



Data Science with Machine Learning – Python

Interview Questions

Starts with statistics then goes towards
Core Python followed by numpy to pandas to scipy and sklearn

VISHWANATHAN NARAYANAN



Data Science with Machine Learning – Python Interview Questions

By
Vishwanathan Narayanan



FIRST EDITION 2019

Copyright © BPB Publication, INDIA

ISBN: 978-93-88176-63-7

All Rights Reserved. No part of this publication can be stored in a retrieval system or reproduced in any form or by any means without the prior written permission of the publishers

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The Author and Publisher of this book have tried their best to ensure that the programmes, procedures and functions described in the book are correct. However, the author and the publishers make no warranty of any kind, expressed or implied, with regard to these programmes or the documentation contained in the book. The author and publisher shall not be liable in any event of any damages, incidental or consequential, in connection with, or arising out of the furnishing, performance or use of these programmes, procedures and functions. Product name mentioned are used for identification purposes only and may be trademarks of their respective companies.

All trademarks referred to in the book are acknowledged as properties of their respective owners.

Distributors:

BPB PUBLICATIONS

20, Ansari Road, Darya Ganj
New Delhi-110002
Ph: 23254990/23254991

BPB BOOK CENTRE

376 Old Lajpat Rai Market,
Delhi-110006
Ph: 23861747

MICRO MEDIA

Shop No. 5, Mahendra Chambers,
150 DN Rd. Next to Capital Cinema,
V.T. (C.S.T.) Station, MUMBAI-400 001
Ph: 22078296/22078297

DECCAN AGENCIES

4-3-329, Bank Street,
Hyderabad-500195
Ph: 24756967/24756400

Published by Manish Jain for BPB Publications, 20, Ansari Road, Darya Ganj, New Delhi-110002 and Printed by Repro India Pvt Ltd, Mumbai

Preface

Data science is one of the hottest topics mainly because of the application areas it is involved and things which were once upon of time, impossible with earlier software has been made easy. This book tries to comprehend the ocean of data science into small book which is mainly intended to be used as last minute revision. Before interview, all the important concepts have been given in simple and understand format.

This book tries to include various terminologies and logic used both as a part of Data Science and Machine learning for last minute revision. As such you can say that this book acts as a companion whenever you want to go for interview.

Simple to use words have been used in the given answers for the questions to help ease of remembering and representation of same. Examples where ever deemed necessary have been provided so that same can be used while giving answers in interview. Author tried to consolidate whatever he came across, on multiple interviews that he attended and put the same in words so that it becomes easy for the reader of the book to give direction on how the interview would be.

With the number of data science jobs increasing, Author is sure that everyone who wants to pursue this field would like to keep this book as a constant companion. Soon, Author will be coming shortly with a new book on R too, so that it makes a complete data science stack.

Happy reading to all the readers, your feedback is highly appreciated.

Foreword

It is not wrong to say that today's dynamic world is driven totally by statistics. With decision making becoming important in being successful the use of software this task has become common. Thanks to the advancement made with respect to technology. While software application always existed for doing the above task , the volume and ability of software programmes to represent complex equation related to statistics and probability was limited.

Thanks to pandas, numpy, scipy and sklearn modules of Python , the above problem faced has been removed to a great extent and the problem is no more a challenge. With complex mathematical concepts easily convertible to algorithms the life of data scientist and analyst has become quite easy.

This book is mainly intended to help people represent their answer in a sensible way to the interviewer. The answers have been carefully rendered in a way to make things quite simple and yet represent the seriousness and complexity of matter. Since data science is incomplete without mathematics we have also included a part of the book dedicated to statistics.

Python has already taught us that small code does not mean lesser powerful the same concept has been adopted to keep the book a powerful weapon for any one attending interview.

Dedication

Dedicated to Pratyangira, Bala, Durga, Mom, Dad, Chitti my aunt, my sister Ishwarya, Sridhar my brother in law and to all my mentors especially Shiv without whom this book would still be a dream. Also the support extended by Shyam Sir, Khadak and BPB Publications is very much appreciated.

Durga has been a great inspiration for this book. She has always been and will me my encouragement to write more books.

Also remember Sudarshan as a friend in need.

Also dedicated to my students from whom I equally learned as I taught them.

Along with all the blessing of almighty is also remembered here without which even a blade of grass does not move

Contents

Preface iii

Foreword v

Dedication vii

Data Science Basic Questions and Terms	1
1. Explain the steps involved in data science?	1
2. Explain variable and different types of variables?	2
3. Explain Categorical measurement?	3
4. Explain Binary variables?	3
5. Explain Nominal measurement?	3
6. Explain Ordinal variable?	3
7. Explain Continuous variables?	3
8. Explain Discrete variables?	3
9. Is it possible to convert continuous values to discrete and vice versa?	3
10. What are interval variables?	3
11. What are ratio variables?	4
12. What are Univariate and Bivariate variables?	4
13. What is measurement error?	4
14. Explain Validity?	4
15. Explain Reliability?	4
16. What are the different ways to test hypotheses?	4
17. Explain the different types of variation?	5
18. Explain repeated-measures design?	5
19. What is independent design?	5
20. Explain the role of randomization w.r.t variation?	6
21. Explain various summary measures?	6
22. Explain alternate hypotheses and null hypotheses?	8
23. What is p value?	8
24. What happens when null hypotheses is rejected?	8
25. Explain directional and non-directional hypotheses?	8

26. Explain fit of model?	9
27. What is relation between sample and population?	9
28. What is estimation?	9
29. Explain deviation score?	9
30. What is variance?	9
31. Explain Standard deviation?	10
32. Explain standard error?	10
33. What is precision?	11
34. Explain confidence intervals?	11
35. Explain confidence level?	11
36. Explain alpha?	11
37. Explain Beta?	11
38. Explain Accuracy?	11
39. Explain Bias?	11
40. What is central limit theorem?	12
41. Explain Absolute value?	12
42. What is degree of freedom?	12
43. Explain cluster sampling?	12
44. Explain Correlation coefficients?	13
45. Explain sample space?	13
46. What is non parametric algorithm?	13
47. How can learning be classified?	13
48. What is classification?	14
49. Explain the steps involved in classification?	14
50. What is regression?	14
51. Explain the similarities and differences between Classification and Regression?	15
52. Explain various terms encountered during classification algorithm? 15	
53. What is logistic regression?	16
54. Explain Naïve Bayes?	16
55. What is Stochastic Gradient Descent?	17
56. Explain decision tree algorithm?	17
57. What is Gini index?	18

58.	Is Gini index the only means which can be used in decision tree?	19
59.	What is Pruning w.r.t. decision tree?	19
60.	What is random forest?	19
61.	Explain the difference between Random forest and decision tree?	20
62.	What is overfitting and underfitting?	20
63.	Explain KNN (K Nearest Neighbor. steps involved, advantage and disadvantage?	20
64.	Explain selection bias?	21
67.	What is Re sampling?	21
68.	Explain tail?	21
69.	Explain the difference between one way test and two way test?	22
70.	Explain degree of freedom?	22
71.	What is predictive modeling?	22
74.	What is Convolutional Neural Network ?	23

Python Programming Questions 25

1.	Is Python Object oriented?	25
2.	Is Python case sensitive?	25
3.	What kind of language is python?	25
4.	What are different versions of Python?	25
5.	Explain different implementations of Python?	26
6.	Is Python loosely typed?	27
7.	How to start a new block in python?	27
8.	How to get data type of particular variable?	27
9.	How many ways can python program be run?	27
10.	Explain the importance of Pylint and Pychecker?	27
11.	Explain Zen of Python?	27
12.	How to print Zen in python?	28
13.	Explain Python data types?	28
14.	How can we switch variables in Python?	29
15.	What is the use of pass statement in python?	29

16.	Is Python pass by value or pass by reference?	29
17.	Does python supports chained operations?	29
18.	Explain ALL and ANY?	30
19.	Explain the difference between IS and ==?	30
20.	Explain supported collection of datatype w.r.t. Python?	30
21.	Create a simple number list?	31
22.	Can you create nested list?	31
23.	Explain CRUD (Create, Update and Delete. operations from list?)	32
24.	Explain operations in dictionary?	33
25.	Explain operation with tuples?	34
26.	Explain del?	34
27.	If del can remove variable can it remove tuple variable?	35
28.	Delete last element in a list?	35
29.	Predict the output of following code	35
30.	What do you mean by list comprehension?	35
31.	Explain the preferred way for looping through list?	35
32.	Find the reverse of the dictionary?	36
33.	How to sort dictionary by value?	36
34.	What is the use of shuffle function?	36
35.	What is the preferred way to get a value based on key in Python?	36
36.	Explain alternate way of merging 2 or more dictionaries without using update method?	37
37.	What is the preferred way of fetching last element/second last and so on from a list?	37
38.	What is the preferred way for reversing a list?	37
39.	Explain various string utility functions in Python?	38
40.	How to check whether two strings are equal?	38
41.	Can string use single quote or double quote?	39
42.	Explain type conversions on collection types?	39
43.	Explain set theory operations supported by set data type?	39
44.	Explain frozenset?	40
45.	Explain functionsin Python?	40
46.	What is a Boolean function?	40

47.	Can we specify data type for arguments as well as return types in python?	41
48.	Explain variable arguments?	41
49.	Write a program to find occurrences or count of characters in given word?	42
50.	What is **kwargs?	42
51.	Write a simple Lambda expression?	42
52.	Lambda forms in Python contain statements? True or False?	42
53.	Explain filter function?	43
54.	Explain steps involved in reading and writing a file?	43
55.	Explain the term “with statement”?	43
56.	Explain the preferred way of reading a big file?	44
57.	Explain modules in python?	44
58.	Explain different ways of importing modules?	44
59.	Can we create our own module?	45
60.	Explain in brief about os module and its corresponding functions?	45
61.	Using os module print the directory structure?	46
62.	Explain dir function?	46
63.	Explain exception handling in python?	47
64.	How to create user defined exception?	48
65.	What is the use of raise statement?	48
66.	How to create own class in Python? Explain constructors?	48
67.	Is it necessary to have the first argument of class function as self? Can't we rename it to any other variable?	49
68.	Explain inheritance in Python?	49
69.	How to determine whether a particular class is sub class?	50
70.	Does python support multiple inheritance?	50
71.	How is diamond problem resolved in case of Python?	50
72.	Does python supports private method and variables?	51
73.	Can __ be used for other purpose than creating private variables or functions?	51
74.	Does python support abstract classes?	52
75.	Differentiate between static methods and class methods in Python?	52
76.	What are named tuple?	53

77. How to sort using lamdas?	53
78. Explain Generators?	53
79. What is generator expression?	54
80. When python programme exits all the memory is released? Say true or false?	54
81. Can a function be passed as parameter to another function?	54
82. Can a function be retuned as result from another function?	55
83. Explain decorator function?	55
84. How can we represent big text in Python?	56
85. What is PEP 8?	57
86. What is anaconda?	57
87. How to install external modules?	57
88. What is Jupyter notebook?	57
89. What is pickling and unpickling?	58
90. Explain importance of setup.py?	58
91. Is it possible to make connections to database using Python?	58
92. Explain meta programming?	60
93. Explain python memory model?	62
Numpy Interview Questions	63
1. What is numpy?	63
2. How to install numpy?	63
3. How to create single dimension numpy array?	63
4. Explain different attributes provided by numpy?	63
5. Explain some utility methods provided by numpy for creating different elements?	64
6. How can we change shape of an object?	65
7. Which all data types are supported in Python?	66
8. Explain various simple mathematical operations which can be done on numpy?	66
9. Explain slicing operation in numpy?	67
10. Explain boolean indexing?	68
11. Perform matrix multiplication using numpy?	69
12. Explain various functions available with numpy?	69

13.	What is broadcast?	70
14.	Explain rules of broadcasting?	71
15.	Explain some statistical measures supported by numpy?	71
16.	Explain functions available in numpy.linalg?	73
17.	How to save numpy data from memory to flat file?	75
18.	What is the use of where and extract?	75
19.	What is the use of ndenumerate?	76
20.	Explain how can we draw a histogram using numpy?	77

