Solution 1:

```
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
ages.sort()
print("The sorted list is: ",ages)
min_age=min(ages)
max_age=max(ages)
print("The min and max ages of the given list are: ",min_age,max_age)

#again adding the min and max age to the list
ages.extend((min_age,max_age))
print("The updated list ",ages)

#median of ages
if len(ages) % 2 == 0:
    median = (ages[len(ages)//2] + ages[len(ages)//2-1])/2
else:
    median = ages[len(ages)//2]
print("The median of ages: ",median)

#average age
average = sum(ages)/len(ages)
print("The average age of the given list: ",average)

#range of ages
range = max_age - min_age
print("The range of given ages",range)
```

Output:

```
The sorted list is:  [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
The min and max ages of the given list are:  19 26
The updated list  [19, 19, 20, 22, 24, 24, 24, 25, 25, 26, 19, 26]
The median of ages:  24.0
The average age of the given list:  22.75
The range of given ages 7
```

Explanation:

Sorted the given list using the built-in sort() function and then found the min and max ages by using the built-in min() and max() functions. Used extend() function, as more than 1 element need to be added. For median if the list having even no. of elements the average should be taken for middle two elements. The average of elements is the summation of elements by using sum() function divided by total no. of elements by using len() function.

Solution 2:

```
#creating empty dictionary
dog={}

#adding key and values to the dictionary
dog={'name':'Bruno','color':'Brown','breed':'Labrador','legs':4,'age':6}
print(dog)

#creating student dictionary and assigning keys & values
student={'first_name':"John",'last_name':"Miller",'gender':"Male",'age':28,'marital status':'Single',
         'skills':["Java", "Angular"],'country':"United States",'city':"Overland Park",'address':"8720 W"}

#length of student dictionary
length=len(student)
print(length)

#value of skills key
student['skills']

#datatype of skills key
print(type(student['skills']))

#adding two skills
student['skills'].extend((('Salesforce'),('Machine Learning')))
print("The student details are: ",student)

#dictionary keys as a list
keys=list(student.keys())
print(keys)

#dictionary values as a list
values=list(student.values())
print(values)
```

Output:

```
{'name': 'Bruno', 'color': 'Brown', 'breed': 'Labrador', 'legs': 4, 'age': 6}
9
<class 'list'>
The student details are:  {'first_name': 'John', 'last_name': 'Miller', 'gender': 'Male', 'age': 28, 'marital statu
s': 'Single', 'skills': ['Java', 'Angular', 'Salesforce', 'Machine Learning'], 'country': 'United States', 'city': 'O
verland Park', 'address': '8720 W'}
['first_name', 'last_name', 'gender', 'age', 'marital status', 'skills', 'country', 'city', 'address']
['John', 'Miller', 'Male', 28, 'Single', ['Java', 'Angular', 'Salesforce', 'Machine Learning'], 'United States', 'Ove
rland Park', '8720 W']
```

Explanation:

Created the empty dictionary as dog and added the keys and values to the dictionary. Created the student dictionary by assigning the keys and values. Length of dictionary is calculated by len() function. Retrieved the value of skills and found the datatype by using type() function. Modified the dictionary using extend() function as more than 1 value need to be added. Retrieved keys and values by built-in keys() and values() function

Solution 3:

```
#creating tuples conatining sister and brother names
brothers=("John","Drake","Simon")
sisters=("Christine","Dalia")

#joining brothers and sisters tuples and assigning it to siblings
siblings=brothers+sisters
print(siblings)

#count of siblings
num_of_siblings=len(siblings)
print(num_of_siblings)

#as the tuples are immutable converting to list and then back to tuple
family_members = list(siblings)
family_members.extend(["Swamy", "Sangeetha"])
family_members = tuple(family_members)
print(family_members)
```

Output:

```
('John', 'Drake', 'Simon', 'Christine', 'Dalia')
5
('John', 'Drake', 'Simon', 'Christine', 'Dalia', 'Swamy', 'Sangeetha')
```

Explanation:

Created two tuples and joined the tuples using '+' operator and assigned it to siblings. As tuples are immutable it's not possible to add or remove elements to it, so converted the tuple to a list, made the modification and then converted it back to a tuple

Solution 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]

#length of set it_companies
length=len(it_companies)
print(length)

#adding 'Twitter' to it_companies
it_companies.add('Twitter')
print(it_companies)

#inserting multiple IT companies at once
it_companies.update({'Capgemini','Cognizant','TCS'})
print(it_companies)

#removing Amazon from the set
it_companies.remove('Amazon')
print(it_companies)

#difference b/w remove and discard
# it_companies.remove('Salesforce') -- throwing KeyError as the element is not found
#in the list whereas discards doesn't throws the error
it_companies.discard('Salesforce')

#joining A and B
C = A.union(B)
print(C)

#A intersection B
D = A.intersection(B)
print(D)

#A subset of B
print(A.issubset(B))
```

```
# are A and B disjoint sets
print(A.isdisjoint(B))

#joining A with B and B with A
E = A.symmetric_difference(B)
print(E)

