TECHVIDYA

#1
LIMITED EDITION

VOL 314

INTERVIEW QUESTIONS

125 MUST HAVE PYTHON QUESTIONS

"STAY UPDATED, STAY AHEAD"

TechVidya is ISO Certified 9001:2015 accredited EdTech Company registered with ROC under the companies act 1956, offers self-paced, online and offline programs. TechVidya pedagogy is rooted in the principle that every young mind should be equipped with exceptional knowledge and skills that benefit them in real life.

YEAR 2023

Company Info
TechVidya
EdTech Company

Company Website
techvidya.education
Official Website

Company Contact
+91 83759 66700
Official Number

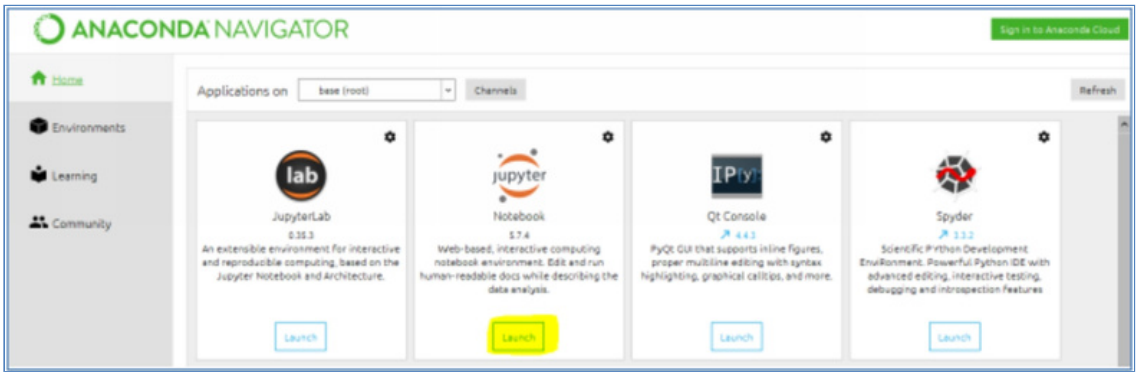
1. What is Python?
   Python is a general purpose interpreted, interactive, object-oriented, and high level language. It was created by Guido Rossum in 1990.

2. What are all these buzz words, interpreted, interactive, etc.?
   By interpreted we mean that you do not need to compile your code before executing it.
   By interactive we mean that you can interact with the interpreter directly to write your program.
   Object oriented technique of programming encapsulates code within objects

3. How to install Python?
   You can download Python directly from the link below
   https://www.python.org/downloads/
   You can get the documentation from the link below
   https://www.python.org/doc/

4. What is Anaconda and Jupyter notebook?
   Anaconda Distribution is the easiest way to perform Python or R on Linux, Windows or any other Operating System. Jupyter is the place where you write codes and document it. We would recommend you the download it from the link below before moving with the coding part
   https://www.anaconda.com/distribution/

   To launch Jupyter notebook you just have to install Anaconda Navigator from the link above and then click the launch Jupyter (Given in the picture below)

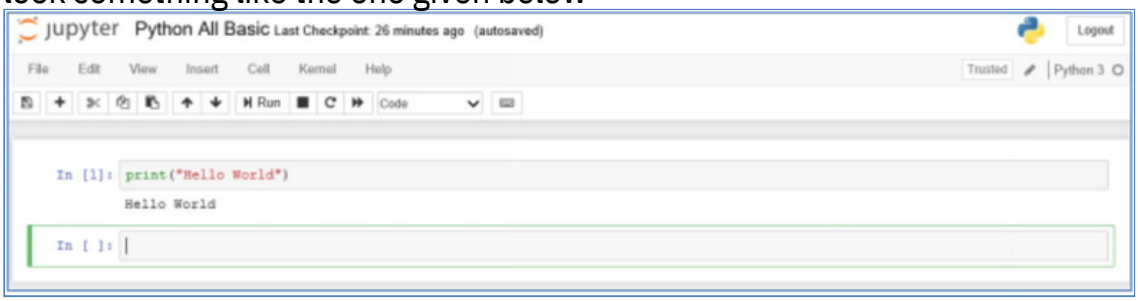


5. Now we have launched Jupyter Notebook, How to write the first program?

Let's start with Hello World program

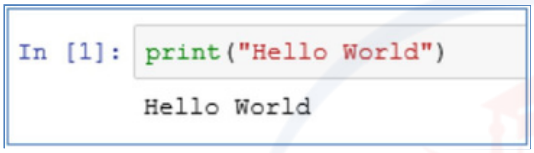
I assume that you have already opened Jupyter notebook from the link below

Creating a new Python file will guide take you to a new page which will look something like the one given below



You can directly write your first Python program to print Hello World

print("Hello World !")

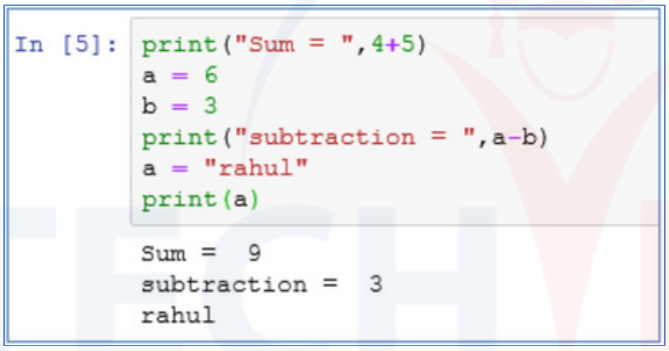


6. Congrats, you wrote your first Python program. Now let's do some basic mathematical operations.

Python is a different type of coding language, here you do not have to specify the data type of a variable. In Java, you have to declare the data type of a variable like int a, double b, etc.

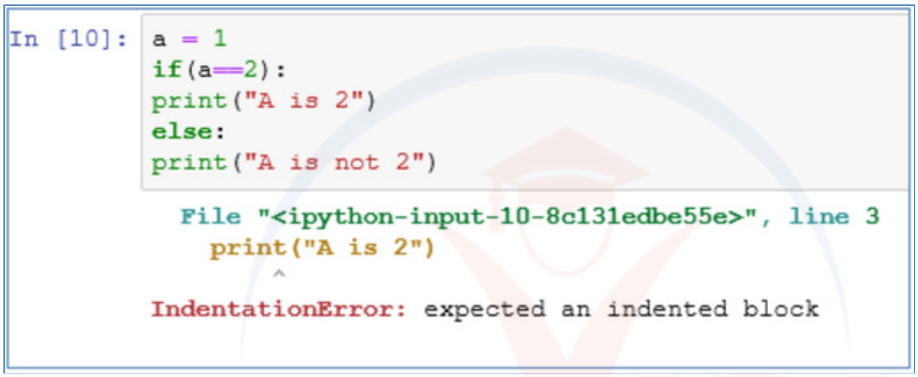
This declaration binds the variable. See the example below to understand how Python works without declaring data type

```
print("Sum = ",4+5)
a = 6
b = 3
print("subtraction = ",a-b)
a = "rahul"
print(a)
```



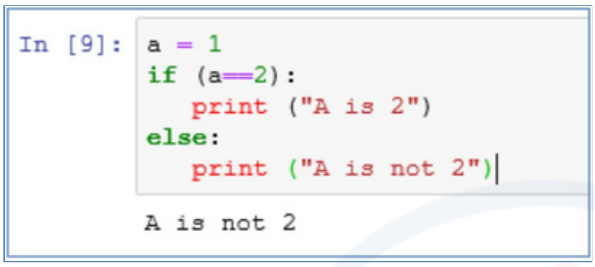
7. What is the most basic rule of writing code in Python?
   Indentation. You need to write your code in a perfect indented manner.
   Example of indentation is below
   a = 1
   if(a==2):
   print("A is 2")
   else:
   print("A is not 2")

   The above code is not indented, so it throws and error



Indented code given below

```
a = 1
if (a==2):
 print ("A is 2")
else:
 print ("A is not 2")
```



We are done with the basics. We know how to install and run Python. We also know how to write a basic program. Let's talk about some data types in Python.

8. What are the data types in Python?
   Python has five standard data types –

a. Number
b. String
c. List
d. Tuple
e. Dictionary

9. We will briefly talk about each of these data types. Let's start with Numbers.
   Number data type stores Numeric Values. There are 4 types of numeric values which we can store in Python

a. Integer – 10,20, 12345, -9876, etc.
b. Long – 0122L, 12432L, etc.
c. Float – 19.45, -43.54, 21+e18, etc.
d. Complex – 3e+2J, 3.14J, etc

10. What are strings?
    Strings are data types which can hold character values. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

11. Take few example of operations on String slicing

```
ex = "TheDataMonk "
#There is a white space at the end of the string
print (ex)
print(ex[0])
print(ex[-1])
print(ex[0:4])
print(ex*2)print(ex,"is a website")
```

Think of the output before looking at the answer given on the next page

```
In [18]: ex = "TheDataMonk "
         print (ex)
         print(ex[0])
         print(ex[-1])
         print(ex[0:4])
         print(ex*2)
         print(ex,"is a website")

         TheDataMonk
         T

         TheD
         TheDataMonk TheDataMonk
         TheDataMonk  is a website
```

12. Write a program to concatenate two strings

    alpha = “Data”
    beta = “Science”
    alpha_beta = alpha+” ”+beta
    print(alpha_beta)

13. What are Lists?

    Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]).

