Python Basic Interview Questions

1. What are the key features of Python?

Python is one of the most popular programming languages used by data scientists and AIML professionals. This popularity is due to the following key features of Python:

- Python is easy to learn due to its clear syntax and readability
- Python is easy to interpret, making debugging easy
- Python is free and Open-source
- It can be used across different languages
- It is an object-oriented language which supports concepts of classes
- It can be easily integrated with other languages like C++, Java and more

2. What are Keywords in Python?

Keywords in Python are reserved words which are used as identifiers, function name or variable name. They help define the structure and syntax of the language.

There are a total of 33 keywords in Python 3.7 which can change in the next version, i.e., Python 3.8. A list of all the keywords is provided below:

Keywords in Python

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield

assert	else	import	pass	
break	except			

3. What are Literals in Python and explain about different Literals?

Literals in Python refer to the data that is given in a variable or constant. Python has various kinds of literals including:

1. String Literals: It is a sequence of characters enclosed in codes. There can be single, double and triple strings based on the number of quotes used. Character literals are single characters surrounded by single or double-quotes.
2. Numeric Literals: These are unchangeable kind and belong to three different types – integer, float and complex.
3. Boolean Literals: They can have either of the two values- True or False which represents '1' and '0' respectively.
4. Special Literals: Special literals are sued to classify fields that are not created. It is represented by the value 'none'.

4. How can you concatenate two tuples?

Solution ->

Let's say we have two tuples like this ->

tup1 = (1,"a",True)

tup2 = (4,5,6)

Concatenation of tuples means that we are adding the elements of one tuple at the end of another tuple.

Now, let's go ahead and concatenate tuple2 with tuple1:

Code

```
<code>tup1=(1,"a",True) tup2=(4,5,6) tup1+tup2 </code>
```

Output

```
(1, 'a', True, 4, 5, 6)
```

All you have to do is, use the '+' operator between the two tuples and you'll get the concatenated result.

Similarly, let's concatenate tuple1 with tuple2:

Code

```
<code>tup1=(1,"a",True) tup2=(4,5,6) tup2+tup1</code>
```

Output

```
tup2+tup1
(4, 5, 6, 1, 'a', True)
```

5. What are functions in Python?

Ans: Functions in Python refer to blocks that have organised, and reusable codes to perform single, and related events. Functions are important to create better modularity for applications which reuse high degree of coding. Python has a number of built-in functions like print(). However, it also allows you to create user-defined functions.

6. How to Install Python?

To Install Python, first go to Anaconda.org and click on "Download Anaconda". Here, you can download the latest version of Python. After Python is installed, it is a pretty straightforward process. The next step is to power up an IDE and start coding in Python. If you wish to learn more about the process, check out this Python Tutorial.

7. What is Python Used For?

Python is one of the most popular programming languages in the world today. Whether you're browsing through Google, scrolling through Instagram, watching videos on YouTube, or listening to music on Spotify, all of these applications make use of Python for their key programming requirements. Python is used across various platforms, applications, and services such as web development.

8. How can you initialize a 5*5 numpy array with only zeroes?

Solution ->

We will be using the .zeros() method

<code>import numpy as np n1=np.zeros((5,5)) n1</code>

Use np.zeros() and pass in the dimensions inside it. Since, we want a 5*5 matrix, we will pass (5,5) inside the .zeros() method.

This will be the output:

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

9. What is Pandas?

Pandas is an open source python library which has a very rich set of data structures for data based operations. Pandas with it's cool features fits in every role of data operation, whether it be academics or solving complex business problems. Pandas can deal with a large variety of files and is one of the most important tools to have a grip on.

10. What are dataframes?

A pandas dataframe is a data structure in pandas which is mutable. Pandas has support for heterogeneous data which is arranged across two axes.(rows and columns).

Reading files into pandas:-

1 Import pandas as pd df=p.read_csv(“mydata.csv”)

Here df is a pandas data frame. read_csv() is used to read a comma delimited file as a dataframe in pandas.

11. What is a Pandas Series?

Series is a one dimensional pandas data structure which can data of almost any type. It resembles an excel column. It supports multiple operations and is used for single dimensional data operations.

Creating a series from data:

Code

```
<code>import pandas as pd data=["1",2,"three",4.0] series=pd.Series(data) print(series) print(type(series))</code>
```

Output

```
0        1
1        2
2    three
3        4
dtype: object
<class 'pandas.core.series.Series'>
```

12. What is pandas groupby?

A pandas groupby is a feature supported by pandas which is used to split and group an object. Like the sql/mysql/oracle groupby it used to group data by classes, entities

which can be further used for aggregation. A dataframe can be grouped by one or more columns.

Code

```
<code>df = pd.DataFrame({'Vehicle':['Etios','Lamborghini','Apache200','Pulsar200'], 'Type':["car","car","motorcycle","motorcycle"]}) df</code>
```

Output

	Vehicle	Type
0	Etios	car
1	Lamborghini	car
2	Apache200	motorcycle
3	Pulsar200	motorcycle

To perform groupby type the following **code:**

```
<code>df.groupby('Type').count()</code>
```

Output

Type	Vehicle
car	2
motorcycle	2

13. How to create a dataframe from lists?

To create a dataframe from lists ,

1)create an empty dataframe

2)add lists as individuals columns to the list

Code

```
<code>df=pd.DataFrame() bikes=["bajaj","tvs","herohonda","kawasaki","bmw"]
cars=["lamborghini","masserati","ferrari","hyundai","ford"] df["cars"]=cars
df["bikes"]=bikes df</code>
```

Output

	cars	bikes
0	lamborghini	bajaj
1	masserati	tvs
2	ferrari	herohonda
3	hyundai	kawasaki
4	ford	bmw

14. How to create dataframe from a dictionary?

A dictionary can be directly passed as an argument to the DataFrame() function to create the data frame.

Code

```
<code>import pandas as pd bikes=["bajaj","tvs","herohonda","kawasaki","bmw"]
cars=["lamborghini","masserati","ferrari","hyundai","ford"] d={"cars":cars,"bikes":bikes}
df=pd.DataFrame(d) df</code>
```

Output

	cars	bikes
0	lamborghini	bajaj
1	masserati	tvs
2	ferrari	herohonda
3	hyundai	kawasaki
4	ford	bmw

15. How to combine dataframes in pandas?

Two different data frames can be stacked either horizontally or vertically by the concat(), append() and join() functions in pandas.

Concat works best when the dataframes have the same columns and can be used for concatenation of data having similar fields and is basically vertical stacking of dataframes into a single dataframe.

Append() is used for horizontal stacking of dataframes. If two tables(dataframes) are to be merged together then this is the best concatenation function.

Join is used when we need to extract data from different dataframes which are having one or more common columns. The stacking is horizontal in this case.

Before going through the **questions**, here's a quick video to help you refresh your memory on Python.

16. What kind of joins does pandas offer?

Pandas has a left join, inner join, right join and an outer join.

17. How to merge data frames in pandas?

Merging depends on the type and fields of different data frames being merged. If data is having similar fields data is merged along axis 0 else they are merged along axis 1.

18. Give the below data frame drop all rows having Nan.

	col1	col2
0	1.0	A
1	2.0	B
2	NaN	C

The dropna function can be used to do that.

<code>df.dropna(inplace=True) df</code>

Output

```
df.dropna(inplace=True)
df
```

	col1	col2
0	1.0	A
1	2.0	B

19. How to access the first five entries of a data frame?

By using the head(5) function we can get the top five entries of a data frame. By default df.head() returns the top 5 rows. To get the top n rows df.head(n) will be used.

20. How to access the last five entries of a data frame?

By using tail(5) function we can get the top five entries of a dataframe. By default df.tail() returns the top 5 rows. To get the last n rows df.tail(n) will be used.

21. How to fetch a data entry from a pandas dataframe using a given value in index?

To fetch a row from dataframe given index x, we can use loc.

Df.loc[10] where 10 is the value of the index.

Code

```
<code>import pandas as pd bikes=["bajaj","tvs","herohonda","kawasaki","bmw"] cars=["lamborghini","masserati","ferrari","hyundai","ford"] d={"cars":cars,"bikes":bikes} df=pd.DataFrame(d) a=[10,20,30,40,50] df.index=a df.loc[10]</code>
```

Output

```
cars     lamborghini
bikes          bajaj
Name: 10, dtype: object
```

22. What are comments and how can you add comments in Python?

Comments in Python refer to a piece of text intended for information. It is especially relevant when more than one person works on a set of codes. It can be used to analyse code, leave feedback, and debug it. There are two types of comments which includes:

1. Single-line comment
2. Multiple-line comment

Codes needed for adding comment

#Note –single line comment

“””Note

Note

Note”””—–multiline comment

23. What is the difference between list and tuples in Python?

Lists are mutable, but tuples are immutable.

24. What is dictionary in Python? Give an example.

A Python dictionary is a collection of items in no particular order. Python dictionaries are written in curly brackets with keys and values. Dictionaries are optimised to retrieve value for known keys.

Example

d={“a”:1,”b”:2}

25. Find out the mean, median and standard deviation of this numpy array -> np.array([1,5,3,100,4,48])

<code>import numpy as np n1=np.array([10,20,30,40,50,60]) print(np.mean(n1)) print(np.median(n1)) print(np.std(n1))</code>

26. What is a classifier?

A classifier is used to predict the class of any data point. Classifiers are special hypotheses that are used to assign class labels to any particular data points. A classifier often uses training data to understand the relation between input variables and the class. Classification is a method used in supervised learning in Machine Learning.

27. In Python how do you convert a string into lowercase?

All the upper cases in a string can be converted into lowercase by using the method: string.lower()

ex: string = ‘GREATLEARNING’ print(string.lower())

o/p: greatlearning

28. How do you get a list of all the keys in a dictionary?

One of the ways we can get a list of keys is by using: dict.keys()

This method returns all the available keys in the dictionary. dict = {1:a, 2:b, 3:c}

dict.keys()

o/p: [1, 2, 3]

29. How can you capitalize the first letter of a string?

We can use the **capitalize()** function to capitalize the first character of a string. If the first character is already in capital then it returns the original string.

Syntax: string_name.capitalize() ex: n = “greatlearning” print(n.capitalize())

o/p: Greatlearning

30. How can you insert an element at a given index in Python?

Python has an inbuilt function called the insert() function.

It can be used used to insert an element at a given index.

