Midterm- CPSC 4800

Python

Instructions

1. You should answer questions in this Jupyter notebook file and submit the file on D2L.
2. The exam is open book. You can look at all the course materials, labs, solutions, etc on D2L. However, you are not allowed to communicate or receive help from anyone else during the exam.
3. The work submitted must be your own. Plagiarism and cheating will be dealt with very seriously.

    D. `If the instructor realizes that a student cheated during the exam, he/she will receive a grade of zero for the final.`

Warning: By submitting an exam file, you promise that the code you have submitted is your own and you did not communicate with or receive help from anyone else.

Question	Point
1	5
2	10
3	10
4	10
5	10
6	10
7	10
8	10
9	10
10	15
Total	**100**

Note: Ensure to write brief docstring for your functions

Question 1

Suppose you have list of prices for a certain products. Your job is to find the difference between the lowest price product and highest priced product.

For example:

Input: 25,10,5,30,2

Output: 28

In [3]:
```
price = [25,10,5,30,2,40]
"""
Find the difference between the lowest price product and highest priced produc
"""
def difference(a):
    for i in a:
        j = max(a) - min(a)
        return j
#Check the function
difference(price)
```

Out[3]:
```
38
```

Question 2

You are given a task to create a mask for sending certain message.Write a code that replaces all the odd places in the given message string with * sign.

For example

input: Python is the best programming language

output: P * t * o *

In [5]:
```
message = "Python is the best programming language"
def replace_odd(b):
    """
    Replaces all the odd places in the given message string with * sign
    """
    masked_b = ""
    for i in range(len(b)):
        if (i+1) %2 != 0: #because python string starts with 0 so to avoid div
            masked_b += b[i] #for character at even places, keep as normal
        else:
            masked_b += "*"
    return masked_b
#Check function
replace_odd(message)
```

Out[5]:
```
'P*t*o* *s*t*e*b*s* *r*g*a*m*n* *a*g*a*e'
```

Question 3

Suppose you have dictionary of employees with employee name and age:

{"Bernita Ahner": 12, "Kristie Marsico": 11, "Sara Pardee": 14, "Fallon Fabiano": 11, "Nidia Dominique": 15}

Write a function to find the oldest employee and display his name and age. Also tell how many years are left for retirement if retirement age is 59. Write docstring for your function

output:

Name = Nidia Dominique

Age = 15

Years to retirement= 44

In [8]:
```
my_employees = {"Bernita Ahner": 12, "Kristie Marsico": 11, "Sara Pardee": 14,
```

In [13]:
```
def find_oldest(my_dict):
    """
    Find the oldest employee and display his name and age
    and tell how many years are left for retirement if retirement age is 59
    """
    oldest_employee = max(my_dict, key=my_dict.get)
    oldest_age = my_dict[oldest_employee]

    years_to_retirement = 59 - oldest_age

    output = print(f"Name = {oldest_employee}\nAge = {oldest_age}\nYears to re
    return output
```

In [14]:
```
#Check function
find_oldest(my_employees)
```

```
Name = Nidia Dominique
Age = 15
Years to retirement = 44
```

Question 4

Given the list:

```
my_list = ["course_1", "course_2", ["programming", ["python", "SAS",
"R"]]]
```

Write the code that would index into the list and return the sublist ["SAS", "R"].

In [20]:
```
my_list = ["course_1", "course_2", ["programming", ["python", "SAS", "R"]]]
my_list[2][1][1:]
```

Out[20]:
```
['SAS', 'R']
```

Question 5

Using python indexing, change key_3 **Gaby** to **Raymon** in the below data structure

In [22]:
```
my_dictionary = [{'id':'1234','key_1': {'key_2': 'Jason', 'key_3': 'Smith'}},
        {'id':'2345','key_1': {'key_2': 'Tom', 'key_3': 'Albert'}},
        {'id':'3456','key_1': {'key_2': 'Martin', 'key_3': 'Arts'}},
             {'id':'6789','key_1': {'key_2': 'Julie', 'key_3': 'Gaby'}}]
```

In [30]:
```
my_dictionary[3]['key_1']['key_3'] = "Raymon"
my_dictionary
```

Out[30]:
```
[{'id': '1234', 'key_1': {'key_2': 'Jason', 'key_3': 'Smith'}},
 {'id': '2345', 'key_1': {'key_2': 'Tom', 'key_3': 'Albert'}},
 {'id': '3456', 'key_1': {'key_2': 'Martin', 'key_3': 'Arts'}},
 {'id': '6789', 'key_1': {'key_2': 'Julie', 'key_3': 'Raymon'}}]
```

Question 6

Write a function order_numbers that compares two numbers and return them in the increasing order. Write docstring for your function. Test out your function per the expected output.

