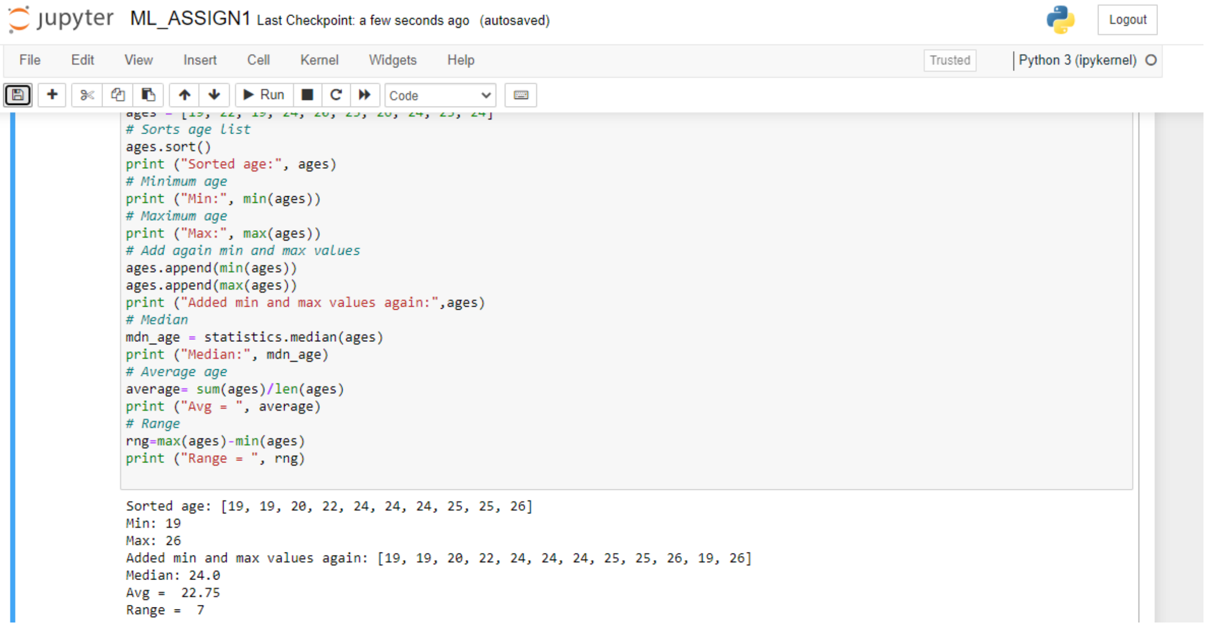
Video Link:
https://1drv.ms/v/s!BFvnPTDPvjkohVk0Wvs19qVpogIb?e=m421KAMVdUyi8tmO4Zr5vA&at=9

Q1) Sort the list of ages, find min and max, average, median and range

```
#Importing library called statistics which helps in calculating mathematical data

import statistics

ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

# Sorts age list in ascending order by default

ages.sort()

print ("Sorted age:", ages) # Displays sorted values

# Minimum age

# Displays min value as we used min() method

print ("Min:", min(ages))

# Maximum age

 # Displays max value as we used max() method

print ("Max:", max(ages))

# Adding again min and max values so we use append() method to insert values to the list

ages.append(min(ages))

ages.append(max(ages))

print ("Added min and max values again:",ages) #Displays the list again with new values

# Median (one middle item or two middle items divided by two, as we imported statistics library it calculates easily and provides the opt)

mdn_age = statistics.median(ages)

print ("Median:", mdn_age)

# Average age

average= sum(ages)/len(ages)

print ("Avg = ", average)

# Range

rng=max(ages)-min(ages)

print ("Range = ", rng)
```



```
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
# Sorts age list
ages.sort()
print ("Sorted age:", ages)
# Minimum age
print ("Min:", min(ages))
# Maximum age
print ("Max:", max(ages))
# Add again min and max values
ages.append(min(ages))
ages.append(max(ages))
print ("Added min and max values again:",ages)
# Median
mdn_age = statistics.median(ages)
print ("Median:", mdn_age)
# Average age
average= sum(ages)/len(ages)
print ("Avg = ", average)
# Range
rng=max(ages)-min(ages)
print ("Range = ", rng)
```

```
Sorted age: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
Min: 19
Max: 26
Added min and max values again: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
Median: 24.0
Avg =  22.75
Range =  7
```