    The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1

14. Do some operations on Lists to understand it

```
listt = ['The','Data','Monk',132,2.4]
print(listt)
print(listt[-1],listt[0],listt[0:3])
print(listt*2)
```

The above operations are same as String, write your output before looking at the answers given on the next page

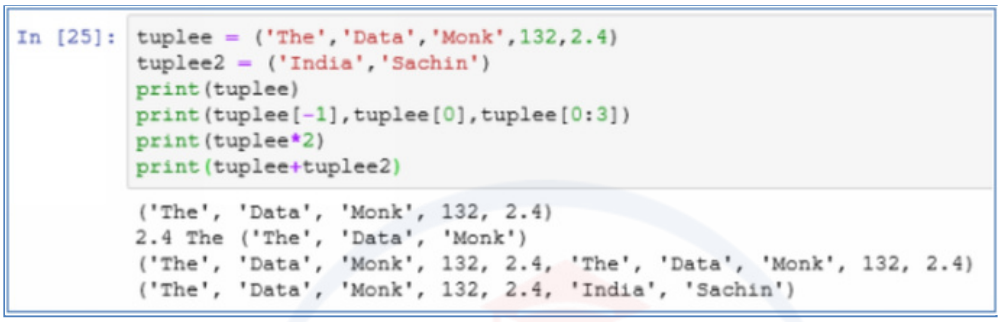
```
In [22]: listt = ['The','Data','Monk',132,2.4]
         print(listt)
         print(listt[-1],listt[0],listt[0:3])
         print(listt*2)

         ['The', 'Data', 'Monk', 132, 2.4]
         2.4 The ['The', 'Data', 'Monk']
         ['The', 'Data', 'Monk', 132, 2.4, 'The', 'Data', 'Monk', 132, 2.4]
```

15. What are tuples? Why do we actually need tuple when we already have List in place?

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

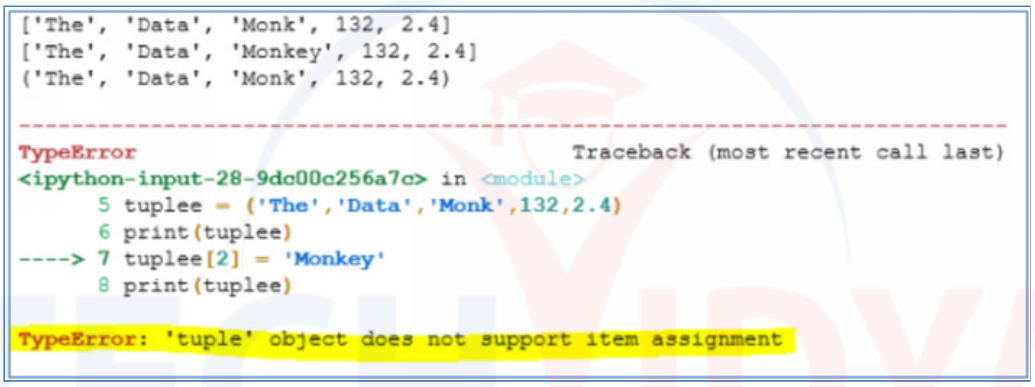
16. Create a tuple and do some operations on the same
    tuplee = ('The','Data','Monk',132,2.4)
    tuplee2 = ('India','Sachin')
    print(tuplee)
    print(tuplee[-1],tuplee[0],tuplee[0:3]
) print(tuplee*2)
    print(tuplee+tuplee2)



```
In [25]: tuplee = ('The','Data','Monk',132,2.4)
         tuplee2 = ('India','Sachin')
         print(tuplee)
         print(tuplee[-1],tuplee[0],tuplee[0:3])
         print(tuplee*2)
         print(tuplee+tuplee2)

         ('The', 'Data', 'Monk', 132, 2.4)
         2.4 The ('The', 'Data', 'Monk')
         ('The', 'Data', 'Monk', 132, 2.4, 'The', 'Data', 'Monk', 132, 2.4)
         ('The', 'Data', 'Monk', 132, 2.4, 'India', 'Sachin')
```

17. Show the difference between List and Tuple using an example
    The most important difference is that you can not change the content of a Tuple, but you can easily do it in case of List

```
listt = ['The','Data','Monk',132,2.4]
print(listt)
listt[2] = 'Monkey'
print(listt)
tuplee = ('The','Data','Monk',132,2.4)
print(tuplee)
tuplee[2] = 'Monkey'
print(tuplee)
```



TypeError – 'tuple' object does not support item assignment

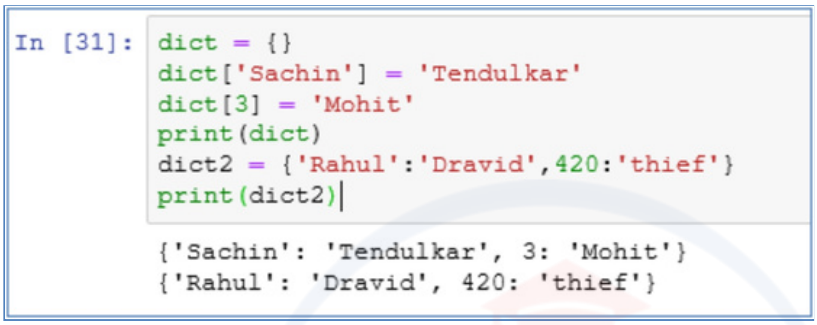
18. WAP to get the first and last element from the basic_list
    basic_list = [‘abc’,’def’,’ghi’]
    first_element = basic_list[0]
    third_element = basic_list[-1]

19. What are dictionaries in Python?
    Python's dictionaries are kind of hash table type. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

    Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

20. Explain some operations on Dictionary

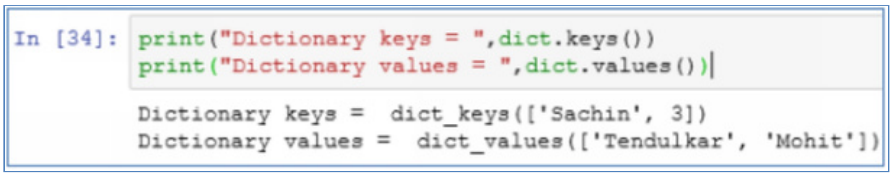
```
dict = {}
dict['Sachin'] = 'Tendulkar'
dict[3] = 'Mohit'
print(dict)
dict2  =  {'Rahul':'Dravid',420:'thief'}
print(dict2)
```



21. What are the keys and values in Dictionary?

Keys are more like indexes and Values are the actual values associated with the keys. In the above example we have a dictionary with keys Sachin and 3, and values Tendulkar and Mohit.

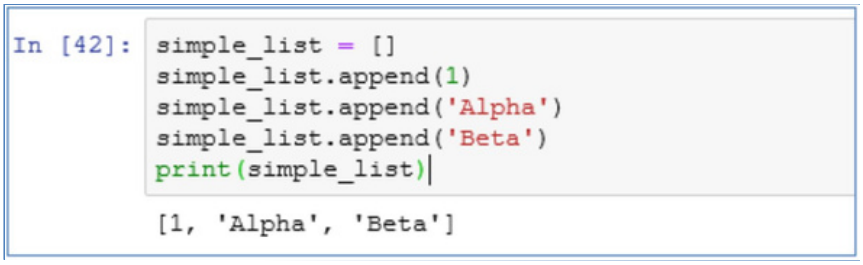
22. How to extract only keys and values from a dictionary?
keys() and values() functions are used to extract keys and values from a dictionary respectively
print("Dictionary keys = ",dict.keys())
print("Dictionary values = ",dict.values())



We have covered all the data types in brief. Till now you should be comfortable with creating basic list, tuple, dictionary, etc. and you can play around with these as well. Let's try some questions to test your skills

23. WAP where you first create an empty list and then add the elements. (Hint: Use append function)

simple_list = []
simple_list.append(1)
simple_list.append('Alpha')
simple_list.append('Beta')
print(simple_list)



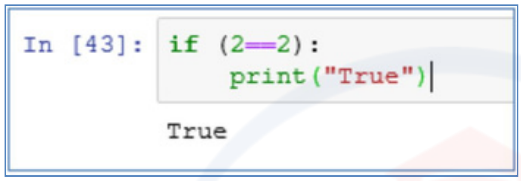
Let's talk about three important topics first i.e. Loop, Functions and Decision Making

24. What are decision making statements in Python?
You already know what we are going to discuss here. Decision making is a snippet of code which helps in anticipating a condition occurring while execution of the program

25. What are the types of decision making statements in Python?
There are three types of decision making concepts in Python

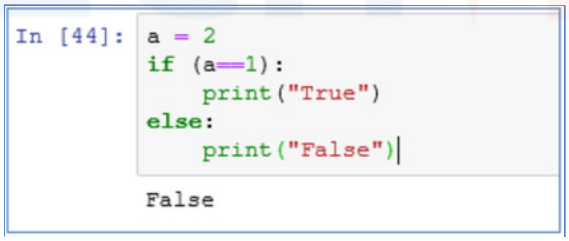
a. If condition

```
if (2==2):
 print("True")
```



