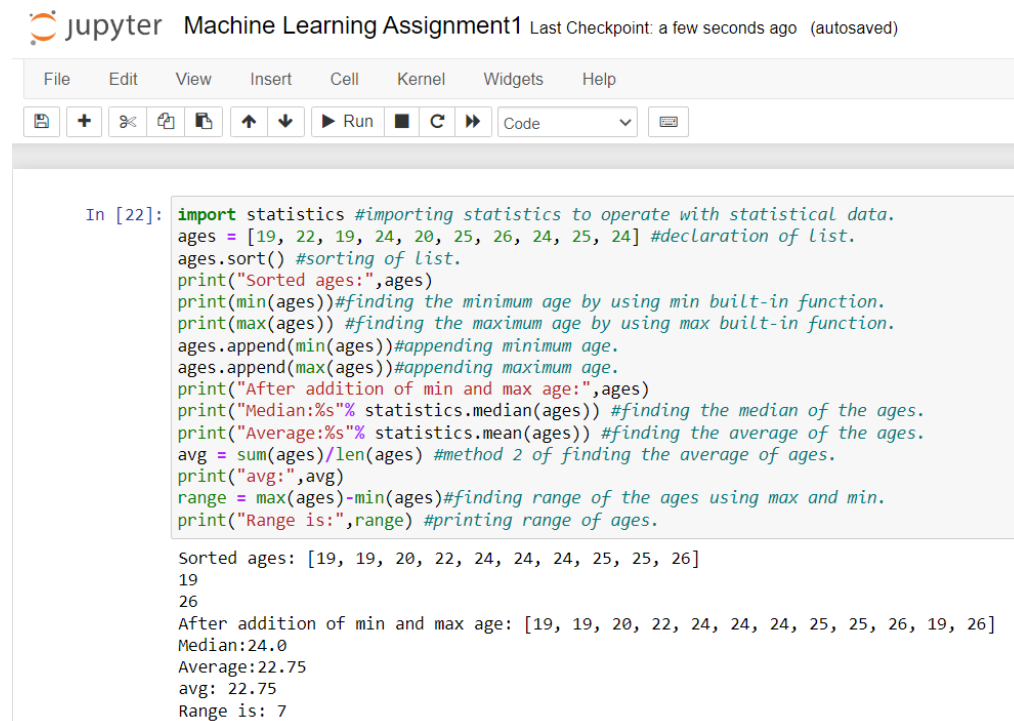


# Machine Learning (Assignment #1)

1. Program to find the min , max , median , average, range of student ages and performing operation like sorting and addition of min and max.
  - The following is a list of 10 students ages: ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
  - Sort the list and find the min and max age
  - Add the min age and the max age again to the list
  - Find the median age (one middle item or two middle items divided by two)
  - Find the average age (sum of all items divided by their number)
  - Find the range of the ages (max minus min)

## Screenshot:



The screenshot shows a Jupyter Notebook titled "Machine Learning Assignment1". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, running, and other functions. The code cell contains the following Python code:

```
In [22]: import statistics #importing statistics to operate with statistical data.
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24] #declaration of list.
ages.sort() #sorting of list.
print("Sorted ages:",ages)
print(min(ages))#finding the minimum age by using min built-in function.
print(max(ages)) #finding the maximum age by using max built-in function.
ages.append(min(ages))#appending minimum age.
ages.append(max(ages))#appending maximum age.
print("After addition of min and max age:",ages)
print("Median:%s"% statistics.median(ages)) #finding the median of the ages.
print("Average:%s"% statistics.mean(ages)) #finding the average of the ages.
avg = sum(ages)/len(ages) #method 2 of finding the average of ages.
print("avg:",avg)
range = max(ages)-min(ages)#finding range of the ages using max and min.
print("Range is:",range) #printing range of ages.
```

The output of the code is displayed below the code cell:

```
Sorted ages: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
19
26
After addition of min and max age: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
Median:24.0
Average:22.75
avg: 22.75
Range is: 7
```

The above screenshot shows the execution of the list, which is ages, imported the statistics module as it has methods like median, mean, mode, which can have used to find the median and average values. Then initialized the ages and then performed different operations like sorting of list, finding maximum and minimum of the ages, and then added those values to the list and lastly calculated median, average, and range using the in-built functions.

#### Source Code:

1. import statistics #importing statistics to operate with statistical data.
2. ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24] #declaration of list.
3. ages.sort() #sorting of list.
4. print("Sorted ages:",ages)
5. print(min(ages))#finding the minimum age by using min built-in function.
6. print(max(ages)) #finding the maximum age by using max built-in function.
7. ages.append(min(ages))#appending minimum age.
8. ages.append(max(ages))#appending maximum age.
9. print("After addition of min and max age:",ages)
10. print("Median:%s"% statistics.median(ages)) #finding the median of the ages.
11. print("Average:%s"% statistics.mean(ages)) #finding the average of the ages.
12. avg = sum(ages)/len(ages) #method 2 of finding the average of ages.
13. print("avg:",avg)
14. range = max(ages)-min(ages)#finding range of the ages using max and min.
15. print("Range is:",range) #printing range of ages.

