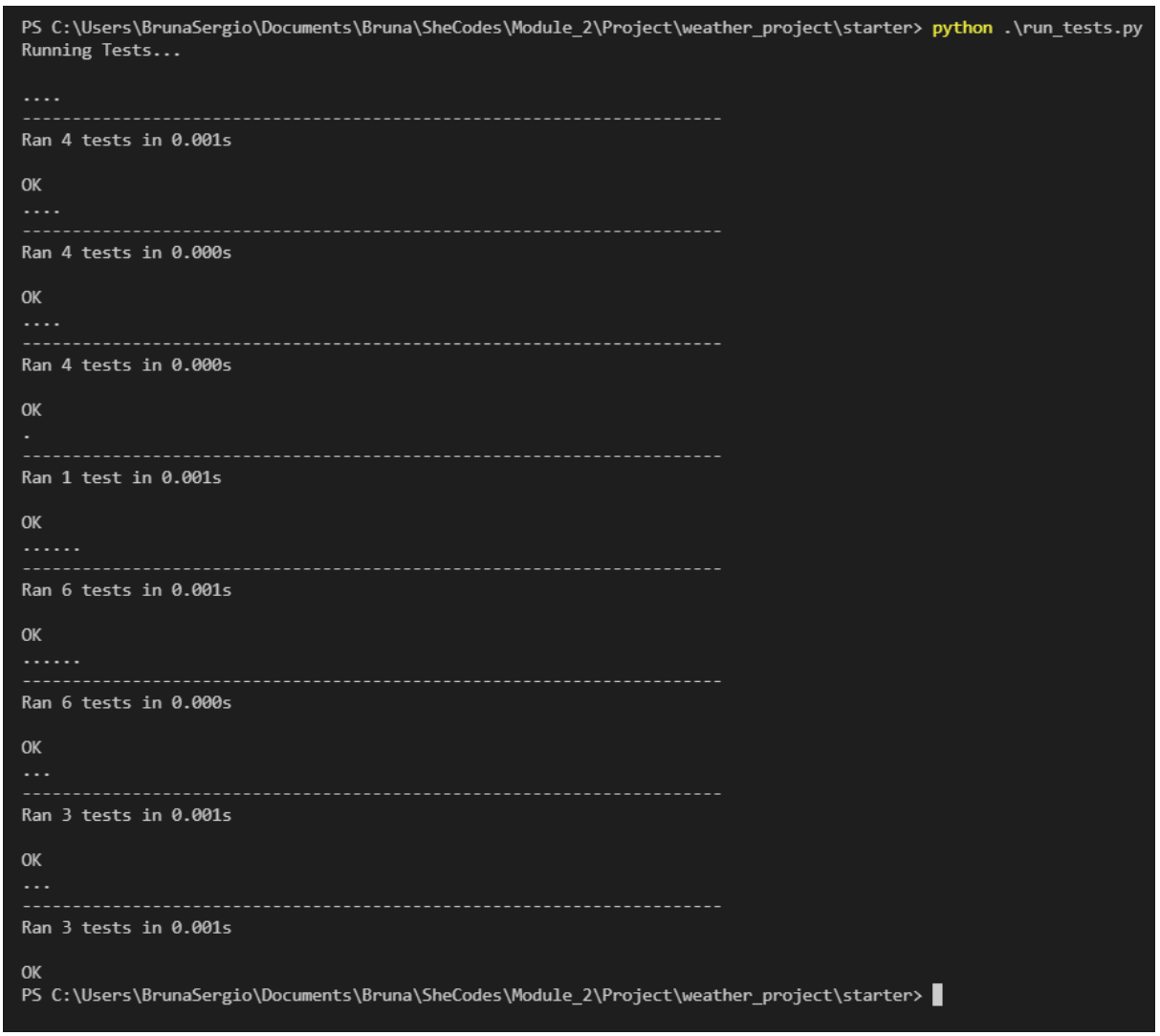
- A text file containing a link to your project repository.

  https://github.com/brunaselymes/weather_Project

- Include a screenshot of your code passing all of the tests in Terminal/Powershell