Syntax: list_name.insert(index, element)

ex: list = [0,1, 2, 3, 4, 5, 6, 7]

#insert 10 at 6th index

list.insert(6, 10)

o/p: [0,1,2,3,4,5,10,6,7]

31. How will you remove duplicate elements from a list?

There are various methods to remove duplicate elements from a list. But, the most common one is, converting the list into a set by using the set() function and using the list() function to convert it back to a list, if required. ex: list0 = [2, 6, 4, 7, 4, 6, 7, 2]

list1 = list(set(list0)) print (“The list without duplicates : ” + str(list1)) **o/p: The list without duplicates : [2, 4, 6, 7]**

32. What is recursion?

Recursion is a function calling itself one or more times in it body. One very important condition a recursive function should have to be used in a program is, it should terminate, else there would be a problem of an infinite loop.

33. Explain Python List Comprehension

List comprehensions are used for transforming one list into another list. Elements can be conditionally included in the new list and each element can be transformed as needed. It consists of an expression leading a for clause, enclosed in brackets. for ex: list = [i for i in range(1000)]
print list

34. What is the bytes() function?

The bytes() function returns a bytes object. It is used to convert objects into bytes objects, or create empty bytes object of the specified size.

35. What are the different types of operators in Python?

Python has the following basic operators:
Arithmetic(Addition(+), Substraction(-), Multiplication(*), Division(/), Modulus(%)), **Relational** (<, >, <=, >=, ==, !=,),
Assignment (=. +=, -=, /=, *=, %=),
Logical (and, or not), Membership, Identity, and Bitwise Operators

36. What is the ‘with statement’?

“with” statement in python is used in exception handling. A file can be opened and closed while executing a block of code, containing the “with” statement., without using the close() function. It essentially makes the code much more easy to read.

37. What is a map() function in Python?

The map() function in Python is used for applying a function on all elements of a specified iterable. It consists of two parameters, function and iterable. The function is taken as an argument and then applied to all the elements of an iterable(passed as the second argument). An object list is returned as a result.

```
def add(n):
return n + n number= (15, 25, 35, 45)
res= map(add, num)
print(list(res))
```

o/p: 30,50,70,90

38. What is __init__ in Python?

init methodology is a reserved method in Python aka constructor in OOP. When an object is created from a class and _init_ methodolgy is called to acess the class attributes.

39. What are the tools present to perform statics analysis?

The two static analysis tool used to find bugs in Python are: Pychecker and Pylint. Pychecker detects bugs from the source code and warns about its style and complexity.While, Pylint checks whether the module matches upto a coding standard.

40. What is the difference between tuple and dictionary?

One major difference between a tuple and a dictionary is that dictionary is mutable while a tuple is not. Meaning the content of a dictionary can be changed without changing it's identity, but in tuple that's not possible.

41. What is pass in Python?

Pass is a statentemen which does nothing when executed. In other words it is a Null statement. This statement is not ignored by the interpreter, but the statement results in no operation. It is used when you do not want any command to execute but a statement is required.

42. How can an object be copied in Python?

Not all objects can be copied in Python, but most can. We ca use the "=" operator to copy an obect to a variable.

ex: var=copy.copy(obj)

43. How can a number be converted to a string?

The inbuilt function str() can be used to convert a nuber to a string.

44. What are module and package in Python?

Modules are the way to structure a program. Each Python program file is a module, importing other attributes and objects. The folder of a program is a package of modules. A package can have modules or subfolders.

45. What is object() function in Python?

In Python the object() function returns an empty object. New properties or methods cannot be added to this object.

46. What is the difference between NumPy and SciPy?

NumPy stands for Numerical Python while SciPy stands for Scientific Python. NumPy is the basic library for defining arrays and simple mathematica problems, while SciPy is used for more complex problems like numerical integration and optimization and machine learning and so on.

47. What does len() do?

len() is used to determine the length of a string, a list, an array, and so on. ex: str = “greatlearning”
print(len(str))
o/p: 13

48. Define encapsulation in Python?

Encapsulation means binding the code and the data together. A Python class for example.

49. What is the type () in Python?

type() is a built-in method which either returns the type of the object or returns a new type object based on the arguments passed.

ex: a = 100
type(a)

o/p: int

50. What is split() function used for?

Split fuction is used to split a string into shorter string using defined seperatos. letters = (” A, B, C”)
n = text.split(“,”)
print(n)

o/p: [‘A’, ‘B’, ‘C’]

51. What are the built-in types does python provide?

Ans. Python has following built-in data types:

Numbers: Python identifies three types of numbers:

1. Integer: All positive and negative numbers without a fractional part
2. Float: Any real number with floating-point representation
3. Complex numbers: A number with a real and imaginary component represented as x+yj. x and y are floats and j is -1(square root of -1 called an imaginary number)

Boolean: The Boolean data type is a data type that has one of two possible values i.e. True or False. Note that ‘T’ and ‘F’ are capital letters.

String: A string value is a collection of one or more characters put in single, double or triple quotes.

List: A list object is an ordered collection of one or more data items which can be of different types, put in square brackets. A list is mutable and thus can be modified, we can add, edit or delete individual elements in a list.

Set: An unordered collection of unique objects enclosed in curly brackets

Frozen set: They are like a set but immutable, which means we cannot modify their values once they are created.

Dictionary: A dictionary object is unordered in which there is a key associated with each value and we can access each value through its key. A collection of such pairs is enclosed in curly brackets. For example {‘First Name’ : ’Tom’ , ’last name’ : ’Hardy’} Note that Number values, strings, and tuple are immutable while as List or Dictionary object are mutable.

52. What is docstring in Python?

Ans. Python docstrings are the string literals enclosed in triple quotes that appear right after the definition of a function, method, class, or module. These are generally used to describe the functionality of a particular function, method, class, or module. We can access these docstrings using the __doc__ attribute. Here is an example:

<code>def square(n): '''Takes in a number n, returns the square of n''' return n**2
print(square.__doc__)</code>

Ouput: Takes in a number n, returns the square of n.

53. How to Reverse a String in Python?

In Python, there are no in-built functions that help us reverse a string. We need to make use of an array slicing operation for the same.

```
1  str_reverse = string[::-1]
```

Learn more: How To Reverse a String In Python

54. How to check Python Version in CMD?

To check the Python Version in CMD, press CMD + Space. This opens Spotlight. Here, type “terminal” and press enter. To execute the command, type python –version or python -V and press enter. This will return the python version in the next line below the command.

55. Is Python case sensitive when dealing with identifiers?

Yes. Python is case sensitive when dealing with identifiers. It is a case sensitive language. Thus, variable and Variable would not be the same.

Python Interview Questions for Experienced Professionals

1. How to create a new column in pandas by using values from other columns?

We can perform column based mathematical operations on a pandas dataframe. Pandas columns containing numeric values can be operated upon by operators.

Code

```
<code>import pandas as pd a=[1,2,3] b=[2,3,5] d={"col1":a,"col2":b} df=pd.DataFrame(d) df["Sum"]=df["col1"]+df["col2"] df["Difference"]=df["col1"]-df["col2"] df</code>
```

Output

	col1	col2	Sum	Difference
0	1	2	3	-1
1	2	3	5	-1
2	3	5	8	-2

2. What are the different functions that can be used by grouby in pandas ?

grouby() in pandas can be used with multiple aggregate functions. Some of which are sum(),mean(), count(),std().

Data is divided into groups based on categories and then the data in these individual groups can be aggregated by the aforementioned functions.

3. How to select columns in pandas and add them to a new dataframe? What if there are two columns with the same name?

If df is dataframe in pandas df.columns gives the list of all columns. We can then form new columns by selecting columns.

If there are two columns with the same name then both columns get copied to the new dataframe.

Code

```
<code>print(d_new.columns) d=d_new[["col1"]] d</code>
```

Output

```
print(d_new.columns)
d=d_new[["col1"]]
d
```

```
Index(['col1', 'col2', 'col1', 'col2'], dtype='object')
```

	col1	col1
0	1	4
1	2	5
2	3	6

4. How to delete a column or group of columns in pandas? Given the below dataframe drop column “col1”.

	col1	col2
0	1	A
1	2	B
2	3	C

drop() function can be used to delete the columns from a dataframe.

```
<code>d={"col1":[1,2,3],"col2":["A","B","C"]} df=pd.DataFrame(d)
df=df.drop(["col1"],axis=1) df</code>
```

Output

	col2
0	A
1	B
2	C

5. Given the following data frame drop rows having column values as A.

	col1	col2
0	1	A
1	2	B
2	3	C

Code

```
<code>d={"col1":[1,2,3],"col2":["A","B","C"]} df=pd.DataFrame(d)
df.dropna(inplace=True) df=df[df.col1!=1] df</code>
```

Output

	col1	col2
1	2	B
2	3	C

6. Given the below dataset find the highest paid player in each college in each team.

	Name	Team	Number	Position	Age	Height	Weight	College	
0	Avery Bradley	Boston Celtics	0.0	PG	25.0	6-2	180.0	Texas	77
1	Jae Crowder	Boston Celtics	99.0	SF	25.0	6-6	235.0	Marquette	67
2	John Holland	Boston Celtics	30.0	SG	27.0	6-5	205.0	Boston University	
3	R.J. Hunter	Boston Celtics	28.0	SG	22.0	6-5	185.0	Georgia State	11
4	Jonas Jerebko	Boston Celtics	8.0	PF	29.0	6-10	231.0	NaN	50
...	...	...	...	...	...	...	...	...	
453	Shelvin Mack	Utah Jazz	8.0	PG	26.0	6-3	203.0	Butler	24
454	Raul Neto	Utah Jazz	25.0	PG	24.0	6-1	179.0	NaN	9
455	Tibor Pleiss	Utah Jazz	21.0	C	26.0	7-3	256.0	NaN	29
456	Jeff Withey	Utah Jazz	24.0	C	26.0	7-0	231.0	Kansas	9
457	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

458 rows × 9 columns

<code>df.groupby(["Team","College"])["Salary"].max()</code>

```
df.groupby(["Team","College"])["Salary"].max()
```

```
Team                College
Atlanta Hawks       Bucknell                 947276.0
                    Creighton               5746479.0
                    Florida                12000000.0
                    Kansas                  2854940.0
                    Louisiana Tech         18671659.0
                                               ...
Washington Wizards  LSU                     1100602.0
                    Michigan State          4000000.0
                    Nevada                  2170465.0
                    North Carolina State     273038.0
                    Virginia Tech            561716.0
Name: Salary, Length: 336, dtype: float64
```

7. Given the above dataset find the min max and average salary of a player collegewise and teamwise.

Code

<code>df.groupby(["Team","College"])["Salary"].max.agg([('max','max'),('min','min'),('count','count'),('avg','min')])</code>

Output

```
df.groupby(["Team","College"])["Salary"].agg([('max', 'max'), ('min', 'min'),('count','co
```

Team	College	max	min	count	avg
Atlanta Hawks	Bucknell	947276.0	947276.0	1	947276.0
	Creighton	5746479.0	5746479.0	1	5746479.0
	Florida	12000000.0	12000000.0	1	12000000.0
	Kansas	2854940.0	2854940.0	1	2854940.0
	Louisiana Tech	18671659.0	18671659.0	1	18671659.0
...	...	...	...	...	...
Washington Wizards	LSU	1100602.0	200600.0	2	200600.0
	Michigan State	4000000.0	4000000.0	1	4000000.0
	Nevada	2170465.0	2170465.0	1	2170465.0
	North Carolina State	273038.0	273038.0	1	273038.0
	Virginia Tech	561716.0	561716.0	1	561716.0

336 rows × 4 columns

8. What is reindexing in pandas?

Reindexing is the process of re-assigning the index of a pandas dataframe.