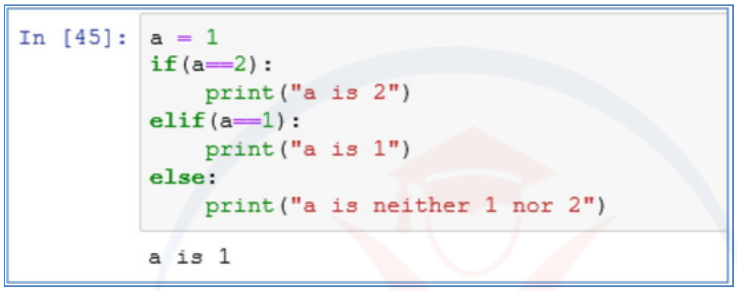
b. If..else condition

```
a = 2
if (a==1):
    print("True")
else:
 print("False")
```



c. Nested if and else

```
a = 1
if(a==2):
 print("a is 2")
elif(a==1):
 print("a is 1")
else:
 print("a is neither 1 nor 2")
```



26. Now, let's talk about loop. What are loops and its types?

A loop statement allows us to execute a statement or group of statements multiple times. There are three types of loops in Python:

a. While loop

b. For loop

c. Nested loop

27. Explain while loop with an example

A while loop is simple, it checks a condition and if the condition is true, it executes the block of the loop until the condition is true. It tests the condition before executing the loop body

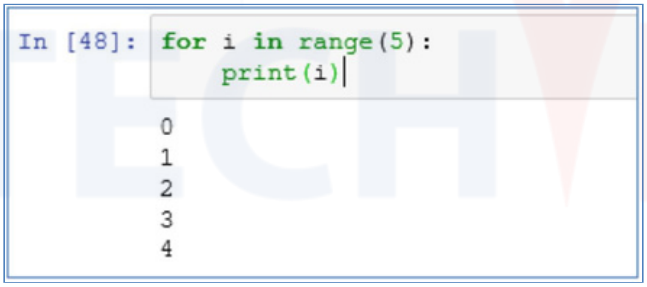
```
a = 5
while(a<9):
 print(a)
 a=a+1
```

28. What is for loop? Give an example

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

range() function takes value from 0. So, a range(3) will have value 0,1 and 2

```
for i in range(5):
 print(i)
```



29. Create a list using loop and fill the list with square of numbers from 1 to 10

```
square_list = []
for number in range(1,11):
 square_list.append(number**2)
```

30. Create a list of 6 elements and slice the first 3

```
basic_element = ['www','The','Data','Monk','dot','com']
basic_slice = basic_element[:3]
```

31. WAP a program to create a list with 3 elements and then copy the exact list in another list.

```
basic_element = ['www','The','Data','Monk','dot','com']
copy_list = basic_element[:]
```

32. WAP to show the use of if-elif-else

```
if salary>20000:
 print("Good Salary")
elif salary<20000
   print("Average   Salary")
else
 print("Salary is 20000")
```

33. WAP to create a dictionary and then iterate over it and print

```
lucky_number = {'Amit':4,'Rahul':6,'Nihar':8} for
name,number in lucky_number.items():
 print(name+'prefers'+str(number))
```

34. WAP to access all the keys in the dictionary.

```
lucky_number = {'Amit':4,'Rahul':6,'Nihar':8} for
name in lucky_number.keys():
 print(name)
```

35. WAP to access all the values in the dictionary.

```
lucky_number = {‘Amit’:4,’Rahul’:6,’Nihar’:8} for
number in lucky_number.values():
 print(str(number))
```

36. Let’s try your input skills, WAP to input a name and age, print it in a sentence

```
name = input("What is your name?")
age = input("What is your age?")
print("Hello "+name+". Your age is "+ age)
```

37. Now, WAP where user will write a text and the loop should stop only when it receives the word “exit”

```
msg = ''
while msg != 'exit':
msg = input("What's your message? ")
print(msg)
```

38. Now write a function where you pass your name and it should print something cool.

```
def cool(name):
  print("Something   cool"+name)
cool('Nitin')
```

39. Guess the Output of the following

```
def cool(name='Kamal'):
 print("You are cool Mr."+name)
cool('Nitin')
cool()
```

```
You are cool Mr.Nitin
You are cool Mr.Kamal
```

The function definition contains a default name i.e. Kamal which will be used in case the function is called from the program but no parameter is passed during the call

40. Write a function which returns sum of two numbers.

```
def add_number(x,y):
 return x+y
sum                    =
add_number(10,20)
print(sum)
```

41. How to read a file and store the lines in your variable.

```
filename = 'abc.txt'
with open(filename) as file_object:
lines = file_object.readlines()
for line in lines:
print(line)
```

42. WAP to write to a file

```
filename = “new.txt”
with open(filename,’w’) as file_object:
file_object.write(“I love India”)
```

43. WAP to append something to a file.

```
filename = “appenddd.txt”
with open(filename,’a’) as file_object:
file_object.write(“I love my India”)
```

44. Exceptions helps us to be prepared for an error which might occur in the program. WAP to showcase how an exception works.

```
x = “What is your age?”
inp = input(x)

try:
inp = int(inp)
except ValueError:
 print(“Sorry, Please Try again latter”)
else:
 print(“That’s a beautiful age “)
```

45. Try the following operations using List
TDM = [‘The’,’Data’,’Monk]
a. Print the last object of the list
print(TDM[-1])
b. Change the last element to Monkey
TDM[-1] = ‘Monkey
c. Remove Monkey from the list
del TDM[-1]
d. GET Monk back to the list
TDM.append(‘Monk’)

46. What is the pop() function? Explain using an example.

```
TDM.append(‘Monkey’)
x = TDM.pop()
print(“The most recent element”+x)
```

47. Print all the Prime numbers less than 20

```
i = 2
while(i < 20):
 j = 2
 while(j <= (i/j)):
 if not(i%j):
 break
 j = j + 1
 if (j > i/j) :
 print (i," is a prime number")  i
= i + 1
```

48. It is very important to know some in-built functions. It will make your task easier. Write the functions which you can apply on a list
    a. List(ex) – It converts a tuple into list
    b. Min(ex) – It gives the minimum value of a list
    c. Max(ex) – It gives the maximum value of a list
    d. Len(ex) – It gives the length of the list
    e. Cmp(ex1,ex2) – It compares two lists

49. Like functions, there are methods also which makes our task easier. Write few methods which are applied on lists.

a. List.append(obj) – To append object to list
b. List.count(obj) – How many times object occurs in list
c. List.extend(seq) – Append the content of a sequence to list
d. List.index(obj) – Returns the lowest index in the list that object appears
e. List.pop(obj=list[-1) – Removes and returns last object of the list

50. Sorting is process to arrange something in numerical or alphabetical order. What is the function to sort a list?

    Permanently sorting a list
    TDM.sort()

    Sort the list in reverse alphabetical order
    TDM.sort(reverse = TRUE)

51. If you want to reverse a list temporarily then which method to use?

    sorted() method is used to temporarily reverse a

    list print(sorted(TDM, reverse = TRUE))

52. How to reverse the order of the list?

    TDM.reverse()
    It will change the index of the elements in the reverse order

53. WAP to make a list of number from 0 to 99.

    list_number = rlist(Range(100))

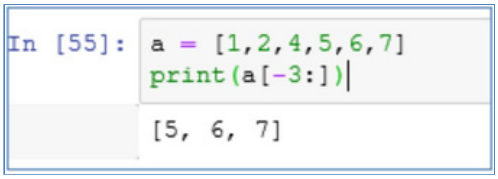
54. WAP to sum all the elements of the list

    number = [1,2,3,4,5,6,7]
    print("Sum is "+str(sum(number)))

    Sum is 28

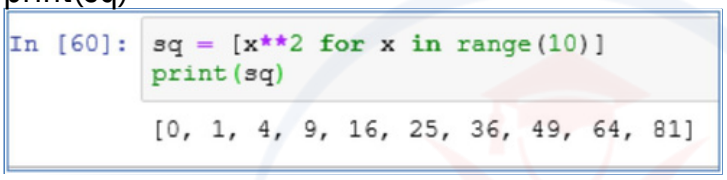
55. Get the last three elements of the list.

```
a = [1,2,4,5,6,7]
print(a[-3:])
```



56. Write a function to print the square of all numbers from 0 to 11

```
sq = [x**2 for x in range(10)]
print(sq)
```



57. Convert all the elements in the upper case using a function.

```
names = ['The', 'Data', 'Monk', 'com', 'www']
upper_names = [name.upper() for name in names]
print(upper_names)
```

['THE', 'DATA', 'MONK', 'COM', 'WWW']

58. Try to solve the above problem using a loop

```
names = ['The', 'Data', 'Monk', 'com', 'www']
upper_names = []
for name in names:
 upper_names.append(name.upper())
print(upper_names)
```

```
['THE', 'DATA', 'MONK', 'COM', 'WWW']
```

59. How does a mutable list works?

```
list_example = ['Amit','Sumit','Rahul']
print(list_example)
list_example[1] = 'Kamal'
print(list_example)
```

```
['Amit', 'Sumit', 'Rahul']
['Amit', 'Kamal', 'Rahul']
```

60. How does a immutable tuple rejects the idea of alteration?

```
tuple_example = ('Amit','Sumit','Rahul')
print(tuple_example)
tiple_example(1) = ('Kamal')
print(tuple_example)

File  "<ipython-input-47-4a338933da64>",  line
3  tiple_example(1) = ('Kamal')
SyntaxError: can't assign to function call
```