Pandas Interview Questions 79

1.	What is Pandas?	79
2.	How does Pandas represent data?	79
3.	How to create Series?	80
4.	How to create Data frame?	80
5.	How are missing values represented in data frame?	81
6.	Explain how indexes can be created?	81
7.	Explain various attributes associated with series?	82
8.	Explain various statistical measures supported by pandas?	83
9.	Explain reindexing?	86
10.	Explain bfill and ffill?	87
11.	What all type of iterations are provided in pandas data frame?	88
12.	Explain how sorting is supported in pandas?	91
13.	How to override default reload option in pandas?	94
14.	Explain various slicing options available with pandas?	94
15.	Explain advanced statistics with pandas?	96
16.	Explain rolling function?	97
17.	How can we handle NA in pandas?	98
18.	Explain group by function?	100
19.	Explain merge functions w.r.t data frame?	101
20.	Explain concat method?	103
21.	Explain how time related range can be generated in pandas?	104
22.	Explain which all data sources can pandas retrieve values?	105
23.	How to print a histogram using pandas?	106

Scipy and its Applications	109
1. Explain Scipy library?	109
2. Explain how can we perform Normality Tests?	109
3. Explain how can we perform correlation test?	110
4. Explain tests pertaining to Parametric Statistical Hypothesis Tests?	111
5. Explain how to test Nonparametric Statistical Hypothesis Tests?	112
6. Implement logistic regression in python?	113
7. Explain how to implement decision tree in python?	113
8. How to implement Random forest in python?	113
9. How to implement support vector machine in python?	114
10. Which all kernels are supported by svm in python?	114
11. Implement KNN algorithm using python?	114
12. How to select k in KNN algorithm?	115
13. How to implement K means in Python?	115
14. How can accuracy of any model be calculated?	115
15. Explain regression metrics?	117
Matplotlib Samples to Remember	119
1. Explain how to draw bar plot?	111
2. How to draw histogram?	120
3. How to draw line chart?	120
4. Draw Pie chart?	121
Statistics with Excel Sheet	123
1. Does Excel has any support for statistics?	123
2. Find correlation using Excel?	124
3. How to get Histogram in excel?	124
4. Explain how to get Descriptive Statistics using Excel?	125
5. Explain how to perform Anova in excel?	126
6. Explain how to perform Rank and Percentile in excel?	127

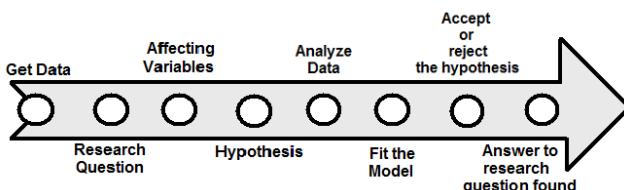
Data Science Basic Questions and Terms

Note: [Q: Question Number and A: Answer]

Q 1: Explain the steps involved in data science?

A: Following are the steps involved:

1. Get data from various data sources available.
2. Generate research question from data.
3. Identify variables present in data. Also, identify important variables or variables to be analyzed as such.
4. Generate hypothesis.
5. Analyze data using graph data like histogram for example.
6. Fit a model from analyzed data.
7. Accept or reject the hypothesis.
8. Research question answer found.



Example of above steps:

1. Get data related to temperature for India reference <https://data.gov.in/catalog/annual-and-seasonal-maximum-temperature-india>

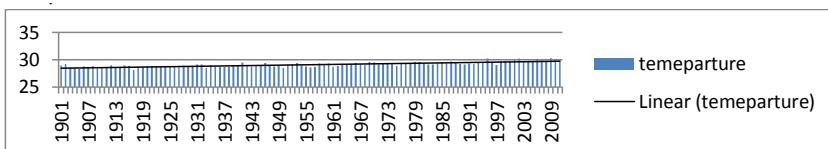
A template of data set:

"YEAR", "ANNUAL", "JAN-FEB", "MAR-MAY", "JUN-SEP", "OCT-DEC"

2 - Data Science with Machine Learning – Python Interview Questions

```
"1901","28.96","23.27","31.46","31.27","27.25"  
"1902","29.22","25.75","31.76","31.09","26.49"  
"1903","28.47","24.24","30.71","30.92","26.26"  
"1904","28.49","23.62","30.95","30.67","26.40"  
"1905","28.30","22.25","30.00","31.33","26.57"  
"1906","28.73","23.03","31.11","30.86","27.29"  
"1907","28.65","24.23","29.92","30.80","27.36"  
"1908","28.83","24.42","31.43","30.72","26.64"  
"1909","28.39","23.52","31.02","30.33","26.88"  
"1910","28.53","24.20","31.14","30.48","26.20"  
"1911","28.62","23.90","30.70","31.14","26.31"  
"1912","28.95","24.88","31.10","31.15","26.57"  
"1913","28.67","24.25","30.89","30.92","26.42"  
"1914","28.66","24.59","30.73","30.84","26.40"  
"1915","28.94","23.22","31.06","31.51","27.18"  
"1916","28.82","24.57","31.88","30.52","26.32"  
"1917","28.11","24.52","30.06","30.24","25.74"  
"1918","28.66","23.57","30.68","31.11","26.77"
```

2. Research question, is the annual temperature in India rising?
3. Variable of interest from the above data set ANNUAL.
4. Hypothesis: Temperature is rising.
5. Analyze data from the above data set.



6. Fit the model.
7. Hypothesis accepted or rejected.

Q 2: Explain variable and different types of variables?

A: Anything which keeps on changing is called variable. Variables are of different type and below are the following:

Dependant/Outcome: A variable being affected, for example annual temperature in above example.

Independent/Predictor: A variable affecting the outcome for e.g. deforestation, pollution, and so on in above example.

Q 3: Explain Categorical measurement?

A: Categorical measurement contains categories i.e. distinct entities.
Example of categories of life on earth is plants, animals, and so on.

Q 4: Explain Binary variables?

A: Binary variables are those in which only two classes exist, like live or dead male or female on or off.

Q 5: Explain Nominal measurement?

A: Nominal measurements are there more than two classes. Such categories can be numbers too.

Q 6: Explain Ordinal variable?

A: These are nominal variables which have logical order. Examples include team ranks in cricket or football, merit list of students appearing for grade students.

Q 7: Explain Continuous variables?

A: These are variables which can take can any value on the measurement scale example includes pitch of voice which can take any possible value within the range.

Q 8: Explain Discrete variables?

A: These are variables which can take fixed values in range. For example, number of customers in a bank.

Q 9: Is it possible to convert continuous values to discrete and vice versa?

A: Yes, based upon the motive of study, it is possible to convert discrete values to continuous and vice versa for example, Level of water in tank can take any value in the range and as such a continuous variable.

But we can approximate the same to three different levels like empty, full, or half empty and this now becomes discrete in nature.

Q 10: What are interval variables?

A: These are variables which are grouped on interval. **E.g.** is age can be divided in range like 1-10, 10-20, 20-30, and so on and, person with particular age would be placed in one of the above groups. When intervals are equal, they represent difference in equal property being measured.

4 – Data Science with Machine Learning – Python Interview Questions

Q 11: What are ratio variables?

A: This is sub type of interval variables where ratio of scales is used for measurement.

For e.g., Water representation in chemistry is H_2O which represent two molecules of hydrogen and one molecule of oxygen. Thus, the ratio of elements is 2: 1.

Q 12: What are Univariate and Bivariate variables?

A: **Univariate variable:** When the variable under consideration is only one then it is called univariate variable study.

Bivariate variable: Involves study of relationship between two variables.

Q 13: What is measurement error?

A: The discrepancy between the measured value and actual value in terms of number is called measurement error.

For e.g. While buying fruits from a vendor in kilograms, if we wanted 1 kilogram of fruits and the vendor's weighing machine showed 1 kilogram when we brought the same. After checking the same in another machine, if the measured value shows 0.1 kilogram less than expected then this difference is what we call as measurement error.

Q 14: Explain Validity?

A: Validity implies whether an instrument measures what it is supposed to measure.

Q 15: Explain Reliability?

A: Reliability implies whether the instrument gives consistent result across different conditions.

For example, if we test the same value twice on the same entity then the results from the instrument should remain same if it has to be reliable. Such tests are known as test-retest.

Q 16: What are the different ways to test hypotheses?

A: There are two ways in which hypotheses can be tested:

1. Correlational research

- This is also known as cross-sectional research
- This involves observing the natural pattern or occurrence to test

- Original occurrences are not manipulated
- 2. Experimental research
 - We select the variables of interest
 - Then we manipulate some aspect of the environment
 - Observe the effect on selected variable

Q 17: Explain the different types of variation?

A: There are two types in variation explained as follows:

Systematic variation:

- Introduced by experimenter
- The participants are tested under different conditions and the difference in condition is introduced by experimenter

For e.g. to test use of woolen clothes w.r.t. temperature, we can test a group of 20 people, in both hot and cold climate. Thus, the difference introduced here is in terms of temperature only.

Unsystematic variation:

- Introduced by random factors that exist between the experimental conditions.
- For e.g. To test use of woolen clothes w.r.t. temperature, we can test a group of 20 people. Of the selected set some might behave differently than expected due to factors like illness and so on.

Q 18: Explain repeated-measures design?

A: Same measure is measured under different conditions on same set participants.

The difference in two conditions can be caused by the following:

- The manipulation/changes that was carried out on the participants
- Factors that might affect the way in which a participant performs from one time to the next

Q 19: What is independent design?

A: Same measure is measured under different conditions on different set of participants:

The differences between the two conditions can be caused by the following:

6 - Data Science with Machine Learning – Python Interview Questions

- The manipulation/changes that were carried out on the participants
- Difference in nature or characteristics of participants in each case

Q 20: Explain the role of randomization w.r.t variation?

A: By using randomization we can ensure that any variation introduced, is due to changes in the conditions/variables introduced rather than any other unexpected changes during the process. Thus, it helps in removing other sources of systematic variation.

Q 21: Explain various summary measures.

A: **Mode:** Represents the value/score which occurs most frequently in data set.

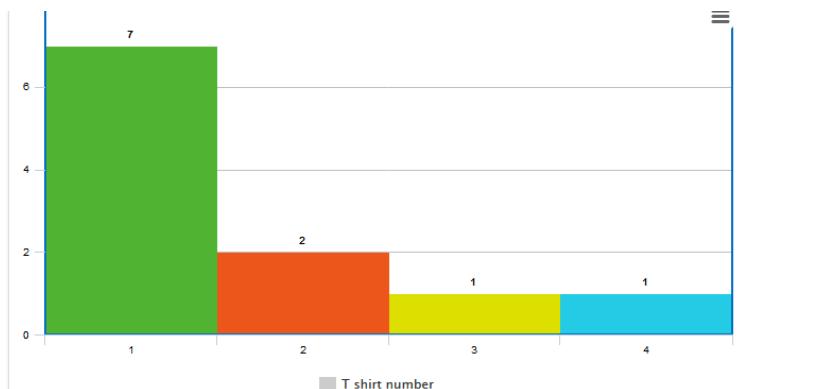
For example: In the values of occurrences of goal in football by players having T-shirt numbers is as follows:

1, 2, 2, 3, 4, 1, 1, 1, 1, 1, 1

If we arrange it in form of table:

T shirt number	Frequency
1	7
2	2
3	1
4	1

Thus the mode in above example is 1 which happens to have maximum frequency of 7. This can be easily determined from histogram as shown in the following screenshot:



Median: This is the middle value which is obtained by ordering the values/scores in ascending order. If the middle value happens to have two numbers then the average is taken as such:

For example:

Median of 2, 3, 4, 5, 6 happens to be 4.

Median of 2, 3, 4, 5 happens to be average of 3 and 4 which is 3.5.

Median is least affected by outliers.

Mean: Represents mathematical average which is sum of all the elements divided by number of elements.

For example, average strike rate of the batsmen in the game of cricket is the average of strike rates in individual matches.

Range of scores: Subtraction between the maximum value and minimum value in range is called range of scores. This indicates dispersion.

Trimmed mean: Represents the mean after removing extreme cases from both the end i.e. from minimum and maximum end. Both the minimum and maximum values may represent values which are not normal and hence represent outlier. So, while finding trimmed mean, we specify the percentage of values to be ignored from both the ends. Hence, trimmed mean gives better representation of data excluding outlier.

Interquartile range (IQR)

It is a measure of variability, based on dividing a data set into quartiles.

Quartiles are the three values that split the sorted data into four equal parts:

- Q1 is the *middle* value in the first half of the rank-ordered data set
- Q2 is the median value in the set
- Q3 is the *middle* value in the second half of the rank-ordered data set

The interquartile range is equal to Q3 minus Q1.

The lower quartile is the median of the lower half of the data.

The upper quartile is the median of the upper half of the data.

Mean absolute deviation: Represents the mean of the absolute value of the deviations from the mean.