Code

```
<code> import pandas as pd bikes=["bajaj","tvs","herohonda","kawasaki","bmw"]
cars=["lamborghini","masserati","ferrari","hyundai","ford"] d={"cars":cars,"bikes":bikes}
df=pd.DataFrame(d) a=[10,20,30,40,50] df.index=a df </code>
```

Output

	cars	bikes
10	lamborghini	bajaj
20	masserati	tvs
30	ferrari	herohonda
40	hyundai	kawasaki
50	ford	bmw

9. What do you understand by lambda function? Create a lambda function which will print the sum of all the elements in this list -> [5, 8, 10, 20, 50, 100]

<code>from functools import reduce sequences = [5, 8, 10, 20, 50, 100] sum = reduce (lambda x, y: x+y, sequences) print(sum)</code>

10. What is vstack() in numpy? Give an example

Ans. vstack() is a function to align rows vertically. All rows must have same number of elements.

Code

<code>import numpy as np n1=np.array([10,20,30,40,50]) n2=np.array([50,60,70,80,90]) print(np.vstack((n1,n2)))</code>

Output

```
[[10 20 30 40 50]
 [50 60 70 80 90]]
```

11. How do we interpret Python?

When a python program is written, it converts the source code written by the developer into intermediate language, which is then coverted into machine language that needs to be executed.

12. How to remove spaces from a string in Python?

Spaces can be removed from a string in python by using strip() or replace() functions. Strip() function is used to remove the leading and trailing white spaces while the replace() function is used to remove all the white spaces in the string:

string.replace(" ","") ex1: str1= "great learning"
print (str.strip())

o/p: great learning

ex2: str2="great learning"
print (str.replace(" ",""))

o/p: greatlearning

13. Explain the file processing modes that Python supports.

There are three file processing modes in Python: read-only(r), write-only(w), read-write(rw) and append (a). So, if you are opening a text file in say, read mode. The preceding modes become "rt" for read-only, "wt" for write and so on. Similarly, a binary file can be opened by specifying "b" along with the file accessing flags ("r", "w", "rw" and "a") preceding it.

14. What is pickling and unpickling?

Pickling is the process of converting a Python object hierarchy into a byte stream for storing it into a database. It is also known as serialization. Unpickling is the reverse of pickling. The byte stream is converted back into an object hierarchy.

15. How is memory managed in Python?

Memory management in python comprises of a private heap containing all objects and data stucture. The heap is managed by the interpreter and the programmer does not have acess to it at all. The Python memory manger does all the memory allocation. Moreover, there is an inbuilt garbage collector that recycles and frees memory for the heap space.

16. What is unittest in Python?

Unittest is a unit testinf framework in Python. It supports sharing of setup and shutdown code for tests, aggregation of tests into collections,test automation, and independence of the tests from the reporting framework.

17. How do you delete a file in Python?

Files can be deleted in Python by using the command os.remove (filename) or os.unlink(filename)

18. How do you create an empty class in Python?

To create an empty class we can use the pass command after the definition of the class object. A pass is a statement in Python that does nothing.

19. What are Python decorators?

Ans. Decorators are functions that take another functions as argument to modify its behaviour without changing the function itself. These are useful when we want to dynamically increase the functionality of a function without changing it. Here is an example :

<code>def smart_divide(func): def inner(a, b): print("Dividing", a, "by", b) if b == 0: print("Make sure Denominator is not zero") return return func(a, b) return inner @smart_divide def divide(a, b): print(a/b) divide(1,0)</code>

Here smart_divide is a decorator function that is used to add functionality to simple divide function.

Take up a data science course and power ahead in your career today!

Python Interview Questions for Advanced Levels

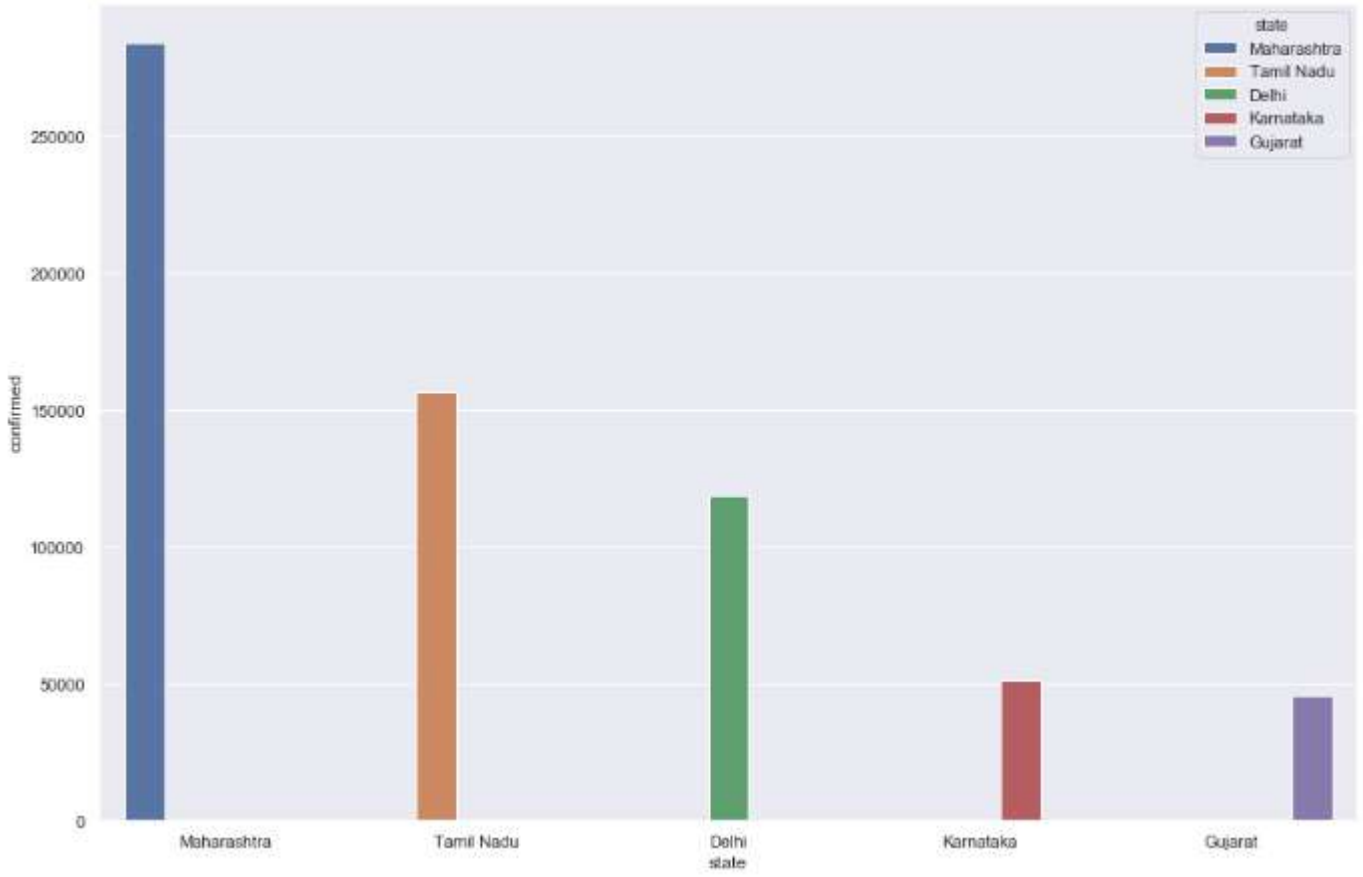
1. You have this covid-19 dataset below:

	Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational
0	1	2020-01-30	6:00 PM	Kerala	1	0
1	2	2020-01-31	6:00 PM	Kerala	1	0
2	3	2020-02-01	6:00 PM	Kerala	2	0
3	4	2020-02-02	6:00 PM	Kerala	3	0
4	5	2020-02-03	6:00 PM	Kerala	3	0

From this dataset, how will you make a bar-plot for the top 5 states having maximum confirmed cases as of 17=07-2020?

sol:

```
<code>#keeping only required columns  df = df[['Date', 'State/UnionTerritory','Cured','Deaths','Confirmed']]  #renaming column names df.columns = ['date', 'state','cured','deaths','confirmed']  #current date  today = df[df.date == '2020-07-17']  #Sorting data w.r.t number of confirmed cases max_confirmed_cases=today.sort_values(by="confirmed",ascending=False) max_confirmed_cases  #Getting states with maximum number of confirmed cases top_states_confirmed=max_confirmed_cases[0:5]  #Making bar-plot for states with top confirmed cases  sns.set(rc={'figure.figsize':(15,10)}) sns.barplot(x="state",y="confirmed",data=top_states_confirmed,hue="state") plt.show()</code>
```



Code explanation:

We start off by taking only the required columns with this command:

df = df[[‘Date’, ‘State/UnionTerritory’,’Cured’,’Deaths’,’Confirmed’]]

Then, we go ahead and rename the columns:

df.columns = [‘date’, ‘state’,’cured’,’deaths’,’confirmed’]

After that, we extract only those records, where the date is equal to 17th July:

today = df[df.date == ‘2020-07-17’]

Then, we go ahead and select the top 5 states with maximum no. of covide cases:

max_confirmed_cases=today.sort_values(by=”confirmed”,ascending=False)

max_confirmed_cases

top_states_confirmed=max_confirmed_cases[0:5]

Finally, we go ahead and make a bar-plot with this:

sns.set(rc={‘figure.figsize’:(15,10)})

sns.barplot(x=”state”,y=”confirmed”,data=top_states_confirmed,hue=”state”)
plt.show()

Here, we are using seaborn library to make the bar-plot. “State” column is mapped onto the x-axis and “confirmed” column is mapped onto the y-axis. The color of the bars is being determined by the “state” column.

