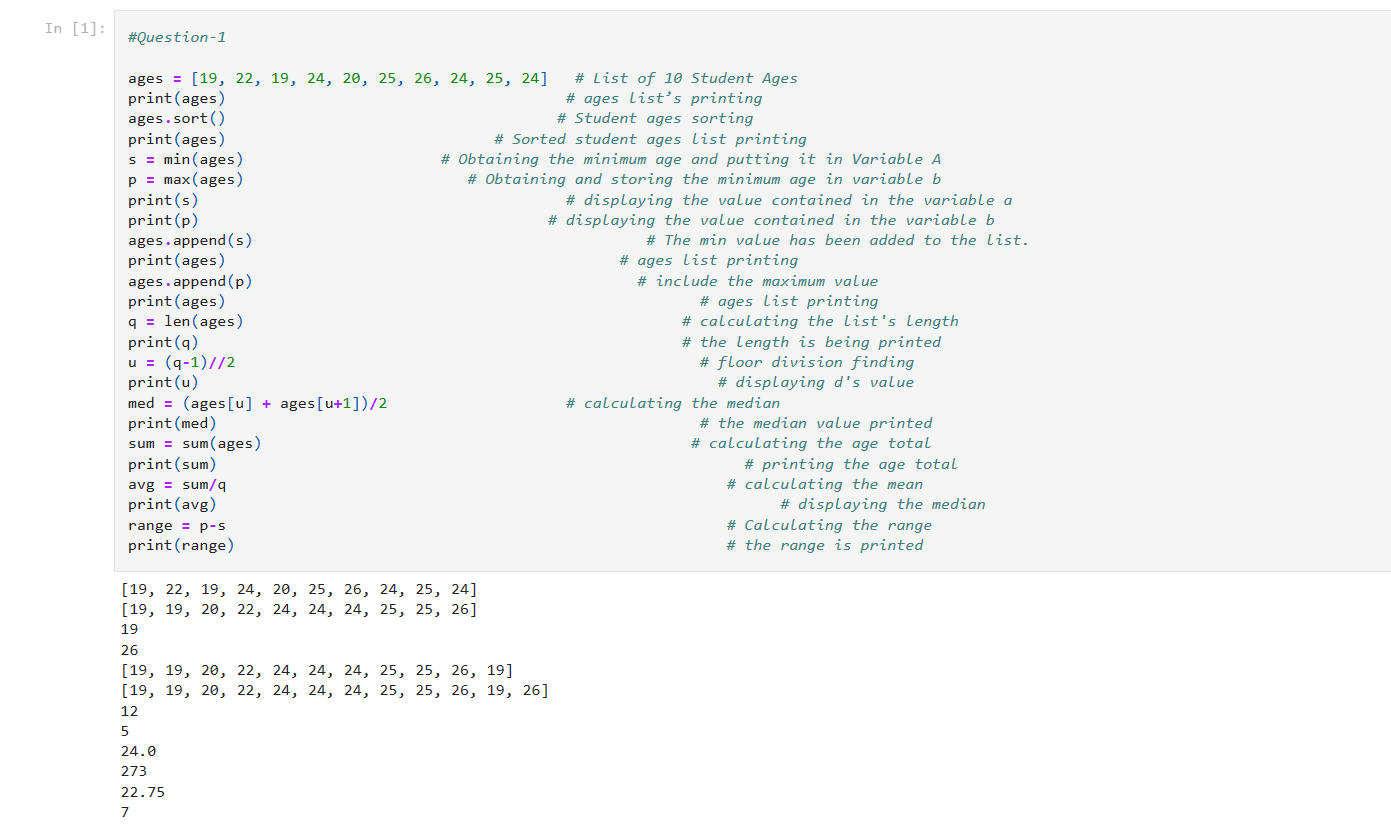
**Name: Shameer Shaik**

**Student ID: - 7007405840**

**GitHub link: https://github.com/SHAMEERSHAIK7/Machine-Learning-CS-5710**

Question 1: -

Ages **=** [19, 22, 19, 24, 20, 25, 26, 24, 25, 24] *# List of 10 Student Ages*  
*print(ages) # ages list’s printing*  
*ages***.**sort() *# Student ages sorting*  
*print(ages) # Sorted student ages list printing*  
*s* **=** min(ages) *# Obtaining the minimum age and putting it in Variable A*  
*p* **=** max(ages) *# Obtaining and storing the minimum age in variable b*  
*print(s) # displaying the value contained in the variable a*  
*print(p) # displaying the value contained in the variable b*  
*ages***.**append(s) *# The min value has been added to the list.*  
*print(ages) # ages list printing*  
*ages***.**append(p) *# include the maximum value*  
*print(ages) # ages list printing*  
*q* **=** len(ages) *# calculating the list's length*  
*print(q) # the length is being printed*  
*u* **=** (q**-**1)**//**2 *# floor division finding*  
*print(u) # displaying d's value*  
*med* **=** (ages[u] **+** ages[u**+**1])**/**2 *# calculating the median*  
*print(med) # the median value printed*  