61. How to add a key-value to a dictionary?

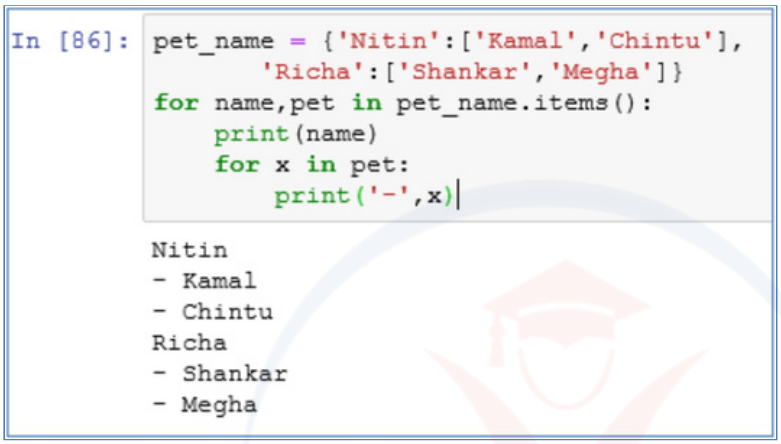
```
dic_ex   =   {'name':'Nitin','last_name':'Kamal'}
dic_ex['x']=0
dic_ex['y']=100
```

62. How to loop through all the key-value pair of a dictionary?

```
stars = {'Nitin':'SRK','Pappu':'Salman','Kamal':'Amir Khan'}
for name, star in stars.items():
 print("Name of employee='+name+'Fav. Baollywood Star='+star)
```

63. Write a code to put a list into dictionary

```
pet_name  = {'Nitin':['Kamal','Chintu'],
 'Richa':['Shankar','Megha']}
for name,pet in pet_name.items():
 print(name)
 for x in pet:
 print('-',x)
```



```
In [86]: pet_name = {'Nitin':['Kamal','Chintu'],
                    'Richa':['Shankar','Megha']}
         for name,pet in pet_name.items():
             print(name)
             for x in pet:
                 print('-',x)

         Nitin
         - Kamal
         - Chintu
         Richa
         - Shankar
         - Megha
```

64. Guess the output of the below snippet of code.

```
cricketer   =   [‘Sachin’,’Dhoni’,’Rahul’]
new = ‘Virat’

if cric not in cricketer:
 print(‘The cricketer does not exist’)
```

Output – The cricketer does not exist

65. You are doing good, you already know how to take input from the user, by using the input() function, now write a program which will ask for the best cricketer and will quit once you get to the right answer.

```
prompt = "\nWho is the best cricketer in the World? "

while True:
 city = input(prompt)
 if city == 'Sachin':
 print('Sachin is awesomeeeee')
 break
 else:
 print("Try again")
```

Output

```
Who is the best cricketer in the World? Rahul
Try again

Who is the best cricketer in the World? Nitin
Try again

Who is the best cricketer in the World? Sachin
Sachin is awesomeeeee
```

66. Use positional argument to create afunction

```
def describe_pet(animal, name):
    print("\nI have a " + animal + ".")
    print("Its name is " + name + ".")

describe_pet('hamster', 'harry')
describe_pet('dog', 'willie')
```

67. Use Keyword argument to create a function

```
def describe_pet(animal, name):
print("\nI have a " + animal + ".")
print("Its name is " + name + ".")

describe_pet(animal='hamster',
name='harry')      describe_pet(name='willie',
animal='dog')
```

68. How to pass a list to a function?

```
def game_name(name):
for x in game_name
print(x)

example   =   [‘Cricket’,’Football’,’TT’]
game_name(example)
```

69. What is *args and **kwargs ? When are these used?

Most of the times when we create a function, we need to specify how many arguments are going to be passed in the function.
*args is used when we don't know how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple to the function.

We use **kwargs in function definitions to pass a keyworded variable-length argument list.
A keyword argument is where you provide a name to the variable as you pass it into the function.

70. When you don't know how many arguments will be passed to a function, then you need to pass a variable number of arguments. Show by an example.

```
def pizza(size, *toppings):
print("\nMaking a " + size + " pizza.")
print("Toppings:")
for topping in toppings:
print("- " + topping)

# Make three pizzas with different toppings.

make_pizza('small', 'pepperoni')
make_pizza('large', 'bacon bits', 'pineapple')
make_pizza('medium', 'mushrooms', 'peppers', 'onions', 'extra cheese')
```

71. How to collect arbitrary number of keyword arguments in a function? Show by an example.

```
def build_profile(first, last, **user_info):
profile = {'first': first, 'last': last}
for key, value in user_info.items():
profile[key] = value
 return profile

user_0 = build_profile('albert', 'einstein', location='princeton')
user_1 = build_profile('marie', 'curie', location='paris', field='chemistry')
print(user_0)
print(user_1)
```

72. How to store a function in a module?
Name the file as pizza.py

```
def make_pizza(size, *toppings):
print("\nMaking a " + size + " pizza.")
print("Toppings:")
for topping in toppings:
print("- " + topping)
```

73. What is the function of describe() method?

Describe method is implemented by dataset.describe() and it gives us the following results
Count
Mean
Standard Deviation
Minimum
25 percentile
50 percentile(Median)
75 percentile and maximum for each of the numerical columns in the data set

74. How to remove duplicates from a list?

We can use the set() data type to remove duplicates from the

```
list itemList = ['1', '2', '3', '3', '6', '4', '5', '6']
new_set = set(itemList)
print(new_set)
```

{'3', '2', '6', '5', '4', '1'}

75. How to remove duplicates from a list without using set data type?

```
items = ['1','2','3','4','4','5','5']
new_list = []
(new_list.append(item) for item in items if item not in new_list]
print(new_list)
```

76. What are the ways to remove data from sets?

We can use the following commands to remove data from set:-
a. discard() – Remove an element from the set of values
b. pop() – It removes and returns an arbitrary element from the set

77. What is a lambda function ?

A lambda function is a small function that can take any number of arguments but can have only one expression.
Example:-

```
x = lambda a:a+10
print(x(10))
```

20

78. Why do we need a lambda function?

Lambda function is like any other function which you can define with the word “def”, but the power of lambda function comes into picture when you have to use a simple function inside a standard function. Example below:-

```
def new_function(x):
 return lambda a : a*x
```

79. How to split a dataset into train and test in python?

Splitting a dataset into train and test is one of the initial stage of most of the machine learning models. Following is how you can split dataset in python:-

```
from sklearn.model_selection import train_test_split

X_train,X_test,Y_train,Y_test =
train_test_split(dataframe_name,target_variable,test_size=0.3,
random_state=42)
```

dataframe_name = the complete dataset as a panda dataframe
target_variable = the name of the target variable
test_size = 0.3 denotes 70-30 split of the dataset in train and test
random_state = 42, Look for the explanation in the next question

80. Use the break statement in a for loop, so that it can stop the iteration as soon as it finds the word “cricket” in the list

```
x  =  [“Football”,”Basketball”,”cricket”,”Ludo”]
for i in x:
print(x)
if i == “cricket”:
break
```

81. Use while loop to print from 1 to 50, but skip at 5,10 and 15.

```
x=0
while(x<51)
x=x+1
    if(x==5)|(x==10)|(x==15):
 continue
 print(x)
```

82. How to join tables in python?

We use the merge function from pandas library. Following is the syntax for the same

Merge the two table
Table 1 State_population
State_name, population

Table 2 State_Capital
Name_State. Capital

m = pd.merge(left=State_population, right = State_Capital, on=None, left_on='State_name',right_on='Name_State')

We use on when the column names of the key column on which you want to merge the data are different. Left_on and right_on are used to specify the column names in the two tables

83. What is globbing? How to use glob function to import all the csv files

Globbing is a way to match patterns to find files in Python by using wild cards such as * and ?
import glob
csv_files = glob.glob('*.csv')

84. Explain split(), sub(), subn() methods of "re" module in Python.
To modify the strings, Python's "re" module is providing 3 methods. They are:
split() – uses a regex pattern to "split" a given string into a list.
sub() – finds all substrings where the regex pattern matches and then replace them with a different string
subn() – it is similar to sub() and also returns the new string along with the no. of replacements

85. What will be the output of the print(str*3) if str="TheDataMonk"?

It will print TheDataMonk three times

```
str='TheDataMonk'
print(str*3)
```

TheDataMonkTheDataMonkTheDataMonk

86. As a Data Scientist, you will come across multiple instance where you need to remove the trailing and leading white spaces. How to strip the string?

```
str = "  TheDataMonk  "
print(str)
print(str.strip())
```

Output

```
 TheDataMonk
TheDataMonk
```

87. What is the output of [1, 2, 3] + [4, 5, 6]?

[1, 2, 3, 4, 5, 6]

88. Write a program in Python to reverse a string without using inbuilt function reverse string?

```
def string_reverse(str1):
rev_str = ''
index = len(str1) #defining index as length of string.
while(index>0):
rev_str = rev_str + str1[index-1]
index = index-1
return(rev_str)
print(string_reverse('1tniop'))
```

89. How to convert a tuple into a list?

```
tup = (‘the’,’Data’,’Monk’)
list_example = list(tup)
print(list_example)
```

```
[‘the’,’Data’,’Monk’]
```

90. How to concatenate two tuples?

Simply you can use the plus(+) operator

```
a = (1,2,3)
b = ('a','f','g')
c = a+b
print(c)
print(type(c))
```

```
(1, 2, 3, 'a', 'f', 'g')
<class 'tuple'>
```

91. What is a global and local variable?

The understanding of global and local variable is very important to use the capability of a programming language.
In layman terms the global variable can be used anywhere in the complete program, whereas a local variable is used only in the vicinity of its declaration. Variables that are defined inside a function has a local scope. This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope.