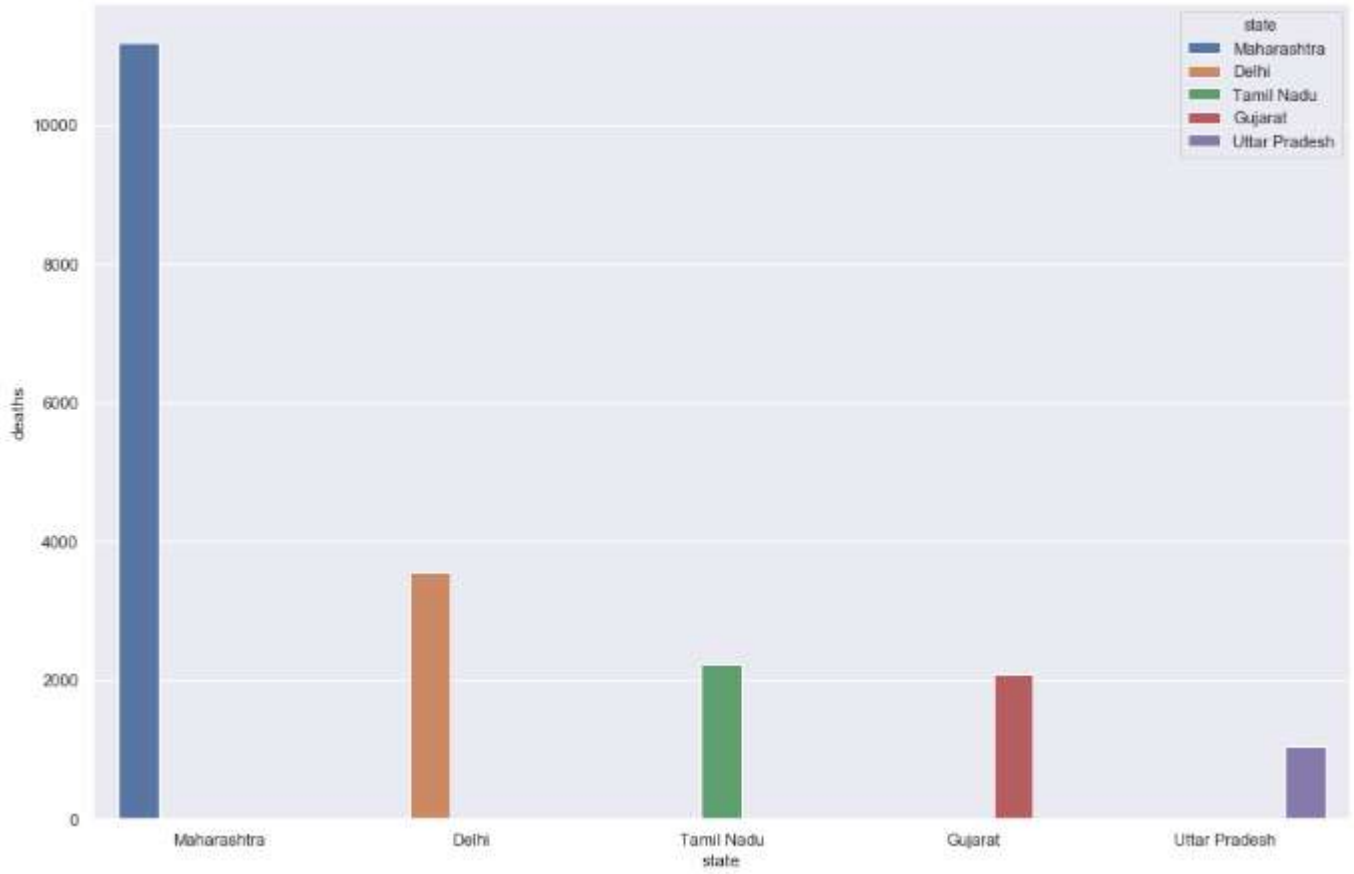
2. From this covid-19 dataset:

	date	state	cured	deaths	confirmed
4210	2020-07-17	Tripura	1604	3	2283
4211	2020-07-17	Uttarakhand	2995	50	3982
4212	2020-07-17	Uttar Pradesh	26675	1046	43441
4213	2020-07-17	West Bengal	21415	1023	36117
4214	2020-07-17	Cases being reassigned to states	0	0	531

How can you make a bar-plot for the top-5 states with the most amount of deaths?

Sol:

```
<code>max_death_cases=today.sort_values(by=”deaths”,ascending=False)
max_death_cases  sns.set(rc={‘figure.figsize’:(15,10)})
sns.barplot(x=”state”,y=”deaths”,data=top_states_death,hue=”state”)
plt.show()</code>
```



Code Explanation:

We start off by sorting our dataframe in descending order w.r.t the “deaths” column:

max_death_cases=today.sort_values(by=”deaths”,ascending=False)

Max_death_cases

Then, we go ahead and make the bar-plot with the help of seaborn library:

sns.set(rc={‘figure.figsize’:(15,10)})

sns.barplot(x=”state”,y=”deaths”,data=top_states_death,hue=”state”)

plt.show()

Here, we are mapping “state” column onto the x-axis and “deaths” column onto the y-axis.

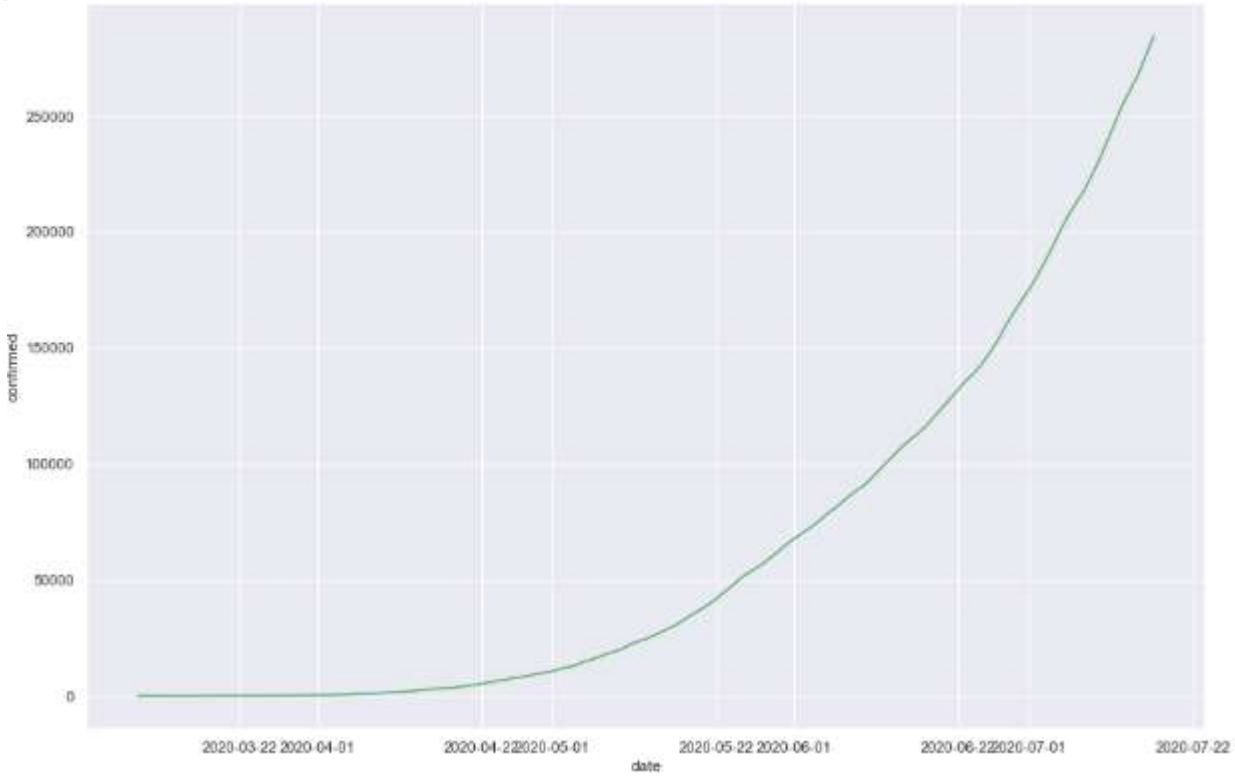
3. From this covid-19 dataset:

	date	state	cured	deaths	confirmed
4210	2020-07-17	Tripura	1604	3	2283
4211	2020-07-17	Uttarakhand	2995	50	3982
4212	2020-07-17	Uttar Pradesh	26675	1046	43441
4213	2020-07-17	West Bengal	21415	1023	36117
4214	2020-07-17	Cases being reassigned to states	0	0	531

How can you make a line plot indicating the confirmed cases with respect to date?

Sol:

<code>maha = df[df.state == 'Maharashtra'] sns.set(rc={'figure.figsize':(15,10)})
sns.lineplot(x="date",y="confirmed",data=maha,color="g") plt.show()</code>



Code Explanation:

We start off by extracting all the records where the state is equal to "Maharashtra":

maha = df[df.state == 'Maharashtra']

Then, we go ahead and make a line-plot using seaborn library:

sns.set(rc={‘figure.figsize’:(15,10)})

sns.lineplot(x=”date”,y=”confirmed”,data=maha,color=”g”)

plt.show()

Here, we map the “date” column onto the x-axis and “confirmed” column onto y-axis.

---

4. On this “Maharashtra” dataset:

	date	state	cured	deaths	confirmed
76	2020-03-09	Maharashtra	0	0	2
91	2020-03-10	Maharashtra	0	0	5
97	2020-03-11	Maharashtra	0	0	2
120	2020-03-12	Maharashtra	0	0	11
133	2020-03-13	Maharashtra	0	0	14
...	...	...	...	...	...
4054	2020-07-13	Maharashtra	140325	10289	254427
4090	2020-07-14	Maharashtra	144507	10482	260924
4126	2020-07-15	Maharashtra	149007	10695	267665
4162	2020-07-16	Maharashtra	152613	10928	275640
4198	2020-07-17	Maharashtra	158140	11194	284281

How will you implement a linear regression algorithm with “date” as independent variable and “confirmed” as dependent variable. That is you have to predict the number of confirmed cases w.r.t date.