Expected output:

```
order_numbers(10, 5)
(5, 10)
```

In [34]:
```
def order_numbers(a,b):
    """
    Compares two numbers and return them in the increasing order
    """
    if a > b:
        return (b,a)
    else:
        return (a,b)
#Check function
order_numbers(40,5)
```

Out[34]:
```
(5, 40)
```

Question 7

Write a function countChar that takes as its arguments two strings, s and c, and return the number of times the character c occurs in the string s. we wiil assume that c is always a string of length 1 (a single character). Here are some samples calls to this fucntion with the correct output given:

```
>> countChar("HELLO", "L")
2
>>countChar ("hello", "m")
0
>> countChar("", "L")
0
```

In [41]:
```
"""
A function that takes as its arguments two strings, s and c,
and return the number of times the character c occurs in the string s
"""
def countChar(s,c):
    count = 0
    for character in s:
        if character == c:
            count += 1
    return count
#Check function
countChar("HELLO", "L")
```

Out[41]: 2

Question 8

Write a function that takes a list of numbers as its argument and return the average of the numbers.

Here are some samples calls to this fucntion with the correct output given:

```
>> average([1, 2, 3, 4, 5, 6])
3.5
>>average([10, 20, 30])
20.0
```

To get the full points, you will need to write a nice docstring above the function that briefly describes what the funciton does and what its inputs and outputs are. You should also test your function according to all the above examples.

In [46]:
```
def average(numbers):
    """
    A function that takes a list of numbers as its argument and return the ave
    """
    if len(numbers) == 0:
        return 0 #Avoid division by zero
    return sum(numbers) / len(numbers)
```

In [54]:
```
#Check function
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
average(numbers)
```

Out[54]: 5.0

Question 9

Go through all the numbers up until 20 (including 20). Print "your name" for every number that's divisible by 3, print your family name for every number divisible by 5, and print 'python' for every number divisible by 3 and by 5! If the number is not divisible either by 3 or 5, print a dash ('-')!

In [55]:
```
for i in range(1, 21): #All the number from 0 to 20
    if i % 3 == 0 and i % 5 == 0:
        print("python")
    elif i % 3 == 0:
        print("your name")
    elif i % 5 == 0:
        print("Nguyen")
    else:
        print("-")
```

```
-
-
your name
-
Nguyen
your name
-
-
your name
Nguyen
-
your name
-
-
python
-
-
your name
-
Nguyen
```

Question 10

- Import the necesary libraries and read the csv file weatherHistory.csv
- Print the head of the dataframe
- Print the complete infomation of data (summary of the dataframe)
- Which type of Weather has the maximum count in the dataset? (Weather type column is the summary column in the dataset)
- What is the standard deviation of the wind speed?
- Out of all the available records which date/dates has/have the lowest humidity?
- Out of all records how many records are equal to the median value of Apparent Temperature?

- Which year has the highest average temperature?

In [56]:
```
import pandas as pd
import numpy as np
import seaborn as sns #import Seaborn
from matplotlib import pyplot as plt
%matplotlib inline
weather = pd.read_csv("weatherHistory (1).csv")
weather.head()
```

Out[56]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visil
0	2006-04-01 00:00:00.000 +0200	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	15.
1	2006-04-01 01:00:00.000 +0200	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	15.
2	2006-04-01 02:00:00.000 +0200	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	14.
3	2006-04-01 03:00:00.000 +0200	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	15.
4	2006-04-01 04:00:00.000 +0200	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	15.

In [57]:
```
#Summary of dataframe
weather.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 96453 entries, 0 to 96452
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Formatted Date            96453 non-null  object 
 1   Summary                   96453 non-null  object 
 2   Precip Type               95936 non-null  object 
 3   Temperature (C)           96453 non-null  float64
 4   Apparent Temperature (C)  96453 non-null  float64
 5   Humidity                  96453 non-null  float64
 6   Wind Speed (km/h)         96453 non-null  float64
 7   Wind Bearing (degrees)    96453 non-null  float64
 8   Visibility (km)           96453 non-null  float64
 9   Loud Cover                96453 non-null  float64
 10  Pressure (millibars)      96453 non-null  float64
 11  Daily Summary             96453 non-null  object 
dtypes: float64(8), object(4)
memory usage: 8.8+ MB
```