92. Explain global and local variable using an example

```
summ = 0#Global Variable
def sum(a,b):
 summ=a+b#Local variable
   print("Local    Variable",summ)
 return sum
sum(40,50)
print("global variable",summ)
```

```
Local Variable 90
global variable 0
```

93. Get the output of the following:

```
print(re.match('kam', 'kamal'))
      print(re.match('kam', 'nitin kamal'))
print(re.search('kam','kamal'))
      print(re.search('kam','nitin kamal'))
```

```
<re.Match object; span=(0, 3), match='kam'>
None
<re.Match object; span=(0, 3), match='kam'>
<re.Match object; span=(6, 9), match='kam'>
```

94. What is the difference between (a-z) and [A-Z]?

This is a very important concept, when you specify (a-z), it will only match the string “a-z”. But when you specify [A-Z] then it covers all the alphabet between upper case A and Z

95. Create a calculator to multiply two numbers in Python.

```
print('Please enter two numbers')
print('Enter 0 to exit')

while True:
x = input("\n First Number")
 if x == 0:
 break
 y = input("\n Second Number")
 if y==0:
 break
result = x*y
print("The product of the two numbers is" + result)
```

96. Write the code to make a simple line graph.

```
import matplotlib.pyplot as plt
x_values = [0, 1, 2, 3, 4, 5]
squares = [0, 1, 4, 9, 16, 25]
plt.plot(x_values, squares)
lt.show()
```

97. Now try creating a scatter plot for some sample data.

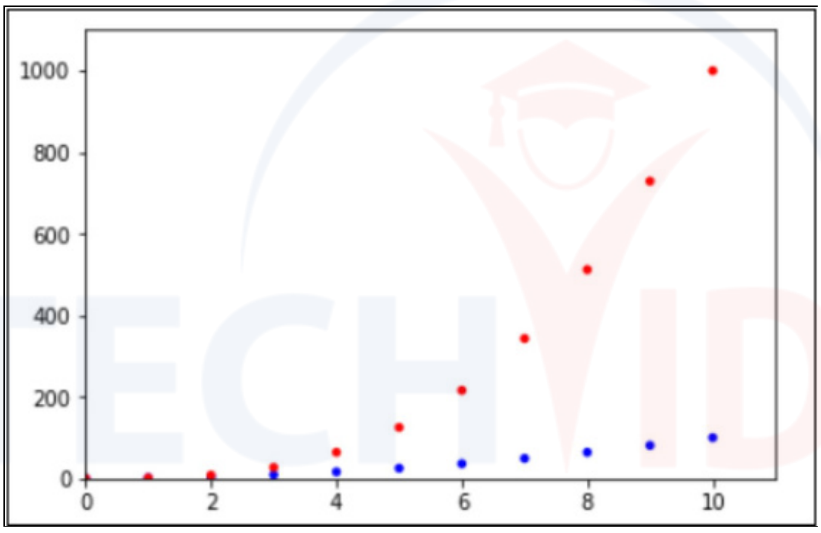
```
import matplotlib.pyplot as plt
x_values = list(range(1000))
squares = [x**2 for x in x_values]
plt.scatter(x_values, squares, s=10)
plt.show()
```

98. How to add titles and labels in a plot, also add scaling axes?

```
import matplotlib.pyplot as plt
x_val = list(range(1000))
sq = [x**2 for x in x_val]

plt.title("Square Numbers")
plt.xlabel("Value")
plt.ylabel("Square of Value")
plt.tick_params(axis='both', which='major', labelsize=14)
plt.axis([0, 1100, 0, 1100000])
plt.show()
]
```

99. How to plot two sets of data?

```
import matplotlib.pyplot as plt
x_values = list(range(11))
squares = [x**2 for x in x_values]
cubes = [x**3 for x in x_values]
plt.scatter(x_values, squares, c='blue',
 edgecolor='none', s=20)
plt.scatter(x_values, cubes, c='red',
 edgecolor='none', s=20)
plt.axis([0, 11, 0, 1100])
plt.show()
```



100.Write a program to calculate number of upper case letters and number of lower case letters?

```
Test on String: ''Tutorials POINT''
def string_test(s):
a = { ''Lower_Case'':0 , ''Upper_Case'':0} #initial count of lower and upper
for ch in s: #for loop
if(ch.islower()): #if-elif-else condition
 a[''Lower_Case''] = a[''Lower_Case''] + 1
 elif(ch.isupper()):
 a[''Upper_Case''] = a [''Upper_Case''] + 1
 else:
 pass

print(''String in testing is: '',s) #printing the statements.
print(''Number of Lower Case characters in String: '',a[''Lower_Case''])
print(''Number of Upper Case characters in String: '',a[''Upper_Case''])
```

101.When to use list, set or dictionaries in Python?

list keeps order, dict and set don't

: when you care about order,
therefore, you must use list (if your choice of containers is limited to these three, of course;-).
dict associates with each key a value, while list and set just contain values: very different use cases, obviously.
set requires items to be hashable, list doesn't: if you have non-hashable items, therefore, you cannot use set and must instead use list.
Set forbids duplicates, list does not.

102. Write a regular expression to split a paragraph every time it finds an exclamation mark.

```
import re
exclamation = r”[!]”
strr = "Data Science comprises of innumerable topics! The aim of this 100 Days series is to get you started assuming ! that you have no prior! knowledge of any of these topics. "
excla = re.split(exclamation,strr)
print(excla)
```

['Data Science comprises of innumerable topics', ' The aim of this 100 Days series is to get you started assuming ', ' that you have no prior', ' knowledge of any of these topics. ']

103. What are classes ?

Python is an Object Oriented Programming Language. In Python, classes are the building blocks of Object Oriented Programming. In a simpler terms, classes is more like any real-world thing, we use classes to create specific instances of things like game, car, etc.

104. Create a class by the name of car and put the name of company, model and year of manufacture as the attributes of the same. Also give a fuel_capacity and fuel_level in the class.

```
class Car():
 def __init__(self, make, model, year):
self.make = make
self.model = model
self.year = year
                # Fuel capacity and level in gallons.
                self.fuel_capacity = 15
self.fuel_level = 0
def fill_tank(self):
self.fuel_level = self.fuel_capacity
print("Fuel tank is full.")
def drive(self):
 """Simulate driving."""
print("The car is moving.")
```

105. How to create and access a class?

Creating an object from a class

my_car = Car('BMW','Z12'.2018)

Accessing attribute values
print(my_car.make)
print(my_car.model)
print(my_car.year)

106. What is class inheritance? Explain with an example

If the class you're writing is a specialized version of another class, you can use inheritance. When one class inherits from another, it automatically takes on all the attributes and methods of the parent class. The child class is free to introduce new attributes and methods, and override

attributes and methods of the parent class. To inherit from another class include the name of the parent class in parentheses when defining the new class.

```
class ElectricCar(Car):
 """A simple model of an electric car."""
 def __init__(self, make, model, year):
 """Initialize an electric car."""
 super().__init__(make, model, year)
 # Attributes specific to electric cars.
 # Battery capacity in kWh.
 self.battery_size = 70
 # Charge level in %.
```

107. What's the Difference between a For Loop and a While Loop?
In Python, a loop iterates over popular data types (like dictionaries, lists, or strings) while the condition is true. This means that the program control will pass to the line immediately following the loop whenever the condition is false. In this scenario, it's not a question of preference, but a question of what your data structures are.

For Loop

In Python (and in almost any other programming language), For Loop is the most common type of loop. For Loop is often leveraged to iterate through the elements of an array.

For example:

For i=0, N_Elements (array) do...

For Loop can also be used to perform a fixed number of iterations and iterate by a given (positive or even negative) increment. It's important to note that by default, the increment will always be one.

While Loop

While Loop can be used in Python to perform an indefinite number of iterations as long as the condition remains true.

For example:

While (condition) do...

When using the While Loop, you have to explicitly specify a counter to keep track of how many times the loop was executed. However, While Loop can't define its own variable. Instead, it has to be previously defined and will continue to exist even after you exit the loop.