*sum* **=** sum(ages) *# calculating the age total*  
*print(sum) # printing the age total*  
*avg* **=** sum**/**q *# calculating the mean*  
*print(avg) # displaying the median*  
*range* **=** p**-**s *# Calculating the range*  
*print(range) # the range is printed*



Question 2:

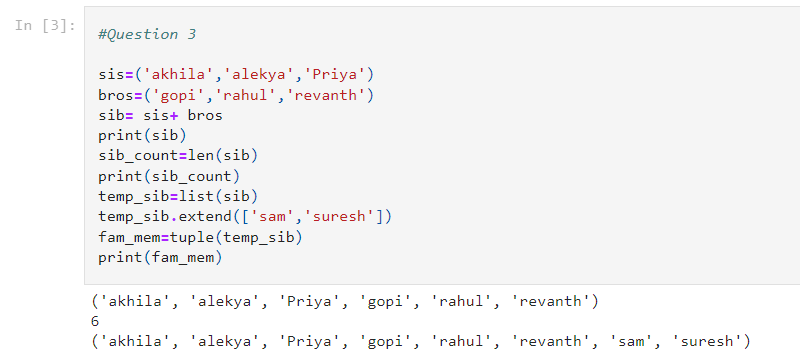
dog**=**{}  
dog['name']**=** 'Rocky'  
dog['color']**=** 'brown'  
dog['breed']**=** 'German Shepherd'  
dog['legs']**=**4  
dog['age']**=**5  
  
person**=**dict()  
person['first\_name']**=**'Shameer'  
person['last\_name']**=** 'Shaik'  
person['gender']**=**'Male'  
person['age']**=**24  
person['Marital\_status']**=**'No'  
person['skills']**=**['C','Python']  
person['country']**=**'India'  
person['city']**=**'Vijayawada'  
person['address']**=** 'Vijayawada Local'

print(person)  
print(len(person))  
skills\_of\_person**=** person**.**get('skills')  
print(person**.**get('skills'))  
print(type(skills\_of\_person))  
person**.**update({'skills':['C','Java','Python','Pyspark','Angular']})  
print(person)  
person\_keys**=** list(person**.**keys())  
print(person\_keys)  
person\_values**=** list(person**.**values())  
print(person\_values)



