# Assignment 1

Name: Rohitha Sai Alla

No: 700734780

Video link: <https://drive.google.com/file/d/1ydzs5YRYD2poR2qDCSg545tFdgQXzO8E/view?usp=sharing>

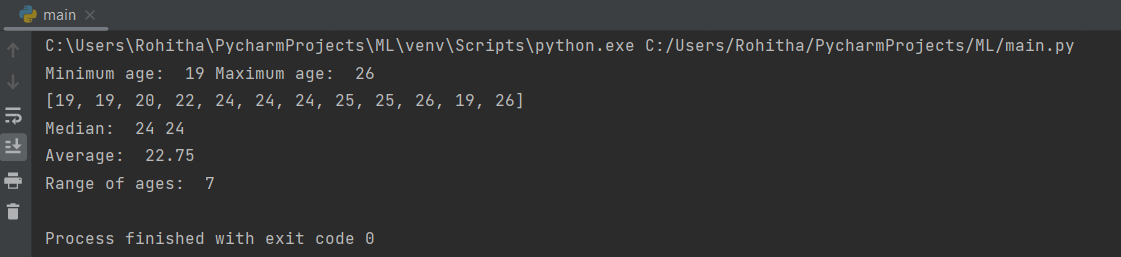
1.

ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]  
#sorting the list  
ages.sort()  
#finding the min and max age  
min = ages[0]  
max = ages[-1]  
print("Minimum age: ", min, "Maximum age: ", max)  
  
#adding min and max age to list  
ages.append(min)  
ages.append(max)  
print(ages)  
  
#finding the median age  
if len(ages) % 2 != 0:  
 med = int((len(ages)+1)/2-1)  
 median = ages[med]  
 print("Median: ", median)  
  
else:  
 med1 = int(len(ages)/2 - 1)  
 med2 = int(len(ages)/2)  
 median = (ages[med1]+ages[med2])/2  
 print("Median: ", ages[med1],ages[med2])  
  
#find average age  
avg = sum(ages) / len(ages)  
print("Average: ",avg)  
  
#finding range of ages  
age\_range = max - min  
print("Range of ages: ",age\_range)

After sorting add the first and last element to the list and checked if the length of list is even or odd.

If list length is even then I am returning two values else one.

Output:



2.

Created new dictionary and updated values as asked.

Performed update operations and added skills as list inside dictionary.

Also updated skills with new extra values. Shown the keys and value pairs.

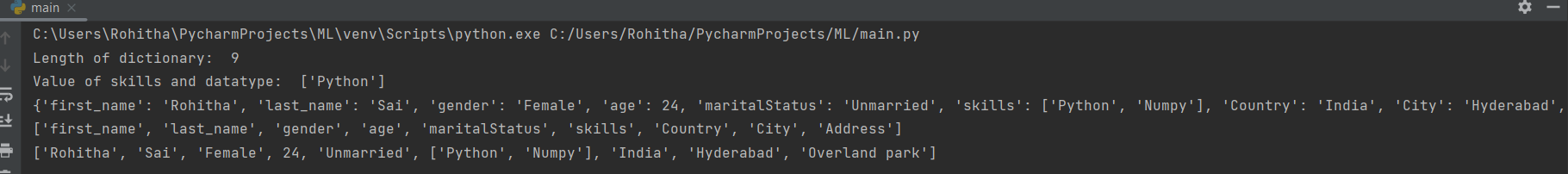
dog = {} # creating empty dictionary  
dog.update({"name": "Husky", "color": "White", "breed": "pug", "legs": "4", "age": "3"}) # adding values  
student\_dict = {  
 "first\_name": "Rohitha",  
 "last\_name": "Sai",  
 "gender": "Female",  
 "age": 24, "maritalStatus": "Unmarried", "skills": ["Python"], "Country": "India", "City": "Hyderabad",  
 "Address": "Overland park"  
}  
print("Length of student dictionary: ", len(student\_dict))  
print("Value of skills and datatype: ", student\_dict["skills"])

# updating skills in dictionary  
student\_dict.update({"skills": ["Python", "Numpy"]})  
print("Modified skills: ",student\_dict)

# Getting dictionary keys as a list  
print(list(student\_dict.keys()))

# Getting dictionary values as a list  
print(list(student\_dict.values()))

Output:



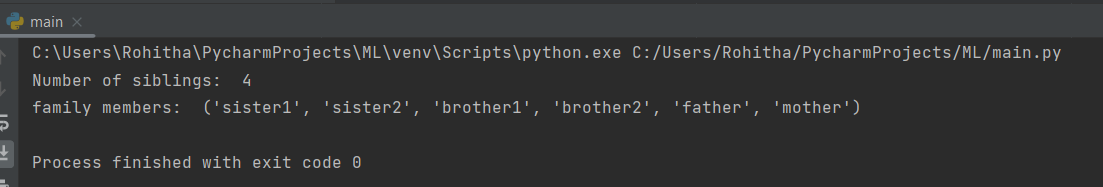
3. Created two tuples sisters and brothers and merged these two into another tuple named siblings.