Mean absolute deviations from median: Represented by absolute value of the deviations from the median.

Outliers: Represents the value which are not normal or within the range and hence data which is corrupted at the time of capture or due to some other reasons. Since it affects all while finding the mean and other summary values they need to represent this data properly.

Q 22: Explain alternate hypotheses and null hypotheses.

A: **Alternative hypotheses:**

- Also called experimental hypotheses
- Denoted by H_1
- It assumes that effect as per prediction would exist in the conclusion

Null hypotheses:

- Denoted by H_0
- It assumes that effect as per prediction would not exist in the conclusion
- Thus, this represents opposite of alternative hypotheses

Q 23: What is p value?

A: It measures the strength of evidence in support of null hypotheses. If this value is less than significance level then null hypotheses is accepted, else rejected.

The range of values that leads the researcher to accept the null hypothesis is called the region of acceptance.

The region other than acceptance is called region of rejection.

Q 24: What happens when null hypotheses is rejected?

A: When a null hypothesis is rejected, it becomes Type 1 error. The probability of Type 1 error occurring is called significance level.

Q 25: Explain directional and non-directional hypotheses.

A: **Directional hypotheses:**

- Gives an indication whether the effect which is being studied would grow positively or negatively

- One tailed test is generally used for such cases

Non directional hypotheses:

- Does not give indication of whether the effect which is being studied would grow positively or negatively
- Two tailed test is generally used for such cases

Q 26: Explain fit of model?

A: It represents the degree till which the determined statistical model represents the data.

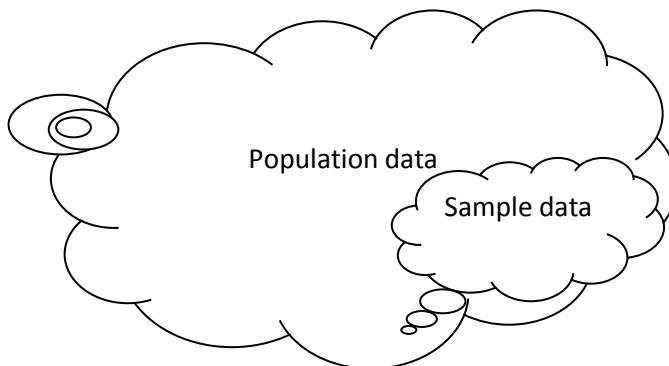
Fit of model can represent either under fit, over fit, or perfect fit as such.

A model which is good fit would have low variance between the calculated value and measured value.

Q 27: What is relation between sample and population?

A: Samples are subset or part of original data or population. If population is very big hence performing analysis on the whole population as such is not easy. Hence, a subset of population data is taken which is known as sample data.

Whether the sample data is true representation of original data set considered is determined with the help of estimation, confidence interval, and so on.



Q 28: What is estimation?

A: By using information available from sample, we can make inferences w.r.t. population, which is what is known as estimation. Parameters used are mean, standard deviation, and so on.

Q 29: Explain deviation score?

A: This is defined as the difference between actual score/value and mean.

Q 30: Explain variance?

A: Variance is the average error between the mean and the measured values.

It indicates the difference between the average value calculated and the observed value as such.

It is an indication of how different individuals in group differ or vary from each other.

The population variance is given:

$$PV = \Sigma (Xi - X)^2 / N$$

- PV is the population variance
- X is the population mean
- Xi is the i^{th} element from the population
- N is the number of elements in the population

The sample variance is given by:

$$SV = \Sigma (xi - x)^2 / (n - 1)$$

- SV is the sample variance
- x is the sample mean
- xi is the i^{th} element from the sample
- n is the number of elements in the sample

Q 31: Explain Standard deviation.

A: Square root of the variance is also called standard deviation.

This is done to keep the measurement same as original one.

They indicate the nearness of the points measured w.r.t. mean.

Smaller the standard deviation will be more nearer to mean and vice versa.

Q 32: Explain standard error.

A: Standard error indicates how well a sample represents the original population. When we break the original population into various small samples, we would like to know the difference between the sample considered and original population. This is represented by standard error.

The smaller the standard error, the closer or true representation of original population.

$$SE = \sqrt{\frac{\sum (\text{sample mean} - \text{overall population mean})^2}{\text{number of samples}}}$$

Q 33: What is precision?

- A: It refers to the closeness between estimates from different samples. Thus indicating the opposite of standard error and are inversely related to Standard error.

Q 34: Explain confidence intervals.

- A: This indicates the boundaries in which the mean value will fall. It is a range of scores constructed, such that the population mean will fall within. They are limits constructed such that for a certain percentage of the time the true value of the population mean will fall within these range.

Q 35: Explain confidence level.

Refers to the percentage of all possible samples that can be expected to include the true population parameter.

Q 36: Explain alpha.

- A: Alpha is defined as 1-confidence interval. This implies probability that the true value remains outside the confidence interval. If confidence interval is 99% then alpha is 1-99% which is 0.01.

Q 37: Explain Beta.

- A: The probability of committing Type 2 error is called beta. Type 2 error is one in which a rejected null hypotheses is accepted. The probability of avoiding Type 2 error is called power of test.

Q 38: Explain Accuracy.

- A: It indicates how much does sample value or parameters matches with the population statistic. If the value of mean of both sample and population are exactly equal then we can say that the sample is fully accurate. If not fully equal then we say that sample is accurate by n limit where n is the difference between sample and population.

Q 39: Explain Bias.

- A: Bias indicates whether the estimation of sample is over fit or under fit w.r.t. population data. For e.g. if the population mean is 4 and

sample mean calculated is 3 then this is under estimate bias. Such estimate in which both sample and population parameter is not equal are called bias estimate.

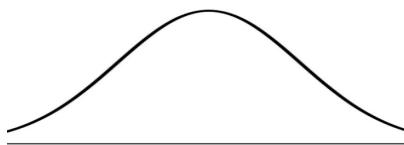
Q 40: What is central limit theorem?

A: It states that the distribution of the mean of any independent, random variable can be approximated to normal if the sample size is large enough.

Generally, the sample size of above 30 or sometimes 40 is taken as reference.

This allows us to approximate bigger samples to normal distribution without having to take hundreds or thousands of distribution.

Standard Normal distribution is preferred as such because mean is equal to zero and variance is one.



Q 41: Explain Absolute value?

A: Absolute value is positive value or magnitude irrespective of its initial sign.

Q 42: What is degree of freedom?

A: It is equal to the number of independent observations in a sample minus number of population parameters to be estimated.

Q 43: Explain cluster sampling.

A: In this method the number of clusters or groups to be formed as pre decided (generally denoted by N) from population data.



The number of elements in each cluster is known and each element from population data is assigned to one cluster. For e.g. clustering can be done on attributes like customer state.

Sampling can be further classified as:

- **One-stage sampling:** All of the elements within selected clusters are included in the sample.
- **Two-stage sampling:** A subset of elements within selected clusters is randomly selected.

Q 44: Explain Correlation coefficients?

- A: Correlation indicates the relationship between two variables.
- As such variables can be positively correlated in which positive change in one variable effect the other variable positively.
 - If variables are negatively correlated the positive change in one variable affects the other variable negatively.
 - The formula for above is given by:
$$\text{Correlation co-efficient} = \frac{\sum (xy)}{\sqrt{[\sum x^2] * [\sum y^2]}},$$
 where x and y are variables under consideration.
 - The value of correlation co-efficient ranges from -1 to +1.

Q 45: Explain sample space.

- A: The outcomes of any statistical experiment are denoted by sample space. Any outcome from such space is called sample point. One or more sample point is called event. When events do not have any sample point in common they are known as mutually exclusive event.

Q 46: What is non parametric algorithm?

- A: Non parametric algorithm does not make any assumptions on data distribution.

Q 47: How can learning be classified?

A: Following are the classifications:

- **Supervised:**
 - o Data is clearly labeled and the algorithms learn to predict the output from the input data
 - o Offline analysis of data possible
 - o The number of classes are predefined

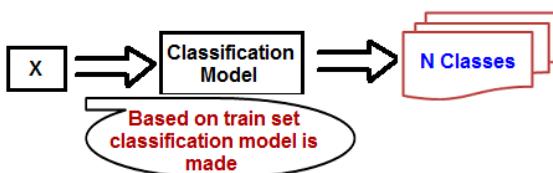
- o Accuracy is high
- o Examples include classification and regression
- **Unsupervised:**
 - o Much amount of data is unlabeled and the algorithms learn to inherent structure from the input data
 - o Analysis is on real time data
 - o Number of classes may be unknown
 - o Accuracy ranges from moderate to high
 - o Examples include Clustering and Association
- **Semi-supervised:**
 - o A mixture of data having label and no labels forms this one
 - o Can be considered as intermediate to above two

Q 48: What is classification?

A: Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

The output variables are often called labels or categories.

The mapping function predicts the class or category for a given observation.



Q 49: Explain the steps involved in classification.



Q 50: What is regression?

A: Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).

A continuous output variable can be a real-value, such as an integer or floating point value and be any measures such as size amount, and so on.

Error indicates the difference between actual value and predicted value in case of regression mostly root mean square error is used w.r.t. regression.



Q 51: Explain the similarities and differences between Classification and Regression.

A: Differences between classification and regression:

Classification	Regression
Predicts classes	Predicts quantity
Can be evaluated using accuracy	Can be evaluated using RMSE (root mean square error)
Discrete class label is the output	Continuous quantity is the output

The similarities are that both try to make prediction based on input values.

Q 52: Explain various terms encountered during classification algorithm.

Classifier is an algorithm that maps the input data to a specific category.

Classification model: After analyzing input data, a classification model is made. The model can take new data as input and predict the class labels/categories.

Feature: A feature is an individual measurable property of a phenomenon being observed.

Binary Classification: Classification task with two possible outcomes.

Multi class classification: Classification with more than two classes is involved. In multi class classification each sample is assigned to one and only one target label.

E.g. an item can be classified as fruit or vegetable not both.

Multi label classification: Classification task where each sample is mapped to a set of target labels having more than one class.

Example: A fruit can be classified as sweet and sour both.

Q 53: What is logistic regression?

A: Logistic regression also known as logit model is most useful for understanding the influence of several independent variables on a single outcome variable.

Function representation:

$$p = 1 / 1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)}$$

It is a kind of predictive analysis.

In this algorithm, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

Disadvantages:

Assumes all predictors are independent of each other which may not always be true.

Works only when the predicted or output variable is binary.

Q 54: Explain Naïve Bayes.

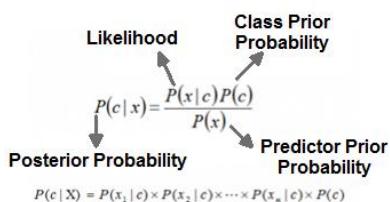
A: Naive Bayes algorithm based on Bayes' theorem with the assumption of independence between every pair of features. This implies the predictors are independent of each other.

Let $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).

$P(c)$ is the prior probability of class.

$P(x|c)$ is the likelihood which is the probability of predictor given class.

$P(x)$ is the prior probability of predictor.



Advantages:

- Are extremely fast
- Training set required is less
- Performs well for categorical values

Disadvantages:

- Zero frequency problems will occur if some of the categories is missed and as a result the effect of same would be neglected
- Assumes independence of predictors which is not always true in real world

Q 55: What is Stochastic Gradient Descent?

A: Gradient is an effective method to calculate parameters or co-efficient which would cause the cost to minimum. The process involves assuming some value as co-efficient and keeping on calculating the slope and adjusting the co-efficient till a minimum cost is found.

But when the logic is applied on very large data set, it becomes slow. Stochastic Gradient Descent is used for this purpose. In this, calculation is done based on batch instances rather than at the end.

Advantages:

- The algorithm is extremely effective
- Easy to use

Disadvantages:

- Requires large number of parameters.

Q 56: Explain decision tree algorithm.

A: Decision tree

Generates the set of rules which depends on attributes or features and based upon classes.

A decision tree is an inverse tree.

The node at the top is called root node and the one without any child is called leaves.

