {factors}")

# Run the test
test_find_factors()
```

3. Write and test a function that determines if a given integer is a prime number. A prime number is an integer greater than 1 that cannot be produced by multiplying two other integers.

```
"""This program has a function that returns True, if a given integer is a prime
number,
and False otherwise. There is also a test function that tests the function with a
list of numbers."""

# Define a function to check if a number is prime.
def is_prime(number):
    count = 0
    # Loop to iterate from 2 to the number (inclusive)
    for i in range(2,number+1):

        # If number is divisible by i, increment count
        if number % i == 0:
            count+=1

    # Prime numbers are greater than 1
    if number <= 1:
        return False

    # If count is 1, it means number has no divisors other than 1 and itself.
    if count == 1:
        return True
    else:
        return False

# Function to test the is_prime function with various numbers
def test_is_prime():
    test_numbers = [2, 3, 4, 17, 20, 23, 29, 35]
```

```
    for num in test_numbers:
        result = is_prime(num)
        print(f"Is {num} a prime number? {result}")

# Run the test
test_is_prime()
```

4. Computers are commonly used in encryption. A very simple form of encryption (more accurately "obfuscation") would be to remove the spaces from a message and reverse the resulting string. Write, and test, a function that takes a string containing a message and "encrypts" it in this way.

```
"""This program has a function that takes string as parameter and returns
reversed string which is a very simple form of encryption. """

# Define function to reverse string.
def text_encryption(text):
    # Removes all spaces from a given string.
    new_text = "".join(text.split())
    # Reverse the string using slicing.
    return new_text[::-1]

# Define function for user input.
def user_input():
    #Prompts user to enter a string and removes spaces from string.
    text = str(input("Enter the text message you want to encrypt: "))
    # Returns text removing spaces
    return(text)

# Main function
def main():
    text = user_input()
    encrypted_text=text_encryption(text)
    print (f"Encrypted text: {encrypted_text}")

# Run program
if __name__ == "__main__":
    main()
```

5. Another way to hide a message is to include the letters that make it up within seemingly random text. The letters of the message might be every fifth characte for example. Write and test a function that does such encryption. It should randomly generate an interval (between 2 and 20), space the message out accordingly, and

should fill the gaps with random letters. The function should return the encrypted message and the interval used. For example, if the message is "send cheese", the random interval is 2, and for clarity the random letters are not random: send cheese s e n d c h e e s e sxyexynxydxy cxyhxyexyexysxye

```
"""This program has a function that takes a message parameter and returns
encrypted message and interval of
characters in message."""

import random
import string

# Function to encrypt a message
def encrypt_message(message):
    # Randomly generate an interval between 2 and 20
    interval = random.randint(2, 20)

    # Create a list to store the encrypted message
    encrypted_message = []

    # Iterate over each character in the message
    for i, char in enumerate(message):
        # Add random characters to the encrypted message
        for _ in range(interval - 1):
            encrypted_message.append(random.choice(string.ascii_letters))
        # Add the message character
        encrypted_message.append(char)

    # Join the list into a single string
    encrypted_message_str = ''.join(encrypted_message)

    return encrypted_message_str, interval

# Short program to test the function
def test_encrypt_message():

    test_message = input("Enter a message to encrypt: ")
    encrypted_message, interval = encrypt_message(test_message)
    print(f"Original message: {test_message}")
    print(f"Encrypted message: {encrypted_message}")
    print(f"Interval used: {interval}")

# Run the test
test_encrypt_message()
```

6. Write a program that decrypts messages encoded as above.

```
"""This program has a function that accepts encrypted message as parameter and
returns
decrypted message so human can understand it."""

# Function to decrypt a message
def decrypt_message(encrypted_message):
    # Extract the characters from the encrypted message at the specified interval
    decrypted_message = encrypted_message.replace("xy","")
    return decrypted_message

# Short program to test the function
def test_decrypt_message():
    encrypted_message = "sxyexynxydxy cxyhxyexyexysxye"
    decrypted_message = decrypt_message(encrypted_message)
    print(f"Encrypted message: {encrypted_message}")
    print(f"Decrypted message: {decrypted_message}")

# Run the test
test_decrypt_message()
```

Week – 7

1. Write and test a function that takes a string as a parameter and returns a sorted list of all the unique letters used in the string. So, if the string is cheese, the list returned should be ['c', 'e', 'h', 's'].