Printed the count of the siblings tuple.

Created new tuple named family\_members and added father and mother to family\_members along with siblings.

# Creating tuple with sisters  
sisters = ("sister1", "sister2")  
  
# Creating tuple with brothers  
brothers = ("brother1", "brother2")  
  
# Merging two tuples  
siblings = sisters + brothers  
  
# Counting the siblings  
print("Number of siblings: ", len(siblings))  
  
# Adding father and mother  
family\_members = siblings + ("father", "mother")  
print("family members: ", family\_members)

Output:



4.

Used len(set\_name) to find length of set.

Used .add() to add new element

For inserting multiple companies at once I have stored them into one set and used .update to add them to existing set.

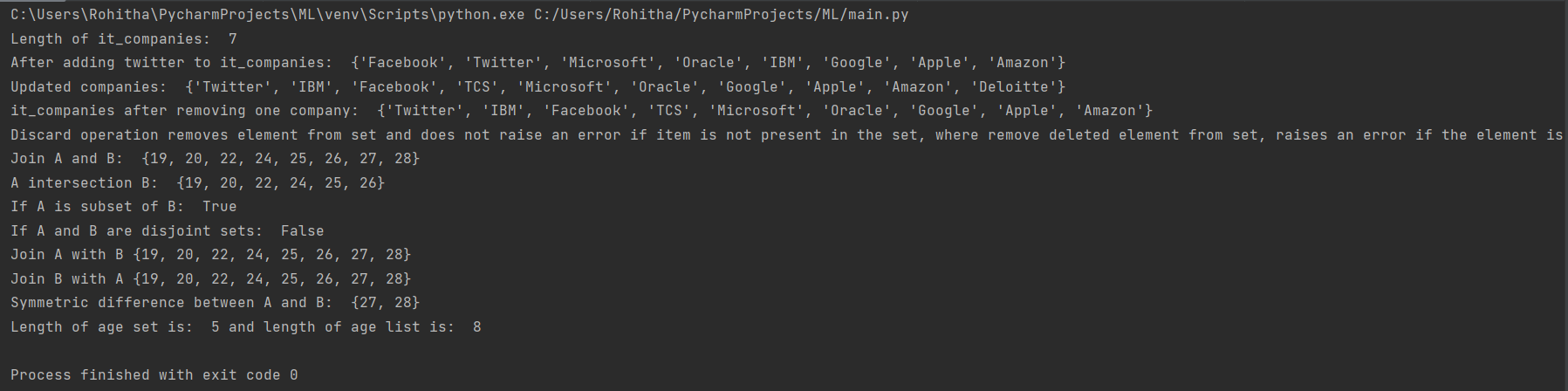
Performed various et operations on it\_companies.

To delete sets completely I have used A.clear()

it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}  
A = {19, 22, 24, 20, 25, 26}  
B = {19, 22, 20, 25, 26, 24, 28, 27}  
age = [22, 19, 24, 25, 26, 24, 25, 24]  
  
# Length of set it\_companies  
print("Length of it\_companies: ", len(it\_companies))  
  
# Adding twitter to it\_companies  
it\_companies.add("Twitter")  
print("After adding twitter to it\_companies: ", it\_companies)  
  
# Inserting multiple it\_companies at once  
new\_companies = {"Deloitte", "TCS"}  
it\_companies.update(new\_companies)  
print("Updated companies: ", it\_companies)  
  
# To remove a company  
it\_companies.remove("Deloitte")  
print("it\_companies after removing one company: ", it\_companies);  
  
# to implement discard  
it\_companies.discard("Deloitte")  
print(  
 "Discard operation removes element from set and does not raise an error if item is not present in the set, where remove deleted element from set, raises an error if the element is not present in the set.")  
  