Each part of decision tree represents something which is indicated:

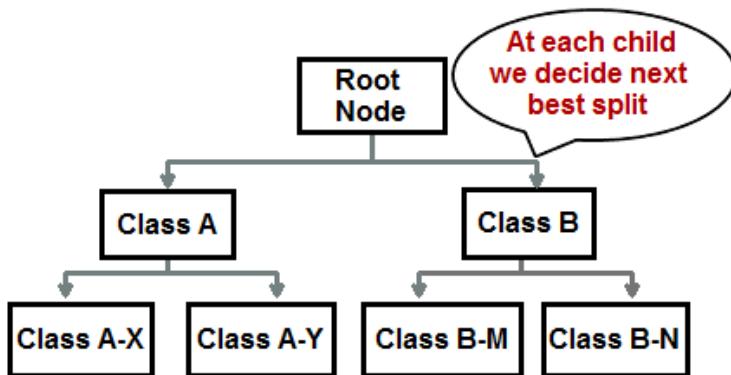
- Each attribute/feature is represented by node
- Rule or decision is represented by link

- Leaf node or node without children represents an the output class

It uses a recursive approach.

The aim is to divide the given data to various groups or classes based upon cost function.

Classification and Regression trees are also known as **CART**.



Advantages:

- Requires small data preparation
- Simple to understand and visualize
- Can handle dataset which can contain numbers
- Can handle dataset which can contain data in the form of categories

Disadvantages:

Outliers can have severe effect on output. Changes in input data does have an important effect on output.

Q 57: What is Gini index?

A: The Gini index is the cost function used to evaluate splits in the dataset.

It gives the measure of impurity.

How effective is the split of data based upon features can be easily determined by Gini index.

A Gini score of 0 is ideal expected result while one having half-half probability is not preferred.

Number of class values

```
probProp=count(class_value) / count(rows)  
gini_index = (1.0 - sum(probProp * probProp)) * (group_size/total_samples)
```

Q 58: Is Gini index the only means which can be used in decision tree?

A: Entropy is another measure which can be used with decision tree.

Q 59: What is Pruning w.r.t. decision tree?

A: Pruning refers to the removal of branches having low importance in terms of classification or regression. The advantages are reduced complexity and reducing over fitting.

The algorithm basically calculates whether removing a particular node causes an impact on prediction precision of the tree.

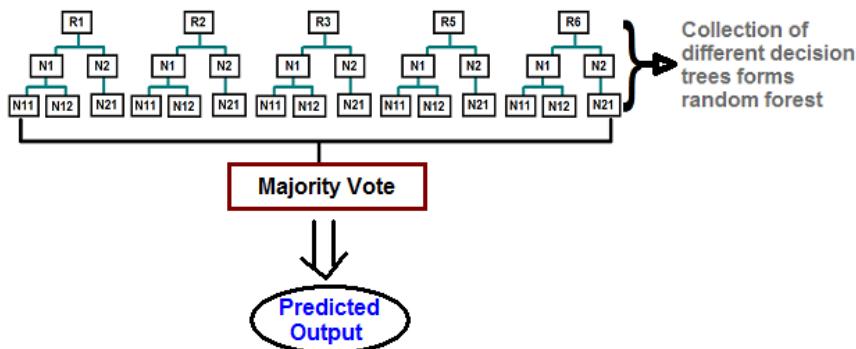
Q 60: What is random forest?

A: Random forest is a meta-estimator that fits a number of decision trees.

It makes use of average to improve the ability to predict the model.

The output from each decision tree is used to finally predict the final model and for doing this, it makes use of majority voting. The data is classified on the class which has majority voting.

To generate this forest, sub samples are used and using sub samples from original data set sub trees are created.



Advantages:

Over fitting is reduced.

Disadvantages:

It's slow as real time updates take time.

Q 61: Explain the difference between Random forest and decision tree.

A:

Decision tree	Random Forest
Is a subpart of Random forest	Consists of lot of decision tree
Over fitting can occur	Over fitting is avoided
Faster	Slower
Easy to interpret	Difficult to interpret

Q 62: What is overfitting and underfitting?

A: Following is the explanation for overfitting and underfitting:

Overfitting:

- Refers to a model which performs very well w.r.t. training set. Such model is able to predict the noise and randomness w.r.t. training set very well but would not perform well when it comes to newly arrived data.
- Occurs with nonparametric and nonlinear models.

Underfitting:

- Refers to a model that does not perform well both with training set and newly arrived data.
- An underfitted model would be directly rejected due to its performance via benchmark and accuracy.

Q 63: Explain KNN (K Nearest Neighbor) steps involved, advantage and disadvantage.

A: KNN abbreviation is K Nearest Neighbor and is explained down below:

- Not an eager learner i.e. learns lazily
- It stores instances of training set.
- The output is determined by majority vote which is given by k nearest neighbors of each point.

Steps involved are as follows:

1. Calculate the distance.

2. Find the closest neighbors.
3. Vote for labels.

Advantages:

- Much faster.
- No need to train a model for generalization.
- Can be used for nonlinear data and regression problems.

Disadvantages:

- Large memory foot print.
- Performance decreases with increase in number of dimensions.

Q 64: Explain selection bias.

A: Selection bias is the bias introduced by the selection of individuals, groups or data for analysis in such a way that proper randomization is not achieved, thereby ensuring that the sample obtained is not representative of the population intended to be analyzed.

They are systematic in nature and produced by the measurement or sampling process.

Q 65: What does selection bias indicate w.r.t. algorithm?

A: It indicates that some important variable has been left off or not considered during the algorithm.

Q 66: What is Bootstrap sample?

A: Bootstrap sample is used to estimate the distribution. A sample taken with replacement from an observed data set is called bootstrap sample.



Q 67: What is Resampling?

A: Resampling combines shuffling and bootstrapping.

Q 68: Explain tail.

A: Tail is the portion of frequency distribution where extreme values are observed at a very low frequency.

Q 69: Explain the difference between one way test and two way test.

A: **One-way test**

It is the hypothesis test where results occur only in one direction.

Two-way test

It is the hypothesis test where results occur in two directions.

Q 70: Explain degree of freedom.

A: It indicates how much value from given samples are supposed to vary to give the required result.

In a sample of 5 we can say 4 values can vary if the mean is known and 4 values are known.

Q 71: What is predictive modeling?

A: Prediction modeling involves building a model which will allow predicting a result/events/behavior whether value of input variables are given. To find out the relationship of input variables and how they affect the output forms the part of regression study.

Q 72: What is time series analysis?

A: Analysis which makes use of data in a particular time interval or periods is called time series analysis. Temporal aspects continue in this type of analysis. It may involve observation of a variable at different time instances (time series data) or collecting different data at same point of time (cross sectional data) or a combination of both (pooled data).

Q 73: What is deep learning?

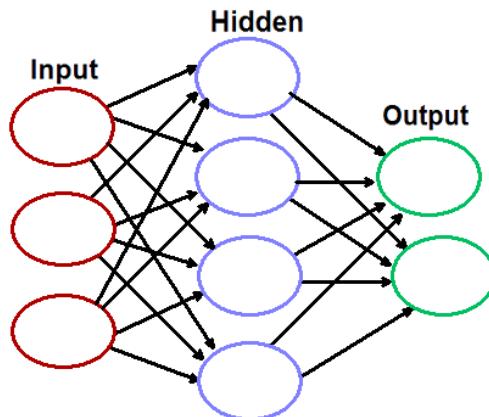
A: Deep learning makes it possible to learn by existing examples. The models are generated by computer by using existing examples.

The underlying relationship of data can be determined with the help of neural network

Neural network consists of various nodes or computers interconnected to perform complex/vast computation.

To achieve deep learning we require large amount of labeled data set as well as large computing capability.

Again, deep learning is a part of machine learning where the scale of operation is much higher.



Q 74: What is Convolutional Neural Network?

A: **CNN** or **ConvNet** stands for Convolutional neural network, is a type of learning in which model learns to perform classification of models. It eliminates the need to perform manual feature extraction. It allows this with the help of many layers many of which can be hidden.

Python Programming Questions

Q 1: Is Python Object oriented?

A: Yes, Python is an object oriented programming language since we can deal with classes and objects.

Q 2: Is Python case sensitive?

A: Yes

Q 3: What kind of language is python?

A: Python is interpreted programming language.

Q 4: What are different versions of Python?

A: Python comes with two major versions 2.x and 3.x.

There are lots of differences between both of them. Following are some of them:

Command/Syntax	Python 3	Python 2
Division 1/3	1/3	0
Print syntax	print("hello")	print "hello"
Type representation type(int)	<class 'type'>	<type 'type'>
Range generation function	Xrange	range
Unicode difference	Supports utf-8 string , bytearray and byte	Supports ASCII str() and unicode function
Preferred input function	Input	raw_input
Raising Exceptions (syntax difference)	raise MyError("some error")	raise MyError, " some error "
Handling Exceptions (syntax difference)	except MyError as err:	except MyError, err:

Command/Syntax	Python 3	Python 2
Return type as iterable objects instead of list	Supported in 3 For example: <ul style="list-style-type: none"> • filter() • zip() • map() • dictionary's .keys(), .values() and .items() method 	Returns list instead of iterators

Q 5: Explain different implementations of Python?

A: Following are different implementations:

- **CPython**: Reference implementation written in C and native Python.
- **Stackless Python**: is an addition or improvement over CPython to support micro threads.
- **PyPy**: is just in time compiler where program is executed on fly rather than compile time.
- MicroPython and CircuitPython supports Microcontrollers.
- Jython compiles into Java byte code, which can then be executed by every Java virtual machine implementation.
- IronPython allows running Python programs on the .NET Common Language Runtime.
- The RPython language can be compiled to C, Java bytecode, or Common Intermediate Language, and is used to build the PyPy interpreter of Python.
- Pyjs compiles Python to JavaScript.
- Cython compiles Python to C and C++.
- Pythran compiles Python to C++.

Q 6: Is Python loosely typed?

A: Python is loosely typed programming language since the same variable can hold different data at different points of time.

For example:

```
a = 20 (holds int)
a="rahul" (holds string now)
```

Same variable can be used to store different value or data belonging to different data types.

Q 7: How to start a new block in python?

A: A new block begins when the line is intended by 4 spaces.

Q 8: How to get data type of a particular variable?

A: Type function is used in the data type of enclosed variable.

For example:

```
>>> a =10  
>>> type(a)  
<class 'int'>
```

Q 9: How many ways can python program be run?

A: Python programs can be run using IDLE which is the integrated IDE or it can also be run from outside using **python.exe** in windows or **python.sh** in UNIX/LINUX based environment.

Q 10: Explain the importance of Pylint and Pychecker.

A: Pylint is used to check coding standards w.r.t. modules.

Pychecker is a static analysis tool that helps find out bugs in the source code.

Q 11: Explain Zen of Python.

Long time Pythoner Tim Peters succinctly channels the BDFL's guiding principles for Python's design into 20 aphorisms, only 19 of which have been written down.

The Zen of Python

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

Q 12: How to print Zen in python?

A: We can print Zen of python by using:

```
import this  
print(this)
```

Q 13: Explain Python data types.

A:

- **Integers:** represent numbers without decimals
- **Floating point:** represent number with decimals
- Complex numbers represents numbers having real and imaginary part
- String collection of characters
- Boolean represents the value as True or False

For example:

```
a = 5  
print(a, "belongs to this type", type(a))
```

```
a = 2.0  
print(a, "belongs to this type", type(a))
```

```
a = 1+2j  
print(a, " has type", type(a))
```

```
a = 'Hello world'  
print(a, " has type", type(a))  
a = True  
print(a, " has type", type(a))
```

```
a = 1+2j
print("Is complex number?",
      isinstance(a,complex))
```

Output:

```
5 belongs to this type <class 'int'>
2.0 belongs to this type <class 'float'>
(1+2j) has type <class 'complex'>
Hello world has type <class 'str'>
True has type <class 'bool'>
Is complex number? True
```

Q 14: How can we switch variables in Python?

A:

```
a = 'Hello'
b = 'World'
print(a, b)
a, b = b, a
print(a, b)
```



Q 15: What is the use of pass statement in python?

A: Pass statement indicates no operation statement. It is generally used to complete the body of classes or functions for alter use.

Q 16: Is Python pass by value or pass by reference?

A: Neither of them is used.

Python passes arguments by value where all values are references to actual objects.

Q 17: Does python supports chained operations?

A: Yes, chained operations are supported by Python.

```
no = 8
if 16 > no > 2:
    print("Chained comparison ")
```

In the above example, 16 is compared with 8 which is further compared with 2.

Q 18: Explain ALL and ANY.

A: ALL returns TRUE, only when all the conditions are TRUE.