#### Output:

Sorted ages: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]

19

26

After addition of min and max age:

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

Median:24.0

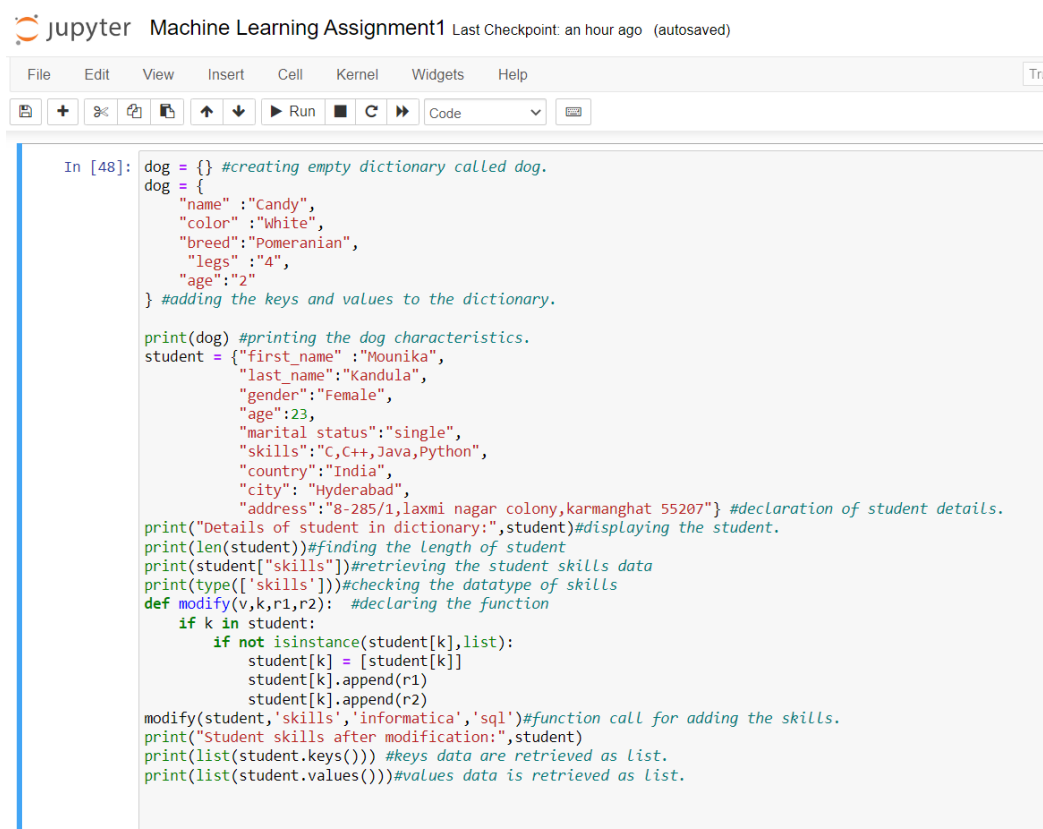
Average:22.75

avg: 22.75

Range is: 7.

2. Program to create the dictionary named dog and then add characters, then create student dictionary adding the keys and values.
  - Create an empty dictionary called dog.
  - Add name, color, breed, legs, age to the dog dictionary.
  - Create a student dictionary and add first\_name, last\_name, gender, age, marital status, skills, country, city and address as keys for the dictionary.
  - Get the length of the student dictionary.
  - Get the value of skills and check the data type, it should be a list.
  - Modify the skills values by adding one or two skills.
  - Get the dictionary keys as a list.
  - Get the dictionary values as a list.