Sol:

```
<code>from sklearn.model_selection import train_test_split
maha[‘date’]=maha[‘date’].map(dt.datetime.toordinal)  maha.head()  x=maha[‘date’]
y=maha[‘confirmed’]  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.linear_model import LinearRegression  lr = LinearRegression()
lr.fit(np.array(x_train).reshape(-1,1),np.array(y_train).reshape(-1,1))
lr.predict(np.array([[737630]]))  </code>
```

Code solution:

We will start off by converting the date to ordinal type:

```
from sklearn.model_selection import train_test_split

maha['date']=maha['date'].map(dt.datetime.toordinal)
```

This is done because we cannot build the linear regression algorithm on top of the date column.

Then, we go ahead and divide the dataset into train and test sets:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

Finally, we go ahead and build the model:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(np.array(x_train).reshape(-1,1),np.array(y_train).reshape(-1,1))

lr.predict(np.array([[737630]]))
```

5. On this customer_churn dataset:

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	..
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	..
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	..
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	..
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	..
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	..

Build a keras sequential model to find out how many customers will churn out on the basis of tenure of customer?

Sol:

```
<code>from keras.models import Sequential  from keras.layers import Dense  model = Sequential()  model.add(Dense(12, input_dim=1, activation='relu'))  model.add(Dense(8, activation='relu'))  model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(x_train, y_train, epochs=150,validation_data=(x_test,y_test))  y_pred = model.predict_classes(x_test)  from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)</code>
```

Code explanation:

We will start off by importing the required libraries:

```
from keras.models import Sequential

from keras.layers import Dense
```

Then, we go ahead and build the structure of the sequential model:

```
model = Sequential()

model.add(Dense(12, input_dim=1, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))
```

Finally, we will go ahead and predict the values:

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

model.fit(x_train, y_train, epochs=150,validation_data=(x_test,y_test))

y_pred = model.predict_classes(x_test)
```

from sklearn.metrics import confusion_matrix

confusion_matrix(y_test,y_pred)

---

6. On this iris dataset:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

Build a decision tree classification model, where dependent variable is “Species” and independent variable is “Sepal.Length”.

Sol:

<code>y = iris[[‘Species’]] x = iris[[‘Sepal.Length’]] from sklearn.model_selection import train_test_split x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4) from sklearn.tree import DecisionTreeClassifier dtc = DecisionTreeClassifier() dtc.fit(x_train,y_train) y_pred=dtc.predict(x_test) from sklearn.metrics import confusion_matrix confusion_matrix(y_test,y_pred)</code>

```
array([[21,  1,  0],
       [ 2, 13,  8],
       [ 0,  3, 12]], dtype=int64)
```

(22+7+9)/(22+2+0+7+7+11+1+1+9)

Code explanation:

We start off by extracting the independent variable and dependent variable:

y = iris[[‘Species’]]

x = iris[[‘Sepal.Length’]]

Then, we go ahead and divide the data into train and test set:

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4)
```

After that, we go ahead and build the model:

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

dtc.fit(x_train,y_train)

y_pred=dtc.predict(x_test)
```

Finally, we build the confusion matrix:

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test,y_pred)
```

```
array([[21,  1,  0],
       [ 2, 13,  8],
       [ 0,  3, 12]], dtype=int64)
```

(22+7+9)/(22+2+0+7+7+11+1+1+9)

7. On this iris dataset:

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa

Build a decision tree regression model where the independent variable is “petal length” and dependent variable is “Sepal length”.

Sol:

```
<code>x= iris[[‘Petal.Length’]]  y = iris[[‘Sepal.Length’]]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)  from sklearn.tree
import DecisionTreeRegressor  dtr = DecisionTreeRegressor()  dtr.fit(x_train,y_train)
y_pred=dtr.predict(x_test)  y_pred[0:5]  from sklearn.metrics import
mean_squared_error  mean_squared_error(y_test,y_pred)</code>
```

---

8. How will you scrape data from the website “cricbuzz”?

Sol:

```
<code>import sys  import time  from bs4 import BeautifulSoup  import requests  import
pandas as pd  try:        #use the browser to get the url. This is suspicious command that
might blow up.      page=requests.get(‘cricbuzz.com’)                    # this might throw
an exception if something goes wrong.  except Exception as e:                    # this
describes what to do if an exception is thrown      error_type, error_obj, error_info =
sys.exc_info()      # get the exception information      print (‘ERROR FOR LINK:’,url)
#print the link that cause the problem      print (error_type, ‘Line:’, error_info.tb_lineno)
#print error info and line that threw the exception                    #ignore
this page. Abandon this and go back.  time.sleep(2)
soup=BeautifulSoup(page.text,’html.parser’)
links=soup.find_all(‘span’,attrs={‘class’:’w_tle’})   links  for i in links:      print(i.text)
print(“\n”)</code>
```

---

9. Write a user-defined function to implement central-limit theorem. You have to implement central limit theorem on this “insurance” dataset:

age	sex	bmi	children	smoker
19	female	27.9	0	yes
18	male	33.77	1	no
28	male	33	3	no
33	male	22.705	0	no
32	male	28.88	0	no
31	female	25.74	0	no
46	female	33.44	1	no
37	female	27.74	3	no

You also have to build two plots on “Sampling Distribution of bmi” and “Population distribution of bmi”.

Sol:

```
<code>df = pd.read_csv('insurance.csv') series1 = df.charges series1.dtype def central_limit_theorem(data,n_samples = 1000, sample_size = 500, min_value = 0, max_value = 1338): """ Use this function to demonstrate Central Limit Theorem. data = 1D array, or a pd.Series n_samples = number of samples to be created sample_size = size of the individual sample min_value = minimum index of the data max_value = maximum index value of the data """ %matplotlib inline import pandas as pd import numpy as np import matplotlib.pyplot as plt import seaborn as sns b = {} for i in range(n_samples): x = np.unique(np.random.randint(min_value, max_value, size = sample_size)) # set of random numbers with a specific size b[i] = data[x].mean() # Mean of each sample c = pd.DataFrame() c['sample'] = b.keys() # Sample number c['Mean'] = b.values() # mean of that particular sample plt.figure(figsize= (15,5)) plt.subplot(1,2,1) sns.distplot(c.Mean) plt.title(f"Sampling Distribution of bmi. \n \u03bc = {round(c.Mean.mean(), 3)} & SE = {round(c.Mean.std(),3)}") plt.xlabel('data') plt.ylabel('freq') plt.subplot(1,2,2) sns.distplot(data) plt.title(f"population Distribution of bmi. \n \u03bc = {round(data.mean(), 3)} & \u03C3 = {round(data.std(),3)}") plt.xlabel('data') plt.ylabel('freq') plt.show() central_limit_theorem(series1,n_samples = 5000, sample_size = 500)</code>
```

Code Explanation:

We start off by importing the insurance.csv file with this command:

df = pd.read_csv('insurance.csv')

Then we go ahead and define the central limit theorem method:

def central_limit_theorem(data,n_samples = 1000, sample_size = 500, min_value = 0, max_value = 1338):

This method comprises of these parameters:

- Data
- N_samples
- Sample_size
- Min_value
- Max_value

Inside this method, we import all the required libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns
```

Then, we go ahead and create the first sub-plot for "Sampling distribution of bmi":

```
plt.subplot(1,2,1)

sns.distplot(c.Mean)

plt.title(f"Sampling Distribution of bmi. \n \u03bc = {round(c.Mean.mean(), 3)} & SE = {round(c.Mean.std(),3)}")

plt.xlabel('data')
```

```
   plt.ylabel('freq')
```

Finally, we create the sub-plot for "Population distribution of bmi":

```
 plt.subplot(1,2,2)

   sns.distplot(data)

   plt.title(f"population Distribution of bmi. \n \u03bc = {round(data.mean(), 3)} &
\u03C3 = {round(data.std(),3)}")

   plt.xlabel('data')

   plt.ylabel('freq')

   plt.show()
```

---

10. Write code to perform sentiment analysis on amazon reviews:

Sol:

```
<code>import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from
tensorflow.python.keras import models, layers, optimizers  import tensorflow  from
tensorflow.keras.preprocessing.text import Tokenizer, text_to_word_sequence  from
tensorflow.keras.preprocessing.sequence import pad_sequences  import bz2  from
sklearn.metrics import f1_score, roc_auc_score, accuracy_score  import re  %matplotlib
inline  def get_labels_and_texts(file):      labels = []      texts = []      for line in
bz2.BZ2File(file):          x = line.decode("utf-8")          labels.append(int(x[9]) – 1)
texts.append(x[10:].strip())      return np.array(labels), texts  train_labels, train_texts =
get_labels_and_texts('train.ft.txt.bz2')  test_labels, test_texts =
get_labels_and_texts('test.ft.txt.bz2')  Train_labels[0]  Train_texts[0]
train_labels=train_labels[0:500]  train_texts=train_texts[0:500]  import re
NON_ALPHANUM = re.compile(r'[\W]')  NON_ASCII = re.compile(r'[^a-z0-1\s]')  def
normalize_texts(texts):      normalized_texts = []      for text in texts:          lower =
text.lower()          no_punctuation = NON_ALPHANUM.sub(r' ', lower)          no_non_ascii
= NON_ASCII.sub(r'', no_punctuation)          normalized_texts.append(no_non_ascii)
```