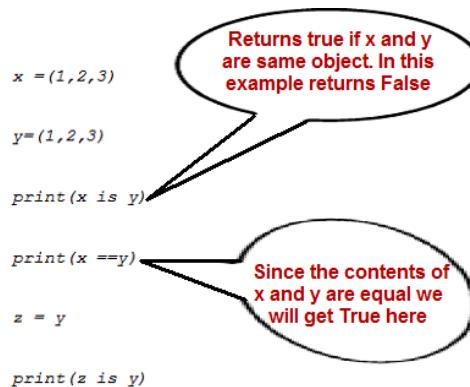
Any return TRUE even when one condition is TRUE.

```
lst= range(20)
print(all(value > 18 for value in lst)) # 
returns false since range starts with 0
lst = range(21,100)
print(all(value > 18 for value in lst)) # 
returns true because range starts with 21
print(any(value > 18 for value in lst)) # 
returns true since 22 is greater than 18 and
is true
print(any(value < 18 for value in lst)) # return
false since not a single value exists which
satisfies given
#condition
```

Q 19: Explain the difference between IS and ==.

A: **IS** : True only if 2 variables point to same object.

==: True if objects which are referred by objects are equal.



Q 20: Explain supported collection of data type w.r.t. Python?

A: Python supports four collections of data types:

List:

- Stores data in a fashion similar to array in other programming language.
- It is mutable and item can be added subtracted or deleted.
- Index starts with zero.
- [] is used to indicate that collection is list.
- It can be homogenous and heterogeneous i.e. store data of same or different data type.
- It can contain duplicates.

Tuples:

- Immutable collection i.e. changes to elements are not possible.
- Indicated by () .
- Again can be homogenous and heterogeneous i.e. store data of same or different data type.
- Duplicates can be present.

Set:

- Duplicates are absent in set.
- It makes use of {} symbol.

Dictionary:

- Stores data in form of key value pair.
- Indicated by {} with each item having : between
- Key has to be immutable object.

Q 21: Create a simple number list?

A: `a =[1,22,33]`

Above creates a simple list having three values.

Index	Value
0	1
1	22
2	33

Q 22: Can you create nested list?

A: Following is a nested list of three elements having list again at index 1 and nested list at location 2:

- `nestedLst = ["hello", [100,101], ['aAA',['pAA','rAA']]]`

- `print('Nested list',nestedLst)`
- `print(nestedLst[0])#prints hello`
- `print(nestedLst[2][1][1])) #prints rAA`

Q 23: Explain CRUD (Create, Update, and Delete) operations from list.

A: **Creation:**

```
lst = [11, 22, 33, 44]
```

Above statement creates a list with the name **lst**.

Empty list creation:

```
lst = []
```

The above statement creates an empty list.

Adding single element in list:

```
lst.append(55)
```

Adding elements of another list:

```
lst.extend([44, 44, 33])
```

Concatenate list:

Concatenate is to join two list.

Following code adds two list and creates a new list c containing elements from both list.

```
a = [1, 2, 3, 4]
b= [5, 65]
c = a + b
print(c) #Outputs 1,2,3,4,5,65
```

Remove element using value:

```
a = [1, 2, 3, 4]
a.remove(3)
print(a) #[1, 2, 4]
```

In above code, element with value 3 is removed.

Delete elements by using index:

```
a = [1, 2, 3, 4]
del a[2] #removes element at index 2
print(a) #[1, 2, 4]
```

Clear or Empty a list:

```
a=[1,2,3,4]
a.clear()
print(a) # prints []
```

Length of list:

```
To print size of list len, function is used.
a=[1,2,3,4]
print( len(a)) #prints 4
```

Q 24: Explain operations in dictionary.

A: Creation:

```
sdict = {1:'one',2:'two'}
```

The above statement creates a dictionary of two items having keys 1 and 2 and corresponding values as one and two respectively.

Creation of empty

```
sdict={}
Add key value pair
sdict={}
sdict[3] ='three'
```

The above statement adds an item in the dictionary with the key as 3 and value as three.

Add or Update elements from another dictionary

Addition of new elements from another dictionary.

```
dict1 ={1:'one',2:'two'}
dict2 ={5:'five'}
dict1.update(dict2)
print(dict1) //prints 1:one,2:two,5:five
```

Updating of existing element from another dictionary.

```
dict1 ={1:'one',2:'two',5:'FIVE'}
dict2 ={5:'five'}
dict1.update(dict2)
print(dict1) # 1:one 2: two 5: five
```

In the above example the value of key 5 is replaced from FIVE to five.

Thus, update is taken from dict2.

Clear or Empty dictionary

```
dict1 = {1:'one', 2:'two', 5:'FIVE'}  
dict1.clear()  
print(dict1) #prints {}  
Length of dictionary  
len(dict) get length of dictionary
```

Q 25: Explain operation with tuples.

A: **Creation**

```
tup = (3, 4)
```

Creation of empty tuple:

```
tup = ()
```

Creation of tuple with single element

This is a bit of challenge because since (1) is taken as number as compared to tuple.

So, to create tuple with one element we make use of the following:

```
tup = (1, )
```

Adding new elements or removing elements from tuples are not possible since tuples are immutable.

Length of tuples:

```
len(a)
```

Q 26: Explain del?

A: The **del** is used to remove variables from memory.

```
a= (1,2,3)  
print(a) #prints 1 2 3  
del a  
print(a) # gives a not defined error
```

Q 27: If del can remove variable can it remove tuple variable?

A: Yes, **del** can remove tuple variable. A tuple has to be immutable till it lives in memory. While **del** removes the object or variable from memory itself hence **del** can remove tuple.

```
a = (1,2)  
del a
```

But if we try to remove an element from tuple using **del**, it won't work.

```
a = (1, 2)
del a[2] #TypeError: 'tuple' object doesn't
support item deletion
```

Q 28: Delete last element in a list.

A speech bubble containing the text: "Deletes the last element. Similarly del lst[-2] would delete second last one and so on". An arrow points from the word "lst" in the code below to this text.

```
lst = [1, 2, 3, 4]
del lst[-1]
print(lst) # [1, 2, 3]
```

Q 29: Predict the output of following code.

```
a = (1, 2, 3, 4)
b= (5, 65)
c = a + b
print(c)
```

A: This question is tricky, mainly because it involves tuples which are immutable. So, the general answer given by many is, it would raise an error which is incorrect. The above code adds two tuples to generate new tuple. Existing tuples are untouched which satisfies immutability property. Hence, output is

(1, 2, 3, 4, 5, 65). So, while reading such questions see if the immutability is affected or not. If not then operation would be successful.

Q 30: What do you mean by list comprehension?

The process of creating a list while performing some operation on the data so that it can be accessed using an iterator is referred to as list comprehension.

Q 31: Explain the preferred way for looping through list?

A: The preferred way of looping through list is with the help of enumerate function.

Enumerate function returns index and corresponding value:



```
mylist = ["a", "b ", "c",  
          for index, item in enumerate(mylist):  
              print(index, item)
```

Output

0 a

1 b

2 c

Q 32: Find the reverse of the dictionary?

A: Reverse of dictionary means key becomes value and value becomes the key.

It is easily achievable in python.



```
countryDict={1:'UK',2:'USA'}  
reversed_dict = {value: key for key, value in countryDict.items()}  
  
print(reversed_dict)
```

Q 33: How to sort dictionary by value?

A:

```
import operator as o  
countryDict = {'CN': 4, 'FR': 5, 'UK': 1, 'USA':  
2, 'IN': 3, 'GM': 6}  
print(sorted(countryDict.items(), key=o.  
itemgetter(1)))
```

Q 34: What is the use of shuffle function?

A: It is used to arrange the values within the list randomly.

Q 35: What is the preferred way to get a value based on key in Python?

A: The preferred way to get a value based on key is to use get method with default value.

The default value will be used whenever key is absent in the dictionary.

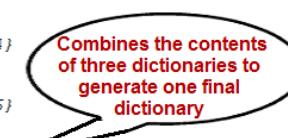
```
countryDict={'UK':1,'USA':2}  
print(countryDict.get('NZ','Country not found'))  
  
print(countryDict.get('UK','Country not found'))
```



Q 36: Explain alternate way of merging 2 or more dictionaries without using update method?

A:

```
countryDict1={'UK':1,'USA':2}  
  
countryDict2={'IN':3,'CN':4}  
  
countryDict3={'FR':5,'GM':6}  
  
countryDict = {**countryDict1, **countryDict2, **countryDict3}  
  
print(countryDict)
```



Q 37: What is the preferred way of fetching last element/second last and so on from a list?

A: Python supports negative indexes on list.

- -1 refers to last element
- -2 refers to second last element

```
l = [1, 2, 3, 4, 5]  
print(l[-1]) #prints 5  
print(l[-2]) #prints 4
```

Q 38: What is the preferred way for reversing a list?

A: The preferred way for returning a list is with help of reversed function.

```
l = [1, 2, 3, 4, 5]
for i in reversed(l):
    print (i)
```



Reversed returns
reverse of given list

The above code will print the following:

5, 4, 3, 2, 1

Q 39: Explain various string utility functions in Python.

A: **capitalize()**: Capitalizes the first letter of string.

isalnum(): Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.

isdigit(): Returns true if string contains only digits and false otherwise.

islower(): Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.

isnumeric(): Returns true if a unicode string contains only numeric characters and false otherwise.

isspace(): Returns true if string contains only whitespace characters and false otherwise.

isupper(): Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.

lower(): Converts all uppercase letters in string to lowercase.

lstrip(): Removes all leading whitespace in string.

max(str): Returns the max alphabetical character from the string str.

replace(old, new [, max]): Replaces all occurrences of old in string with new or at most max occurrences if max given.

strip([chars]): Performs both **lstrip()** and **rstrip()** on string.

swapcase(): Inverts case for all letters in string.

Q 40: How to check whether two strings are equal.

A: `==` operator can be used to check whether strings are equal or not.

Q 41: Can string use single quote or double quote?

A: Both single quote or double quote is allowed but combination of both is not.

Q 42: Explain type conversions on collection types.

A:

Convert from	Convert to	Syntax
list	Set	B =set([1,2,3])
set	Tuple	B= tuple({5,6,7})
nested list	Dictionary	dict([[1,2],[3,4]])
set	List	list({1,2})

Q 43: Explain set theory operations supported by set data type.

A:

Operation	Details
isdisjoint	Returns true if both the sets do not have anything in common
A.issubset(B)	Returns true if A is subset of B
B. issuperset(A)	Returns true if B is superset of A
A & B or A.intersection(B)	Returns common elements between A and B
A – B Or A.difference(B)	Returns the difference between A and B

Sample code for above operations:

```

setA = {11,22,33,44,5}
setB = {44,22,1}
setC = {55}
setD = {44,22,1,447}
print('isdisjoint',setA.isdisjoint(setC))
print('Is setB subset of setD ',setB.issubset(setD))
print('Is setD superset of setD ',setD.issuperset(setB))
print('common elements are ',setA & setB)
print('common elements are ',setA.

```

```
intersection(setB))
print('Elements in A but not in B ', setA-setB)
print('Elements in B but not in A ', setB-setA)
print('Elements in A but not in B ', setA.
difference(setB))
print('Elements in B but not in A ', setB.
difference(setA))
```

Output:

```
isdisjoint True
Is setB subset of setD  True
Is setD superset of setD  True
common elements are  {44, 22}
common elements are  {44, 22}
Elements in A but not in B  {33, 11, 5}
Elements in B but not in A  {1}
Elements in A but not in B  {33, 11, 5}
Elements in B but not in A  {1}
```

Q 44: Explain frozenset?

A: frozenset gives immutability to set i.e. nothing in it can be modified.

Q 45: Explain functions in Python?

A: A function is a group of statements intended to do particular task.

In Python we can define functions with the help of **def**.

```
def fun():
    print('hello')
```

Above creates a function fun which prints **hello** message.

```
def fun1(ip):
    print('hello', ip)
```

Above creates a function with the name fun1 which takes parameter ip and prints corresponding hello message with parameters.

```
def area(r):
    return 3.14* r * r
```

Above creates a function which returns result with the help of return type.

Q 46: What is a Boolean function?

A: Boolean function is one which either returns True or False.

```
def isdivisible(num, den):
```

```
if num%den==0:  
    return True  
else:  
    return False  
print(isdivisible(10,2))
```

The above function returns true or false based upon whether the numerator is divisible by denominator.

Q 47: Can we specify data type for arguments as well as return types in python?

A: Recent version of Python has support for type annotations which allows doing the same.

Data type of parameters
are passed as well as
return type mentioned
with -> symbol

```
def mul(x: float, y: int) -> float:  
  
    return x * y  
  
print(mul(9,9.9)) # print 89.100000
```

Q 48: Explain variable arguments?