Q2) Create a dictionary

# Dog dictionary is created with given key and values

dog = {'name':'Tommy','color':'white','breed':'husky','legs':'4','age':'2'}

print ("Dog Dictionary Created:",dog)

# Student dictionary is created with given key and values

student = {'first_name':'Srujana','last_name':'Makutam','Gender':'Female','age':'22','marital_status':'single',

'skills':'dancer','Country':'India','City':'Hyderabad','Address':'1/180'}

print ("Student Dictionary Created:",student)

# Create another dictionary for skills

skills = {'dancer':'1','singer':'2','coder':'3'}

print ("Skills Dictionary Created:",skills)

# Find the length of student dictionary

print ("Length of student:", len(student))

# Check the datatype of skills

print ("Datatype fo skills:",type(skills))

# Get values of skills dictionary

print ("Values of skills:",skills.values())

# Add one item to skills

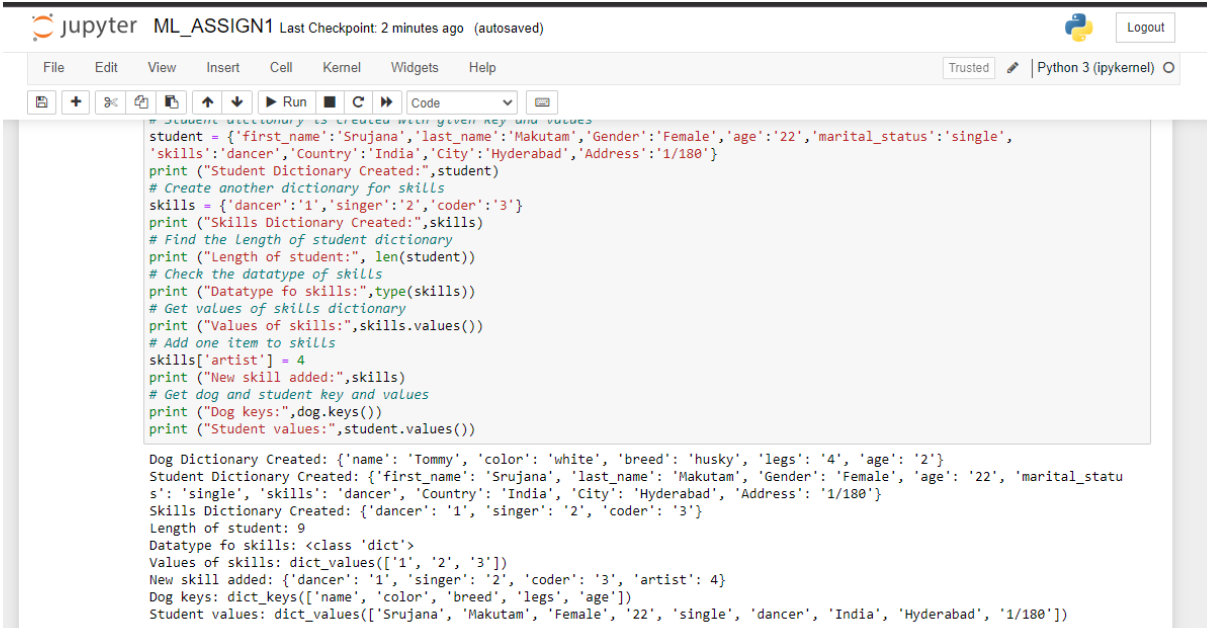
skills['artist'] = 4

print ("New skill added:",skills)