Question 3:-

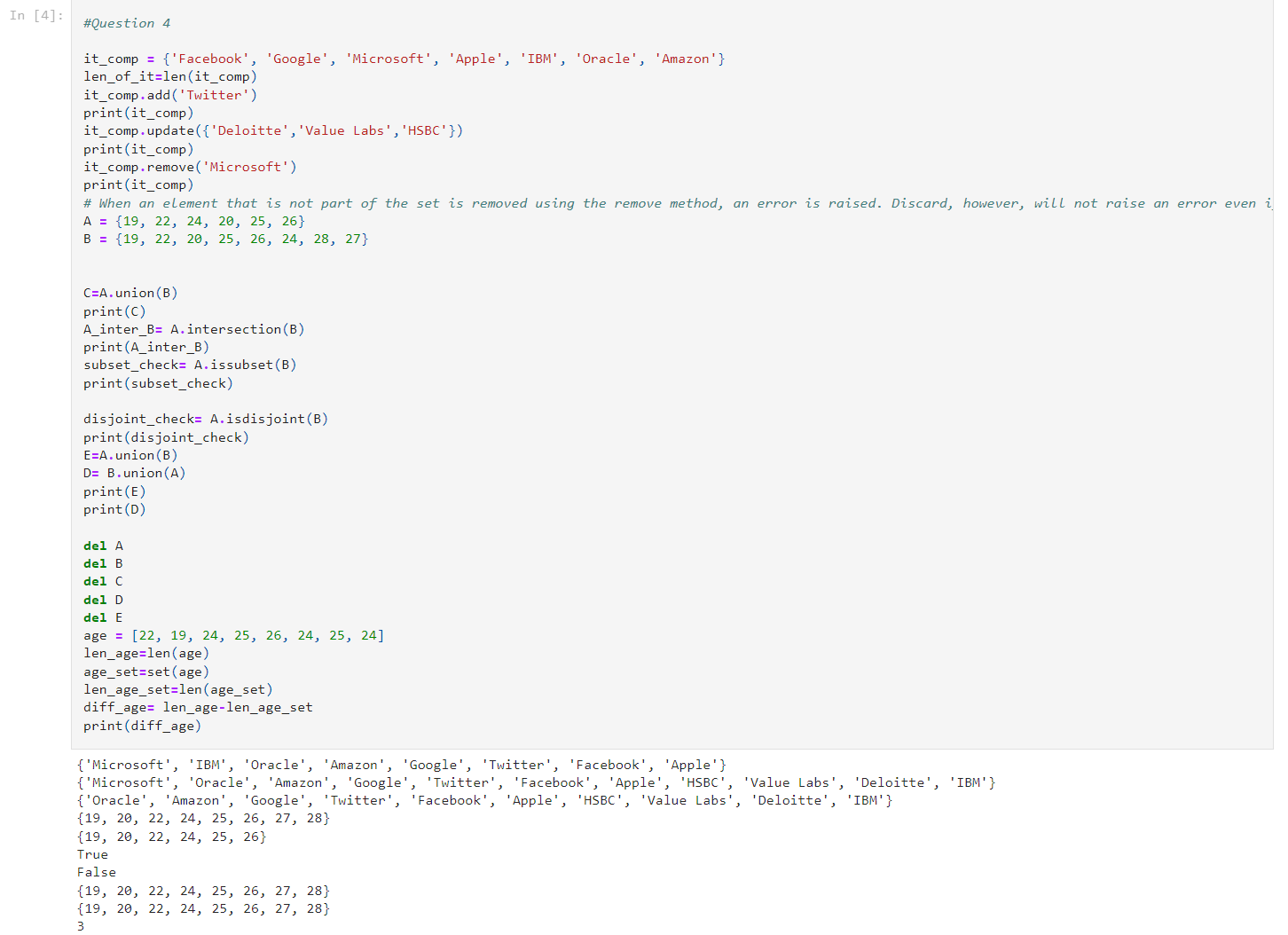
sis**=**('akhila','alekya','Priya')  
bros**=**('gopi','rahul','revanth')  
sib**=** sis**+** bros  
print(sib)  
sib\_count**=**len(sib)  
print(sib\_count)  
temp\_sib**=**list(sib)  
temp\_sib**.**extend(['sam','suresh'])  
fam\_mem**=**tuple(temp\_sib)  
print(fam\_mem)



Question 4:-

it\_comp **=** {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}  
len\_of\_it**=**len(it\_comp)  
it\_comp**.**add('Twitter')  
print(it\_comp)  
it\_comp**.**update({'Deloitte','Value Labs','HSBC'})  
print(it\_comp)  
it\_comp**.**remove('Microsoft')  
print(it\_comp)  
# When an element that is not part of the set is removed using the remove method, an error is raised. Discard, however, will not raise an error even if no element is present.  
A **=** {19, 22, 24, 20, 25, 26}  
B **=** {19, 22, 20, 25, 26, 24, 28, 27}  
  
  
C**=**A**.**union(B)  
print(C)  
A\_inter\_B**=** A**.**intersection(B)  
print(A\_inter\_B)  
subset\_check**=** A**.**issubset(B)  
print(subset\_check)  
  
disjoint\_check**=** A**.**isdisjoint(B)  
print(disjoint\_check)  
E**=**A**.**union(B)  
D**=** B**.**union(A)  
print(E)  
print(D)