- Functions:

```
import csv

from datetime import  datetime


DEGREE_SYBMOL = u"\N{DEGREE SIGN}C"
```

```
def format_temperature(temp):
    """Takes a temperature and returns it in string format with the degrees
        and celcius symbols.

    Args:
        temp: A string representing a temperature.
    Returns:
        A string contain the temperature and "degrees celcius."
    """
    return f"{temp}{DEGREE_SYBMOL}"


def convert_date(iso_string):
    """Converts and ISO formatted date into a human readable format.

    Args:
        iso_string: An ISO date string..
    Returns:
        A date formatted like: Weekday Date Month Year e.g. Tuesday 06 July
2021
    """
    return datetime.fromisoformat(iso_string).strftime("%A %d %B %Y")


def convert_f_to_c(temp_in_farenheit):
    """Converts an temperature from farenheit to celcius.
```

```
      Args:
        temp_in_farenheit: float representing a temperature.
    Returns:
        A float representing a temperature in degrees celcius, rounded to
1dp.
    """
    return round((float(temp_in_farenheit)-32) * (float(5/9)), 1)


def calculate_mean(weather_data):
    """Calculates the mean value from a list of numbers.

    Args:
        weather_data: a list of numbers.
    Returns:
        A float representing the mean value.
    """
    sumOfNumbers = 0
    for number in weather_data:
        sumOfNumbers += float(number)
    return sumOfNumbers / len(weather_data)


def load_data_from_csv(csv_file):
    """Reads a csv file and stores the data in a list.

    Args:
        csv_file: a string representing the file path to a csv file.
    Returns:
        A list of lists, where each sublist is a (non-empty) line in the csv
file.
```

```
    """

    linesOfFile = []
    with open(csv_file) as csv_file:
        csv_reader = csv.reader(csv_file)
        header = next(csv_reader)
        for line in csv_reader:
            if line != []:
                linesOfFile.append([line[0],int(line[1]), int(line[2])])
        return linesOfFile

def find_min(weather_data):
    """Calculates the minimum value in a list of numbers.

    Args:
        weather_data: A list of numbers.
    Returns:
        The minium value and it's position in the list.
    """
    min = None
    index = None
    for i, value in enumerate(weather_data):
        if min is None or float(value) <= min:
            min = float(value)
            index = i
    return (min, index) if min is not None else ()

def find_max(weather_data):
```

```
    """Calculates the maximum value in a list of numbers.

    Args:
        weather_data: A list of numbers.
    Returns:
        The maximum value and it's position in the list.
    """
    max = None
    index = None
    for i, value in enumerate(weather_data):
        if max is None or float(value) >= max:
            max = float(value)
            index = i
    return (max, index) if max is not None else ()

def generate_summary(weather_data):
    """Outputs a summary for the given weather data.

    Args:
        weather_data: A list of lists, where each sublist represents a day of
weather data.
    Returns:
        A string containing the summary information.
    """
    output = ""
    list_min = []
    list_max = []
    dates = []
```

```
    output += f"{len(weather_data)} Day Overview\n"

    for day in weather_data:

        list_min.append(convert_f_to_c(day[1]))

        list_max.append(convert_f_to_c(day[2]))

        dates.append(day[0])

    output += f'  The lowest temperature will be
{format_temperature(find_min(list_min)[0])}, and will occur on
{convert_date(dates[list_min.index(find_min(list_min)[0])])}.\n'

    output += f'  The highest temperature will be
{format_temperature(find_max(list_max)[0])}, and will occur on
{convert_date(dates[list_max.index(find_max(list_max)[0])])}.\n'

    output += f'  The average low this week is
{format_temperature(round(calculate_mean(list_min),1))}.\n'

    output += f'  The average high this week is
{format_temperature(round(calculate_mean(list_max),1))}.\n'

    return output



def generate_daily_summary(weather_data):

    """Outputs a daily summary for the given weather data.



    Args:

        weather_data: A list of lists, where each sublist represents a day of
weather data.

    Returns:

        A string containing the summary information.

    """

    results = ""

    for data in weather_data:

        results += f'---- {convert_date(data[0])} ----\n'

        results += f"  Minimum Temperature:
{format_temperature(convert_f_to_c(int(data[1])))}\n "
```

```
        results += f" Maximum Temperature:
{format_temperature(convert_f_to_c(int(data[2])))}\n"

        results += '\n'

    return results
```