A: Variable arguments in python can be achieved with the help of *args.

```
def fun(*args):  
    print(args)
```

Function defined
with variable
arguments

```
fun()
```

Function call with
no. 1,2 and variable
arguments

```
fun(1)
```

```
fun(1,2)
```

Q 49: Write a program to find occurrences or count of characters in given word.

A:

```
import collections as c
c
counters = c.Counter('Vishwanathan')

print(counters)#Counter({'a': 3, 'h': 2, 'n': 2, 'v': 1, 'i': 1, 's': 1,
's': 1, 'w': 1, 't': 1})
```

Maintains character involved in given string and corresponding count

Q 50: What is **kwargs?

A: **kwargs is also variable arguments but in the form of dictionary input also known as keyworded, variable-length argument.

```
def fun(**kwargs):
    print(kwargs)
```

keyworded,
variable-length
argument used

fun()

fun(a=1)

fun(a=1,b=2)

Value passed to
kwargs using
a=1

Q 51: Write a simple Lambda expression?

A:

```
sumlambda = lambda x, y : x + y
print(sumlambda(2, 3))
```

In the above example, we have defined lambda expression which takes two parameters and performs the addition. The return result is x+ y but no need to return explicitly since it contains single statement only.

Q 52: Lambda forms in Python contain statements? True or False?

A: Statement is true as lambdas in Python have to be single line expressions.

Q 53: Explain filter function?

A: The filter function is a utility function which can skip values based on some condition.

```
a = [1, 2, 3, 4, 5, 6]
for i in filter(lambda x : x % 2 == 0, a):
    print(i)
```

Checks whether the number in list is even in which case output is allowed and vice versa

Q 54: Explain steps involved in reading and writing a file?

A: In both the case, three steps are involved.

1. Opening the file.
2. Reading or writing the file using corresponding functions (read or write).
3. Closing the file.

Write example:

```
a = open('hello.txt', 'a') # opens a file in write mode
a.write('welcome to files with python') # writes this line in file
a.close() # closes the file
```

The above code writes the welcome message to **hello.txt** file.

Read example:

```
a = open('hello.txt', 'r') # opens a file in read mode
text = a.read() # reads the entire contents of file in memory
print(text) #prints the read text
a.close() # closes the file
```

Q 55: Explain the term “withstatement”?

A: The **with** statement would be useful whenever two related operations block on each other.

```
with expression [as variable]:
```

with-block

In the above statement, we see that **with** block would be used within the context of variable opened with the help of **with** statement.

```
with open('output.txt', 'w') as f:  
    f.write(' | Python! ')
```

In the above example, the file would be closed as soon as the scope of **with** block gets over the advantage of being not to close the file explicitly using close block.

Q 56: Explain the preferred way of reading a big file?

A:

```
with open("sampleread.txt", "r+") as ip:  
    for singleLine in ip:  
        print(singleLine)
```

The above code opens the file in read mode with the help of with statement. So, as soon as file operations get over, the file would be automatically closed.

Further the above code loads only a single line in memory at a time thus saving memory or making it possible to run this program even with computer having lower memory footprint.

Q 57: Explain modules in python.

A: Modules both inbuilt and user defined are ways of writing reusable set of code or reusing common set of code.

Python provides set of many inbuilt modules like mathos, sys etc.

To use a module, we will have to import the same with the help of import statement or “from ..import” statement.

Q 58: Explain different ways of importing modules.

Modules can be imported using import statement or *from ..import* statement.

The difference lies in fact whether we want the entire module or particular part of it.

The *from..import* statement is preferred when we want to import particular module.

```
from math import pi  
print(pi)
```

The **import** statement is preferred when the entire module is to be imported.

```
import math  
print(math.pi)
```

As you can see in import statement we have to use the module name wherever we want to get variables or methods from modules.

Q 59: Can we create our own module?

A: Yes, we can create our own module. Any python file itself is a module.

Consider for example following **mathop.py** file containing following code:

```
def sum(a,b) :  
    return a+b  
def sub(a,b) :  
    return (a-b)
```

In another file named **usage.py** we can write the following code:

```
import mathop  
print(mathop.sum(99,100))  
print(mathop.sub(99,100))  
print(dir(mathop))
```

The above code reads from the modules and performs the required operations.

If the files were in different directory, for example **com** folder then we could have imported by giving:

```
com.modulename
```

Q 60: Explain in brief about os module and its corresponding functions.

A: os module provides various functions to perform operating system related functions especially w.r.t. file operations.

Following are some of the important functions:

os.chdir("D:/pythoneg/intro/pyFolder") : It is used to change the current working directory.

os.mkdir('pyFolder') : It is used to create new directory.

os.remove(): It removes a particular file.

os.getcwd(): It gets the current working directory.

os.environ: It prints environment related details.

os.system(' dir>> 1.txt'): It execute a command line related command for example, executes dir command which is windows command prompt command, to list directories.

os.walk(dir): It lists down sub directory and files within a particular directory.

Q 61: Using os module print the directory structure.

A:

```
import os
rootDir = 'D:/nodejstraining/NodeJSExample/
NodeJSExample'
for dirName, subdirList, fileList in
os.walk(rootDir):
    print('Found directory: %s' % dirName)
    for fname in fileList:
        print('\t%s' % fname)
```

In the above example we print the directory structure with the help of **os.walk** function which can list down directory name, sub directory list and file list within a particular directory.

Q 62: Explain dir function.

A: The dir function is used to print various functions which are exposed by a module.

For e.g **dir(math)** prints all the functions within **math** module.

```
import math
print(dir(math))
```

Output:

```
['__doc__', '__loader__', '__name__', '__
package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh',
'ceil', 'copysign', 'cos', 'cosh', 'degrees',
'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose',
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma',
'log', 'log10', 'log1p', 'log2', 'modf', 'nan',
'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'tau', 'trunc']
```

Q 63: Explain exception handling in python.

A: Exception handling is supported with the help of **try..except** block.

Any statements which can logically cause problems would be placed in **try** block.

In case of exception occurring, corresponding handling is done in **except** block.

A single try can contain multiple except block.

```
try:

    ip = input('Enter no')

    no=int(ip) Possible exception point

    print('You entered',no)

except ValueError as e2: Specific exception handling
    print('value error',e2) Generic exception
except Exception as e : Alias print complete exception details
    print('some error',e)
else: Alias print complete exception details
    print('valid')
```

In the above example, we place the code of accepting input from user and type casting the same to **int** within the **try** block. We have two **except** block one catching **Exception** block and another generic **Exception**. Based upon the exception raised, the corresponding **except** block would be called.

To print the complete details of Exception we make use of alias i.e. **except Exception as e :** makes alias of the exception.

In case of no exception, the **else** block would be executed.

Q 64: How to create user defined exception?

A: User defined exceptions are supported in Python.

To create our own exception, we will have to create class with the name of Exception we want, which extends from Exception and correspondingly use it with the help of **raise** statement.

```
class NotEqualError(Exception):
    pass
    n1 = 10
    n2 = 10

    try:
        if n1 != n2:
            raise NotEqualError

    else:
        print('Equal')

    except NotEqualError as e:
        print('NotEqualError', e)
```

Used to complete class body

Extends exception class

Causing NotEqualError to occur with the help of raise.
Raise is equivalent to throw in other programming

Q 65: What is the use of raise statement?

A: The **raise** statement is used to generate exception at run time, based upon some condition or constraint imposed.

Q 66: How to create own class in Python? Explain constructors.

A: We can create our own class in Python. The constructor of the class is specified with the help of **__init__ method**.

Any function created as a part of class which is a non-static and non-class method, should have **self** as the first argument which represents the object.

Following is the code to create a class with the name Complex which consists of two members **real** and **img** and one display function.

```
class Complex:
    Class definition starts

    def __init__(self, real, img):
        self.real = real
        self.img = img
        Constructor with self as the first argument and real and img are other parameters

    def display(self):
        Display function containing self as first argument
        print(self.real, self.img)

Object of Complex class created 3 represents real part and 4 represents imaginary
c = Complex(3, 4) ← Object reference
c.display()
```

Q 67: Is it necessary to have the first argument of class function as **self**? Can't we rename it to any other variable?

A: The name of the variable can be anything but it always represents the object. The **self** is the preferred and a general practice naming convention.

Q 68: Explain inheritance in Python.

A: Python supports inheritance with classes. A class extending another class needs to use **()**.

Also, super method needs to be used to call base class constructor and methods.

```

class A: ← Base class with
          constructor and display
          method.
          Constructor takes self
          and name as parameters
          self.name=name

def display(self):
    print(self.name)

class B(A): ← B is a child of A
            ← Base class
            constructor
            invoked
            ← Child property
            initialized
            super().__init__(name)
            self.age=age

def display(self): ← Display function
                  ← overridden to print
                  ← age and name
                  print(self.name, self.age)

bb = B('raj',33)

bb.display()

```

Q 69: How to determine whether a particular class is sub class?

A: **issubclass(ChildClass,ParentClass)**

The above function would return true only if the **ChildClass** has inherited from **ParentClass**.

Q 70: Does python support multiple inheritance?

A: Yes, Python supports multiple inheritance.

A particular class can extend from any number of super classes.

class Child(Parent1,Parent2,.....,ParentN):

Q 71: How is diamond problem resolved in case of Python?

A: Diamond problem occurs whenever Child class inherits same method with different body from two different parents.

In the case of Python this is resolved by order in which parent is inherited. Parent which is inherited first is given more importance. In the following example we have two parents both having hello method.

```

class Parent1:

    def hello(self):
        print("Hello from Parent1")

class Parent2:

    def hello(self):
        print("Hello from Parent2")

class Child(Parent1,Parent2):
    pass

c= Child()
c.hello()

```

The diagram illustrates the method resolution order in Python. It shows a class hierarchy with Parent1 at the top, followed by Parent2, and then Child which inherits from both. A call to c.hello() is shown at the bottom. Two callouts explain the behavior: one points to Parent1 with the text "Parent1 is inherited first", and another points to the call to c.hello() with the text "This prints Hello from Parent1 indicating importance given to Parent1".

Q 72: Does python support private method and variables?

A: Private method and variables are supported in Python. Any variable or method having __ becomes private in nature and cannot be accessed directly by using object.

To get their value we will have to use class name reference from the object.

The main aim of these variables and functions is to ensure that the sub classes do not accidentally override them.

Q 73: Can __ be used for other purpose than creating private variables or functions?

A: __ has many use cases.

Following are some of them:

1. Ignoring while assigning values from tuples or list.

a, __, b = (11, 220, 54)

In the above example, a gets value 11 and b gets 54. The __ is used for ignoring the central value 220.

2. Used to separate numbers no = 3_200. This gives better understanding and readability. The number represented here is 3200.

Q 74: Does python support abstract classes?

A: Python supports abstract classes with the help of module named **abc**.

An abstract class contains abstract functions and cannot be instantiated.

The abc module contains ABC which stands for Abstract base class and abstract method to do necessary things.

```
import abc

class SampleAbstract(abc.ABC):
    Extends ABC

    def __init__(self, someValue):
        self.value = someValue

    super().__init__()

    @abc.abstractmethod
    def abstractM1(self):
        Indicates
        abstractM1 is an
        abstract method

    pass
```

The classes which extend the above class have to give body to abstract method else even they cannot be instantiated.

Q 75: Differentiate between static methods and class methods in Python.

A:

Static method	Class method
Needs no specific parameters though static function can be parameterized.	Takes cls as first parameter. cls represents the class on which the method is created.
Can't modify class state.	Can modify and access class state.
Created by using @staticmethod annotation.	Created by using @classmethod annotation.
Used to create utilities.	Used to create factories.
No such feature.	Is a descriptor, wrapping a function, and can call the resulting object on a class.
Self-contained code.	Code access till Class level.

Q 76: What are named tuple?

A: Named tuple allows us to create classes at run time with attributes.

```
from collections import namedtuple

Point = namedtuple('Point', ['x', 'y'])
p1=Point(1, 2)
p2=Point(1,72)
Creates a dynamic class with
name Point having attributes
x and y. First parameter is the
name of class and second
one is the list of attributes
print(p1)

print(p2)
```

Q 77: How to sort using lamdas?

A:

```
from collections import namedtuple

Emp = namedtuple('Emp', ['name', 'age'])
List of Emp
objects created
employees = [Emp('Vishva', 30), Emp('Raj', 28), Emp('Joe', 42)]
Sorting using lambda
expression on age
attribute of Emp. Default
sorting is ascending
print(sorted(employees, key=lambda p: p.age)) ←
print(sorted(employees, key=lambda p: p.age,reverse=True))
Sorting in
descending order
using reverse
```

Q 78: Explain Generators?