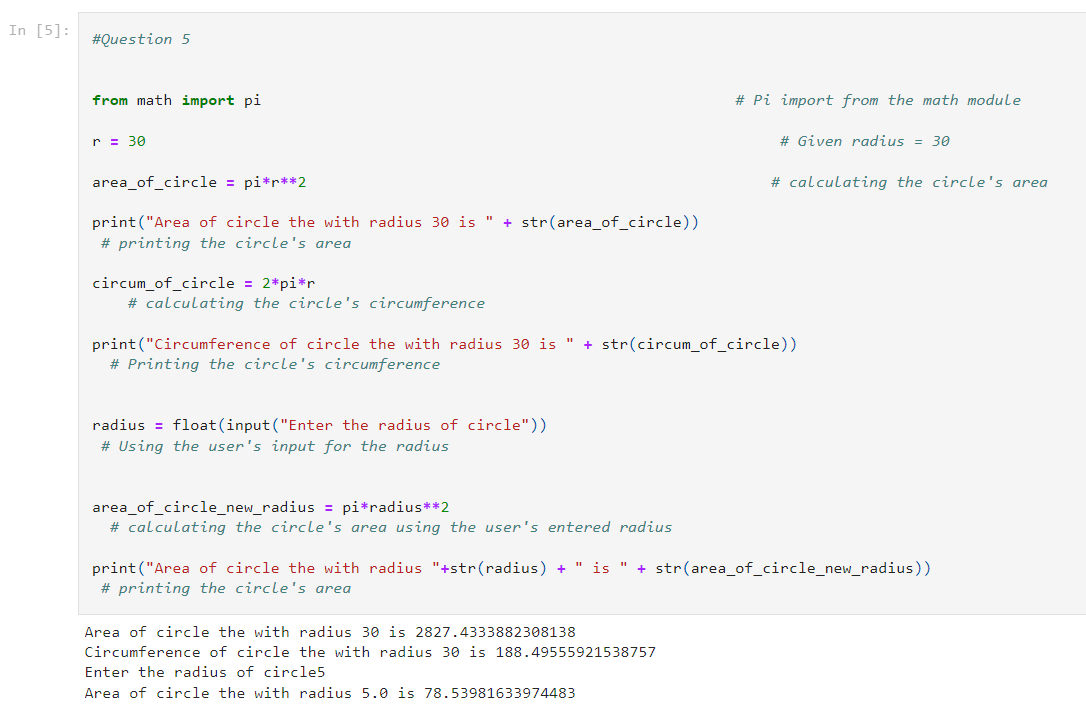
del A   
del B  
del C  
del D  
del E  
age **=** [22, 19, 24, 25, 26, 24, 25, 24]  
len\_age**=**len(age)  
age\_set**=**set(age)  
len\_age\_set**=**len(age\_set)  
diff\_age**=** len\_age**-**len\_age\_set  
print(diff\_age)



Question 5:-

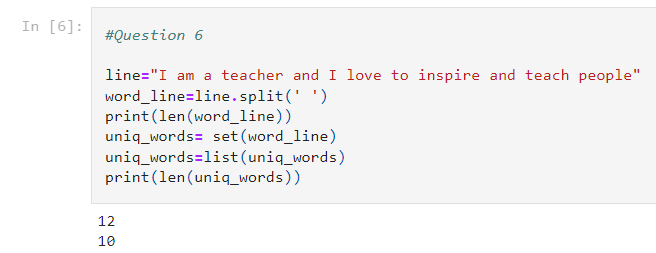
**from** math **import** pi *# Pi import from the math module*  
  
*r* **=** 30 *# Given radius = 30*  
  
*area\_of\_circle* **=** pi**\***r**\*\***2 *# calculating the circle's area*  
  
*print("Area of circle the with radius 30 is "* **+** str(area\_of\_circle))   
 *# printing the circle's area*

*circum\_of\_circle* **=** 2**\***pi**\***r   
 *# calculating the circle's circumference*  
  
*print("Circumference of circle the with radius 30 is "* **+** str(circum\_of\_circle))   
 *# Printing the circle's circumference*  
  
  
*radius* **=** float(input("Enter the radius of circle"))   
 *# Using the user's input for the radius*  
  
  
*area\_of\_circle\_new\_radius* **=** pi**\***radius**\*\***2   
 *# calculating the circle's area using the user's entered radius*  
  
*print("Area of circle the with radius "***+**str(radius) **+** " is " **+** str(area\_of\_circle\_new\_radius))  
 *# printing the circle's area*



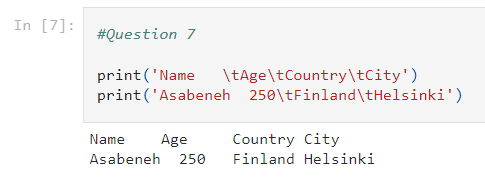
Question 6:-

*line***=**"I am a teacher and I love to inspire and teach people"  
word\_line**=**line**.**split(' ')  
print(len(word\_line))  
uniq\_words**=** set(word\_line)  
uniq\_words**=**list(uniq\_words)  
print(len(uniq\_words))