# Join A and B  
print("Join A and B: ", A.union(B))  
  
# Intersection A and B  
C = A.intersection(B)  
print("A intersection B: ", C)  
  
# To check if A is subset of B  
print("If A is subset of B: ", A.issubset(B))  
  
# A and B are disjoint sets  
print("If A and B are disjoint sets: ", A.isdisjoint(B))  
  
# Join A with B and B with A  
print("Join A with B", A.union(B))  
print("Join B with A", B.union(A))  
  
# Symmetric difference between A and B  
print("Symmetric difference between A and B: ", A.symmetric\_difference(B))  
  
# Delete the sets completely  
A.clear()  
B.clear()  
C.clear()  
  
# Convert age to set  
age\_set = set(age)  
print("Length of age set is: ", len(age\_set), "and length of age list is: ", len(age))

Discard operation removes the element from set and remove is also used for the same. But discard does not raise an error if element is not present in the set. Where remove raises an error is element is not present in set.

Output:



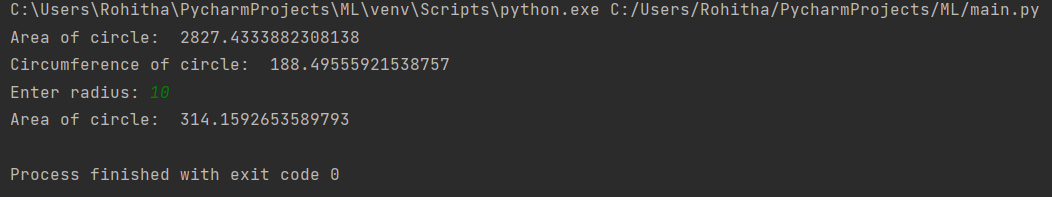
Length of age set is 5 and length of age list is 8

5.

import math  
  
# Value given in question  
radius = 30  
# Calculate area of circle and assign value to \_area\_of\_circle\_  
\_area\_of\_circle\_ = math.pi \* radius \* radius  
print("Area of circle: ", \_area\_of\_circle\_)  
# Calculate circumference of circle and assign value to \_circum\_of\_circle\_  
\_circum\_of\_circle\_ = 2 \* math.pi \* radius  
print("Circumference of circle: ", \_circum\_of\_circle\_)  
  
# Take radius as per user input and calculate area  
radius = int(input("Enter radius: "))  
print("Area of circle: ", math.pi \* radius \* radius)

Calculated area and circumference with given radius and computed with user input.

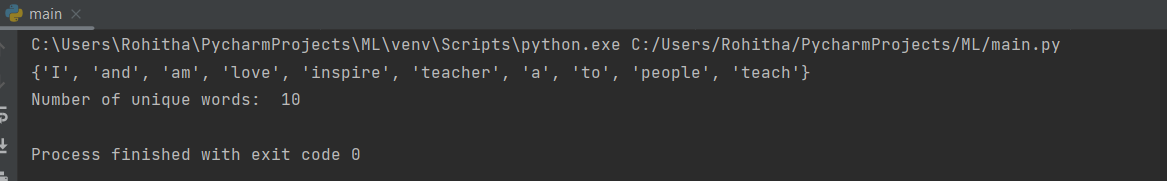
Output:



6. I have converted the given string to a set inorder to get unique words

x = "I am a teacher and I love to inspire and teach people"  
# find unique words in the sentence  
words = x.split(" ")  
unique\_words = set(words)  
print(unique\_words)  
  
# To find unique words  
print("Number of unique words: ", len(unique\_words))

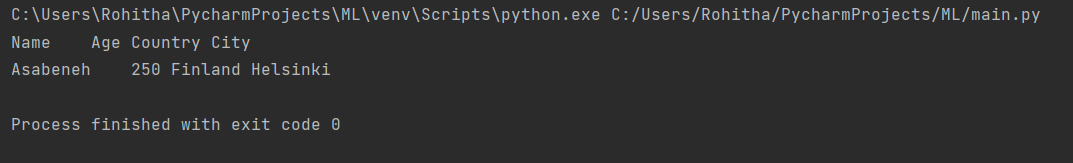
Output:



7. Used \t – tab escape to give spaces in the words.

#Using tab escape sequence  
print("Name\tAge\tCountry\tCity")  
print("Asabeneh\t250\tFinland\tHelsinki")

Output:



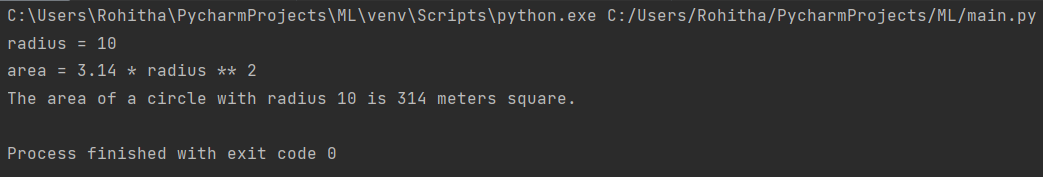
8.Used \n to navigate to next line.