Get dog and student key and values

print ("Dog keys:",dog.keys())

print ("Student values:",student.values())



```
# Student dictionary is created with given key and values
student = {'first_name':'Srujana','last_name':'Makutam','Gender':'Female','age':'22','marital_status':'single',
'skills':'dancer','Country':'India','City':'Hyderabad','Address':'1/180'}
print ("Student Dictionary Created:",student)
# Create another dictionary for skills
skills = {'dancer':'1','singer':'2','coder':'3'}
print ("Skills Dictionary Created:",skills)
# Find the length of student dictionary
print ("Length of student:", len(student))
# Check the datatype of skills
print ("Datatype fo skills:",type(skills))
# Get values of skills dictionary
print ("Values of skills:",skills.values())
# Add one item to skills
skills['artist'] = 4
print ("New skill added:",skills)
# Get dog and student key and values
print ("Dog keys:",dog.keys())
print ("Student values:",student.values())
```

```
Dog Dictionary Created: {'name': 'Tommy', 'color': 'white', 'breed': 'husky', 'legs': '4', 'age': '2'}
Student Dictionary Created: {'first_name': 'Srujana', 'last_name': 'Makutam', 'Gender': 'Female', 'age': '22', 'marital_statu
s': 'single', 'skills': 'dancer', 'Country': 'India', 'City': 'Hyderabad', 'Address': '1/180'}
Skills Dictionary Created: {'dancer': '1', 'singer': '2', 'coder': '3'}
Length of student: 9
Datatype fo skills: <class 'dict'>
Values of skills: dict_values(['1', '2', '3'])
New skill added: {'dancer': '1', 'singer': '2', 'coder': '3', 'artist': 4}
Dog keys: dict_keys(['name', 'color', 'breed', 'legs', 'age'])
Student values: dict_values(['Srujana', 'Makutam', 'Female', '22', 'single', 'dancer', 'India', 'Hyderabad', '1/180'])
```

Q3) Create tuple of sisters and brothers

my_sisters = ('Sanjana', 'Sheethal','Shivani','Spoorthi')

my_brothers = ('Akhil','Suchith','Vandith','Vaishnav')

Create another tuple as siblings and join the sister’s and brother’s tuple

siblings = my_sisters + my_brothers

Displays siblings’ output and length of siblings

print("Siblings:", siblings)

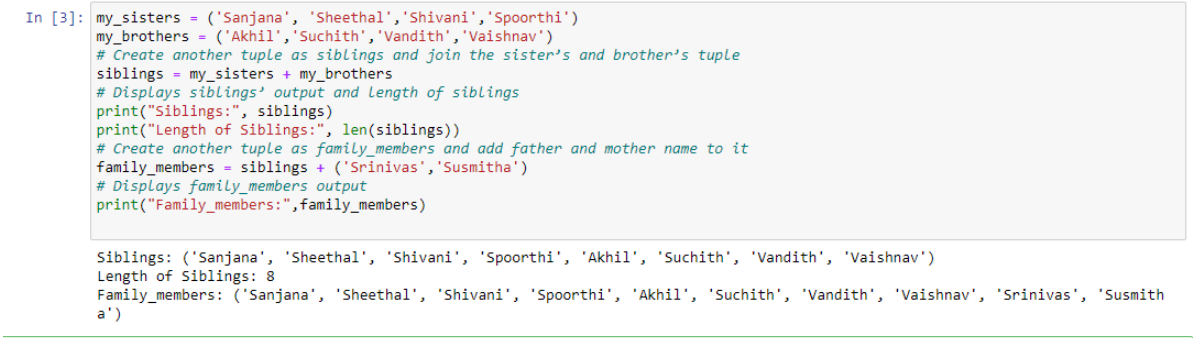
print("Length of Siblings:", len(siblings))

Create another tuple as family_members and add father and mother name to it

family_members = siblings + ('Srinivas','Susmitha')