## Screenshot:



The screenshot shows a Jupyter Notebook window titled "Machine Learning Assignment1" with a "Last Checkpoint: an hour ago (autosaved)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The code cell contains the following Python code:

```
In [48]: dog = {} #creating empty dictionary called dog.
dog = {
    "name" : "Candy",
    "color" : "White",
    "breed" : "Pomeranian",
    "legs" : "4",
    "age" : "2"
} #adding the keys and values to the dictionary.

print(dog) #printing the dog characteristics.
student = {"first_name" : "Mounika",
           "last_name" : "Kandula",
           "gender" : "Female",
           "age" : 23,
           "marital status" : "single",
           "skills" : "C,C++,Java,Python",
           "country" : "India",
           "city" : "Hyderabad",
           "address" : "8-285/1,laxmi nagar colony,karmanghat 55207"} #declaration of student details.
print("Details of student in dictionary:", student) #displaying the student.
print(len(student)) #finding the length of student
print(student["skills"]) #retrieving the student skills data
print(type(['skills'])) #checking the datatype of skills
def modify(v,k,r1,r2): #declaring the function
    if k in student:
        if not isinstance(student[k], list):
            student[k] = [student[k]]
            student[k].append(r1)
            student[k].append(r2)
modify(student, 'skills', 'informatica', 'sql') #function call for adding the skills.
print("Student skills after modification:", student)
print(list(student.keys())) #keys data are retrieved as List.
print(list(student.values())) #values data is retrieved as List.
```

```
{'name': 'Candy', 'color': 'White', 'breed': 'Pomeranian', 'legs': '4', 'age': '2'}
Details of student in dictionary: {'first_name': 'Mounika', 'last_name': 'Kandula', 'gender': 'Female', 'age': 23, 'marital status': 'single', 'skills': 'C,C++,Java,Python', 'country': 'India', 'city': 'Hyderabad', 'address': '8-285/1,laxmi nagar colony, karmanghat 55207'}
9
C,C++,Java,Python
<class 'list'>
Student skills after modification: {'first_name': 'Mounika', 'last_name': 'Kandula', 'gender': 'Female', 'age': 23, 'marital status': 'single', 'skills': ['C,C++,Java,Python', 'informatica', 'sql'], 'country': 'India', 'city': 'Hyderabad', 'address': '8-285/1,laxmi nagar colony,karmanghat 55207'}
['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
['Mounika', 'Kandula', 'Female', 23, 'single', ['C,C++,Java,Python', 'informatica', 'sql'], 'India', 'Hyderabad', '8-285/1,laxmi nagar colony,karmanghat 55207']
```

We have created an empty dictionary called as dog, and then added the key and values to it, namely name ,color ,breed ,legs ,age to it. Then we have created an other dictionary called student added the keys and values to it .Calculated the length of the student dictionary using the len function. Retrieved the values of skills and checked its datatype which we got as a list, modified the skills by adding few skills like informatica ,sql to it. Extracting the keys and values from the dictionary as a list.

#### Source Code:

1. dog = {}      #creating empty dictionary called dog.
2. dog = {
3. "name" : "Candy",
4. "color" : "White",
5. "breed" : "Pomeranian",
6. "legs" : "4",
7. "age" : "2"
8. }      #adding the keys and values to the dictionary.
  
9. print(dog)    #printing the dog characteristics.
10. student = {"first\_name" : "Mounika",
11. "last\_name" : "Kandula",
12. "gender" : "Female",
13. "age" : 23,
14. "marital status" : "single",
15. "skills" : "C,C++,Java,Python",
16. "country" : "India",
17. "city" : "Hyderabad",
18. "address" : "8-285/1,laxmi nagar colony,karmanghat 55207"}    #declaration of student details.
19. print("Details of student in dictionary:", student)    #displaying the student.
20. print(len(student))    #finding the length of student
21. print(student["skills"])    #retrieving the student skills data

```

22. print(type(['skills'])) #checking the datatype of skills
23. def modify(v,k,r1,r2): #declaring the function
24. if k in student:
25. if not isinstance(student[k],list):
26. student[k] = [student[k]]
27. student[k].append(r1)
28. student[k].append(r2)
29. modify(student,'skills','informatica','sql') #function call for adding the skills.
30. print("Student skills after modification=",student)
31. print(list(student.keys())) #keys data are retrieved as list.
32. print(list(student.values())) #values data is retrieved as list.

```

### Output:

```

{'name': 'Candy', 'color': 'White', 'breed': 'Pomeranian', 'legs': '4', 'age': '2'}
Details of student in dictionary: {'first_name': 'Mounika', 'last_name': 'Kandula', 'gender': 'Female', 'age': 23, 'marital status': 'single', 'skills': 'C,C++,Java,Python', 'country': 'India', 'city': 'Hyderabad', 'address': '8-285/1,laxmi nagar colony,karmanghat 55207'}
9
C,C++,Java,Python
<class 'list'>
Student skills after modification={'first_name': 'Mounika', 'last_name': 'Kandula', 'gender': 'Female', 'age': 23, 'marital status': 'single', 'skills': ['C,C++,Java,Python', 'informatica', 'sql'], 'country': 'India', 'city': 'Hyderabad', 'address': '8-285/1,laxmi nagar colony,karmanghat 55207'}
['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
['Mounika', 'Kandula', 'Female', 23, 'single', ['C,C++,Java,Python', 'informatica', 'sql'], 'India', 'Hyderabad', '8-285/1,laxmi nagar colony,karmanghat 55207']

```

3. Program to create a tuple and finding its length the assign it to other tuple.

- Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)
- Join brothers and sisters tuples and assign it to siblings
- How many siblings do you have?
- Modify the siblings tuple and add the name of your father and mother and assign it to family\_members

## Screenshot:

```
In [56]: sisters = ("Mounika","Manasa","Manisha","Praneetha","Yashu")#creating the tuple called sisters
brothers = ("sai anvesh","rahul","vinay","vijay")#creating the tuple called sisters
print(sisters)
print(brothers)
siblings = sisters+brothers #join of brother and sister.
print(siblings)
print("Count of my siblings:",len(siblings))#count of siblings.
siblings = ("Venkateshwarlu","Renuka")+siblings #adding the name of father and mother.
family_members = siblings #assigning siblings to family_members.
print("Family Members:",family_members)

('Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu')
('sai anvesh', 'rahul', 'vinay', 'vijay')
('Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu', 'sai anvesh', 'rahul', 'vinay', 'vijay')
Count of my siblings: 9
Family Members: ('Venkateshwarlu', 'Renuka', 'Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu', 'sai anvesh', 'rahul', 'vinay', 'vijay')
```

In the above code we have created two tuples namely sisters and brothers and then using the addition of the brothers and sisters tuples to a siblings have been performed then using the count method we have retrieved the total number of siblings. Lastly the addition of the father and mother details to the siblings have been performed and then assigned it to a family\_members. When here if we print the family\_members it displays all the family details.

## Source Code:

1. sisters = ("Mounika","Manasa","Manisha","Praneetha","Yashu")#creating the tuple called sisters
2. brothers = ("sai anvesh","rahul","vinay","vijay")#creating the tuple called sisters
3. print(sisters)
4. print(brothers)
5. siblings = sisters+brothers #join of brother and sister.
6. print(siblings)
7. print("Count of my siblings:",len(siblings))#count of siblings.
8. siblings = ("Venkateshwarlu","Renuka")+siblings #adding the name of father and mother.
9. family\_members = siblings #assigning siblings to family\_members.
10. print("Family Members:",family\_members)

## Output:

```
('Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu')
('sai anvesh', 'rahul', 'vinay', 'vijay')
```

('Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu', 'sai anvesh', 'rahul', 'vinay', 'vijay')

Count of my siblings: 9

Family Members: ('Venkateshwarlu', 'Renuka', 'Mounika', 'Manasa', 'Manisha', 'Praneetha', 'Yashu', 'sai anvesh', 'rahul', 'vinay', 'vijay')

4.

- `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]`
- Find the length of the set `it_companies`
- Add 'Twitter' to `it_companies`
- Insert multiple IT companies at once to the set `it_companies`
- Remove one of the companies from the set `it_companies`
- What is the difference between remove and discard
- Join A and B
- Find A intersection B
- Is A subset of B
- Are A and B disjoint sets
- Join A with B and B with A
- What is the symmetric difference between A and B
- Delete the sets completely
- Convert the ages to a set and compare the length of the list and the set.

## Screenshot:

```
In [69]: 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]
print(len(it_companies))#Finding the Length of the it_companies.
it_companies.add('Twitter')#addition of value.
print(it_companies)
it_companies.update(["TCSL","WIPRO","INFOSYS","Delloite","Accenture"])#adding the multiple values.
print(it_companies)
it_companies.remove("TCSL")#removing the value from set.
#the difference between remove and discard is that if the item to remove does not exist,discard will not raise an error,
#whereas the remove will.
it_companies.discard("capgemini")#though the item does not exist in the set the discard will not through any error whereas remove
print(it_companies)
print(A.union(B))#join of A and B
print(A.intersection(B))#intersection of A and B
print(A.issubset(B)) #finding whether A is subset of B or not
print(A.isdisjoint(B))#checking the disjoint sets
print(A.union(B))#A join B
print(B.union(A))#B join A
print(A.symmetric_difference(B)) #checking the symmetric difference between A and B.
del A #deleting the sets completely.
del B #deleting the set completely.
print(A)#this will throw an error because the set no longer exists.
print(B)
ages=set(age)#converting List of age to set ages.
if(len(ages)>len(age)): #comparing the Length of set and List.
    print("set size is greater than list size")
elif(len(ages)<len(age)):
    print("list size is greater than set size")
else:
    print("Both are equal")
```

#After performing the deletion of set when we try to print it throws the NameError because it no longer exist.

The below output depicts it. When I tried to print set A which I have deleted, it shows the error.

```
7
{'Microsoft', 'IBM', 'Amazon', 'Apple', 'Oracle', 'Google', 'Twitter', 'Facebook'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'TCSL', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'Delloite'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'Delloite'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26, 27, 28}
{27, 28}

-----
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14812\1438172324.py in <module>
     22 del A #deleting the sets completely.
     23 del B #deleting the set completely.
--> 24 print(A)#this will throw an error because the set no longer exists.
     25 print(B)
     26 ages=set(age)#converting list of age to set ages.

NameError: name 'A' is not defined
```



```

print(A.symmetric_difference(B)) #checking the symmetric difference between A and B.
#del A #deleting the sets completely.
#del B #deleting the set completely.
#print(A)#this will throw an error because the set no longer exists.
#print(B)
ages=set(age)#converting list of age to set ages.
if(len(ages)>len(age)): #comparing the length of set and list.
    print("set size is greater than list size")
elif(len(ages)<len(age)):
    print("list size is greater than set size")
else:
    print("Both are equal")

```

```

7
{'Microsoft', 'IBM', 'Amazon', 'Apple', 'Oracle', 'Google', 'Twitter', 'Facebook'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'TCSL', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'De
lloite'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'Delloite'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26, 27, 28}
{27, 28}
Both are equal

```

To find out the either the length of the list is greater or the set of ages we have firstly converted the age list to set then using if operator founded out which one is greater,the comparison operation has been operated.

In the beginning of the code using len function the it\_companies length has been retrieved,to the set the addition of 'Twitter' operation has been taken place.The multiple values are added to set using the update ,then the random item has been removed that is TCSL.On the set A and B the union ,intersection, issubset , isdisjoint, symmetric\_difference,all these checks have been retrieved using the respective methods, and then deletion of set A and B completely.Which gives an error when we try to print.

### Source Code:

1. it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}
2. A = {19, 22, 24, 20, 25, 26}
3. B = {19, 22, 20, 25, 26, 24, 28, 27}
4. age = [22, 19, 24, 25, 26, 24, 25, 24]
5. print(len(it\_companies))#Finding the length of the it\_companies.
6. it\_companies.add('Twitter')#addition of value.
7. print(it\_companies)
8. it\_companies.update(["TCSL","WIPRO","INFOSYS","Delloite","Accenture"])#adding the multiple values.
9. print(it\_companies)

```

10. it_companies.remove("TCSL")#removing the value from set.
11. #the difference between remove and discard is that if the item to remove does not
    exist,discard will not raise an error,
12. #whereas the remove will.
13. it_companies.discard("capgemini")#though the item does not exist in the set the discard
    will not through any error whereas remove does.
14. print(it_companies)
15. print(A.union(B))#join of A and B
16. print(A.intersection(B))#intersection of A and B
17. print(A.issubset(B)) #finding whether A is subset of B or not
18. print(A.isdisjoint(B))#checking the disjoint sets
19. print(A.union(B))#A join B
20. print(B.union(A))#B join A
21. print(A.symmetric_difference(B)) #checking the symmetric difference between A and B.
22. del A #deleting the sets completely.
23. del B #deleting the set completely.
24. print(A)#this will throw an error because the set no longer exists.
25. print(B)
26. ages=set(age)#converting list of age to set ages.
27. if(len(ages)>len(age)): #comparing the length of set and list.
28. print("set size is greater than list size")
29. elif(len(age)>len(ages)):
30. print("list size is greater than set size")
31. else:
32. print("Both are equal")

```

### Output:

7

```

{'Microsoft', 'IBM', 'Amazon', 'Apple', 'Oracle', 'Google', 'Twitter', 'Facebook'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'TCSL', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'Delloite'}
{'Microsoft', 'INFOSYS', 'WIPRO', 'Accenture', 'Apple', 'Oracle', 'Amazon', 'Google', 'Twitter', 'Facebook', 'IBM', 'Delloite'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26, 27, 28}
{27, 28}

```

---

```

NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14812\1438172324.py in <module>
    22 del A #deleting the sets completely.
    23 del B #deleting the set completely.
--> 24 print(A)#this will throw an error because the set no longer exists.
    25 print(B)
    26 ages=set(age)#converting list of age to set ages.

```

**NameError:** name 'A' is not defined  
list size is greater than set size

5

- The radius of a circle is 30 meters.
- Calculate the area of a circle and assign the value to a variable name of `_area_of_circle_`
- Calculate the circumference of a circle and assign the value to a variable name of `_circum_of_circle_`
- Take radius as user input and calculate the area .

**Screenshot:**

```

In [71]: import math
r = int(input("Enter Radius Of Circle : ")) #taking the radius as the user input.
area_of_circle=math.pi*r*r #calculating the area of circle.
print("Area Of Circle Is : ",round(area_of_circle,4))
circum_of_circle=2*math.pi*r # calculating the circumference of the circle.
print("Circumference Of Circle Is : ",round(circum_of_circle,3))

Enter Radius Of Circle : 30
Area Of Circle Is : 2827.4334
Circumference Of Circle Is : 188.496

```

For calculating the area of the circle and the circumference of circle, have imported a module named `math` as it has many methods and constants values for instance in this program we have used `pi`. and then took the radius values as a user input. After calculating the area and circumference we have assigned those values to the `_area_of_circle_`, and the `_circumference_of_circle_`.

### Source Code:

1. import math
2. r = int(input("Enter Radius Of Circle : ")) #taking the radius as the user input.
3. area\_of\_circle=math.pi\*r\*r #calculating the area of circle.
4. print("Area Of Circle Is : ",round(area\_of\_circle,4))
5. circum\_of\_circle=2\*math.pi\*r # calculating the circumference of the circle.
6. print("Circumference Of Circle Is : ",round(circum\_of\_circle,3))

### Output:

Enter Radius Of Circle : 30

Area Of Circle Is : 2827.4334

Circumference Of Circle Is : 188.496

6

“I am a teacher and I love to inspire and teach people”

- How many unique words have been used in the sentence? Use the split methods and set to get the unique words.

### Screenshot:

```
In [81]: sentence = "I am a teacher and I love to inspire and teach people"
words = sentence.split()#split function to divide words.
print(words)
sentence_set=set(words)# assign to set
print(sentence_set)
y=len(sentence_set)#to find out unique words.
print("No of unique words:",y)

['I', 'am', 'a', 'teacher', 'and', 'I', 'love', 'to', 'inspire', 'and', 'teach', 'people']
{'and', 'am', 'a', 'I', 'teach', 'love', 'people', 'inspire', 'to', 'teacher'}
```

```
In [83]: print("Name\tAge\tCountry City") #for header
```

The main aim of the program is to find the unique words in the sentence by using the split methods and set. In the above screenshot split is used to divide the words and then the set is used to find out the unique words then the count of unique words is performed.

### Source Code:

1. sentence = "I am a teacher and I love to inspire and teach people"
2. words = sentence.split()#split function to divide words.
3. print(words)
4. sentence\_set=set(words)# assign to set
5. print(sentence\_set)
6. y=len(sentence\_set)#to find out unique words.
7. print("No of unique words:",y)

### Output:

```
['I', 'am', 'a', 'teacher', 'and', 'I', 'love', 'to', 'inspire', 'and', 't  
each', 'people']  
{'and', 'am', 'a', 'I', 'teach', 'love', 'people', 'inspire', 'to', 'teach  
er'}  
No of unique words: 10
```

7.

Use a tab escape sequence to get the following lines. Name Age Country City Asabeneh 250  
Finland Helsinki

### Screenshot:

```
In [83]: print("Name\tAge\tCountry City") #for header  
         print("Asabeneh\t250\tFinland Helsinki") #for values
```

Name	Age	Country	City
Asabeneh	250	Finland	Helsinki

'\t' the tab key is used to display the given format.

### Source code:

1. print("Name\tAge\tCountry City") #for header
2. print("Asabeneh\t250\tFinland Helsinki") #for values

### Output :

Name	Age	Country	City
Asabeneh	250	Finland	Helsinki

8

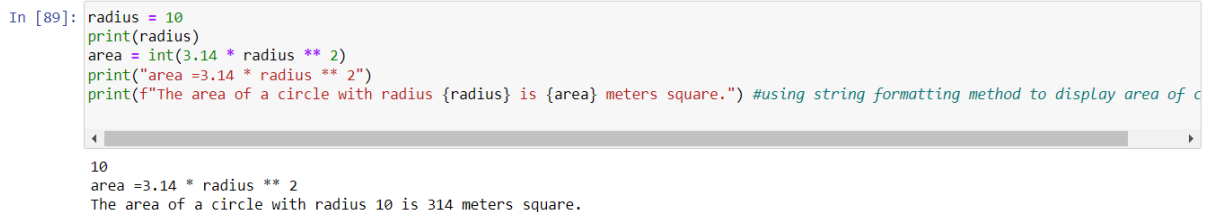
Use the string formatting method to display the following:

radius = 10

area = 3.14 \* radius \*\* 2

“The area of a circle with radius 10 is 314 meters square.”

#### Screenshot:



```
In [89]: radius = 10
print(radius)
area = int(3.14 * radius ** 2)
print("area =3.14 * radius ** 2")
print(f"The area of a circle with radius {radius} is {area} meters square.") #using string formatting method to display area of c
```

10  
area =3.14 \* radius \*\* 2  
The area of a circle with radius 10 is 314 meters square.

The aim of the program is to display the area of the circle by using the string formatting method.

#### Source Code:

1. radius = 10
2. print(radius)
3. area = int(3.14 \* radius \*\* 2)
4. print("area =3.14 \* radius \*\* 2")
5. print(f"The area of a circle with radius {radius} is {area} meters square.") #using string formatting method to display area of circle.

#### Output:

10

area =3.14 \* radius \*\* 2

The area of a circle with radius 10 is 314 meters square.

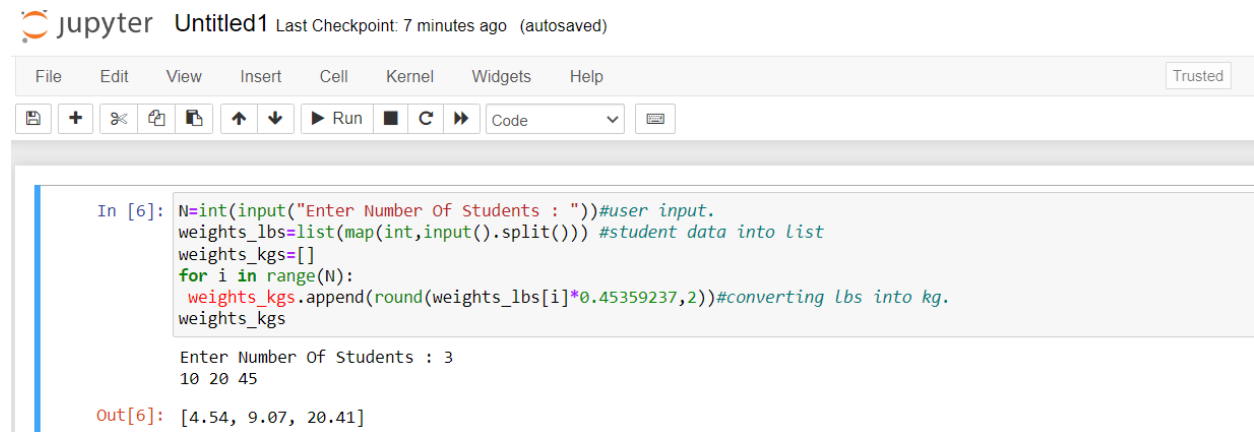
9

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)

Ex: L1: [150, 155, 145, 148]

Output: [68.03, 70.3, 65.77, 67.13]

### Screenshot:



The screenshot shows a Jupyter Notebook window titled 'Untitled1' with a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code cell contains the following Python code:

```
In [6]: N=int(input("Enter Number Of Students : "))#user input.  
weights_lbs=list(map(int,input().split())) #student data into list  
weights_kgs=[]  
for i in range(N):  
    weights_kgs.append(round(weights_lbs[i]*0.45359237,2))#converting lbs into kg.  
weights_kgs
```

The output of the code is displayed below the code cell:

```
Enter Number Of Students : 3  
10 20 45  
Out[6]: [4.54, 9.07, 20.41]
```

In the above program we have taken the details of weight in lbs,into a list as a user input then we have converted into kilograms in an other list using the loop.

### Source Code:

1. `N=int(input("Enter Number Of Students : "))#user input.`
2. `weights_lbs=list(map(int,input().split())) #student data into list`
3. `weights_kgs=[]`
4. `for i in range(N):`
5. `weights_kgs.append(round(weights_lbs[i]*0.45359237,2))#converting lbs into kg.`
6. `weights_kgs`

### Output:

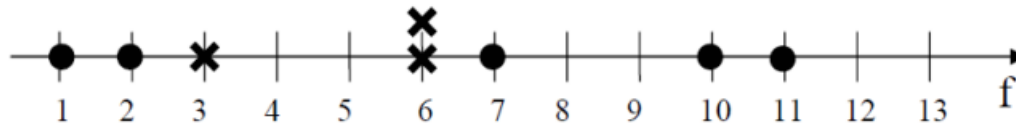
Enter Number Of Students : 3

10 20 45

[4.54, 9.07, 20.41]

10

The diagram below shows a dataset with 2 classes and 8 data points, each with only one feature value, labeled  $f$ . Note that there are two data points with the same feature value of 6. These are shown as two 'x's one above the other. Provide stepwise mathematical solution, do not write code for it.



1. Divide this data equally into two parts. Use first part as training and second part as testing. Using KNN classifier, for  $K=3$ , what would be the predicted outputs for the test samples? Show how you arrived at your answer.
2. Compute the confusion matrix for this and calculate accuracy, sensitivity and specificity values.

**Screenshot:**

```
In [12]: import numpy as num
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
A = num.array([[1, 0], [2, 0], [3, 0], [6, 0], [6, 0], [7, 0], [10, 0], [11, 0]])
print("The Dataset: \n", A, "\n Size: ", len(A))
B = num.array([0, 0, 1, 1, 1, 0, 0, 0])

print("\n Class label for the datasets A B:", B)
A_train, a_test, B_train, b_test = train_test_split(A, B, test_size=0.50, random_state=0, shuffle=False)
print("\n Training Data A: \n", A_train)
print("\n Training Data label B: \n", B_train)
stand_scalar = StandardScaler()#scaling is used to normalize the range of variables or feature data ,prior performing the data st
A_train = stand_scalar.fit_transform(A_train) #Train
a_test = stand_scalar.transform(a_test) #test
classifier = KNeighborsClassifier(n_neighbors=3) #The KNN value is set to 3
classifier.fit(A_train, B_train)# Fitting the data A and B to classifier
B_pred = classifier.predict(a_test)#The Label for the A-test data it predictis.
print('\n Predicted Classs labels for testing data B:', B_pred) # Confusion matrix with tested data
confusion_mt_re = confusion_matrix(b_test, B_pred) #matrix with tested data.
print("\n Confusion Matrix\n", confusion_mt_re)
tl_value = sum(sum(confusion_mt_re)) #the value of P+N can be calculatd using the logic sum(sum(sum(conf_matrix))
acc = (confusion_mt_re[0, 0] + confusion_mt_re[1, 1]) / tl_value #finding the accuracy.
print('\n Accuracy value: ', acc) ## Accuracy TN+TP / P+N where P is positive and N is negative.
sensitivity = confusion_mt_re[1, 1] / (confusion_mt_re[1, 0] + confusion_mt_re[1, 1]) #finding the sensitivity
print('Sensitivity value : ', sensitivity) # Sensitivity TP/(TP+FN)
sp = confusion_mt_re[0, 0] / (confusion_mt_re[0, 0] + confusion_mt_re[0, 1])# the formula to find the specificity
print('Specificity value: ', sp)# Specificity TN/(TN+FP)
```



```

print('Sensitivity value: ', sensitivity) # sensitivity TP/(TP+FN)
sp = confusion_mt_re[0, 0] / (confusion_mt_re[0, 0] + confusion_mt_re[0,1])# the formula to find the specificity
print('Specificity value:',sp)# Specificity TN/(TN+FP)

```

The Dataset:

```

[[ 1  0]
 [ 2  0]
 [ 3  0]
 [ 6  0]
 [ 6  0]
 [ 7  0]
 [10  0]
 [11  0]]
Size: 8

```

Class label for the datasets A B: [0 0 1 1 1 0 0 0]

Training Data A:

```

[[1 0]
 [2 0]
 [3 0]
 [6 0]]

```

Training Data label B: [0 0 1 1]

Predicted Classs labels for testing data B: [1 1 1 1]

Confusion Matrix

```

[[0 3]
 [0 1]]

```

Accuracy value: 0.25

Sensitivity value : 1.0

Specificity value: 0.0

Imported required libraries from library set. Declared A and B variables to take data in the arrays and then printing the datasets. Then created labels for the datasets A and B by taking test size and state values and then used scalar function to normalize the data and finally using KNN and setting the value of the k Nearest to 3,calculated the accuracy, sensitivity and also the specificity values using the confusion matrix.

## Source Code:

1. import numpy as num
2. from sklearn.model\_selection import train\_test\_split
3. from sklearn.neighbors import KNeighborsClassifier
4. from sklearn.preprocessing import StandardScaler
5. from sklearn.metrics import confusion\_matrix
6. A = num.array([[1, 0], [2, 0], [3, 0], [6, 0], [6, 0], [7, 0], [10, 0], [11, 0]])
7. print("The Dataset: \n", A, "\n Size: ", len(A))
8. B = num.array([0, 0, 1, 1, 1, 0, 0, 0])
9. print("\n Class label for the datasets A B:", B)
10. A\_train, a\_test, B\_train, b\_test = train\_test\_split(A, B, test\_size=0.50, random\_state=0, shuffle=False)
11. print("\n Training Data A: \n", A\_train)

```

12. print("\n Training Data label B:", B_train)
13. stand_scaler = StandardScaler()#scaling is used to normalize the range of variables or
    feature data ,prior performing the data standardization is seen.
14. A_train = stand_scaler.fit_transform(A_train) #Train
15. a_test = stand_scaler.transform(a_test) #test
16. classifier = KNeighborsClassifier(n_neighbors=3) #The KNN value is set to 3
17. classifier.fit(A_train, B_train)# Fitting the data A and B to classifier
18. B_pred = classifier.predict(a_test)#The label for the A-test data it predicts.
19. print('\nPredicted Classs labels for testing data B:', B_pred) # Confusion matrix with
    tested data
20. confusion_mt_re= confusion_matrix(b_test, B_pred) #matrix with tested data.
21. print("\nConfusion Matrix\n", confusion_mt_re)
22. tl_value = sum(sum(confusion_mt_re)) #the value of P+N can be calculatd using the
    logic sum(sum(sum(conf_matrix))
23. acc = (confusion_mt_re[0, 0] + confusion_mt_re[1, 1]) / tl_value #finding the accuracy.
24. print('\nAccuracy value: ', acc) # # Accuracy TN+TP / P+N where P is positive and N is
    negative.
25. sensitivity = confusion_mt_re[1, 1] / (confusion_mt_re[1, 0] + confusion_mt_re[1,
26. 1]) #finding the sensitivity
27. print('Sensitivity value : ', sensitivity) # Sensitivity TP/(TP+FN)
28. sp = confusion_mt_re[0, 0] / (confusion_mt_re[0, 0] + confusion_mt_re[0,1])# the
    formula to find the specificity
29. print('Specificity value:',sp)# Specificity TN/(TN+FP)

```

## Output:

The Dataset:

```

[[ 1 0]
 [ 2 0]
 [ 3 0]
 [ 6 0]
 [ 6 0]
 [ 7 0]
 [10 0]
 [11 0]]

```

Size: 8

Class label for the datasets A B: [0 0 1 1 1 0 0 0]

Training Data A:

[[1 0]

[2 0]

[3 0]

[6 0]]

Training Data label B: [0 0 1 1]

Predicted Classs labels for testing data B: [1 1 1 1]

Confusion Matrix

[[0 3]

[0 1]]

Accuracy value: 0.25

Sensitivity value : 1.0

Specificity value: 0.0

**Video : <https://drive.google.com/file/d/1YWMxfSEGIEsPycjkzOc0hie4PV1vf8r4/view?usp=sharing>**