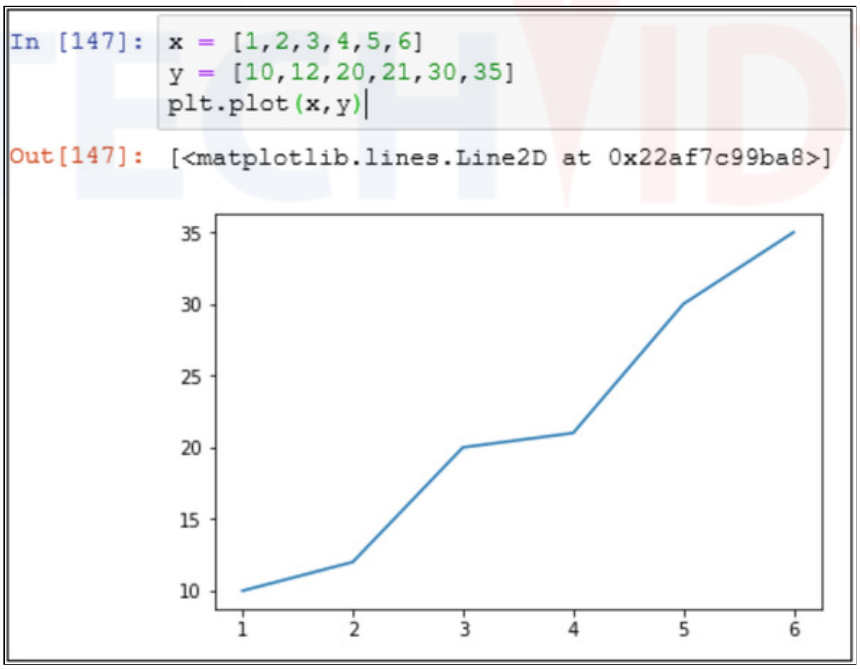
When compared to For Loop, While Loop is inefficient because it's much slower. This can be attributed to the fact that it checks the condition after each iteration. However, if you need to perform one or more conditional checks in a For Loop, you will want to consider using While Loop instead (as these checks won't be required).

108. In Python, How is Memory Managed?
In Python, memory is managed in a private heap space. This means that all the objects and data structures will be located in a private heap. However, the programmer won't be allowed to access this heap. Instead, the Python interpreter will handle it. At the same time, the core API will enable access to some Python tools for the programmer to start coding.

The memory manager will allocate the heap space for the Python objects while the inbuilt garbage collector will recycle all the memory that's not being used to boost available heap space.

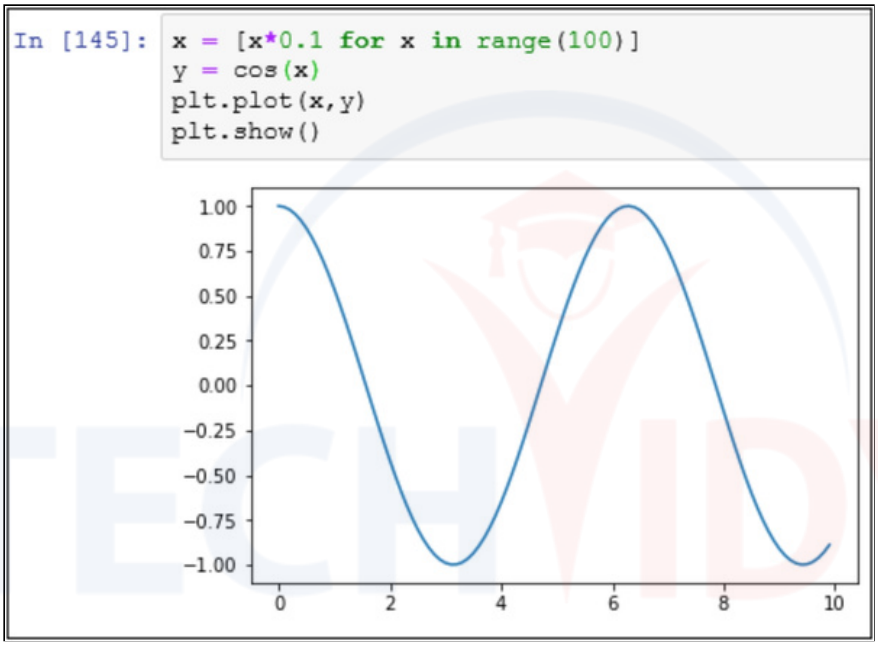
109.Make a line chart
```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6]
y = [10,12,20,21,30,35]
plt.plot(x,y)
```

Here is what you will get



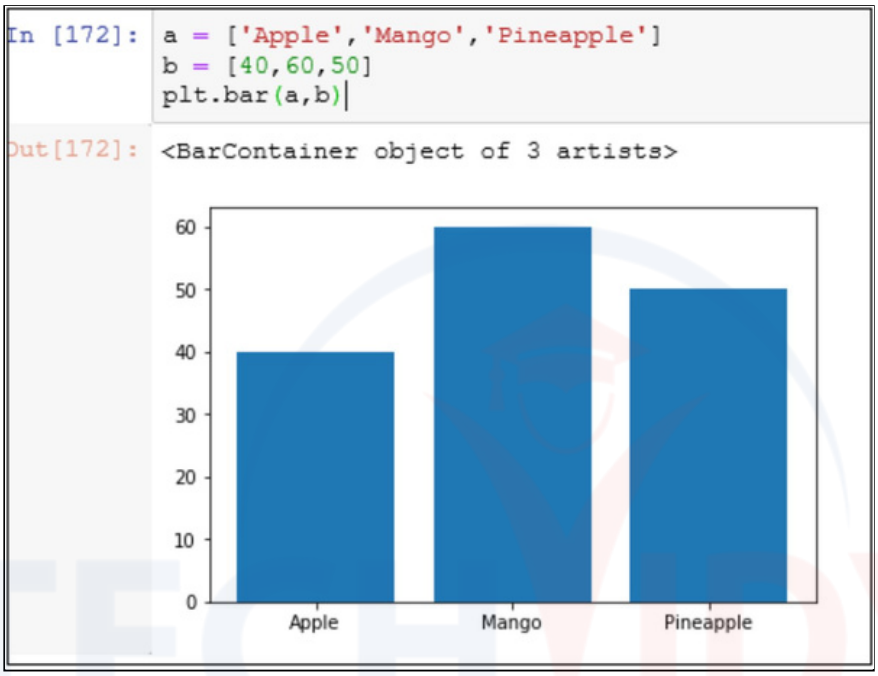
110. Plot a sin graph using line plot

```
import matplotlib.pyplot as plt
from numpy import cos

x = [x*0.01 for x in range(100)]
y = cos(x)
plt.plot(x,y)
plt.show()
```



111. Plot a bar chart

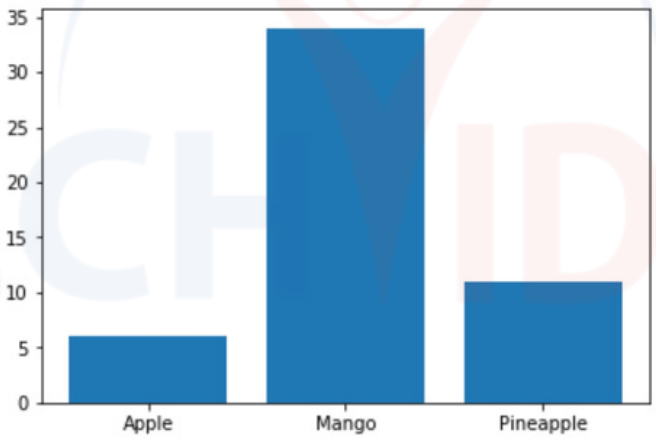
```
import matplotlib.pyplot as plt
a  =  ['Apple','Mango','Pineapple']
b = [40,60,50]
plt.bar(a,b)
```



112. Use random values between 1 and 100 to create the same graph.

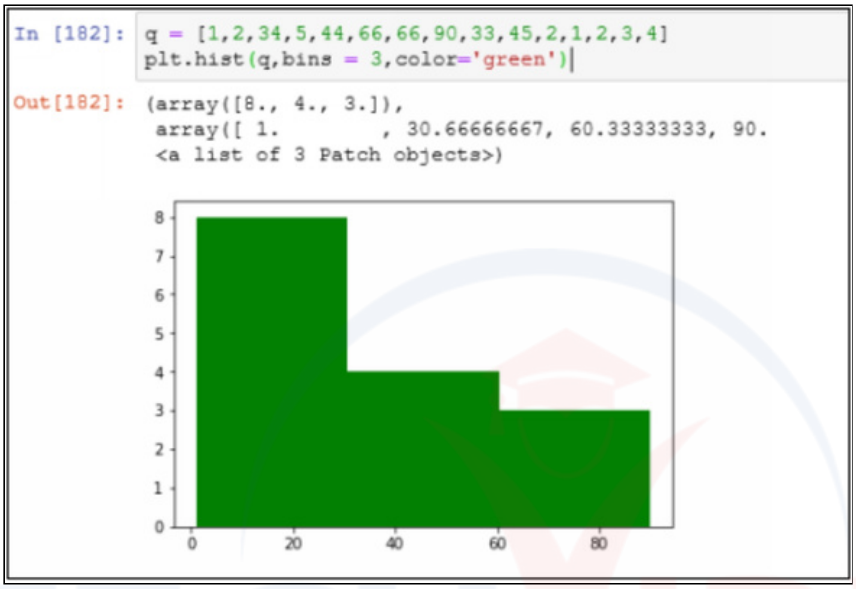
import matplotlib.pyplot as plt
from random import seed
from random import randint
seed(123)
x = ['Apple','Mango','Pineapple']
y = [randint(0,100),randint(0,100),randint(0,100)]
plt.bar(x,y)

```
In [171]: from random import seed
          from random import randint
          seed(123)
          x = ['Apple','Mango','Pineapple']
          y = [randint(0,100),randint(0,100),randint(0,100)]
          plt.bar(x,y)
Out[171]: <BarContainer object of 3 artists>
```