Displays family_members output

print("Family_members:",family_members)



```
In [3]: my_sisters = ('Sanjana', 'Sheethal','Shivani','Spoorthi')
        my_brothers = ('Akhil','Suchith','Vandith','Vaishnav')
        # Create another tuple as siblings and join the sister’s and brother’s tuple
        siblings = my_sisters + my_brothers
        # Displays siblings’ output and length of siblings
        print("Siblings:", siblings)
        print("Length of Siblings:", len(siblings))
        # Create another tuple as family_members and add father and mother name to it
        family_members = siblings + ('Srinivas','Susmitha')
        # Displays family_members output
        print("Family_members:",family_members)
```

```
Siblings: ('Sanjana', 'Sheethal', 'Shivani', 'Spoorthi', 'Akhil', 'Suchith', 'Vandith', 'Vaishnav')
Length of Siblings: 8
Family_members: ('Sanjana', 'Sheethal', 'Shivani', 'Spoorthi', 'Akhil', 'Suchith', 'Vandith', 'Vaishnav', 'Srinivas', 'Susmith
a')
```

Q4) Length of the set

```
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
print("Length of it_companies:", len(it_companies))
#Add twitter
it_companies.add('Twitter')
print("After adding another item:",it_companies)
#Add multiple it_companies
it_companies.update({'Infosys','Capgemini','Wipro','TCS'})
print("After adding multiple items:",it_companies)
#Remove
it_companies.remove('TCS')
print("After removing one company:",it_companies)
#Discard
it_companies.discard('TCS')
print("After discarding  company:",it_companies)
#Discard doesn't raise any error if any item is not present in the set
#Join A & B
A = {19, 22, 24, 20, 25, 26}
B = {19, 22, 20, 25, 26, 24, 28, 27}
print("Join A and B:", A.union(B))
#Intersection
print("Intersection of A and B:", A.intersection(B))
#Subset
print("Subset of A and B:", A.issubset(B))
#Disjoint
print("Disjoint:", A.isdisjoint(B))
#Convert list to set
age = [22, 19, 24, 25, 26, 24, 25, 24]
print("Converting list to set:", set(age))
#Length of set
print("Length of set:",len(set(age)))
#Length of list
```

print("Length of list:",len(age))

#Symmetric diff- returns values which are not in common with other set

print("Symmetric diff:",A.symmetric_difference(B))