Used string formatting to get the float values in desired format.

To not get decimal places for radius I have used %s. (to get same output as shown in question)

#print using string formatting  
radius = 10  
area = 3.14 \* (radius \*\* 2)  
print("radius = %1.0f\narea = 3.14 \* radius \*\* 2\nThe area of a circle with radius %s is %1.0f meters square."%(radius, radius, area))

Output:



9.

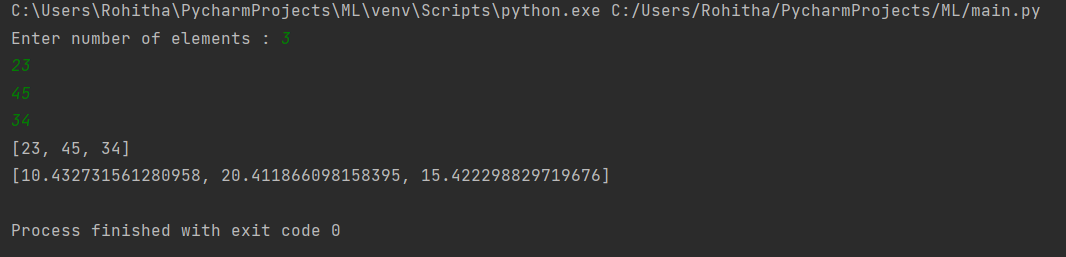
Read N value from user and used a for loop to take those values from user.

Stored all these values in a list.

Later divided each list item with 2.2046 to convert from lbs to kgs.

# to read n number of list elements as input  
L1 = []  
N = int(input("Enter number of elements : "))  
  
# to take values till the number of elements  
for i in range(0, N):  
 lbs = int(input())  
  
 L1.append(lbs)  
  
# converting from lbs to kgs  
print(L1)  
list\_kg = [item / 2.2046 for item in L1]  
print(list\_kg)

Output:



10.

Initially took all points marked as values to x-axis and all points as points into dataframe called data.

Divided the data to test and train.

Used sklearn module to compute knn value for 3.

Computed predict value, confusion matrix.

From confusion matrix, calculated false positive, true negative, true positive, false negative.

Using those values sensitivity, specificity are derived.

import numpy as np  
import pandas as pd  
from matplotlib import pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn import metrics  
  
# giving dataframe the data  
data = {'x-axis': ['1', '2', '3', '6', '6', '7', '10', '11'], 'points': ['o', 'o', 'x', 'x', 'x', 'o', 'o', 'o']}  
df = pd.DataFrame(data)  
print(df)  
  
# arranging data in plottable form  
x = df.loc[:, ["x-axis"]]  
dots = df.loc[:, ["points"]]  
x\_train, x\_test, dot\_train, dot\_test = train\_test\_split(x, dots, random\_state=0, train\_size=0.5) # divided to test and train  
  
# importing sklearn for knn  
from sklearn.neighbors import KNeighborsClassifier  
  
knn = KNeighborsClassifier(n\_neighbors=3) # taking knn as 3  
knn.fit(x\_train, dot\_train) # fit with train data  
predictoutput = knn.predict(x\_test) # predict with test data  
print("Predicted output for test samples: ", predictoutput)  
  
# to calculate accuracy  
acc\_knn = round(knn.score(x\_test, dot\_test) \* 100, 2)  
print('KNN accuracy is:', acc\_knn)  
  
# confusion matrix  
confusion\_matrix = metrics.confusion\_matrix(x\_test, dot\_test)  
print("Confusion matrix: ", confusion\_matrix)  
  
confusion\_matrix = np.matrix(confusion\_matrix)  
FP = confusion\_matrix.sum(axis=0) - np.diag(confusion\_matrix)  
FN = confusion\_matrix.sum(axis=1) - np.diag(confusion\_matrix)  
TP = np.diag(confusion\_matrix)  
TN = confusion\_matrix.sum() - (FP + FN + TP)  
  
# sensitivity values  
sensitivity\_value = TP/(TP+FN)  
print("Sensitivity value: ", sensitivity\_value)  
  
# specificity value  
specificity = TN/(TN+FP)  
print("Specificity value: ", specificity)

Output:

