Assignment - 1

Machine Learning (CS 5710) CRN: 22002

Vinay Kumar Camarushi (700740428)

Link to the video recording :
https://drive.google.com/file/d/1QuBmS75VNiP5o1_pJxoug6G7xg5mUpOk/view?usp=sharing

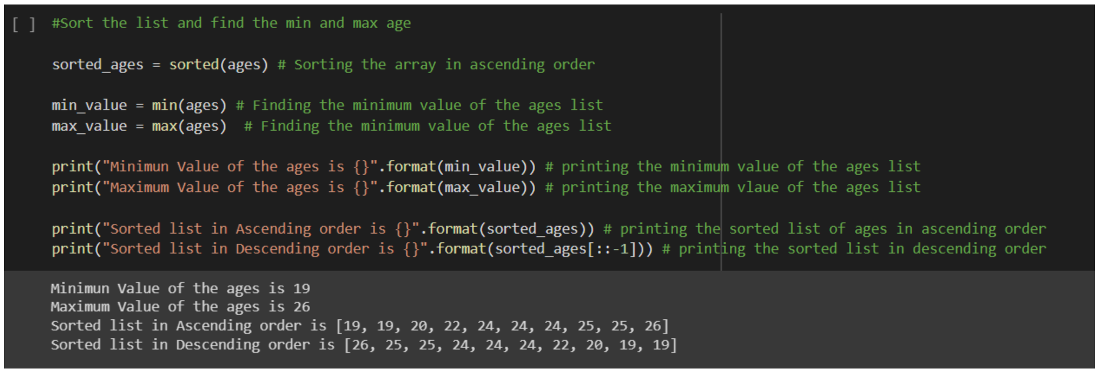
GitHub Link: https://github.com/VXC04280/ML_Assignment_1

Question – 1

The following is a list of 10 students ages:

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

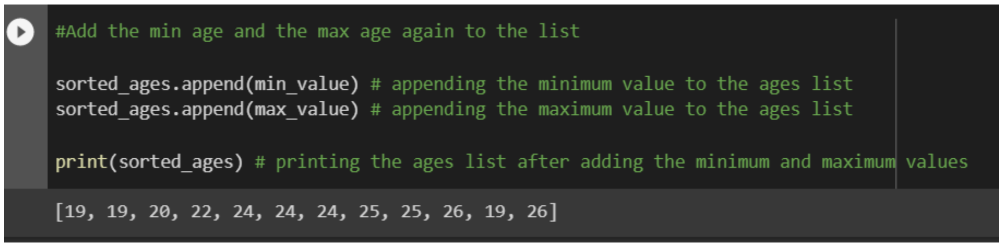
- **Sort the list and find the min and max age**
  - ⇨ Here we have sorted the list using sorted() method and stored the sorted list in sorted_ages list.
  - ⇨ We used min() method to get the minimum value of the list and max() method to get the maximum value of he list.
  - ⇨ The we printed the list in the minimum, maximum values of the list along with the sorted list in Ascending order and Descending order.



```
[ ] #Sort the list and find the min and max age

    sorted_ages = sorted(ages) # Sorting the array in ascending order

    min_value = min(ages) # Finding the minimum value of the ages list
    max_value = max(ages)  # Finding the minimum value of the ages list

    print("Minimun Value of the ages is {}".format(min_value)) # printing the minimum value of the ages list
    print("Maximum Value of the ages is {}".format(max_value)) # printing the maximum vlaue of the ages list

    print("Sorted list in Ascending order is {}".format(sorted_ages)) # printing the sorted list of ages in ascending order
    print("Sorted list in Descending order is {}".format(sorted_ages[::-1])) # printing the sorted list in descending order
```

```
Minimun Value of the ages is 19
Maximum Value of the ages is 26
Sorted list in Ascending order is [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
Sorted list in Descending order is [26, 25, 25, 24, 24, 24, 22, 20, 19, 19]
```

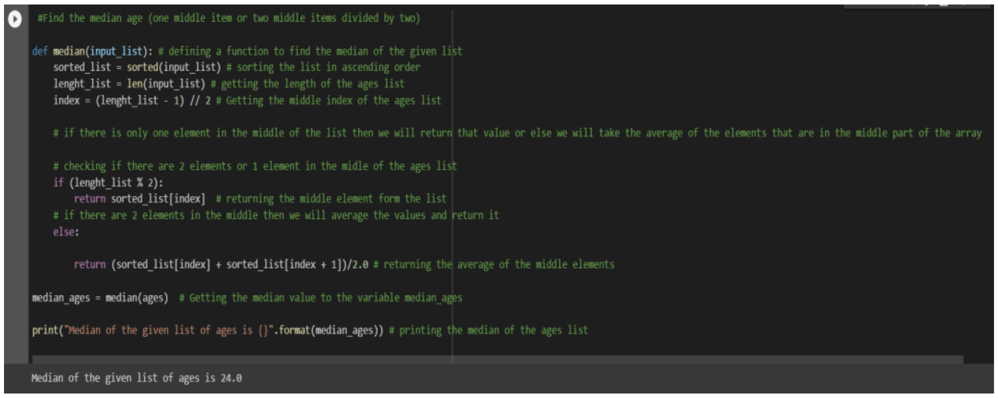
- **Add the min age and the max age again to the list**
  - ⇨ Here we used the append method to add the minimum and maximum values of the list to the sorted list sorted_ages.
  - ⇨ We then printed the list to which the minimum and maximum values of the list are added.



```
#Add the min age and the max age again to the list

sorted_ages.append(min_value) # appending the minimum value to the ages list
sorted_ages.append(max_value) # appending the maximum value to the ages list

print(sorted_ages) # printing the ages list after adding the minimum and maximum values
```

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

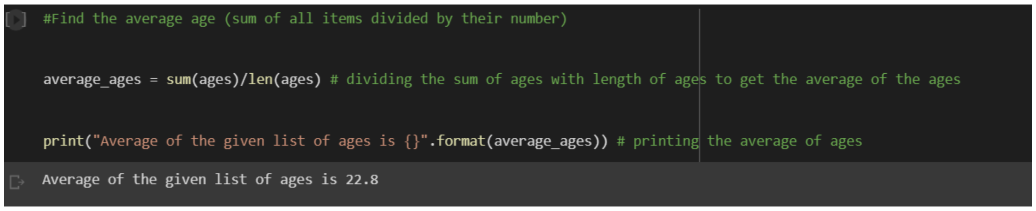
- **Find the median age (one middle item or two middle items divided by two)**
    - ⇨ Here we used a user defined method ‘median’ to calculate the median of the list.
    - ⇨ The operations done inside the median method is
        - Sort the given list.
        - Find the length of the list.
        - Find the middle index of the list
        - To find out if there is one middle value or 2 middle values in the list.
        - If there is one middle value, it would be the median of the list.
        - If there are 2 middle values, then the average of the 2 middle values would be the mean.
    - ⇨ And then call the method, get the median value, store it in a variable named **median_ages** and print the median values.



```
#Find the median age (one middle item or two middle items divided by two)

def median(input_list): # defining a function to find the median of the given list
    sorted_list = sorted(input_list) # sorting the list in ascending order
    lenght_list = len(input_list) # getting the length of the ages list
    index = (lenght_list - 1) // 2 # Getting the middle index of the ages list

    # if there is only one element in the middle of the list then we will return that value or else we will take the average of the elements that are in the middle part of the array

    # checking if there are 2 elements or 1 element in the midle of the ages list
    if (lenght_list % 2):
        return sorted_list[index]  # returning the middle element form the list
    # if there are 2 elements in the middle then we will average the values and return it
    else:

        return (sorted_list[index] + sorted_list[index + 1])/2.0 # returning the average of the middle elements

median_ages = median(ages)  # Getting the median value to the variable median_ages

print("Median of the given list of ages is {}".format(median_ages)) # printing the median of the ages list
```

```
Median of the given list of ages is 24.0
```

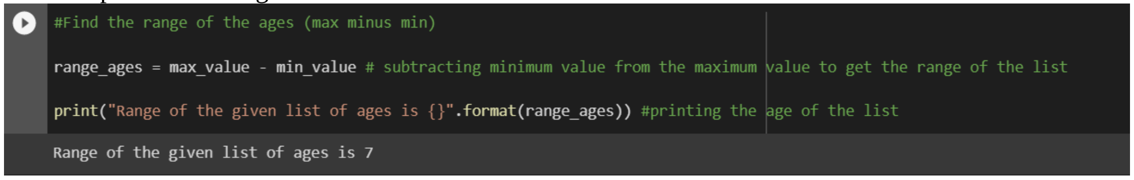
- **Find the average age (sum of all items divided by their number)**
    - ⇨ Here we calculated the sum of the values of the list by using sum() method.
    - ⇨ We calculated the number of elements of the list using len() method.
    - ⇨ We divided the sum of the list with length of the list to get the Average of the list.
    - ⇨ We then printed the Average of the list.



```
#Find the average age (sum of all items divided by their number)


average_ages = sum(ages)/len(ages) # dividing the sum of ages with length of ages to get the average of the ages


print("Average of the given list of ages is {}".format(average_ages)) # printing the average of ages
```

```
Average of the given list of ages is 22.8
```

- **Find the range of the ages (max minus min)**
Solution:
    - ⇨ We used the maximum value and minimum values of the list which was calculated previously and subtracted the minimum value from the maximum value to get the range of the list and stored it in a variable named ranges_ages.
    - ⇨ We then printed the range value.

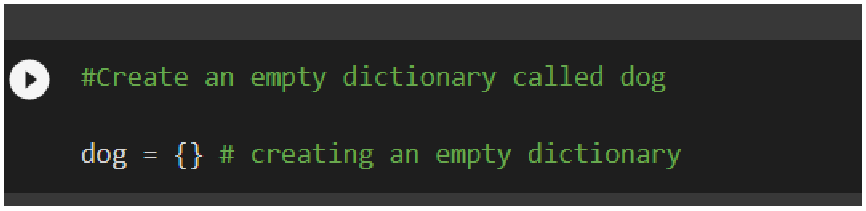


```
#Find the range of the ages (max minus min)

range_ages = max_value - min_value # subtracting minimum value from the maximum value to get the range of the list

print("Range of the given list of ages is {}".format(range_ages)) #printing the age of the list
```

```
Range of the given list of ages is 7
```

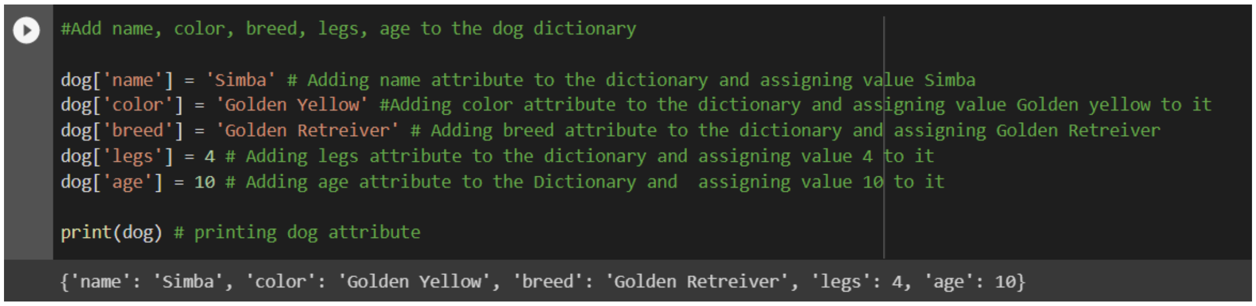
Question – 2

- **Create an empty dictionary called dog.**
  - ⇨ Here we are creating an empty dictionary named dog.



```
#Create an empty dictionary called dog

dog = {} # creating an empty dictionary
```

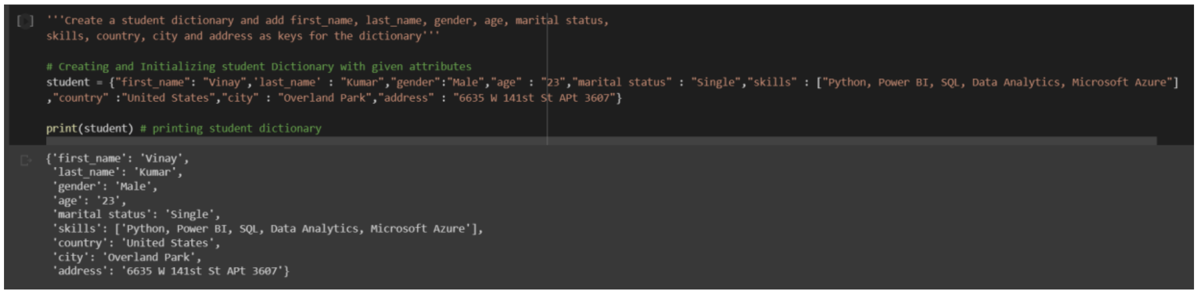
- **Add name, color, breed, legs, age to the dog dictionary**
  - ⇨ Here we added the given attributes to the dictionary one by one using the assignment operator.
  - ⇨ And then we printed the dictionary.



```
#Add name, color, breed, legs, age to the dog dictionary

dog['name'] = 'Simba' # Adding name attribute to the dictionary and assigning value Simba
dog['color'] = 'Golden Yellow' #Adding color attribute to the dictionary and assigning value Golden yellow to it
dog['breed'] = 'Golden Retreiver' # Adding breed attribute to the dictionary and assigning Golden Retreiver
dog['legs'] = 4 # Adding legs attribute to the dictionary and assigning value 4 to it
dog['age'] = 10 # Adding age attribute to the Dictionary and  assigning value 10 to it

print(dog) # printing dog attribute
```

```
{'name': 'Simba', 'color': 'Golden Yellow', 'breed': 'Golden Retreiver', 'legs': 4, 'age': 10}
```

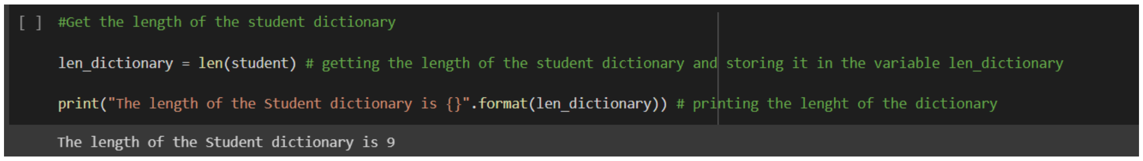
- **Create a student dictionary and add first_name, last_name, gender, age, marital status, skills, country, city and address as keys for the dictionary**
  - ⇨ Here we are Initializing student Dictionary with given attributes
  - ⇨ And then printing the dictionary.



```
'''Create a student dictionary and add first_name, last_name, gender, age, marital status,
skills, country, city and address as keys for the dictionary'''

# Creating and Initializing student Dictionary with given attributes
student = {"first_name": "Vinay",'last_name' : "Kumar","gender":"Male","age" : "23","marital status" : "Single","skills" : ["Python, Power BI, SQL, Data Analytics, Microsoft Azure"]
,"country" :"United States","city" : "Overland Park","address" : "6635 W 141st St APt 3607"}

print(student) # printing student dictionary
```

```
{'first_name': 'Vinay',
 'last_name': 'Kumar',
 'gender': 'Male',
 'age': '23',
 'marital status': 'Single',
 'skills': ['Python, Power BI, SQL, Data Analytics, Microsoft Azure'],
 'country': 'United States',
 'city': 'Overland Park',
 'address': '6635 W 141st St APt 3607'}
```

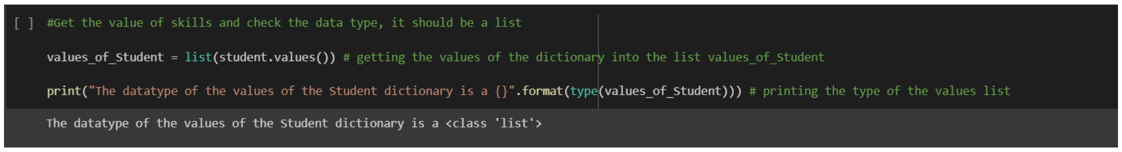
- **Get the length of the student dictionary**
  - ⇨ Here we got the length of the student Dictionary using the len() method.



```
#Get the length of the student dictionary

len_dictionary = len(student) # getting the length of the student dictionary and storing it in the variable len_dictionary

print("The length of the Student dictionary is {}".format(len_dictionary)) # printing the lenght of the dictionary
```

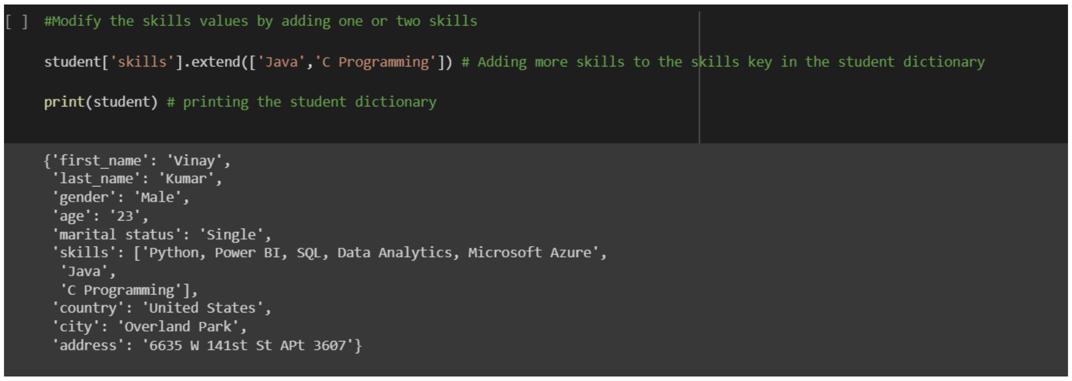
```
The length of the Student dictionary is 9
```

- **Get the value of skills and check the data type, it should be a list**
    - ⇨ Here we got the values of the dictionary using values() method.
    - ⇨ To check the type of the values we used type() method.



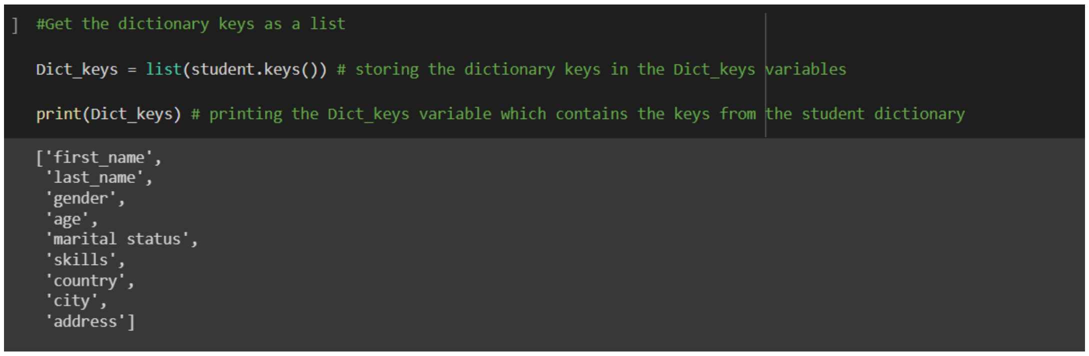
```
[ ] #Get the value of skills and check the data type, it should be a list

    values_of_Student = list(student.values()) # getting the values of the dictionary into the list values_of_Student

    print("The datatype of the values of the Student dictionary is a {}".format(type(values_of_Student))) # printing the type of the values list

    The datatype of the values of the Student dictionary is a <class 'list'>
```

- **Modify the skills values by adding one or two skills**
    - ⇨ We used the extend method to add values to the skills key in the Student Dictionary.



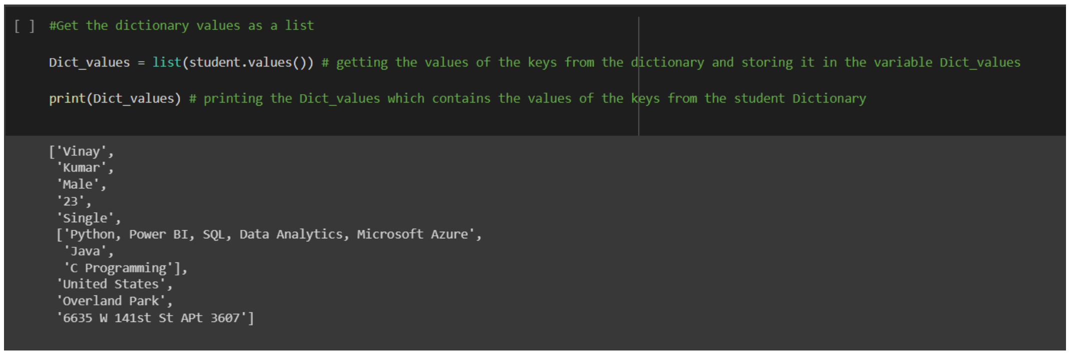
```
[ ] #Modify the skills values by adding one or two skills

    student['skills'].extend(['Java','C Programming']) # Adding more skills to the skills key in the student dictionary

    print(student) # printing the student dictionary

    {'first_name': 'Vinay',
     'last_name': 'Kumar',
     'gender': 'Male',
     'age': '23',
     'marital status': 'Single',
     'skills': ['Python, Power BI, SQL, Data Analytics, Microsoft Azure',
      'Java',
      'C Programming'],
     'country': 'United States',
     'city': 'Overland Park',
     'address': '6635 W 141st St APt 3607'}
```

- **Get the dictionary keys as a list**
    - ⇨ We used the keys() method to get the keys and type casted it to a list.



```
] #Get the dictionary keys as a list

  Dict_keys = list(student.keys()) # storing the dictionary keys in the Dict_keys variables

  print(Dict_keys) # printing the Dict_keys variable which contains the keys from the student dictionary

  ['first_name',
   'last_name',
   'gender',
   'age',
   'marital status',
   'skills',
   'country',
   'city',
   'address']
```

- **Get the dictionary values as a list**
    - ⇨ We used the values() method to get the values of the keys and type casted it to a list.

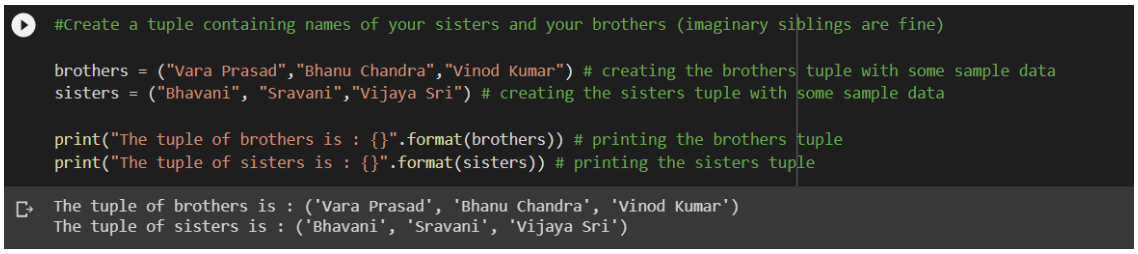


```
#Get the dictionary values as a list

Dict_values = list(student.values()) # getting the values of the keys from the dictionary and storing it in the variable Dict_values

print(Dict_values) # printing the Dict_values which contains the values of the keys from the student Dictionary
```

```
['Vinay',
 'Kumar',
 'Male',
 '23',
 'Single',
 ['Python, Power BI, SQL, Data Analytics, Microsoft Azure',
  'Java',
  'C Programming'],
 'United States',
 'Overland Park',
 '6635 W 141st St APt 3607']
```

Question – 3

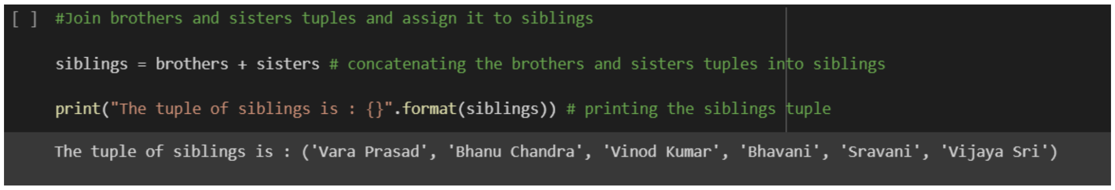
- **Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)**
  - ⇨ We initialized brothers and sisters tuples with sample data.



```
#Create a tuple containing names of your sisters and your brothers (imaginary siblings are fine)

brothers = ("Vara Prasad","Bhanu Chandra","Vinod Kumar") # creating the brothers tuple with some sample data
sisters = ("Bhavani", "Sravani","Vijaya Sri") # creating the sisters tuple with some sample data

print("The tuple of brothers is : {}".format(brothers)) # printing the brothers tuple
print("The tuple of sisters is : {}".format(sisters)) # printing the sisters tuple
```

```
The tuple of brothers is : ('Vara Prasad', 'Bhanu Chandra', 'Vinod Kumar')
The tuple of sisters is : ('Bhavani', 'Sravani', 'Vijaya Sri')
```

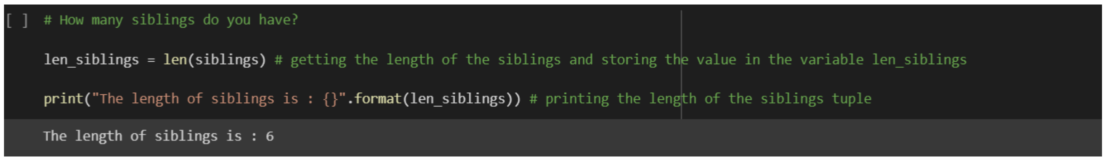
- **Join brothers and sisters tuples and assign it to siblings**
  - ⇨ We joined the brothers and sisters tuples using the + operator and stored it in siblings tuple.



```
#Join brothers and sisters tuples and assign it to siblings

siblings = brothers + sisters # concatenating the brothers and sisters tuples into siblings

print("The tuple of siblings is : {}".format(siblings)) # printing the siblings tuple
```

```
The tuple of siblings is : ('Vara Prasad', 'Bhanu Chandra', 'Vinod Kumar', 'Bhavani', 'Sravani', 'Vijaya Sri')
```

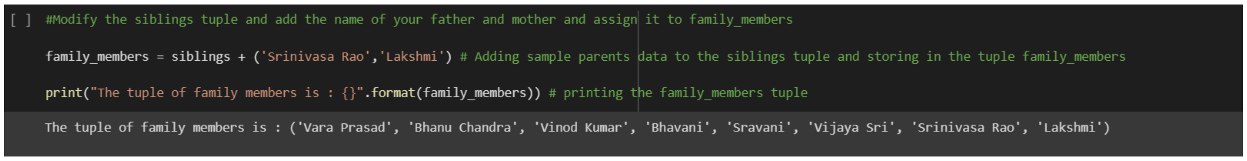
- **How many siblings do you have?**
  - ⇨ Calculated the number of siblings using len() method.



```
# How many siblings do you have?

len_siblings = len(siblings) # getting the length of the siblings and storing the value in the variable len_siblings

print("The length of siblings is : {}".format(len_siblings)) # printing the length of the siblings tuple
```

```
The length of siblings is : 6
```

- **Modify the siblings tuple and add the name of your father and mother and assign it to family_members**
  - ⇨ Father and Mother got added to the tuple using + operator and the resulting tuple is stored in the family_members tuple.



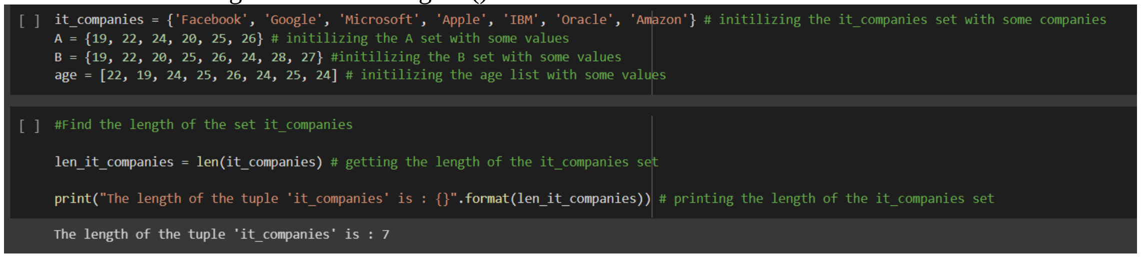
```
#Modify the siblings tuple and add the name of your father and mother and assign it to family_members

family_members = siblings + ('Srinivasa Rao','Lakshmi') # Adding sample parents data to the siblings tuple and storing in the tuple family_members

print("The tuple of family members is : {}".format(family_members)) # printing the family_members tuple
```

```
The tuple of family members is : ('Vara Prasad', 'Bhanu Chandra', 'Vinod Kumar', 'Bhavani', 'Sravani', 'Vijaya Sri', 'Srinivasa Rao', 'Lakshmi')
```

Question – 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**
  - ⇨ We found out the length of the set using len() method.

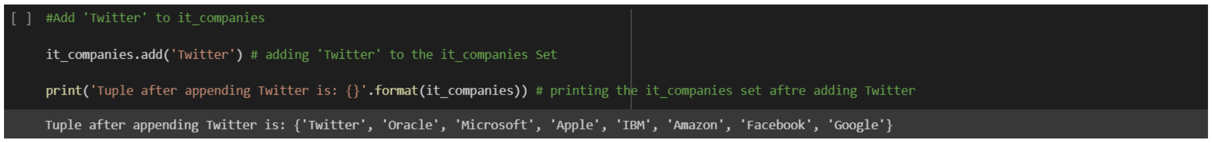


```
it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'} # initilizing the it_companies set with some companies
A = {19, 22, 24, 20, 25, 26} # initilizing the A set with some values
B = {19, 22, 20, 25, 26, 24, 28, 27} #initilizing the B set with some values
age = [22, 19, 24, 25, 26, 24, 25, 24] # initilizing the age list with some values
```

```
#Find the length of the set it_companies

len_it_companies = len(it_companies) # getting the length of the it_companies set

print("The length of the tuple 'it_companies' is : {}".format(len_it_companies)) # printing the length of the it_companies set
```

```
The length of the tuple 'it_companies' is : 7
```

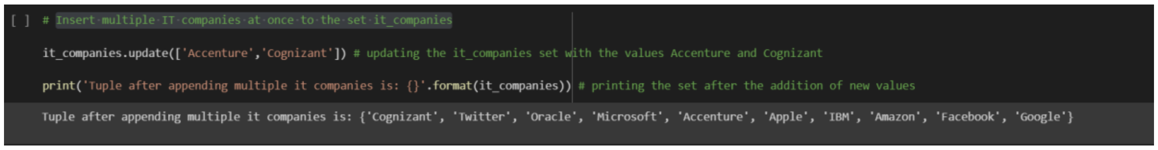
- **Add 'Twitter' to it_companies**
  - ⇨ Twitter got added using the add() method.



```
#Add 'Twitter' to it_companies

it_companies.add('Twitter') # adding 'Twitter' to the it_companies Set

print('Tuple after appending Twitter is: {}'.format(it_companies)) # printing the it_companies set aftre adding Twitter
```

```
Tuple after appending Twitter is: {'Twitter', 'Oracle', 'Microsoft', 'Apple', 'IBM', 'Amazon', 'Facebook', 'Google'}
```

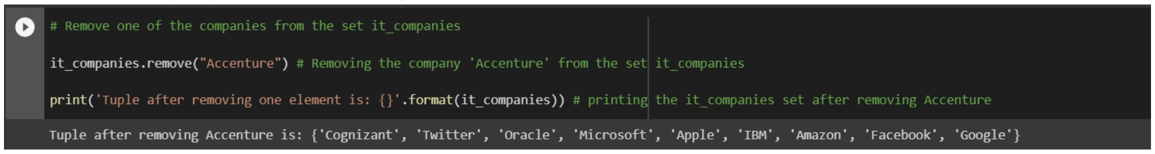
- **Insert multiple IT companies at once to the set it_companies**
  - ⇨ Multiple values are added to the tuple using update() method.



```
# Insert multiple IT companies at once to the set it_companies

it_companies.update(['Accenture','Cognizant']) # updating the it_companies set with the values Accenture and Cognizant

print('Tuple after appending multiple it companies is: {}'.format(it_companies)) # printing the set after the addition of new values
```

```
Tuple after appending multiple it companies is: {'Cognizant', 'Twitter', 'Oracle', 'Microsoft', 'Accenture', 'Apple', 'IBM', 'Amazon', 'Facebook', 'Google'}
```

- **Remove one of the companies from the set it_companies**
  - ⇨ One of the company is removed using remove() method.



```
# Remove one of the companies from the set it_companies

it_companies.remove("Accenture") # Removing the company 'Accenture' from the set it_companies

print('Tuple after removing one element is: {}'.format(it_companies)) # printing the it_companies set after removing Accenture
```

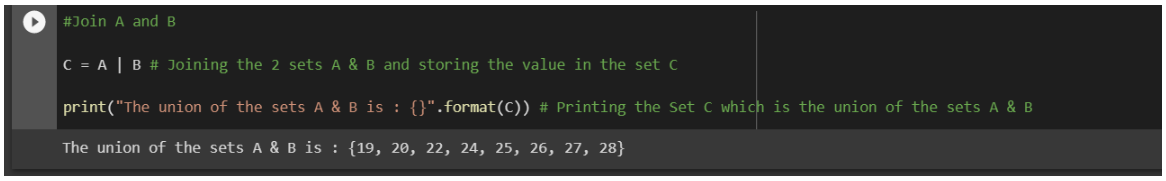
```
Tuple after removing Accenture is: {'Cognizant', 'Twitter', 'Oracle', 'Microsoft', 'Apple', 'IBM', 'Amazon', 'Facebook', 'Google'}
```

- **What is the difference between remove and discard ?**

⇨ Both of the methods do the same work (i.e.), removes the element from the set only if the element is present in the set. the only difference is remove() method throws an error/exception if the element is not present in the set which is supposed to be removed and the discard method does not throw the exception.

- **Join A and B**

⇨ The sets A & B are joined using the union operator '|'.

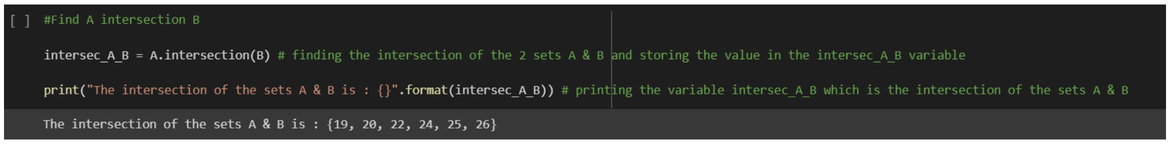


```
#Join A and B

C = A | B # Joining the 2 sets A & B and storing the value in the set C

print("The union of the sets A & B is : {}".format(C)) # Printing the Set C which is the union of the sets A & B
```

```
The union of the sets A & B is : {19, 20, 22, 24, 25, 26, 27, 28}
```

- **Find A intersection B**

⇨ Intersection of the 2 sets are done using the intersection() method.

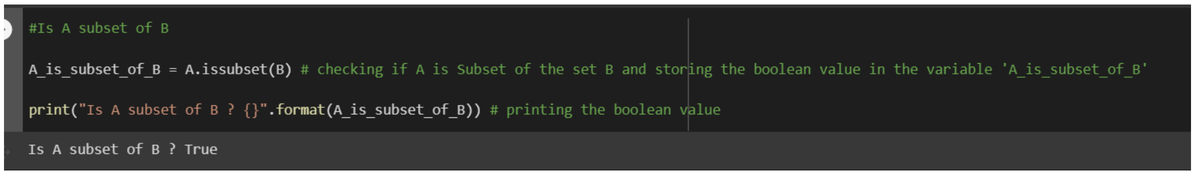


```
[ ] #Find A intersection B

    intersec_A_B = A.intersection(B) # finding the intersection of the 2 sets A & B and storing the value in the intersec_A_B variable

    print("The intersection of the sets A & B is : {}".format(intersec_A_B)) # printing the variable intersec_A_B which is the intersection of the sets A & B
```

```
The intersection of the sets A & B is : {19, 20, 22, 24, 25, 26}
```

- **Is A subset of B**

⇨ The above statement is verified by using issubset() method.



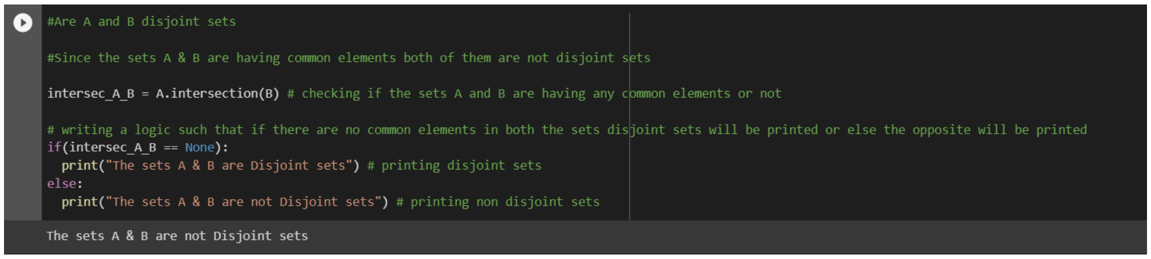
```
#Is A subset of B

A_is_subset_of_B = A.issubset(B) # checking if A is Subset of the set B and storing the boolean value in the variable 'A_is_subset_of_B'

print("Is A subset of B ? {}".format(A_is_subset_of_B)) # printing the boolean value
```

```
Is A subset of B ? True
```

- **Are A and B disjoint sets**

⇨ 2 sets are said to be disjoint if there are no common elements in the sets.

⇨ Here intersection is done for both the sets and if there are common elements, then not disjoint sets is printed or else, disjoint set is printed using an if loop.

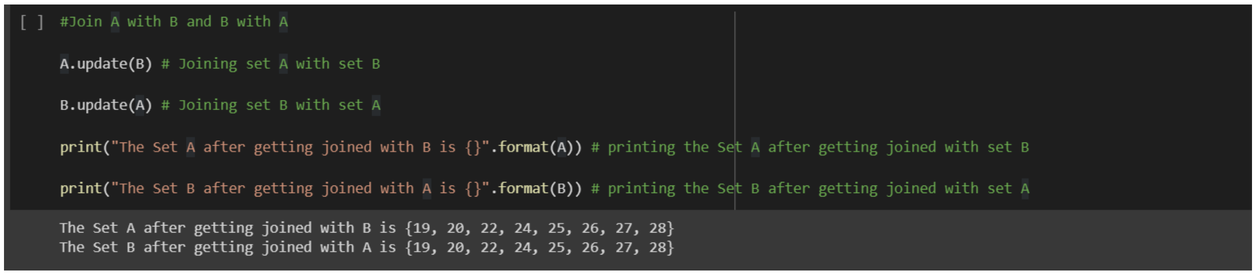


```
#Are A and B disjoint sets

#Since the sets A & B are having common elements both of them are not disjoint sets

intersec_A_B = A.intersection(B) # checking if the sets A and B are having any common elements or not

# writing a logic such that if there are no common elements in both the sets disjoint sets will be printed or else the opposite will be printed
if(intersec_A_B == None):
  print("The sets A & B are Disjoint sets") # printing disjoint sets
else:
  print("The sets A & B are not Disjoint sets") # printing non disjoint sets
```

```
The sets A & B are not Disjoint sets
```

- **Join A with B and B with A**

⇨ B is joined to A using the update() method and A is joined to B using the same method.

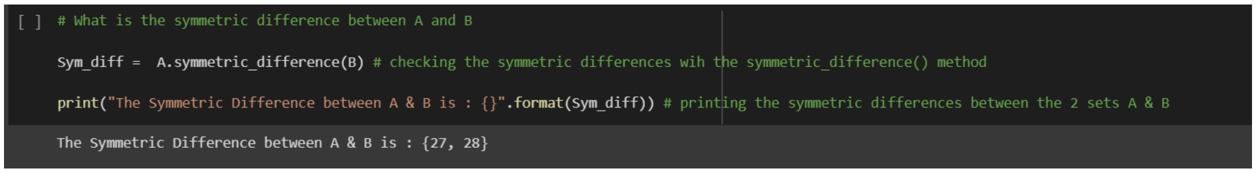


```
[ ] #Join A with B and B with A

    A.update(B) # Joining set A with set B

    B.update(A) # Joining set B with set A

    print("The Set A after getting joined with B is {}".format(A)) # printing the Set A after getting joined with set B

    print("The Set B after getting joined with A is {}".format(B)) # printing the Set B after getting joined with set A
```

```
The Set A after getting joined with B is {19, 20, 22, 24, 25, 26, 27, 28}
The Set B after getting joined with A is {19, 20, 22, 24, 25, 26, 27, 28}
```

- **What is the symmetric difference between A and B**

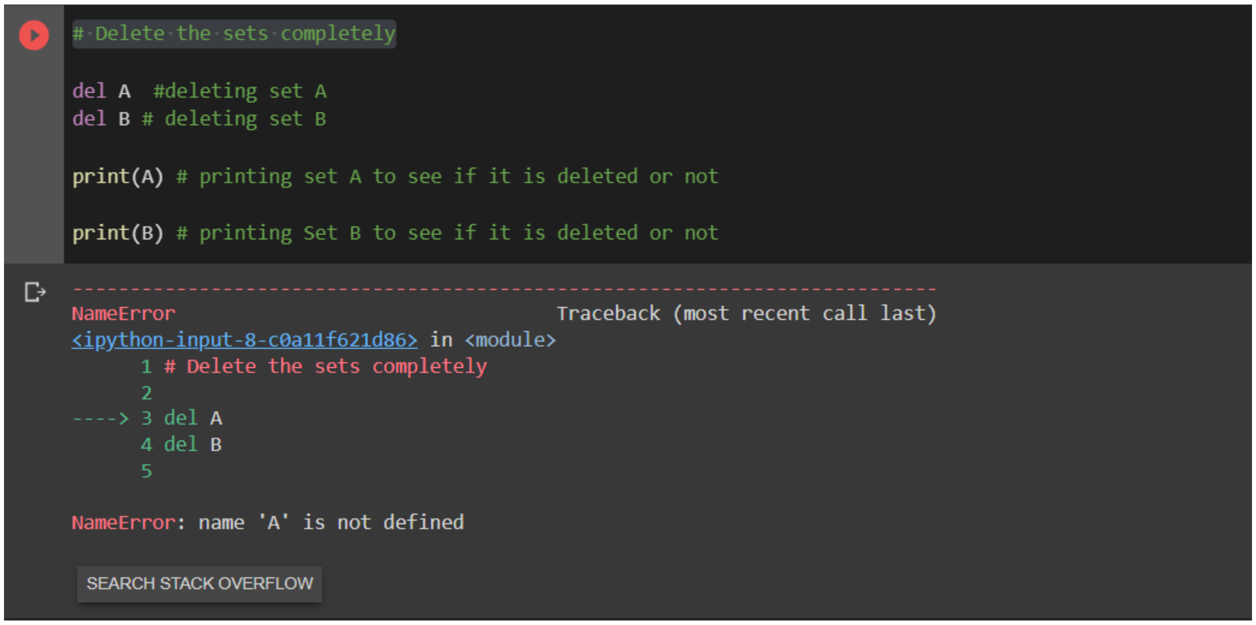
⇨ Symmetric difference is calculated using the symmetric_difference() method.



```
[ ] # What is the symmetric difference between A and B

    Sym_diff =  A.symmetric_difference(B) # checking the symmetric differences wih the symmetric_difference() method

    print("The Symmetric Difference between A & B is : {}".format(Sym_diff)) # printing the symmetric differences between the 2 sets A & B

    The Symmetric Difference between A & B is : {27, 28}
```

- **Delete the sets completely**

⇨ Both the sets are deleted using the del method.



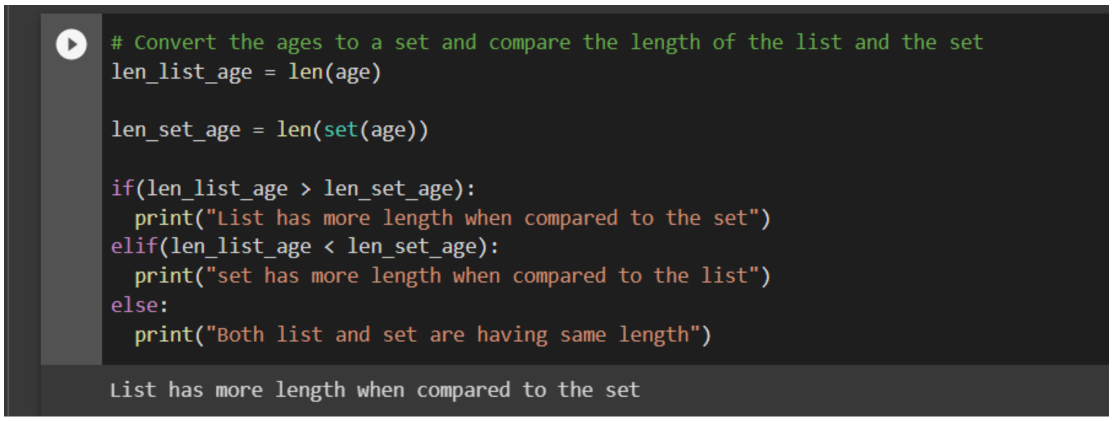
```
# Delete the sets completely

del A  #deleting set A
del B # deleting set B

print(A) # printing set A to see if it is deleted or not

print(B) # printing Set B to see if it is deleted or not
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-c0a11f621d86> in <module>
      1 # Delete the sets completely
      2
----> 3 del A
      4 del B
      5

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

SEARCH STACK OVERFLOW

- **Convert the ages to a set and compare the length of the list and the set**

⇨ The length of the set and list are calculated using the len() method.



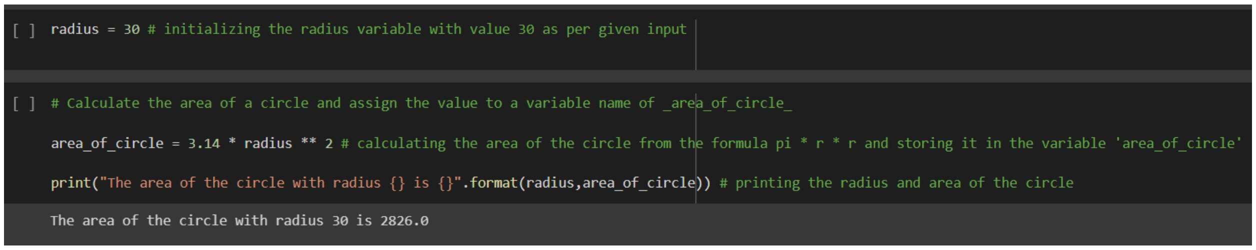
```
# Convert the ages to a set and compare the length of the list and the set
len_list_age = len(age)

len_set_age = len(set(age))

if(len_list_age > len_set_age):
  print("List has more length when compared to the set")
elif(len_list_age < len_set_age):
  print("set has more length when compared to the list")
else:
  print("Both list and set are having same length")
```

```
List has more length when compared to the set
```

Question – 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_**
  - ⇨ Here the area of the circle is calculated using pi*r*r and the value is stored in the variable area_of_circle.

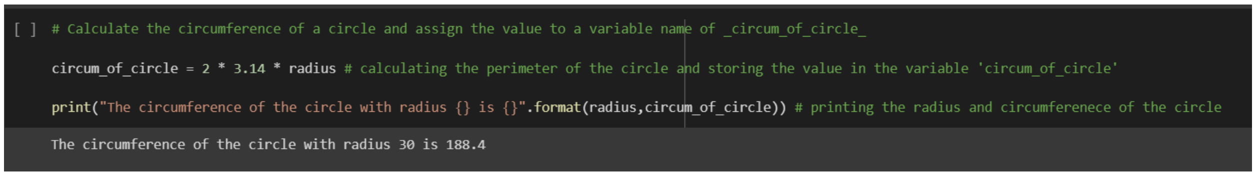


```
[ ] radius = 30 # initializing the radius variable with value 30 as per given input
```

```
[ ] # Calculate the area of a circle and assign the value to a variable name of _area_of_circle_

    area_of_circle = 3.14 * radius ** 2 # calculating the area of the circle from the formula pi * r * r and storing it in the variable 'area_of_circle'

    print("The area of the circle with radius {} is {}".format(radius,area_of_circle)) # printing the radius and area of the circle
```

```
The area of the circle with radius 30 is 2826.0
```

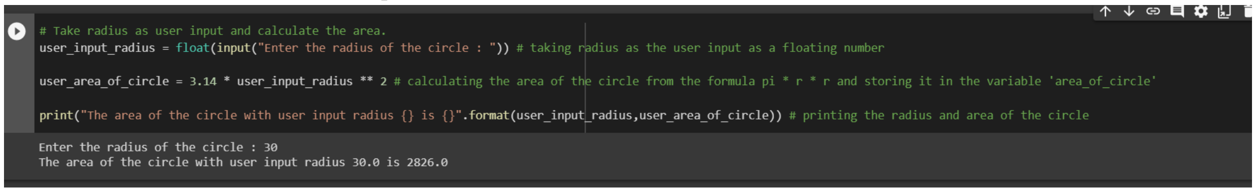
- **Calculate the circumference of a circle and assign the value to a variable name of _circum_of_circle_**
  - ⇨ Here the perimeter of the circle is calculated using 2*pi*r and the value is stored in the variable circum _of_circle.



```
[ ] # Calculate the circumference of a circle and assign the value to a variable name of _circum_of_circle_

    circum_of_circle = 2 * 3.14 * radius # calculating the perimeter of the circle and storing the value in the variable 'circum_of_circle'

    print("The circumference of the circle with radius {} is {}".format(radius,circum_of_circle)) # printing the radius and circumferenece of the circle
```

```
The circumference of the circle with radius 30 is 188.4
```

- **Take radius as user input and calculate the area.**
  - ⇨ Here the radius is taken as a user input and the area of the circle is calculated.



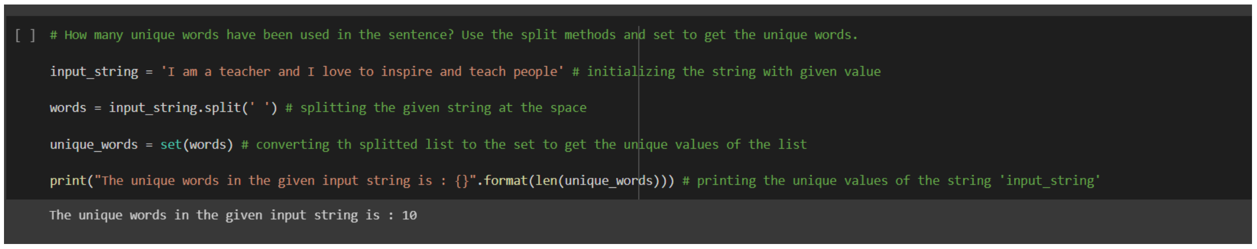
```
# Take radius as user input and calculate the area.
user_input_radius = float(input("Enter the radius of the circle : ")) # taking radius as the user input as a floating number

user_area_of_circle = 3.14 * user_input_radius ** 2 # calculating the area of the circle from the formula pi * r * r and storing it in the variable 'area_of_circle'

print("The area of the circle with user input radius {} is {}".format(user_input_radius,user_area_of_circle)) # printing the radius and area of the circle
```

```
Enter the radius of the circle : 30
The area of the circle with user input radius 30.0 is 2826.0
```

Question – 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.**
  - ⇨ Here the string is divided at spaces using split() method and the resulting list is converted to a set to get the unique values of the string and len() method is used to get the number of unique words.



```
[ ] # How many unique words have been used in the sentence? Use the split methods and set to get the unique words.

    input_string = 'I am a teacher and I love to inspire and teach people' # initializing the string with given value

    words = input_string.split(' ') # splitting the given string at the space

    unique_words = set(words) # converting th splitted list to the set to get the unique values of the list

    print("The unique words in the given input string is : {}".format(len(unique_words))) # printing the unique values of the string 'input_string'
```

```
The unique words in the given input string is : 10
```

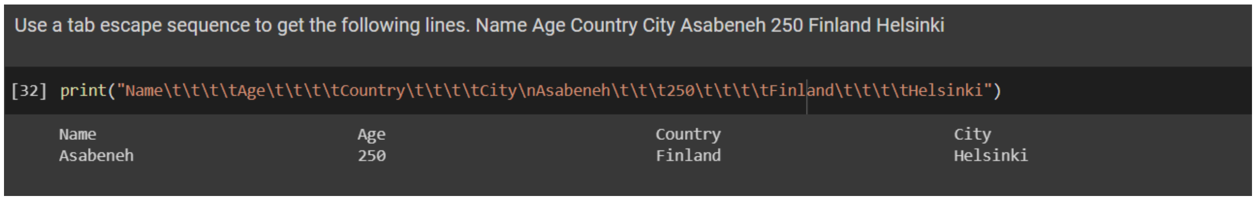
Question – 7

Use a tab escape sequence to get the following lines.

Name Age Country City

Asabeneh 250 Finland Helsinki

⇨ Here \t is used for a tab and \n is used for new line in the print statement.



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

```
[32] print("Name\t\t\t\tAge\t\t\t\tCountry\t\t\t\tCity\nAsabeneh\t\t\t250\t\t\t\tFinland\t\t\t\tHelsinki")
```

```
Name                                    Age                                     Country                                 City
Asabeneh                                250                                     Finland                                 Helsinki
```

Question – 8

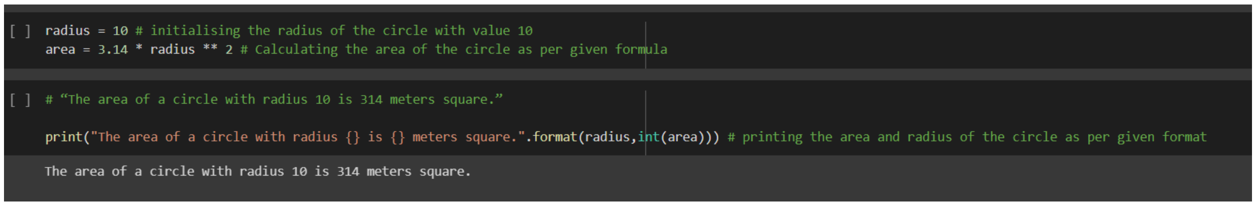
Use the string formatting method to display the following:

radius = 10

area = 3.14 * radius ** 2

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

- ⇨ .format() method is used to get the desired output.



```
radius = 10 # initialising the radius of the circle with value 10
area = 3.14 * radius ** 2 # Calculating the area of the circle as per given formula
```

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

print("The area of a circle with radius {} is {} meters square.".format(radius,int(area))) # printing the area and radius of the circle as per given format
```

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

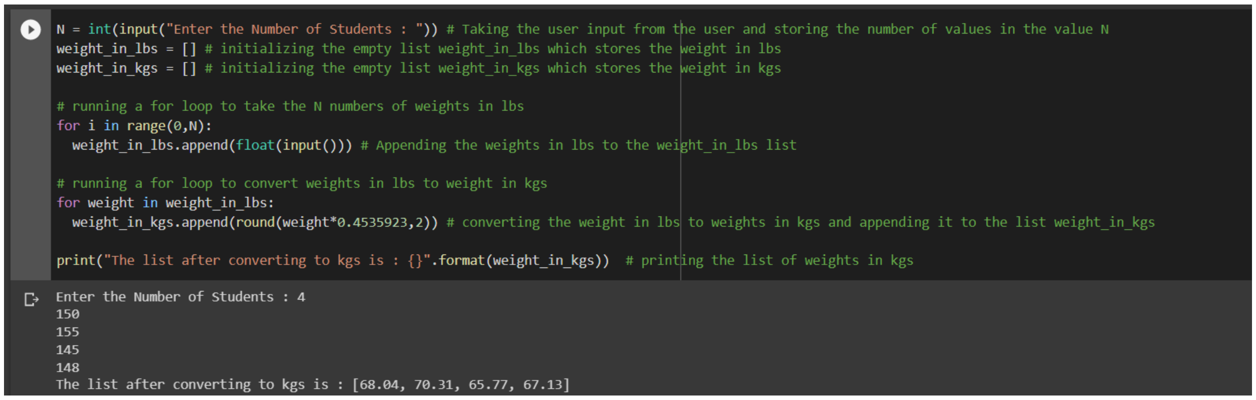
Question – 9

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]

- ⇨ Here 2 empty lists are initialized to store the weights in lbs and kgs.
- ⇨ The number of weights are taken as user input.
- ⇨ Then the values of user input weights in lbs is added to the lbs list.
- ⇨ Then using a for loop the values in lbs are converted to kgs and gets added to list of kgs.

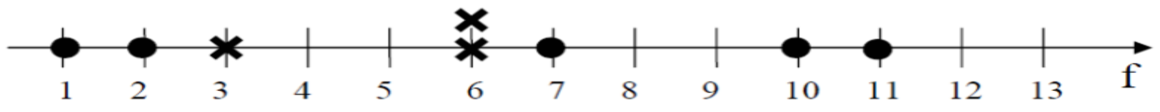


```
N = int(input("Enter the Number of Students : ")) # Taking the user input from the user and storing the number of values in the value N
weight_in_lbs = [] # initializing the empty list weight_in_lbs which stores the weight in lbs
weight_in_kgs = [] # initializing the empty list weight_in_kgs which stores the weight in kgs

# running a for loop to take the N numbers of weights in lbs
for i in range(0,N):
  weight_in_lbs.append(float(input())) # Appending the weights in lbs to the weight_in_lbs list

# running a for loop to convert weights in lbs to weight in kgs
for weight in weight_in_lbs:
  weight_in_kgs.append(round(weight*0.4535923,2)) # converting the weight in lbs to weights in kgs and appending it to the list weight_in_kgs

print("The list after converting to kgs is : {}".format(weight_in_kgs))  # printing the list of weights in kgs
```

```
Enter the Number of Students : 4
150
155
145
148
The list after converting to kgs is : [68.04, 70.31, 65.77, 67.13]
```

Question - 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.**

Solution

As per the given data, let's assume that 0 represents a dot and 1 represents a cross.

So the data points would be (1, 0), (2, 1), (3, 1), (6, 1), (6, 1), (7, 0), (10, 0), (11, 0).

By dividing the dataset into equal parts randomly we get,

The training set to be (2, 1), (6, 1), (7, 0), (11, 0).

The test set to be (1, 0), (3, 1), (6, 1), (10, 0).

Now by calculating the distances of each test point from each of the train set, we would get the following table.

Here we would consider 3 nearest points from the 4 training points and predict the y co-ordinate value of test data point with majority value in the y co-ordinate of the training points.

As we see for the first test point, we have two 1's and one 0 in the nearest 3 points of the training set, hence we predicted the y co-ordinate value of test point to be 1.

Similarly for the second test point, we have two 1's and one 0 in the nearest 3 points of the training set, hence we predicted the y co-ordinate value of test point to be 1.

Similarly for the third test point, we have two 1's and one 0 in the nearest 3 points of the training set, hence we predicted the y co-ordinate value of test point to be 1.

Similarly for the fourth test point, we have two 0's and one 1 in the nearest 3 points of the training set, hence we predicted the y co-ordinate value of test point to be 0.

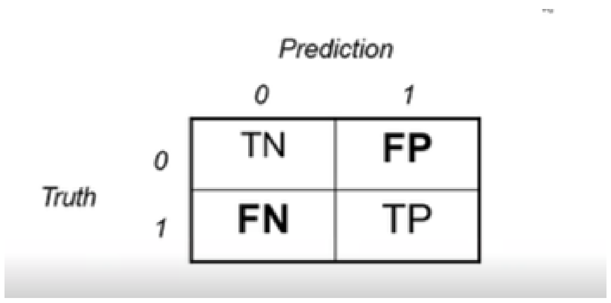
Train Set → Test Set ↓	(2, 1)	(6, 1)	(7, 0)	(11, 0)	KNN classifier, for K=3 (predicted output)	Actual Output
(1, 0)	1.414	5.099	6.0	10	1	0
(3, 1)	1.0	3.1	5.099	9.055	1	1
(6, 1)	4.0	0.0	1.414	5.099	1	1
(10, 0)	8.06	4.123	3	4	0	0

(Note: The distances between the points is calculated by the Euclidean Formula

The distance between the points (x1,y1) and (x2,y2) is distance = ((x1 - x2)^2 + (y1 - y2)^2)^0.5)

2. Compute the confusion matrix for this and calculate accuracy, sensitivity and specificity values

Solution:



		Prediction	
		0	1
Truth	0	TN	**FP**
	1	**FN**	TP

As per the above table, the confusion matrix is [1 1]

[0 2]

- ⇨ **TP = 2**
- ⇨ **TN = 1**
- ⇨ **FP = 1**
- ⇨ **FN = 0**

Accuracy:

Accuracy = (TP+TN)/(P+N) = ¾ = 0.75 %

Sensitivity:

Sensitivity = TP/(TP+FN) = TP/P = 2/(2+0) = 100 %

Specificity:

Specificity = TN/(FP+TN) = TN/N = 1/(2) = 50 %