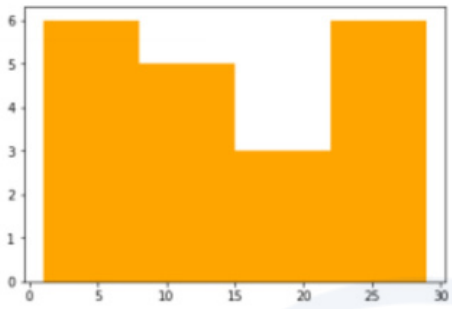
113. A simple histogram plot

q = [1,2,34,5,44,66,66,90,33,45,2,1,2,3,4]
plt.hist(q,bins = 3,color='green')



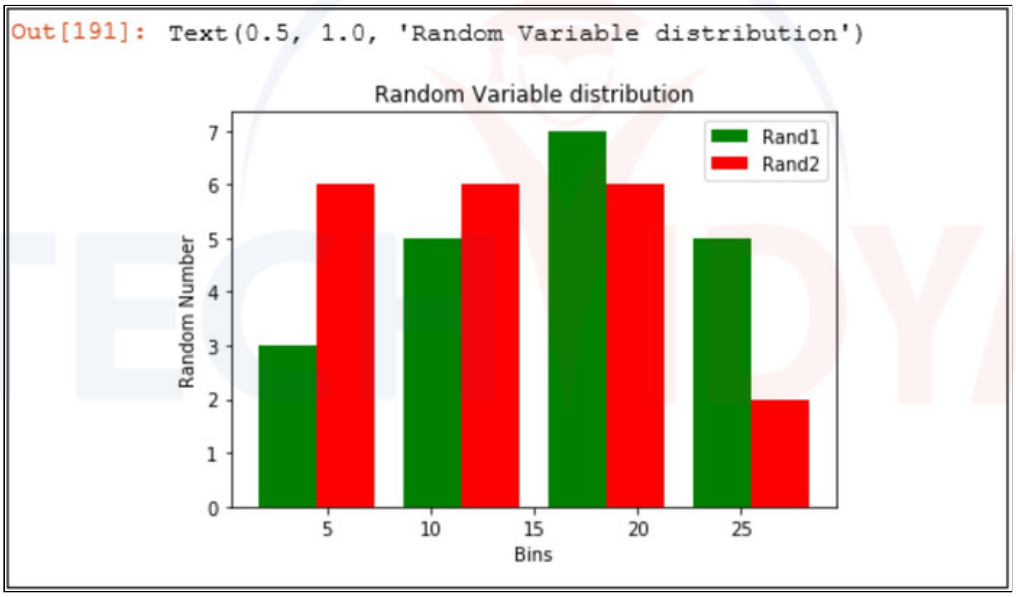
import random
my_rand =
random.sample(range(1,30),20)
print(my_rand)
print(type(my_rand))
plt.hist(my_rand,bins=4,color='orange')

```
In [183]: import random
          my_rand = random.sample(range(1,30),20)
          print(my_rand)
          print(type(my_rand))
          plt.hist(my_rand,bins=4,color='orange')

          [25, 14, 9, 4, 2, 13, 18, 23, 11, 21, 29, 6, 5, 20, 27, 22, 12, 26, 3, 1]
          <class 'list'>

Out[183]: (array([6., 5., 3., 6.]),
           array([ 1.,  8., 15., 22., 29.]),
           <a list of 4 Patch objects>)
```



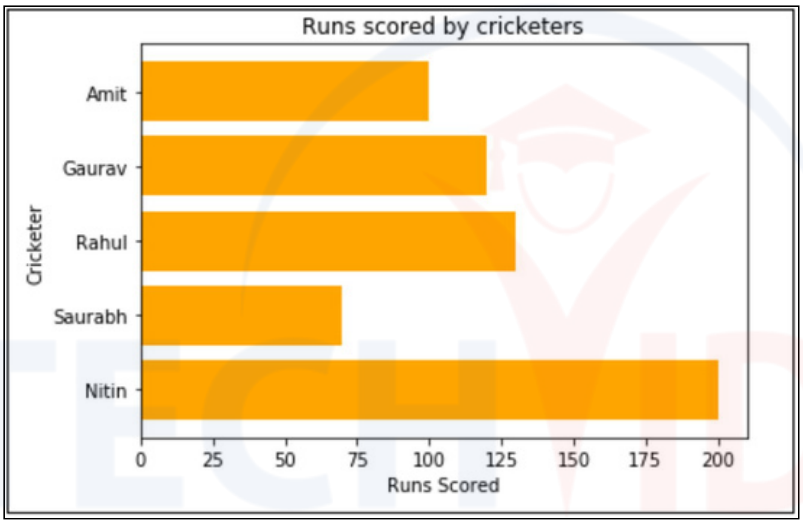
115. In Histogram also you can add more than one data points to make parallel bars.

```
import random
my_rand = random.sample(range(1,30),20)
my_rand2 = random.sample(range(1,25),20)
print(my_rand)
print(type(my_rand))
plt.hist([my_rand,my_rand2],bins=4,color=['green','red'])
legend = ['Rand1','Rand2']
plt.legend(legend)
plt.xlabel("Bins")
plt.ylabel("Random Number")
plt.title("Random Variable distribution")
```



116. Horizontal Histogram

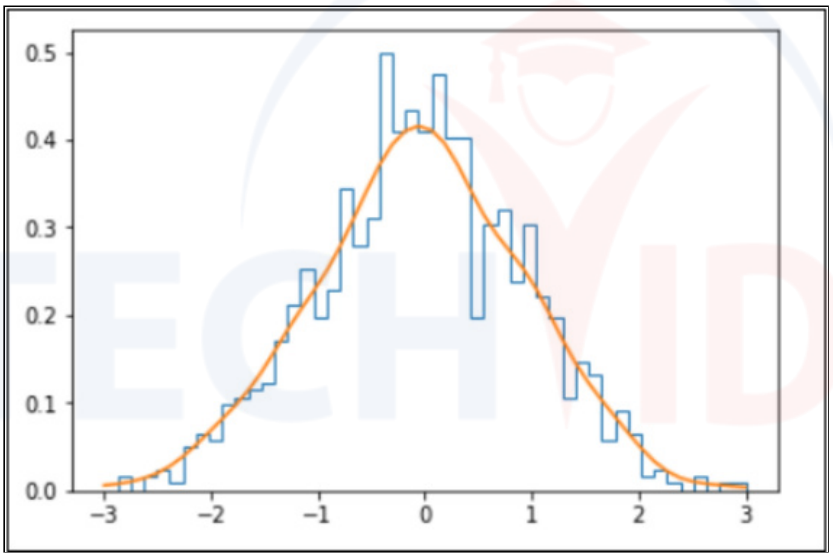
```
import numpy as np
import matplotlib.pyplot as plt
name = ['Nitin','Saurabh','Rahul','Gaurav','Amit']
run = [200,70,130,120,100]
plt.barh(name,run,color='orange')
plt.xlabel("Runs Scored")
plt.ylabel("Cricketer")
plt.title("Runs scored by cricketers")
plt.show()
```



117. Line Histogram

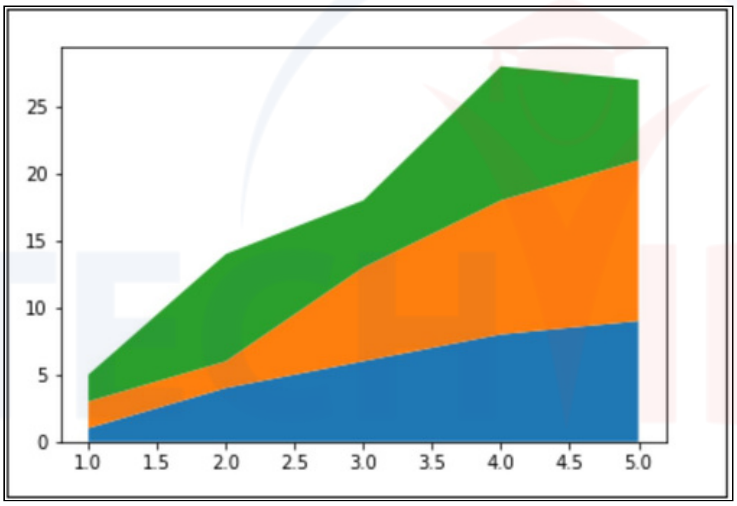
Now let's create a line histogram with some random data

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
noise = np.random.normal(0, 1, (1000, ))
density = stats.gaussian_kde(noise)
n, x, _ = plt.hist(noise, bins=np.linspace(-3, 3, 50),histtype=u'step',
density=True)
plt.plot(x, density(x))
plt.show()
```



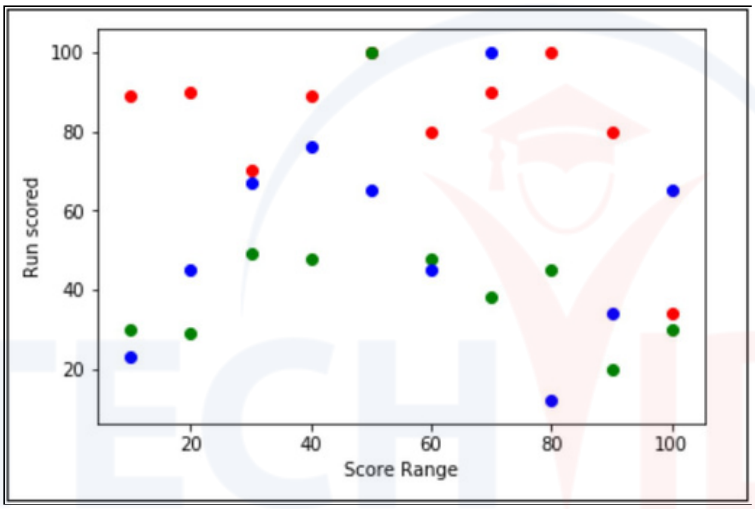
118. Let’s create a basic area chart with some dummy data