A: Generators are used to save memory as they store only some values in memory at a time rather than storing entire dataset in memory. Generators make use of yield function which stores the previous state to remember.

Generators are invoked using next function.

For example, if we want to process the entire file having million records, generator function would bring only one record at a time in the memory.

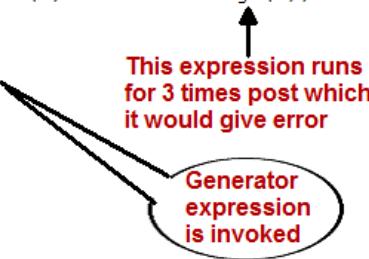
```
def fixed_generator(value, max_repeats):  
  
    for i in range(max_repeats):  
  
        yield value ← Generator function  
                           makes use of yield  
  
iterator = fixed_generator('Vishwa', 3)  
  
print(next(iterator))  
print(next(iterator))  
print(next(iterator))  
print(next(iterator))
```



Q 79: What is generator expression?

A: The above generators concept can also be applied to expressions.

```
genexpr = ('Hello'+str(i) for i in range(3))  
  
print(next(genexpr))  
print(next(genexpr))  
print(next(genexpr))  
print(next(genexpr))  
print(next(genexpr))
```



Q 80: When python program exits, all the memory is released? Say true or false?

A) The answer is false because objects that are referenced from global namespaces of Python modules are not always de-allocated when Python exits.

Q 81: Can a function be passed as parameter to another function?

A: Yes, it is possible to do the same in Python.

```
def say_hello(a): ← The parameter a can  
    print("Hello from say_hello")  
  
    print("Start Calling function passed")  
  
    a() ← Parameter passed is  
          called as function  
  
    print("Finished Calling function passed")  
  
def say_hi ():  
    print("Hi from say_hi")  
  
say_hello(say_hi) ← say_hi is passed  
                   as argument to  
                   say_hello
```

Q 82: Can a function be returned as result from another function?

A: Yes, the result of a function can be another function.

```
def say_hello():  
    print("Hello")  
  
    def say_hi ():  
        print("Hi")  
  
        return say_hi  
  
    a = say_hello() ← say_hi is the  
                      inner function  
                      within say_hello  
  
    a () ← The say_hi is  
           returned as result in  
           say_hello function  
  
    print("ddddd") ← The say_hi is  
                      returned as result in  
                      say_hello function  
  
    a () ← say_hi is  
           called
```

Q 83: Explain decorator function.

A: Decorator function allows performing wrapper around a function. It acts as an interceptor which allows performing pre-processing and post-processing.

```

Decorator function  

takes another  

function as argument  

def decorator_func(some_func):  

    def wrapper_func(): ←  

        Inner wrapper function to perform  

        pre-processing and post  

        processing before or after calling  

        function passed as argument  

        print("Wrapper function started")  

        some_func()  

        print("Wrapper function ended")  

        return wrapper_func ←  

Wrapper function is  

returned as result  

Decorator function  

call which will wrap  

say_hello function  

def say_hello():  

    print ("Hello")  

a = decorator_func(say_hello)  

a()  

Same as earlier one  

def wrapper_func(hello_var, world_var):  

    return a(hello_var, world_var)  

say_hello being annotated with  

above created decorator. A  

call to say_hello will cause  

decorator to be called  

@decorator_func ←  

def say_hello(hello_var, world_var):  

    print(hello_var + " " + world_var)  

    say_hello("Welcome ", "here")

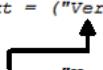
```

Q 84: How can we represent big text in Python?

A: Representation of big text is quite simple w.r.t. Python, unlike other programming language.

```
longText = ("Very very very very long text "
           "Very very very very long text "
           "Very very very very long text ")
print(longText)
```

**Represented
by string
within ()**



Q 85: What is PEP 8?

A: PEP 8 is a coding convention, a set of recommendation, about how to make your Python code more readable. It can be said as globally accepted standards for python programming.

Q 86: What is anaconda?

A: Anaconda is easy to use distribution which comes bundled up with lot of modules used in data science world. It provides a packaged environment to get started easily. It also manages dependencies.

Q 87: How to install external modules?

A: External modules can be easily installed with the help of **pip** tool which comes bundled with Python.

The command to execute is **python -m pip install module name**.

For e.g. `python -m pip install numpy`

In anaconda environment we can install with **conda** command.

`conda install --name environment_name module`

Q 88: What is Jupyter notebook?

A: Jupyter Notebook is a web application that allows you to create and share documents that contain:

- Python code
- visualizations
- explanatory text of code

It can be installed as an external module with the help of **pip**.

`python -m pip install jupyter`

Once installed, it can be started by executing command **jupyter notebook**.

The above command will start the web server with notebook opened on web browser.

Notebooks consist of a linear sequence of cells. There are four basic cell types:

- **Code cells:** Input and output of live code that is run in the kernel
- **Markdown cells:** Narrative text with embedded LaTeX equations
- **Heading cells:** 6 levels of hierarchical organization and formatting
- **Raw cells:** Unformatted text that is included

Q 89: What is pickling and unpickling?

Pickling is the process of converting from python format to intermediate string formats which can be then be stored on file.

Unpickling is the process of bringing or loading the stored python objects back into memory.

Q 90: Explain the importance of setup.py?

A: The **setup.py** allows setting up dependencies easily. While generating one project, it is quite natural to use various libraries. When the same code is shipped to another developer or deployment environment we can make use of **setup.py** which stores the dependencies on various modules which were used in project.

Q 91: Is it possible to make connections to database using Python?

A: Python supports connection to database with Db-API modules.

The mechanism to query remains same but difference lies in the classes to be imported to suit various databases.

To connect to database (in our case MySQL) we use the following code:

```
import mysql.connector

mydb = mysql.connector.connect(
    host="localhost",
    port=3306,
    user="root",
    passwd="root",
    database='pymysql'
)
```

To fetch data from database table:

```

mycursor = mydb.cursor()
Create a cursor to
execute query

sql = "SELECT * FROM vishwa ORDER BY name DESC"

mycursor.execute(sql)
SQL is executed
here

myresult = mycursor.fetchall()
Results of query is
fetched here and
stored in myresult
variable

for x in myresult:
Looping through
resultset to print
data row by row or
tuple by tuple

print(x)

```

Delete example:

```

mycursor = mydb.cursor()

sql = "DELETE FROM vishwa WHERE address = 'Chembur'

mycursor.execute(sql)
Commit to finally commit
mydb.commit() ← the transaction in database

```

Insert example:

Insert can be done using **execute** or **executemany**. Difference lies in fact that execute just takes one record while executemany takes many records.

```

sql = "INSERT INTO vishwa (name, address) VALUES (%s, %s)"

val = [
    ('Vishva', 'Chembur'),
    ('Shiv', 'Mulund')
]

mycursor.executemany(sql, val)
mydb.commit()

```

Execute many to
insert more than one
records in database

Update example:

```
sql = "UPDATE vishwa SET address = %s WHERE name = %s"

val = ("Mulund", "Vishwa")
mycursor.execute(sql, val)
```

Parameters passed
as positions the
values of which are
set in execute query

Q 92: Explain meta programming?

A: This is a programming technique in which program itself is treated as data. Such languages do have knowledge of itself, and can change or manipulate itself during run time.

Following are some of the examples of meta programming in Python:

1. Adding attribute and method to class.

```
class A:
    pass

A.f = 42
```

Adds field or attribute
with name f on class A.
Default value set is 42

```
x = A()

x.f=78 ← Default value changed
from 42 to 78 for object
print(x.f) # prints 78
```

Added a lambda
with name sayHi in
A dynamically

```
A.sayHi = lambda self: print("Hi from me!")

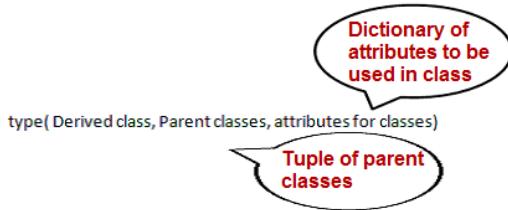
x.sayHi() # prints Hi from me
```

print(x1.f) # prints 42 the default value.

2. Creating inheritance chain using type.

The type method can not only be used to get the type of variable but can also be used for meta programming.

Details of overloaded type, is as follows:



For example:

```
B = type(B, (A,), dict(f=100))
```

The above statement creates **class B** which inherits from **Class A** and has field f with value **100**.

3. Creating dynamic constructor or linking constructor.

```

class A:
    pass
    def new(cls):
        x = object.__new__(cls)
        x.f = 100
        return x
    A.__new__ = new
    print(t.f)
  
```

Method takes class as argument and creates object using __new__. Also creates new field f. Created Object is returned back

Links the created new method to the __new__ function which is present in class A. When object of A is created this gets called

4. Complete meta class example:

```

class Meta(type):
    def __new__(cls, name, bases, dict):
        x = super().__new__(cls, name, bases, dict)
        x.f = 100
        return x

class A(metaclass=Meta):
    pass
    print(A.f) ## prints 100 set up in Meta class
  
```

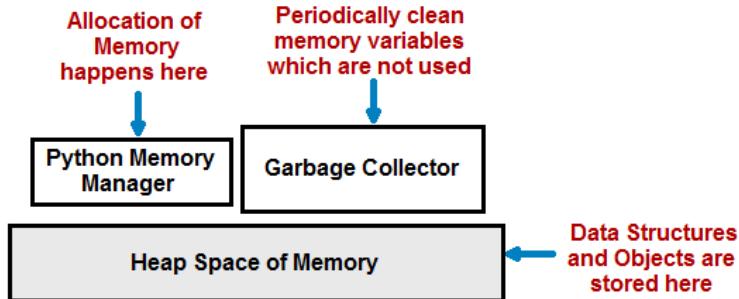
Indicates this class is child of type

New method created which follows the same signature as type method

A class is given metaclass attribute in which the above created type would be called. Meta class would be called

Q 93: Explain python memory model.

A:



Numpy Interview Questions

Q 1: What is numpy?

A: The **numpy** is a module which is responsible for effectively storing and processing the same at a faster rate as compared to normal array. The advantage of **numpy** is support of large number of in built mathematical operations as compared to other programming language. Also, the support to represent n dimensions is also possible with **numpy**.

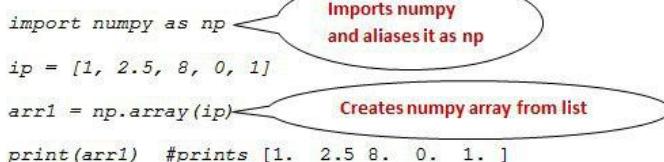
Q 2: How to install numpy?

A: Since **numpy** is an external module it can be installed using **pip**.
`python -m pip install numpy`

Q 3: How to create single dimension numpy array?

A:

```
import numpy as np
ip = [1, 2.5, 8, 0, 1]
arr1 = np.array(ip)
print(arr1) #prints [1. 2.5 8. 0. 1.]
```



Q 4: Explain different attributes provided by numpy?

A: **numpy** provides the following attributes:

ndim: As **numpy** provides **n** dimensions, we can get how many dimensions currently the array is having with **ndim**.

shape: Indicates number of rows and columns which again can be in different dimensions.

dtype: Indicates data type of elements stored in **numpy**.

```
import numpy as np  
  
ip = [[1, 2, 3, 4], [5, 6, 7, 8]]  
  
arr2 = np.array(ip)  
  
print(arr2)  
  
print('dimension',arr2.ndim)  
  
print('shape',arr2.shape)  
  
print('type',arr2.dtype)
```

Prints number of dimensions of given sample array

Shape indicates rows and columns and that too till n dimensions

Output:

```
[[1 2 3 4]  
 [5 6 7 8]]  
  
dimension 2  
  
shape (2, 4)  
  
type int32
```

Data types of elements stored in numpy array

Q 5: Explain some utility methods provided by numpy for creating different elements?