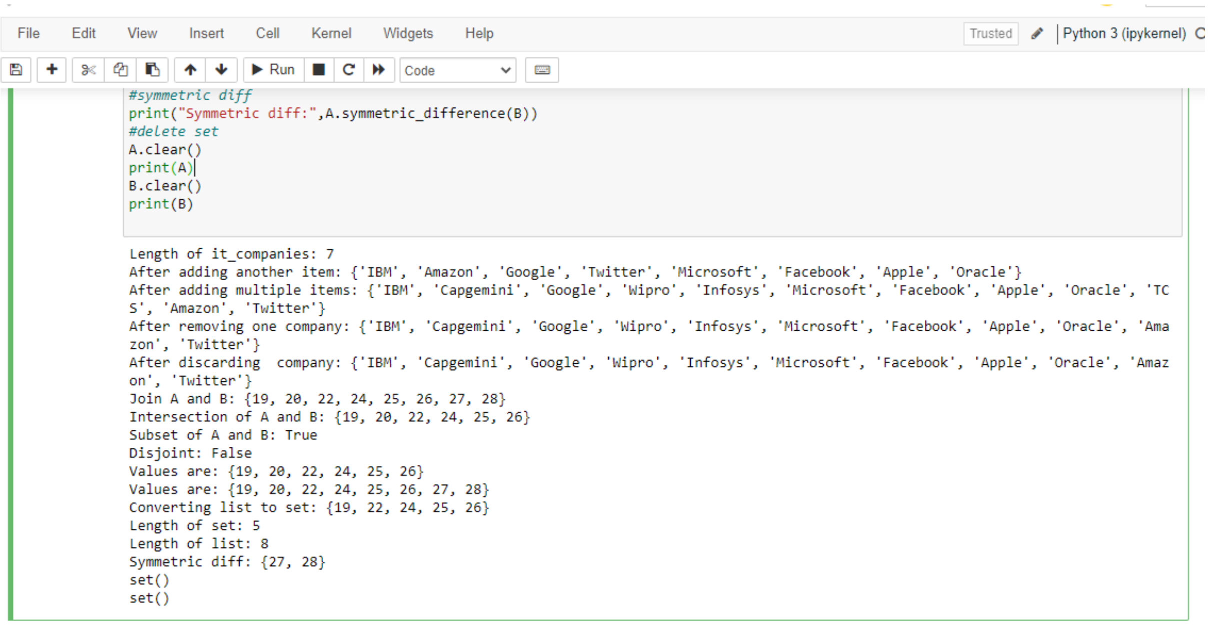
#delete set

A.clear()

print(A)

B.clear()

print(B)



```
#symmetric diff
print("Symmetric diff:",A.symmetric_difference(B))
#delete set
A.clear()
print(A)
B.clear()
print(B)
```

```
Length of it_companies: 7
After adding another item: {'IBM', 'Amazon', 'Google', 'Twitter', 'Microsoft', 'Facebook', 'Apple', 'Oracle'}
After adding multiple items: {'IBM', 'Capgemini', 'Google', 'Wipro', 'Infosys', 'Microsoft', 'Facebook', 'Apple', 'Oracle', 'TC
S', 'Amazon', 'Twitter'}
After removing one company: {'IBM', 'Capgemini', 'Google', 'Wipro', 'Infosys', 'Microsoft', 'Facebook', 'Apple', 'Oracle', 'Ama
zon', 'Twitter'}
After discarding  company: {'IBM', 'Capgemini', 'Google', 'Wipro', 'Infosys', 'Microsoft', 'Facebook', 'Apple', 'Oracle', 'Amaz
on', 'Twitter'}
Join A and B: {19, 20, 22, 24, 25, 26, 27, 28}
Intersection of A and B: {19, 20, 22, 24, 25, 26}
Subset of A and B: True
Disjoint: False
Values are: {19, 20, 22, 24, 25, 26}
Values are: {19, 20, 22, 24, 25, 26, 27, 28}
Converting list to set: {19, 22, 24, 25, 26}
Length of set: 5
Length of list: 8
Symmetric diff: {27, 28}
set()
set()
```

Q5) Calculate area of circle and circumference of circle

# Initialise r where r value can be read from user inpt

r = int(input("enter r:"))

# Calculate area of circle and circumference of circle

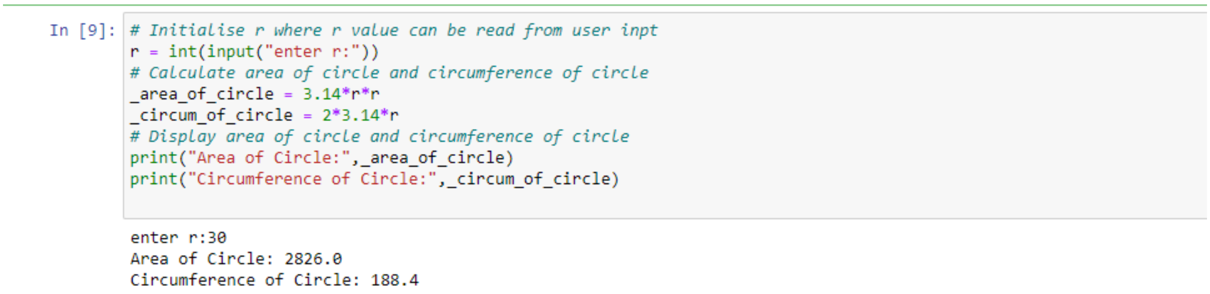
_area_of_circle = 3.14*r*r

_circum_of_circle = 2*3.14*r

# Display area of circle and circumference of circle

print("Area of Circle:",_area_of_circle)

print("Circumference of Circle:",_circum_of_circle)



```
In [9]: # Initialise r where r value can be read from user inpt
        r = int(input("enter r:"))
        # Calculate area of circle and circumference of circle
        _area_of_circle = 3.14*r*r
        _circum_of_circle = 2*3.14*r
        # Display area of circle and circumference of circle
        print("Area of Circle:",_area_of_circle)
        print("Circumference of Circle:",_circum_of_circle)

        enter r:30
        Area of Circle: 2826.0
        Circumference of Circle: 188.4
```