```
Import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Data
x=range(1,6)
y=[ [1,4,6,8,9], [2,2,7,10,12], [2,8,5,10,6] ]

# Plot
plt.stackplot(x,y, labels=['A','B','C'])
plt.legend(loc='upper left')
plt.show()
```



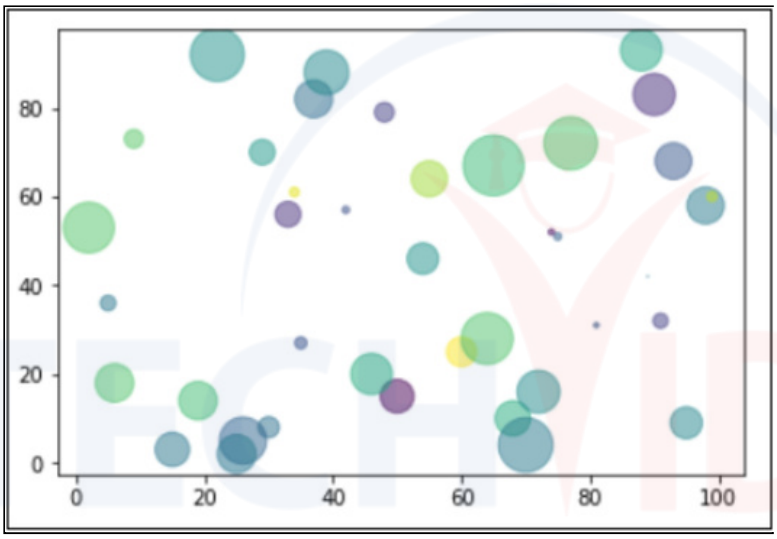
119. Scatter plot

```
sachin = [89, 90, 70, 89, 100, 80, 90, 100, 80, 34]
kohli = [30, 29, 49, 48, 100, 48, 38, 45, 20, 30]
dhoni = [23,45,67,76,65,45,100,12,34,65]
run = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(run, sachin, color='red')
plt.scatter(run, kohli, color='green')
plt.scatter(run,dhoni,color='blue')
plt.xlabel('Score Range')
plt.ylabel('Run scored')
plt.show()
```



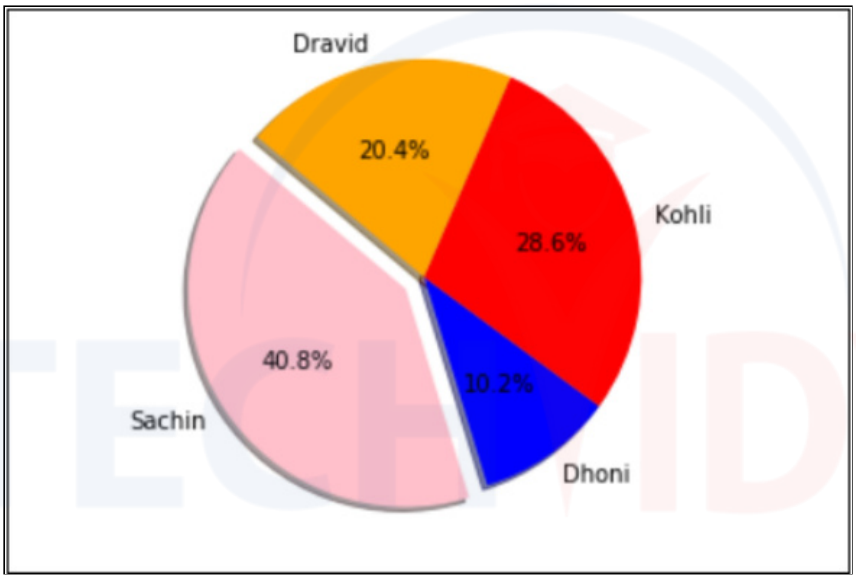
120. Below is one more scatter plot where you give weighted area and the size of the circle will be on the basis of the circle

```
import numpy as np
np.random.seed(123)
x = random.sample(range(1,100),40)
y = random.sample(range(1,100),40)
colors = np.random.rand(N)
area = (30*np.random.rand(N))**2
plt.scatter(x,y,s=area,c=colors,alpha=0.5)
plt.show()
```



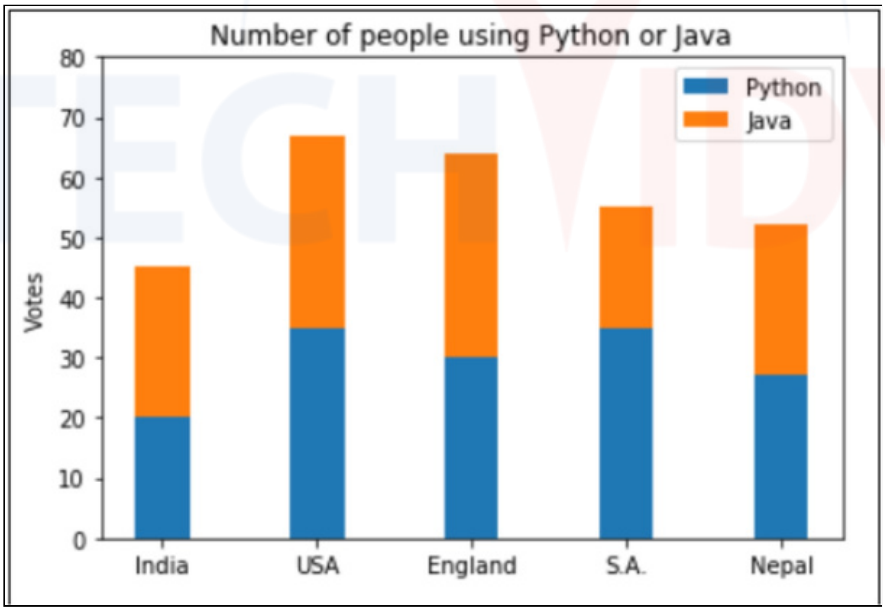
121. Create a pie chart for the number of centuries scored by Sachin, Dhoni, Dravid, and Kohli.

```
labels = 'Sachin','Dhoni','Kohli','Dravid'
size = [100,25,70,50]
colors = ['pink','blue','red','orange']
explode = (0.1,0,0,0)
plt.pie(size,explode=explode,labels=labels,colors=colors,autopct='%1.1f%
%',shadow=True,startangle=140)
plt.axis('equal')
plt.show()
```



122. Create a stacked chart to demonstrate the number of people voting for either Python or Java in 5 countries, namely, India, USA, England, S.A., Nepal

```
import numpy as np
import matplotlib.pyplot as plt
Python = (20, 35, 30, 35, 27)
Java = (25, 32, 34, 20, 25)
width = 0.35 # the width of the bars: can also be len(x) sequence p1
= plt.bar(ind, Python, width)
p2 = plt.bar(ind, Java, width,bottom=Python)
plt.ylabel('Votes')
plt.title('Number of people using Python or Java')
plt.xticks(ind, ('India', 'USA', 'England', 'S.A.', 'Nepal'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Python', 'Java'))
plt.show()
```



123. When to use which graph?

One of the most important thing is to understand when to use which graph and a list of all the graphs in your knowledge.

There are four types of information which we can display using any plot:-
1. Distribution
2. Comparison
3. Relationship
4. Composition

1. Distributionshows how diversely the data is distributed in your data set. How many people are from which state of the country?

a Histogram – If you have few data point
b. Line Histogram – When you have a lot of data points
c. Scatter plot – When you have to show the distribution of 2-3 variables

2. Comparison – When you have to compare something over 2 or more categories

a. Variable width chart – When you have to compare two variables per item
b. Tables with embedded charts – When there are many categories, basically a matrix of charts
c. Horizontal or Vertical Histogram – When there are few categories in a data set
d. If you want to compare something over time
i. Line Chart
ii. Bar Vertical Chart
iii. Many categories line chart

3. Relationship Charts – When you want to see the relationship between two or more variables then you have to use relationship charts

a. Scatter Plot
b. Scatter plot bubble chart

4. Composition Charts – When you have to show a percentage or composition of variables.

a. Pie Chart – Very basic plot when there are 3-6 categories
b. Stacked 100% bar chart with sub component – When you have to show components of components
c. Stacked 100% bar chart – When you have to look into the contribution of each component.
d. Stacked area chart – When relative and absolute difference matters

124. What will be the output of the below Python code –

```
def multipliers ():
 return [lambda x: i * x for i in range (4)]
 print [m (2) for m in multipliers ()]
```

The output for the above code will be [6, 6,6,6]. The reason for this is that because of late binding the value of the variable i is looked up when any of the functions returned by multipliers are called.

125. What is the difference between range & xrange?
For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.

This means that xrange doesn't actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you'd like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It's a memory hungry beast.

TECHVIDYA

TECHVIDYA

ISO 9001:2015 Accredited Company

"Stay Updated, Stay Ahead"