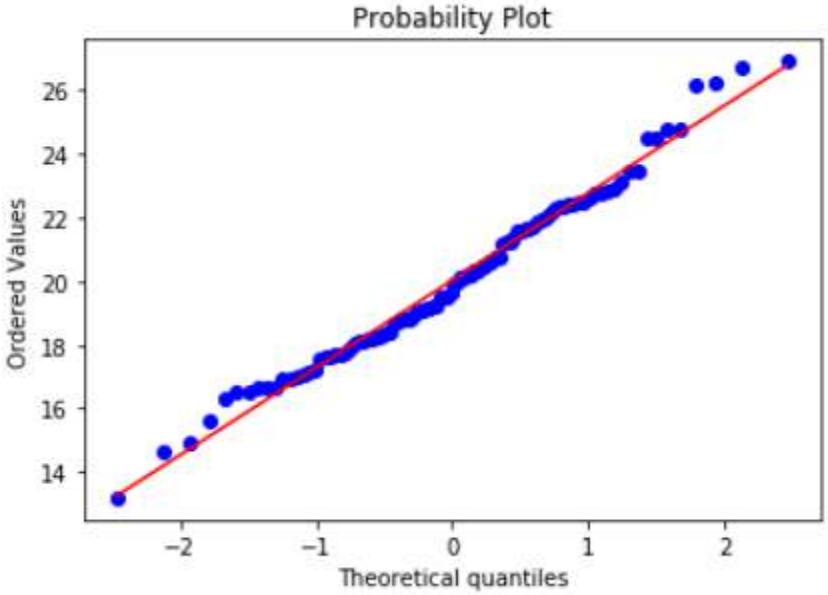
```
return normalized_texts  train_texts = normalize_texts(train_texts)  test_texts =
normalize_texts(test_texts)  from sklearn.feature_extraction.text import
CountVectorizer  cv = CountVectorizer(binary=True)  cv.fit(train_texts)  X =
cv.transform(train_texts)  X_test = cv.transform(test_texts)  from sklearn.linear_model
import LogisticRegression  from sklearn.metrics import accuracy_score  from
sklearn.model_selection import train_test_split  X_train, X_val, y_train, y_val =
train_test_split(      X, train_labels, train_size = 0.75)  for c in [0.01, 0.05, 0.25, 0.5, 1]:
lr = LogisticRegression(C=c)      lr.fit(X_train, y_train)      print ("Accuracy for C=%s: %s"
% (c, accuracy_score(y_val, lr.predict(X_val))))  lr.predict(X_test[29])</code>
```

---

11. Implement a probability plot using numpy and matplotlib:

sol:

```
<code>import numpy as np  import pylab  import scipy.stats as stats  from matplotlib
import pyplot as plt  n1=np.random.normal(loc=0,scale=1,size=1000)
np.percentile(n1,100)  n1=np.random.normal(loc=20,scale=3,size=100)
stats.probplot(n1,dist="norm",plot=pylab)  plt.show()</code>
```



---

12. Implement multiple linear regression on this iris dataset:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

The independent variables should be “Sepal.Width”, “Petal.Length”, “Petal.Width”, while the dependent variable should be “Sepal.Length”.

Sol:

<code>import pandas as pd iris = pd.read_csv(“iris.csv”) iris.head() x = iris[[‘Sepal.Width’,’Petal.Length’,’Petal.Width’]] y = iris[[‘Sepal.Length’]] from sklearn.model_selection import train_test_split x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.35) from sklearn.linear_model import LinearRegression lr = LinearRegression() lr.fit(x_train, y_train) y_pred = lr.predict(x_test) from sklearn.metrics import mean_squared_error mean_squared_error(y_test, y_pred)</code>

Code solution:

We start off by importing the required libraries:

import pandas as pd

iris = pd.read_csv(“iris.csv”)

iris.head()

Then, we will go ahead and extract the independent variables and dependent variable:

x = iris[[‘Sepal.Width’,’Petal.Length’,’Petal.Width’]]

y = iris[[‘Sepal.Length’]]

Following which, we divide the data into train and test sets:

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.35)
```

Then, we go ahead and build the model:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(x_train, y_train)

y_pred = lr.predict(x_test)
```

Finally, we will find out the mean squared error:

```
from sklearn.metrics import mean_squared_error

mean_squared_error(y_test, y_pred)
```

---

13. From this credit fraud dataset:

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787
0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425
1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654
1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024
2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739

Find the percentage of transactions which are fraudulent and not fraudulent. Also build a logistic regression model, to find out if the transaction is fraudulent or not.

Sol:

```
<code>nfcount=0  notFraud=data_df['Class']  for i in range(len(notFraud)):    if notFraud[i]==0:      nfcount=nfcount+1  nfcount      per_nf=(nfcount/len(notFraud))*100 print('percentage of total not fraud transaction in the dataset: ',per_nf)  fcount=0 Fraud=data_df['Class']  for i in range(len(Fraud)):    if Fraud[i]==1:      fcount=fcount+1 fcount      per_f=(fcount/len(Fraud))*100  print('percentage of total fraud transaction in the dataset: ',per_f)  x=data_df.drop(['Class'], axis = 1)#drop the target variable y=data_df['Class']  xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.2, random_state = 42)   logisticreg = LogisticRegression()  logisticreg.fit(xtrain, ytrain) y_pred = logisticreg.predict(xtest)  accuracy= logisticreg.score(xtest,ytest)  cm = metrics.confusion_matrix(ytest, y_pred)  print(cm) </code>
```

```
[[56829     35]
 [   45     53]]
```

---

14. Implement a simple CNN on the MNIST dataset using Keras. Following which, also add in drop out layers.

Sol:

```
<code>from __future__ import absolute_import, division, print_function  import numpy as np  # import keras  from tensorflow.keras.datasets import cifar10, mnist  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten, Reshape  from tensorflow.keras.layers import Convolution2D, MaxPooling2D  from tensorflow.keras import utils  import pickle  from matplotlib import pyplot as plt  import seaborn as sns  plt.rcParams['figure.figsize'] = (15, 8)  %matplotlib inline  # Load/Prep the Data  (x_train, y_train_num), (x_test, y_test_num) = mnist.load_data()  x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype('float32')  x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype('float32') x_train /= 255  x_test /= 255  y_train = utils.to_categorical(y_train_num, 10)  y_test = utils.to_categorical(y_test_num, 10)  print('— THE DATA —')  print('x_train shape:', x_train.shape)  print(x_train.shape[0], 'train samples')  print(x_test.shape[0], 'test samples')  TRAIN = False  BATCH_SIZE = 32  EPOCHS = 1  # Define the Type of Model model1 = tf.keras.Sequential()  # Flatten Imgaes to Vector  model1.add(Reshape((784,), input_shape=(28, 28, 1)))  # Layer 1  model1.add(Dense(128, kernel_initializer='he_normal', use_bias=True))  model1.add(Activation("relu"))  # Layer 2  model1.add(Dense(10, kernel_initializer='he_normal', use_bias=True)) model1.add(Activation("softmax"))  # Loss and Optimizer model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])  # Store Training Results  early_stopping = keras.callbacks.EarlyStopping(monitor='val_acc', patience=10, verbose=1, mode='auto')
```

```
callback_list = [early_stopping]# [stats, early_stopping]  # Train the model
model1.fit(x_train, y_train, nb_epoch=EPOCHS, batch_size=BATCH_SIZE,
validation_data=(x_test, y_test), callbacks=callback_list, verbose=True)  #drop-out
layers:     # Define Model     model3 = tf.keras.Sequential()     # 1st Conv Layer
model3.add(Convolution2D(32, (3, 3), input_shape=(28, 28, 1)))
model3.add(Activation('relu'))     # 2nd Conv Layer     model3.add(Convolution2D(32, (3,
3)))     model3.add(Activation('relu'))     # Max Pooling
model3.add(MaxPooling2D(pool_size=(2,2)))     # Dropout
model3.add(Dropout(0.25))     # Fully Connected Layer     model3.add(Flatten())
model3.add(Dense(128))     model3.add(Activation('relu'))     # More Dropout
model3.add(Dropout(0.5))     # Prediction Layer     model3.add(Dense(10))
model3.add(Activation('softmax'))     # Loss and Optimizer
model3.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])     # Store Training Results     early_stopping =
tf.keras.callbacks.EarlyStopping(monitor='val_acc', patience=7, verbose=1, mode='auto')
callback_list = [early_stopping]     # Train the model     model3.fit(x_train, y_train,
batch_size=BATCH_SIZE, nb_epoch=EPOCHS,              validation_data=(x_test, y_test),
callbacks=callback_list)</code>
```

---

15. Implement a popularity based recommendation system on this movie lens dataset:

movieId	title	genres
1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
2	Jumanji (1995)	Adventure|Children|Fantasy
3	Grumpier Old Men (1995)	Comedy|Romance
4	Waiting to Exhale (1995)	Comedy|Drama|Romance
5	Father of the Bride Part II (1995)	Comedy

```
<code>import os  import numpy as np    import pandas as pd  ratings_data =
pd.read_csv("ratings.csv")    ratings_data.head()   movie_names =
pd.read_csv("movies.csv")    movie_names.head()    movie_data =
pd.merge(ratings_data, movie_names, on='movieId')
movie_data.groupby('title')['rating'].mean().head()
movie_data.groupby('title')['rating'].mean().sort_values(ascending=False).head()
movie_data.groupby('title')['rating'].count().sort_values(ascending=False).head()
ratings_mean_count = pd.DataFrame(movie_data.groupby('title')['rating'].mean())
ratings_mean_count.head()  ratings_mean_count['rating_counts'] =
pd.DataFrame(movie_data.groupby('title')['rating'].count())  ratings_mean_count.head()
</code>
```