#symmetric difference between A and B
F = A.symmetric_difference(B)
print(F)

#deleting the sets completely
del A
del B
del C
del D
del E
del F

#converting to set and comparing the lengths
age_set=set(age)
print(len(age))
print(len(age_set))
```

Output:

```
7
{'Apple', 'IBM', 'Microsoft', 'Google', 'Twitter', 'Amazon', 'Facebook', 'Oracle'}
{'TCS', 'IBM', 'Microsoft', 'Twitter', 'Amazon', 'Cognizant', 'Capgemini', 'Apple', 'Facebook', 'Google', 'Oracle'}
{'TCS', 'IBM', 'Microsoft', 'Twitter', 'Cognizant', 'Capgemini', 'Apple', 'Facebook', 'Google', 'Oracle'}
{19, 20, 22, 24, 25, 26, 27, 28}
{19, 20, 22, 24, 25, 26}
True
False
{27, 28}
{27, 28}
8
5
```

Explanation:

In this question we have used all the built-in functions to perform the operations and as well as to retrieve the data.

The difference between remove and discard is when the remove() function is used if the element is not found in the list KeyError is thrown whereas if discard() function used no error is thrown.

Solution 5:

```
import math

Radius = 30
#calculating area of circle
_area_of_circle = math.pi * (Radius ** 2)
print("The area of circle with the given radius: ",round(_area_of_circle,2))

#calculating perimeter of circle
_perimeter_of_circle = 2 * math.pi * Radius
print("The perimeter of circle: ",round(_perimeter_of_circle,2))

#calculating area of circle with the inpt radius
radius = float (input ("Please enter the radius of the circle: "))
_area_of_circle_ipradius = math.pi * (radius ** 2)
print("The area of circle with inpurt radius: ",round(_area_of_circle_ipradius,2))
```

Output:

```
The area of circle with the given radius:  2827.4333882308138
The perimeter of circle:  188.49555921538757
Please enter the radius of the circle: 13.5
The area of circle with inpurt radius:  572.5552611167398
```

Explanation:

To calculate the area of circle we have imported math package as we need to use the value of pi and calculated the area and perimeter of circle using the standard formulas. Used the round() function to round off the decimal values.

Solution 6:

```
statement = "I am a teacher and I love to inspire and teach people"
#using split method to split the given statement
split_words = statement.split()

#set function is used remove the duplicate elements
unique_words = set(split_words)

#calculating the count of unique words
print(len(unique_words))
```

Output:

```
10
```

Explanation:

To find the unique words in a given statement, used the split() function to split the words individually and then converted to set by passing in the set() function to remove the duplicate elements. Then calculated the count of unique words using len() function.

Solution 7:

```
print("Name\tAge\tCountry\tCity")
print("Asabeneh\t250\tFinland\tHelsinki")
```

Output:

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

Explanation:

The tab escape sequence \t is used to create a tab space between the words, so used \t in both the lines to create tab space between the elements.

Solution 8:

```
radius = 10
area = 3.14 * (radius ** 2)
print("The area of a circle with radius %r is %a meters square"%(radius,area))
```

Output:

```
The area of a circle with radius 10 is 314.0 meters square
```

Explanation:

The string formatting method is used to insert the values of the variables into the string. It can be implemented in multiple ways, above is one way.
Other way is: print("The area of a circle with radius {} is {} meters square.".format(radius, area)), whereas the curly braces {} are used as placeholders for the values, and the values are passed as arguments to the format()
Another way: print(f"The area of a circle with radius {radius} is {area} meters square.")

Solution 9:

```
# creating an empty list for lbs

weight_in_lbs = []

# number of elements as input
n = int(input("Enter number of students : "))

# iterating till the range
for i in range(n):
    lbs_lst = float(input("Enter the weights of students in lbs: "))
    # adding the weights
    weight_in_lbs.append(lbs_lst)

print("The weights of",n,"students in lbs is: ",weight_in_lbs)

# creating an empty list for kgs
weight_in_kgs=[]
for i in weight_in_lbs:
    weight_in_kgs.append(round(i/2.205, 2))

print("The weights of same",n,"students in kgs is: ",(weight_in_kgs))
```

Output:

```
Enter number of students : 4
Enter the weights of students in lbs: 150
Enter the weights of students in lbs: 155
Enter the weights of students in lbs: 145
Enter the weights of students in lbs: 148
The weights of 4 students in lbs is:  [150.0, 155.0, 145.0, 148.0]
The weights of same 4 students in kgs is:  [68.03, 70.29, 65.76, 67.12]
```

Explanation:

Created empty list for weights in lbs and declared n variable to input the no. of weights to be entered. Then iterated over lbs list to append the entered values to the list. Then converted to kgs by dividing with the conversion value while iterating over the appended lbs list.

Video Link:
https://drive.google.com/drive/folders/12SAXk_eW3mgxQ9Mj8rZKUSNWLbQCg-WX?usp=sharing

GitHub Link:
https://github.com/shruthikatkam26/MachinelearningICP1.git