```
"""This program has function that takes a string as a parameter and returns a
sorted list
of all the unique letters used in the string"""

# Define a function to get unique letters from a string
def unique_character(user_text):
    # Converts the string to a set to remove duplicates, Sort the list of unique
characters and returns it.
    return sorted(list(set(user_text)))

# Test the function with a string
print(unique_character("cheese"))
```

2. Write and test three functions that each take two words (strings) as parameters and return sorted lists (as defined above) representing respectively: Letters that appear

in at least one of the two words. Letters that appear in both words. Letters that appear in either word, but not in both. Hint: These could all be done programmatically, but consider carefully what topic we have been discussing this week! Each function can be exactly one line.

```
"""This program has three functions that takes two word strings as parameters
First function returns sorted list of letters that appear in at least one of the
two words.
Second function returns a sorted list of letters that appear in both words.
Third function returns a sorted list of letters that appear in the first word but
not in the second word."""


# Define function to return letters that appear in at least one of the two words
def letters_in_at_least_one(word1, word2):
    # Convert the words to sets to remove duplicates, perform union operation,
and then convert to sorted.
    return sorted(list(set(word1) | set(word2)))

# Define function to return letters that appear in both words
def letters_in_both(word1, word2):
    # Convert the words to sets to remove duplicates, perform intersection
operation,
    # and then convert back to sorted list
    return sorted(list(set(word1) & set(word2)))

# Define function to return letters that appear in either word, but not in both
def letters_in_either_but_not_both(word1, word2):
    # Convert the words to sets to remove duplicates, perform symmetric
difference operation,
    # and then convert back to sorted list
    return sorted(list(set(word1) ^ set(word2)))

# Test the functions
word1 = "Computer"
word2 = "Programming"
print("Letters in at least one of the words:", letters_in_at_least_one(word1,
word2))
print("Letters in both words:", letters_in_both(word1, word2))
print("Letters in either word, but not in both:",
letters_in_either_but_not_both(word1, word2))
```

3. Write a program that manages a list of countries and their capital cities. It should prompt the user to enter the name of a country. If the program already "knows" the

name of the capital city, it should display it. Otherwise it should ask the user to enter it. This should carry on until the user terminates the program (how this happens is up to you).

```
"""This program prompts user to input a country name. If the program already
"knows"
the name of the capital city, it displays it. Otherwise it asks the user to
enter it and adds to dict infinetly. if user enters exits the program terminates.
"""

# Define a dictionary to store country-capital pairs
countries_capitals = {}

# Define a flag to track whether the user wants to exit the program
while True:
    # Prompt the user to enter the name of a country or 'exit' to terminate the
program
    country = input("Enter the name of a country (or type 'exit' to quit):
").strip().lower()

    if country == 'exit':
        print("Exiting the program. Goodbye!")
        break

    # Convert the first letter of each word in the country name to uppercase
    country = country.title()

    if country in countries_capitals:
        # If the country is already known, display its capital
        print(f"The capital city of {country} is {countries_capitals[country]}.")
    else:
        # If the country is unknown, ask the user for its capital
        capital = input(f"I don't know the capital city of {country}. Please
enter it: ").strip()
        # Store the capital city in the dictionary
        countries_capitals[country] = capital
        print(f"Thank you! The capital city of {country} has been added as
{capital}.")
```

4. One approach to analysing some encrypted data where a substitution is suspected is frequency analysis. A count of the different symbols in the message can be us to identify the language used, and sometimes some of the letters. In English, the most common letter is "e", and so the symbol representing "e" should appear most in the encrypted text. Write a program that processes a string representing a message and

reports the six most common letters, along with the number of times they appear. Case should not matter, so "E" and "e" are considered the same. Hint: There are many ways to do this. It is obviously a dictionary, but we will want zero counts, so some initialisation is needed. Also, sorting dictionaries is tricky, so best to ignore that initially, and then check the usual resources for the runes

```
"""This program has a function that takes a message string as a parameter,
and returns the count of six most common letters of that message."""

from collections import Counter


def frequency_analysis(message):
    # Convert the message to lower case to ignore case sensitivity
    message = message.lower()

    # Filter out non-alphabet characters
    filtered_message = ''.join(filter(str.isalpha, message))

    # Count the frequency of each letter
    letter_counts = Counter(filtered_message)

    # Get the six most common letters
    most_common_letters = letter_counts.most_common(6)
    return most_common_letters

# Test the function
def test_frequency_analysis():
    message = "I like programming and I love coding in python."
    result = frequency_analysis(message)
    for letter, count in result:
        print(f"Letter: '{letter}', Count: {count}")

# Run the test
test_frequency_analysis()
```