title	rating	rating_counts
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

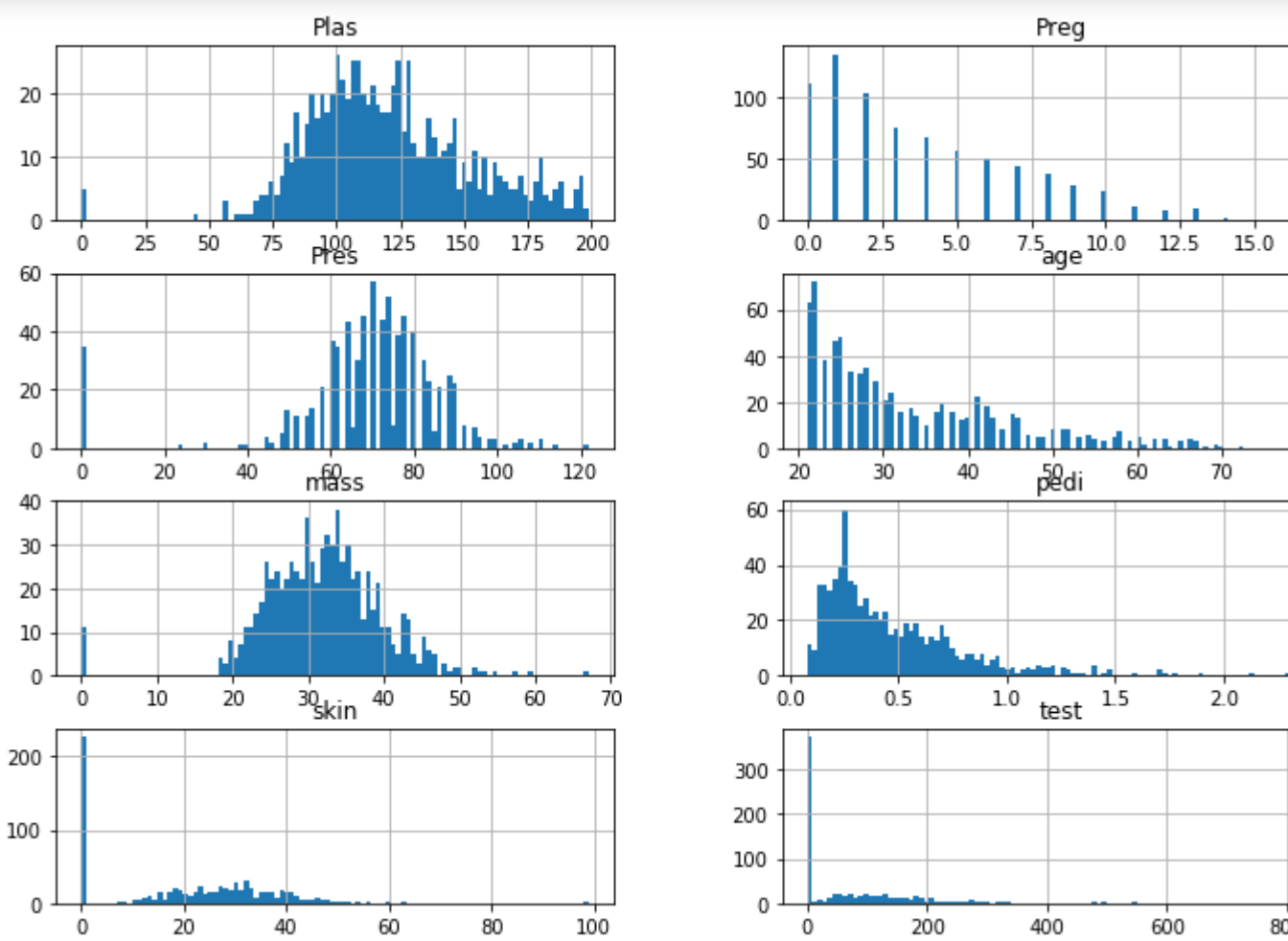
---

16. Implement the naive bayes algorithm on top of the diabetes dataset:

Preg	Plas	Pres	skin	test	mass	pedi	age	class
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

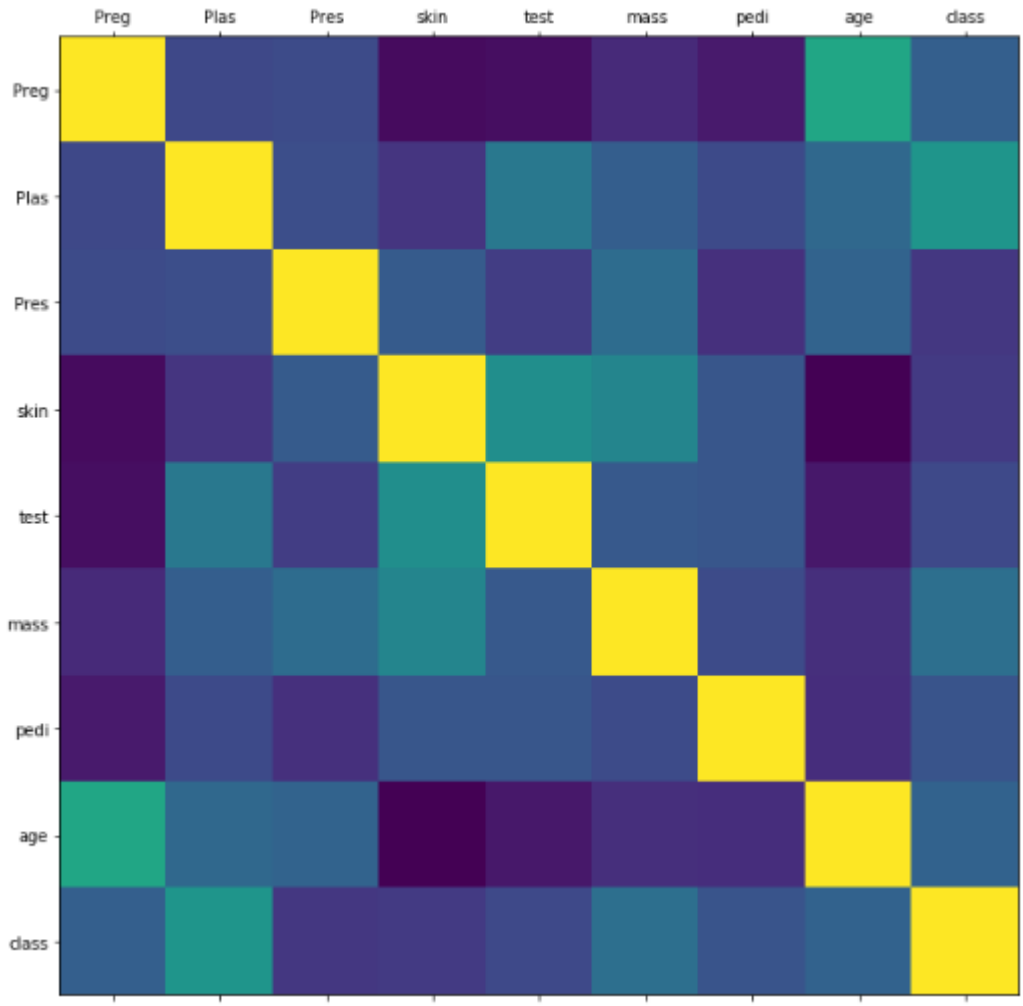
Sol:

<code>import numpy as np # linear algebra import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv) import matplotlib.pyplot as plt # matplotlib.pyplot plots data %matplotlib inline import seaborn as sns pdata = pd.read_csv("pima-indians-diabetes.csv") columns = list(pdata)[0:-1] # Excluding Outcome column which has only pdata[columns].hist(stacked=False, bins=100, figsize=(12,30), layout=(14,2)); # Histogram of first 8 columns</code>



However we want to see correlation in graphical representation so below is function for that

<code>def plot_corr(df, size=11): corr = df.corr() fig, ax = plt.subplots(figsize=(size, size)) ax.matshow(corr) plt.xticks(range(len(corr.columns)), corr.columns) plt.yticks(range(len(corr.columns)), corr.columns) plot_corr(pdata)</code>




```
<code>from sklearn.model_selection import train_test_split  X =
pdata.drop('class',axis=1)     # Predictor feature columns (8 X m)  Y = pdata['class']   #
Predicted class (1=True, 0=False) (1 X m)  x_train, x_test, y_train, y_test =
train_test_split(X, Y, test_size=0.3, random_state=1)  # 1 is just any random seed
number  x_train.head()  from sklearn.naive_bayes import GaussianNB # using Gaussian
algorithm from Naive Bayes  # creatw the model  diab_model = GaussianNB()
diab_model.fit(x_train, y_train.ravel())  diab_train_predict = diab_model.predict(x_train)
from sklearn import metrics  print("Model Accuracy:
{0:.4f}".format(metrics.accuracy_score(y_train, diab_train_predict)))  print()
diab_test_predict = diab_model.predict(x_test)  from sklearn import metrics
print("Model Accuracy: {0:.4f}".format(metrics.accuracy_score(y_test,
diab_test_predict)))  print()  print("Confusion Matrix")
cm=metrics.confusion_matrix(y_test, diab_test_predict, labels=[1, 0])  df_cm =
pd.DataFrame(cm, index = [i for i in ["1","0"]],                columns = [i for i in ["Predict
1","Predict 0"]])  plt.figure(figsize = (7,5))  sns.heatmap(df_cm, annot=True)</code>
```

Python OOPS Interview Questions

1. What do you understand by object oriented programming in Python?

Object oriented programming refers to the process of solving a problem by creating objects. This approach takes into account two key factors of an object- attributes and behaviour.

2. How are classes created in Python? Give an example

```
class Node(object):
  def __init__(self):
    self.x=0
    self.y=0
```

Here Node is a class

3. What is inheritance in Object oriented programming? Give an example of multiple inheritance.

Inheritance is one of the core concepts of object-oriented programming. It is a process of deriving a class from a different class and form a hierarchy of classes that share the same attributes and methods. It is generally used for deriving different kinds of exceptions, create custom logic for existing frameworks and even map domain models for database.

Example

```
class Node(object):
  def __init__(self):
    self.x=0
    self.y=0
```

Here class Node inherits from the object class.

Wish to upskill? Take up a data science course and learn now!

4. What is multi-level inheritance? Give an example for multi-level inheritance?

<code>If class A inherits from B and C inherits from A it's called multilevel inheritance. class B(object): def __init__(self): self.b=0 class A(B): def __init__(self): self.a=0 class C(A): def __init__(self): self.c=0</code>

Python Programs for Interview

1. How can you find the minimum and maximum values present in a tuple?

Solution ->

We can use the min() function on top of the tuple to find out the minimum value present in the tuple:

<code>tup1=(1,2,3,4,5) min(tup1) </code>

Output

1

We see that the minimum value present in the tuple is 1.

Analogous to the min() function is the max() function, which will help us to find out the maximum value present in the tuple:

<code>tup1=(1,2,3,4,5) max(tup1) </code>

Output

5

We see that the maximum value present in the tuple is 5

2. If you have a list like this -> [1,"a",2,"b",3,"c"]. How can you access the 2nd, 4th and 5th elements from this list?

Solution ->

We will start off by creating a tuple which will comprise of the indices of elements which we want to access:

```
indices = (1,3,4)
```

Then, we will use a for loop to go through the index values and print them out:

```
for i in indices:
    print(a[i])
```

```
a
b
3
```

Below is the entire code for the process:

<code>indices = (1,3,4) for i in indices: print(a[i])</code>

3. If you have a list like this -> [“sparta”,True,3+4j,False]. How would you reverse the elements of this list?

Solution ->

We can use the reverse() function on the list:

<code>a.reverse() a</code>

```
a.reverse()
a
```

```
[False, (3+4j), True, 'sparta']
```

4. If you have dictionary like this – > fruit={“Apple”:10,”Orange”:20,”Banana”:30,”Guava”:40}. How would you update the value of ‘Apple’ from 10 to 100?

Solution ->

This is how you can do it:

<code>fruit["Apple"]=100 fruit</code>

```
fruit["Apple"]=100
fruit
```

```
{'Apple': 100, 'Orange': 20, 'Banana': 30, 'Guava': 40}
```

Give in the name of the key inside the parenthesis and assign it a new value.

5. If you have two sets like this -> s1 = {1,2,3,4,5,6}, s2 = {5,6,7,8,9}. How would you find the common elements in these sets.