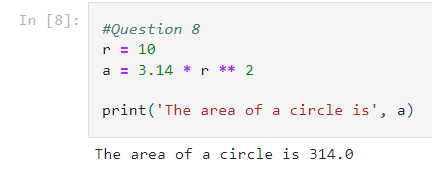
Question 7:

print('Name \tAge\tCountry\tCity')  
print('Asabeneh 250\tFinland\tHelsinki')



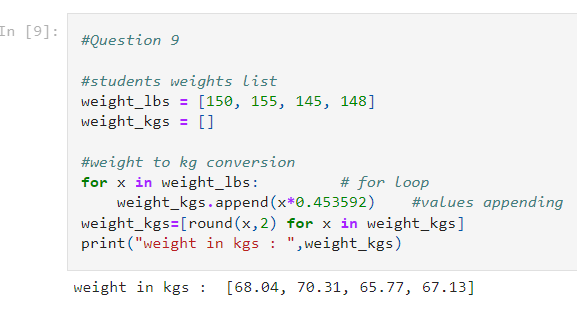
Question 8:-

r **=** 10  
a **=** 3.14 **\*** r **\*\*** 2  
  
print('The area of a circle is', a)



Question 9:-

*#students weights list*  
*weight\_lbs* **=** [150, 155, 145, 148]  
weight\_kgs **=** []  
  
#weight to kg conversion  
for x **in** weight\_lbs: *# for loop*  
 weight\_kgs**.**append(x**\***0.453592) *#values appending*  
*weight\_kgs***=**[round(x,2) **for** x **in** weight\_kgs]  
print("weight in kgs : ",weight\_kgs)



Question 10:-

**import** numpy *#Bringing in the required libraries*   
*import* sklearn  
from sklearn.model\_selection **import** train\_test\_split  
from sklearn.neighbors **import** KNeighborsClassifier  
from sklearn.metrics **import** accuracy\_score  
from collections **import** Counter  
import numpy **as** np  
from sklearn.linear\_model **import** LogisticRegression  
a **=** np**.**array([1,2,3,6,6,7,10,11]) *#constructing arrays*  
*y* **=** np**.**array([0,0,1,1,1,0,0,0])  
x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(a,y,test\_size**=**0.5,random\_state**=**4,stratify **=** y)  
x\_train **=** x\_train**.**reshape(**-**1,1)  
x\_test **=** x\_test**.**reshape(**-**1,1)  
y\_train **=** y\_train**.**reshape(**-**1,1)  
y\_test **=** y\_test**.**reshape(**-**1,1)  
print(x\_train,x\_test,y\_train,y\_test)  
def KNNAlgo(a,b,k,d):  
 dist **=** []   
 *#The Euclidean distance calculation*  
 dist\_ind **=** np**.**sqrt(np**.**sum((a**-**b)**\*\***2, axis**=**1))  
 print(dist\_ind)  
 *#combining the distance and the output label*  
 main\_arr **=** np**.**column\_stack((d,dist\_ind))  
 print(main\_arr)  
 *#Distances are arranged in descending order.*  
 main **=** main\_arr[main\_arr[:,1]**.**argsort()]  
 *#Based on the value of k, calculating the frequency of the labels*  
 count **=** Counter(main[0:k,0])  
 keys, vals **=** list(count**.**keys()), list(count**.**values())  
 **if** len(vals)**>**1:  
 **if** vals[0]**>**vals[1]:  
 **return** int(keys[0])  
 **else**:  
 **return** int(keys[1])  
 **else**:  
 **return** int(keys[0])

k**=**3  
knn **=** KNeighborsClassifier(n\_neighbors**=**k)  
knn**.**fit(x\_train,y\_train)  
ypred **=** knn**.**predict(x\_test)   
for i **in** x\_test: *#estimating the results' values*  
 result **=** KNNAlgo(x\_train,i,3,y\_train)  
 print(result)  
print(y\_test,ypred)

A **=** confusion\_matrix(y\_test, y\_test\_pred)  
print(A)

from sklearn.metrics **import** accuracy\_score, confusion\_matrix  
confusion\_matrix(y\_test, ypred)  
accuracy\_score(y\_test,ypred) *#determining the precision*  
*A* **=** confusion\_matrix(y\_test, ypred)  
print(A)  
sensitivity **=** (A[1][1])**/**(A[1][1] **+** A[1][0]) *#figuring out the sensitivity*  
*print(sensitivity)*   
*specificity* **=** (A[0][0])**/**(A[0][0]**+**A[0][1]) *#Making a Specificity Calculation*  
*print(specificity)*