Q6) Unique words using split method

Unique

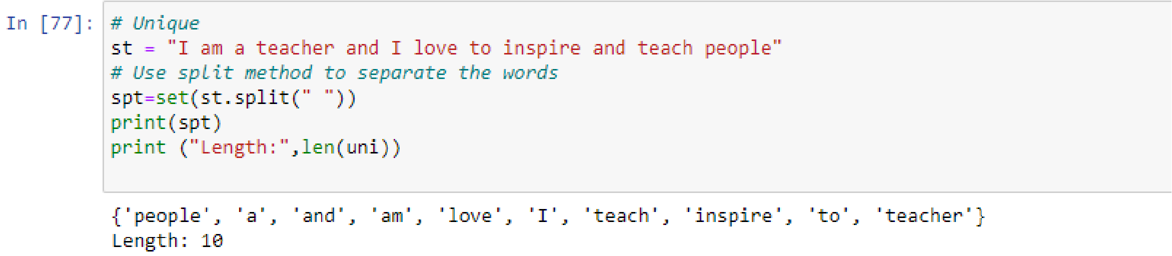
st = "I am a teacher and I love to inspire and teach people"

Use split method to separate the words and set to get the unique values

spt=set(st.split(" "))

print(spt)

print ("Length:",len(uni))

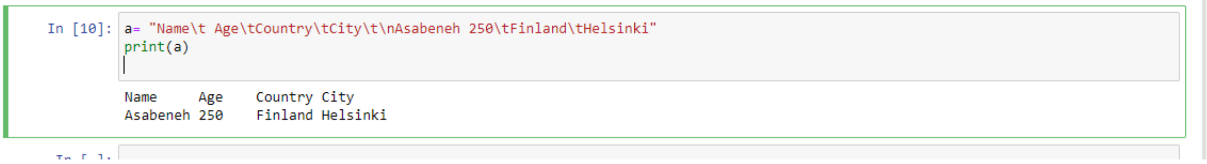


```
In [77]: # Unique
         st = "I am a teacher and I love to inspire and teach people"
         # Use split method to separate the words
         spt=set(st.split(" "))
         print(spt)
         print ("Length:",len(uni))

         {'people', 'a', 'and', 'am', 'love', 'I', 'teach', 'inspire', 'to', 'teacher'}
         Length: 10
```

Q7) Used tab and escape to display them in the given format

a= "Name\t Age\tCountry\tCity\t\nAsabeneh 250\tFinland\tHelsinki"

print(a)



```
In [10]: a= "Name\t Age\tCountry\tCity\t\nAsabeneh 250\tFinland\tHelsinki"
         print(a)

         Name     Age     Country City
         Asabeneh 250     Finland Helsinki
```

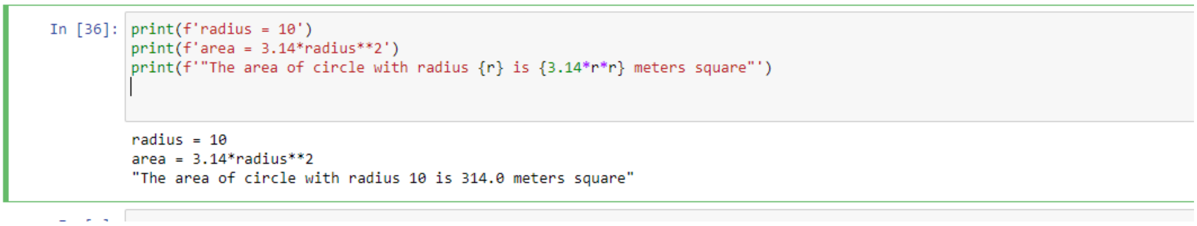
Q8) Use the string formatting method to display the following:

#Using String format method

print(f'radius = 10')

print(f'area = 3.14*radius2')**

print(f"The area of circle with radius {r} is {3.14*r*r} meters square")



```
In [36]: print(f'radius = 10')
         print(f'area = 3.14*radius**2')
         print(f'"The area of circle with radius {r} is {3.14*r*r} meters square"')

radius = 10
area = 3.14*radius**2
"The area of circle with radius 10 is 314.0 meters square"
```

Q9) Write a program, which reads weights (lbs.) of N students into a list and convert these weights to kilograms in a separate list using Loop. N: No of students (Read input from user)

#Creating a list(L1) for weights(lbs) of N students

L1=[int(num) for num in input().split(" ")]

#Creating another list called W_kg

W_kg=[]

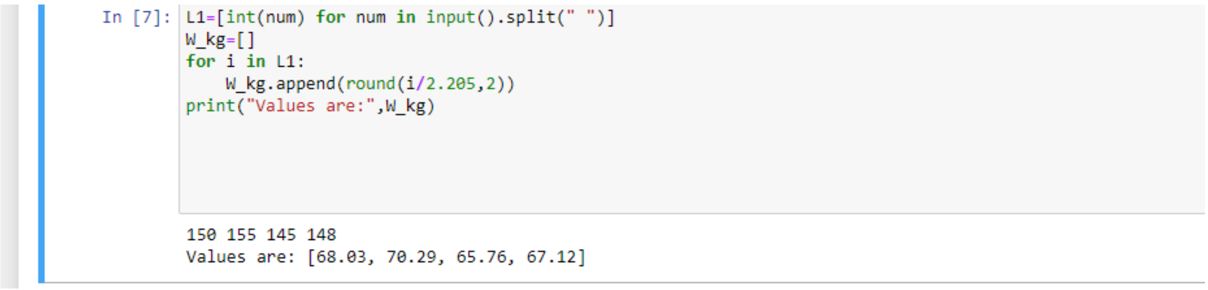
#Using for loop to iterate the values and appending the list

for i in L1:

W_kg.append(round(i/2.205,2))

#Displaying the values in kgs after conversion

print ("Values are:",W_kg)



```
In [7]: L1=[int(num) for num in input().split(" ")]
        W_kg=[]
        for i in L1:
            W_kg.append(round(i/2.205,2))
        print("Values are:",W_kg)

150 155 145 148
Values are: [68.03, 70.29, 65.76, 67.12]
```

Test Set

10)

f	1	2	3	6	6	7	10	11
Label	1	1	0	0	0	1	1	1

Train set

1) Using KNN classifier where k=3

$$d = \sqrt{(x-x_1)^2}$$

$(\overset{x}{6}, \overset{x_1}{6})$ $(\overset{x}{6}, \overset{x_1}{3})$ $(\overset{x}{6}, \overset{x_1}{2})$ $(\overset{x}{6}, \overset{x_1}{1})$ are the points need to be calculated

i.e

$d = \sqrt{(6-6)2} = 0$ (6,6)

$d = \sqrt{(6-3)^2} = \sqrt{9} = 3$ (6,3)

$d = \sqrt{(6-2)^2} = \sqrt{4^2} = 4$ (6,2)

} nearest

$d = \sqrt{(6-1)^2} = \sqrt{5^2} = 5$ (6,1)

i.e (0,0,1)

Max = 0 (o/p is also 0)

calculate for rest points which are also 0 (predicted)

2) Confusion matrix

Accuracy = (TP+TN)/(TN+FP+FN+TP)

Sensitivity = TP/(TP+FN)

Specificity = TN/(FP+TN)

	0	1
0	TN=1	FP=0
1	FN=3	TP=0

A = (0+1)/(1+0+3+0)

= 1/4 ⇒ 25%

S = 0/(0+3) = 0

Sp = 1/(0+1) = 1