Solution ->

You can use the intersection() function to find the common elements between the two sets:

<code>s1 = {1,2,3,4,5,6} s2 = {5,6,7,8,9} s1.intersection(s2)</code>

```
s1 = {1,2,3,4,5,6}
s2 = {5,6,7,8,9}

s1.intersection(s2)
```

```
{5, 6}
```

We see that the common elements between the two sets are 5 & 6.

6. Write a program to print out the 2-table using while loop.

Solution ->

Below is the code to print out the 2-table:

Code

```
<code>i=1 n=2 while i<=10:    print(n,"*", i, "=", n*i)    i=i+1</code>
```

Output

```
i=1
n=2
while i<=10:
    print(n," * ", i, " = ",n*i)
    i=i+1
```

```
2  *  1  =  2
2  *  2  =  4
2  *  3  =  6
2  *  4  =  8
2  *  5  =  10
2  *  6  =  12
2  *  7  =  14
2  *  8  =  16
2  *  9  =  18
2  *  10  =  20
```

We start off by initializing two variables ‘i’ and ‘n’. ‘i’ is initialized to 1 and ‘n’ is initialized to ‘2’.

Inside the while loop, since the ‘i’ value goes from 1 to 10, the loop iterates 10 times.

Initially n*i is equal to 2*1, and we print out the value.

Then, ‘i’ value is incremented and n*i becomes 2*2. We go ahead and print it out.

This process goes on until i value becomes 10.

7. Write a function, which will take in a value and print out if it is even or odd.

Solution ->

The below code will do the job:

```
<code>def even_odd(x):     if x%2==0:         print(x," is even")     else:         print(x, " is odd")</code>
```

Here, we start off by creating a method, with the name 'even_odd()'. This function takes a single parameter and prints out if the number taken is even or odd.

Now, let's invoke the function:

even_odd(5)

```
even_odd(5)
```

```
5  is odd
```

We see that, when 5 is passed as a parameter into the function, we get the output -> '5 is odd'.

8. Write a python program to print the factorial of a number.

Solution ->

Below is the code to print the factorial of a number:

```
<code>factorial = 1 #check if the number is negative, positive or zero if num<0:
print("Sorry, factorial does not exist for negative numbers") elif num==0:     print("The factorial of 0 is 1") else     for i in range(1,num+1):         factorial = factorial*i     print("The factorial of",num,"is",factorial)</code>
```

We start off by taking an input which is stored in ‘num’. Then, we check if ‘num’ is less than zero and if it is actually less than 0, we print out ‘Sorry, factorial does not exist for negative numbers’.

After that, we check,if ‘num’ is equal to zero, and it that’s the case, we print out ‘The factorial of 0 is 1’.

On the other hand, if ‘num’ is greater than 1, we enter the for loop and calculate the factorial of the number.

9. Write a python program to check if the number given is a palindrome or not

Solution ->

Below is the code to Check whether the given number is palindrome or not:

```
<code>n=int(input("Enter number:")) temp=n rev=0 while(n>0)    dig=n%10 rev=rev*10+dig    n=n//10 if(temp==rev):    print("The number is a palindrome!") else: print("The number isn't a palindrome!")</code>
```

We will start off by taking an input and store it in ‘n’ and make a duplicate of it in ‘temp’. We will also initialize another variable ‘rev’ to 0.

Then, we will enter a while loop which will go on until ‘n’ becomes 0.

Inside the loop, we will start off by dividing ‘n’ with 10 and then store the remainder in ‘dig’.

Then, we will multiply ‘rev’ with 10 and then add ‘dig’ to it. This result will be stored back in ‘rev’.

Going ahead, we will divide ‘n’ by 10 and store the result back in ‘n’

Once the for loop ends, we will compare the values of ‘rev’ and ‘temp’. If they are equal, we will print ‘The number is a palindrome’, else we will print ‘The number isn’t a palindrome’.

10. Write a python program to print the following pattern ->

1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

Solution ->

Below is the code to print this pattern:

```
<code>#10 is the total number to print for num in range(6):     for i in range(num):
print(num,end=" ")#print number     #new line after each row to display pattern correctly
print("\n")</code>
```

We are solving the problem with the help of nested for loop. We will have an outer for loop, which goes from 1 to 5. Then, we have an inner for loop, which would print the respective numbers.

11. Pattern questions. Print the following pattern

#

# #

# # #

# # # #

# # # # #

Solution –>

```
<code>def pattern_1(num):  # outer loop handles the number of rows  # inner loop handles the number of columns  # n is the number of rows.  for i in range(0, n):  # value of j depends on i  for j in range(0, i+1):  # printing hashes  print("#",end="")  # ending line after each row  print("\r")  num = int(input("Enter the number of rows in pattern: ")) pattern_1(num)</code>
```

12. Print the following pattern

```
 #
   # #
  # # #
 # # # #
# # # # #
```

Solution –>

```
<code>  Code: def pattern_2(num):  # define the number of spaces  k = 2*num - 2  # outer loop always handles the number of rows  # let us use the inner loop to control the number of spaces  # we need the number of spaces as maximum initially and then decrement it after every iteration  for i in range(0, num):  for j in range(0, k):  print(end=" ")  # decrementing k after each loop  k = k - 2  # reinitializing the inner loop to keep a track of the number of columns  # similar to pattern_1 function  for j in range(0, i+1):  print("# ", end="")  # ending line after each row  print("\r")  num = int(input("Enter the number of rows in pattern: ")) pattern_2(num)</code>
```

13. Print the following pattern:

```
0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
```

Solution –>

<code>Code: def pattern_3(num): # initialising starting number number = 1 # outer loop always handles the number of rows # let us use the inner loop to control the number for i in range(0, num): # re assigning number after every iteration # ensure the column starts from 0 number = 0 # inner loop to handle number of columns for j in range(0, i+1): # printing number print(number, end=" ") # increment number column wise number = number + 1 # ending line after each row print("\r") num = int(input("Enter the number of rows in pattern: ")) pattern_3(num)</code>

14. Print the following pattern:

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

Solution –>

<code>Code: def pattern_4(num): # initialising starting number number = 1 # outer loop always handles the number of rows # let us use the inner loop to control the number for i in range(0, num): # commenting the reinitialization part ensure that numbers are printed continuously # ensure the column starts from 0 number = 0 # inner loop to handle number of columns for j in range(0, i+1): # printing number print(number, end=" ") # increment number column wise number = number + 1 # ending line after each row print("\r") num = int(input("Enter the number of rows in pattern: ")) pattern_4(num)</code>

15. Print the following pattern:

A

B B

C C C

D D D D

Solution –>

<code>def pattern_5(num): # initializing value of A as 65 # ASCII value equivalent number = 65 # outer loop always handles the number of rows for i in range(0, num): # inner loop handles the number of columns for j in range(0, i+1): # finding the ascii equivalent of the number char = chr(number) # printing char value print(char, end=" ") # incrementing number number = number + 1 # ending line after each row print("\r") num = int(input("Enter the number of rows in pattern: ")) pattern_5(num)</code>

16. Print the following pattern:

A

B C

D E F

G H I J

K L M N O

P Q R S T U

Solution –>

<code>def pattern_6(num): # initializing value equivalent to 'A' in ASCII # ASCII value number = 65 # outer loop always handles the number of rows for i in range(0, num): # inner loop to handle number of columns # values changing acc. to outer loop for j in range(0, i+1): # explicit conversion of int to char # returns character equivalent to ASCII. char = chr(number) # printing char value print(char, end=" ") # printing the next character by incrementing number = number +1 # ending line after each row print("\r") num = int(input("enter the number of rows in the pattern: ")) pattern_6(num)</code>

17. Print the following pattern

```
#
 # #
 # # #
# # # #
# # # # #
```

Solution –>

<code>Code: def pattern_7(num): # number of spaces is a function of the input num k = 2*num - 2 # outer loop always handle the number of rows for i in range(0, num): # inner loop used to handle the number of spaces for j in range(0, k): print(end=" ") # the variable holding information about number of spaces # is decremented after every iteration k = k - 1 # inner loop reinitialized to handle the number of columns for j in range(0, i+1): # printing hash print("# ", end="") # ending line after each row print("\r") num = int(input("Enter the number of rows: ")) pattern_7(n)</code>

18. Given the below dataframes form a single dataframe by vertical stacking.

```
   col1 col2
0     1    A
1     2    B
2     3    C
   col1 col2
0     4    D
1     5    E
2     6    F
```

We use the pd.concat and axis as 0 to stack them horizontally.

Code

<code>import pandas as pd d={"col1":[1,2,3],"col2":['A','B','C']} df1=pd.DataFrame(d) d={"col1":[4,5,6],"col2":['D','E','F']} df2=pd.DataFrame(d) d_new=pd.comcat([df1,df2],axis=0) d_new</code>

Output

```
import pandas as pd
d={"col1":[1,2,3],"col2":['A','B','C']}
df1=pd.DataFrame(d)

d={"col1":[4,5,6],"col2":['D','E','F']}
df2=pd.DataFrame(d)
d_new=pd.concat([df1,df2],axis=0)
d_new
```

	col1	col2
0	1	A
1	2	B
2	3	C
0	4	D
1	5	E
2	6	F

19. Given the below dataframes stack them horizontally to form a single data frame.

```
   col1 col2
0     1    A
1     2    B
2     3    C
   col1 col2
0     4    D
1     5    E
2     6    F
```

We use the pd.concat and axis as 0 to stack them horizontally.

Code

```
<code>import pandas as pd d={"col1":[1,2,3],"col2":['A','B','C']} df1=pd.DataFrame(d)
d={"col1":[4,5,6],"col2":['D','E','F']} df2=pd.DataFrame(d)
d_new=pd.comcat([df1,df2],axis=1) d_new</code>
```

Output

```
import pandas as pd
d={"col1":[1,2,3],"col2":['A','B','C']}
df1=pd.DataFrame(d)

d={"col1":[4,5,6],"col2":['D','E','F']}
df2=pd.DataFrame(d)
d_new=pd.concat([df1,df2],axis=1)
d_new
```

	col1	col2	col1	col2
0	1	A	4	D
1	2	B	5	E
2	3	C	6	F

20. If you have a dictionary like this -> d1={“k1”:10,”k2”:20,”k3”:30}. How would you increment values of all the keys ?

<code>d1={"k1":10,"k2":20,"k3":30} for i in d1.keys(): d1[i]=d1[i]+1</code>

21. How can you get a random number in python?

Ans. To generate a random, we use a random module of python. Here are some examples To generate a floating-point number from 0-1

<code>import random n = random.random() print(n) To generate a integer between