In [58]:
```
#Statistical summary of dataframe
weather.describe()
```

Out[58]:

	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Visibility (km)	C
count	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96453.000000	96
mean	11.932678	10.855029	0.734899	10.810640	187.509232	10.347325	
std	9.551546	10.696847	0.195473	6.913571	107.383428	4.192123	
min	-21.822222	-27.716667	0.000000	0.000000	0.000000	0.000000	
25%	4.688889	2.311111	0.600000	5.828200	116.000000	8.339800	
50%	12.000000	12.000000	0.780000	9.965900	180.000000	10.046400	
75%	18.838889	18.838889	0.890000	14.135800	290.000000	14.812000	
max	39.905556	39.344444	1.000000	63.852600	359.000000	16.100000	

In [65]:
```
# "Partly Cloudy" has the maximum count in the dataset
weather['Summary'].value_counts()
```

Out[65]:
```
Partly Cloudy                             31733
Mostly Cloudy                             28094
Overcast                                  16597
Clear                                     10890
Foggy                                      7148
Breezy and Overcast                         528
Breezy and Mostly Cloudy                    516
Breezy and Partly Cloudy                    386
Dry and Partly Cloudy                        86
Windy and Partly Cloudy                      67
Light Rain                                   63
Breezy                                       54
Windy and Overcast                           45
Humid and Mostly Cloudy                      40
Drizzle                                      39
Breezy and Foggy                             35
Windy and Mostly Cloudy                      35
Dry                                          34
Humid and Partly Cloudy                      17
Dry and Mostly Cloudy                        14
Rain                                         10
Windy                                         8
Humid and Overcast                            7
Windy and Foggy                               4
Windy and Dry                                 1
Dangerously Windy and Partly Cloudy           1
Breezy and Dry                                1
Name: Summary, dtype: int64
```

In [67]:
```
#Standard deviation of Wind Speed is 6.91 km/h
weather['Wind Speed (km/h)'].describe()
```

Out[67]:
```
count    96453.000000
mean        10.810640
std          6.913571
min          0.000000
25%          5.828200
50%          9.965900
75%         14.135800
max         63.852600
Name: Wind Speed (km/h), dtype: float64
```

In [76]:
```
# Date of the lowest humidity is 2008/02/17
weather[weather['Humidity'] == weather['Humidity'].min()]['Formatted Date'].he
```

Out[76]:
```
19958    2008-02-17 14:00:00.000 +0100
Name: Formatted Date, dtype: object
```

In [89]:
```
# There are 19 records that are equal to the median value of Apparent Temperat
weather[weather['Apparent Temperature (C)'].median() == weather['Apparent Temp
```

Out[89]:
```
Formatted Date              19
Summary                     19
Precip Type                 19
Temperature (C)             19
Apparent Temperature (C)    19
Humidity                    19
Wind Speed (km/h)           19
Wind Bearing (degrees)      19
Visibility (km)             19
Loud Cover                  19
Pressure (millibars)        19
Daily Summary               19
dtype: int64
```

In [105]:
```
#Which year has the highest average temperature?
from datetime import datetime
weather['Formatted Date'] = pd.to_datetime(weather['Formatted Date'],utc=True)
weather['Year'] = weather['Formatted Date'].dt.year #Add the Year column to da

# Group by year and calculate the average temperature
average_temperature_by_year = weather.groupby('Year')['Temperature (C)'].mean(

# Find the year with the highest average temperature
year_with_highest_average_temp = average_temperature_by_year.idxmax()
print(year_with_highest_average_temp)
```

```
2014
```

In [106]:
```
#Check the final dataframe
weather.head(5)
```

Out[106]:

	Formatted Date	Summary	Precip Type	Temperature (C)	Apparent Temperature (C)	Humidity	Wind Speed (km/h)	Wind Bearing (degrees)	Vi
0	2006-03-31 22:00:00+00:00	Partly Cloudy	rain	9.472222	7.388889	0.89	14.1197	251.0	1
1	2006-03-31 23:00:00+00:00	Partly Cloudy	rain	9.355556	7.227778	0.86	14.2646	259.0	1
2	2006-04-01 00:00:00+00:00	Mostly Cloudy	rain	9.377778	9.377778	0.89	3.9284	204.0	1
3	2006-04-01 01:00:00+00:00	Partly Cloudy	rain	8.288889	5.944444	0.83	14.1036	269.0	1
4	2006-04-01 02:00:00+00:00	Mostly Cloudy	rain	8.755556	6.977778	0.83	11.0446	259.0	1

Type *Markdown* and LaTeX: α^2