A: **np.zeros(length)** creates a numpy array at all zeroes occupying the length specified. Can also take rows and columns and create **n** dimensions filled with zeroes.

```
ip4 =np.zeros((2, 3))  
  
print(ip4)
```

Creates 2 rows and 3 columns filled with zeroes

Output:

```
[ [0. 0. 0.]  
 [0. 0. 0.]
```

np.ones: Can create either single dimension or **n** dimension, all filled with ones.

```
ip4 = np.ones((2, 3))
print(ip4)
```

Output:

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

np.eye: Creates a representation in which diagonal elements are zeroes.

```
ip8 = np.eye(3, 3)
print(ip8)
```

Output:

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

np.arange: Create a single or **n** dimension array in which numbers are populated starting from 0 to the number specified as parameter.

```
ip6=np.arange((5))
print(ip6)
```

Output:

```
[0 1 2 3 4]
```

Q 6: How can we change shape of an object?

A: Shape of the object can be changed with shape attribute or reshape method.

Shape attribute:

```
import numpy as np

a = np.array([[1,2,3],[4,5,6]])
print('shape',a.shape)           Will give
                                shape as (2, 3)

print('before',a)

a.shape = (3,2)                 Changing the shape
                                from 2,3 to 3,2

print('after change',a)
```

Output:

```
shape (2, 3)
```

Before,

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

After change,

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

reshape:

Allows the dimensions to be specified based on which, the shape can be changed.

```
a = np.array([[1,2,3], [4,5,6]])
b = a.reshape(3,2)
print('reshape', b)
```

Output:

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

Q 7: Which all data types are supported in Python?

A: int8, uint8 i1, u1 int16, uint16 i2, u2 int32, uint32 i4, u4 int64, uint64 : Signed and unsigned 8-bit (1 byte) 16-bit integer types or 32-bit or 64-bit integer types

float16 float32 float64, float128 Floating point with 16,32,64 or 128 bits

complex64, complex128, complex256 Complex numbers represented by two 32, 64, or 128

bool ?: Boolean type storing True and False values

object O : Python object type

string_S : Fixed-length string type (1 byte per character)

unicode_U : Fixed-length unicode type (platform specific number of bytes)

Q 8: Explain various simple mathematical operations which can be done on numpy?

A:

```

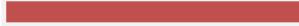
import numpy as np

arr1 = np.array([[1., 2., 3.], [4., 5., 6.]])

arr2= np.array([[2., 3., 4.], [4., 5., 6.]])  

print(arr1 + arr2)  

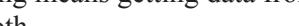
print(arr1 - arr2)  

print(arr1 * arr2)  

print(1 / arr1)  



```

Addition
Subtraction
Multiplication Division
Raise to supported

Q 9: Explain slicing operation in numpy?

A: Slicing means getting data from **numpy** array via row or columns or both.

General slicing on single dimension [start:end].

Start element is inclusive end element is exclusive.

```

import numpy as np

arr = np.arange(10)

print(arr)
  

print(arr[5:8])

```

Cuts from
index 5 to 8

Output:

[5 6 7]

For 2 dimensions and more than 2 dimensions, we can specify slicing by row and column indexes. In such case, following is the slicing.

[rowStart:rowEnd,columnStart:columnEnd]

- If **rowStart** is not specified, starts from zero
- If **rowEnd** is not specified, goes all the way till end

- If **columnStart** not specified, starts from zero
- If **columnEnd** not specified, goes till end

For example:

```
import numpy as np

x = np.array([[ 0,  1,  2], [ 3,  4,  5], [ 6,  7,  8], [ 9, 10, 11]])

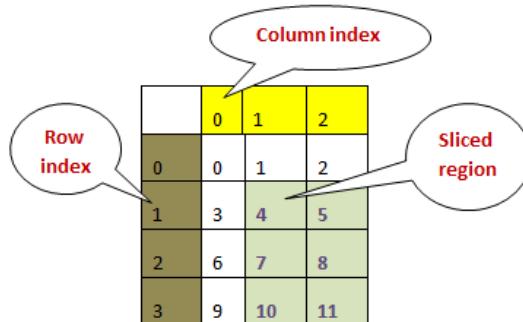
print(x)

z = x[1:4,1:3]
print(z)
```

**Slices row 1 to 4
Column 1 to 3**

2 dimension matrix represented by

0	1	2
3	4	5
6	7	8
9	10	11



Q 10: Explain Boolean indexing?

A: Boolean indexing return True or False based on some condition.
Consider the following numpy array with names.

```
names = np.array(['Raj', 'Rahul', 'Mahesh', 'Ravi', 'Rahul'])
```

```
print(names == 'Rahul')
```

**Return all the indexes as
True where name is Rahul**

Output:

```
[False  True False False  True]
```

Raj	Rahul	Mahesh	Ravi	Rahul
False	True	False	False	True

Boolean Indexing can also be masked that is, it can have more than one condition.

```
(names == 'Rahul') | (names == 'Raj')
```

The above code returns true for both Rahul and Raj.

Q 11: Perform matrix multiplication using numpy?

A: Matrix multiplication can be easily achieved with the help of dot function.

```
import numpy as np

arr = np.arange(15).reshape((3, 5))

print(arr)

print(arr.T)

print('mul', np.dot(arr, arr.T))
```

Q 12: Explain various functions available with numpy?

A: **numpy** supports the following function. The advantage of given functions are that they are applied on all the elements of **numpy** array.

abs, fabs: Returns the absolute value element-wise for .

Fabs can be used for non-complex-valued data with additional speed.

sqrt: Square root of each element is calculated.

square: Square of each element is calculated.

exp: Exponent ex of each element is calculated.

log: log10, log2, log1p Natural logarithm (base e), log base 10, log base 2, and log(1 + x), respectively.

sign: sign of each element: 1 (positive), 0 (zero), or -1 (negative) is calculated.

ceil: Determines the ceiling of each element,

floor: Determines the floor of each element,

rint: Round elements to the nearest integer, preserving the dtype.

modf: Return fractional and integral parts of given input

isnan:

Returns true if the value is NaN else false

isfinite, isninf: Return true indicating whether each element is finite or infinite, respectively.

Regular and hyperbolic trigonometric functions

cos, cosh, sin, sinh,tan, tanh ,arccos, arccosh, arcsin,arcsinh, arctan, arctanh

maximum, fmax Element-wise maximum. fmax ignores NaN

minimum, fmin Element-wise minimum. fmin ignores NaN

mod Element-wise modulus (remainder of division)

copysign Copy sign of values in second argument to values in first argument

Q 13: What is broadcast?

A: The ability of **numpy** to treat arrays with different dimensions or shape in a uniform way is known as broadcasting.

We can do broadcasting with the help of broadcast method.

```
import numpy as np
```

```
x = np.array([[1], [2], [3]])
```

```
y = np.array([4, 5, 6])
```

```
b = np.broadcast(y,x)
```

Will repeat each element
of y w.r.t x for example 4
will be repeated with all
elements of x and so on

```
for t in b:
```

```
    print(t)
```

Output:

```
(4, 1)
(5, 1)
(6, 1)
(4, 2)
(5, 2)
(6, 2)
(4, 3)
(5, 3)
(6, 3)
```

Q 14: Explain rules of broadcasting.

- Whenever two arrays have different dimensions the shape of the one with fewer dimensions is adjusted by padding with ones on its leading (left) side.
- If there is a difference in the shape of the two arrays, it does not match in any dimension, then dimension is stretched to match the other shape for that array with shape one.
- An error is raised in case the dimension and sizes disagree, and neither is equal to 1.

Q 15: Explain some statistical measures supported by numpy.

A:

amin: Find the minimum value row wise or column wise

mean: Finds the mean value of given dataset

std: Finds standard deviation of given dataset

var: Finds variance of given dataset

average: Finds weighted average in which average is calculated based upon frequency of occurrence of variables

percentile: Calculates various percentile intervals mostly around 25,50 and 75%

ptp: Calculates peak to peak values in a given dataset

median: Calculates median from given dataset

```
import numpy as np

a = np.array([[3,7,5],[8,4,3],[2,4,9]])

print('a')
print(a)                                1 indicates  
                                         calculates based  
                                         on row

print('amin 1')
print (np.amin(a,1))                     0 indicates  
                                         calculate based  
                                         on column

print('amin 0')

print (np.amin(a,0) )

print('ptp')

print (np.ptp(a) )

print('ptp axis 0')

print (np.ptp(a, axis = 0) )
print('percentile axis 1')
print (np.percentile(a,50, axis = 1) )
print('percentile axis 0')
print (np.percentile(a,50, axis = 0) )
print('median')

print (np.median(a, axis = 0) )
print('mean')
print (np.mean(a, axis = 0) )
a = np.array([1,2,3,4])
wts = np.array([4,3,2,1])
print('average')
print (np.average(a,weights = wts) )
print('std')
```

```
print (np.std([1,2,3,4]))
print('var')
print (np.var([1,2,3,4]))
```

Output:

```
a
[[3 7 5]
 [8 4 3]
 [2 4 9]]
amin 1
[3 3 2]
amin 0
[2 4 3]
ptp
7
ptp axis 0
[6 3 6]
percentile axis 1
[5. 4. 4.]
percentile axis 0
[3. 4. 5.]
median
[3. 4. 5.]
mean
[4.33333333 5.           5.66666667]
average
2.0
std
1.118033988749895
var
1.25
```

Q 16: Explain functions available in numpy.linalg.

A:

dot(a, b[, out]): Dot product of two arrays.

linalg.multi_dot(arrays) : Compute the dot product of two or more arrays in a single function call, while automatically selecting the fastest evaluation order.

vdot(a, b): Return the dot product of two vectors.

inner(a, b): Inner product of two arrays.

outer(a, b[, out]): Compute the outer product of two vectors.

matmul(a, b[, out]): Matrix product of two arrays.

tensordot(a, b[, axes]): Compute tensor dot product along specified axes for arrays $\geq 1\text{-D}$.

einsum(subscripts, *operands[, out, dtype, ...]):

Evaluates the Einstein summation convention on the operands.

linalg.matrix_power(M, n): Raise a square matrix to the (integer) power **n**.

kron(a, b): Kronecker product of two arrays.

linalg.cholesky(a): Cholesky decomposition.

linalg.qr(a[, mode]): Compute the qr factorization of a matrix.

linalg.svd(a[, full_matrices, compute_uv]): Singular Value Decomposition.

Matrix eigenvalues

linalg.eig(a): Compute the eigenvalues and right eigenvectors of a square array.

linalg.eigh(a[, UPLO]): Return the eigenvalues and eigenvectors of a Hermitian or symmetric matrix.

linalg.eigvals(a): Compute the eigenvalues of a general matrix.

linalg.eigvalsh(a[, UPLO]): Compute the eigenvalues of a Hermitian or real symmetric matrix.

linalg.norm(x[, ord, axis, keepdims]): Matrix or vector norm.

linalg.cond(x[, p]): Compute the condition number of a matrix.

linalg.det(a): Compute the determinant of an array.

linalg.matrix_rank(M[, tol]): Return matrix rank of array using SVD method.

linalg.slogdet(a): Compute the sign and (natural) logarithm of the determinant of an array.

trace(a[, offset, axis1, axis2, dtype, out]): Return the sum along diagonals of the array.

linalg.solve(a, b): Solve a linear matrix equation, or system of linear scalar equations.

linalg.tensorsolve(a, b[, axes]): Solve the tensor equation $a \times = b$ for x .

linalg.lstsq(a, b[, rcond]): Return the least-squares solution to a linear matrix equation.

linalg.inv(a): Compute the (multiplicative) inverse of a matrix.

linalg.pinv(a[, rcond]): Compute the (Moore-Penrose) pseudo-inverse of a matrix.

linalg.tensorinv(a[, ind]): Compute the ‘inverse’ of an N-dimensional array.

Q 17: How to save numpy data from memory to flat file?

A: numpy data can be stored in **npy** format files which are custom serialized or they can be stored in text file.

```
import numpy as np
a = np.array([1,2,3,4,5])
np.save('outfile',a)
b = np.load('outfile.npy')
print (b)

np.savetxt('out.txt',a)
b = np.loadtxt('out.txt')
print (b)
```

The diagram illustrates the following code snippets with callouts explaining their purpose:

- Top Snippet:**

```
import numpy as np
a = np.array([1,2,3,4,5])
np.save('outfile',a)
```

Saves numpy data in outfile.npy file
- Middle Snippet:**

```
b = np.load('outfile.npy')
```

Loads data from npy file into memory
- Bottom Snippet:**

```
np.savetxt('out.txt',a)
b = np.loadtxt('out.txt')
```

Saves data in text file format
- Final Line:**

```
print (b)
```

Loads data from text file format to numpy in memory

Q 18: What is the use of where and extract?

A: Both **where** and **extract** are used to match data based on the conditions.

Where sample:

```
import numpy as np

x = np.arange(9.).reshape(3, 3)

print(x)
y = np.where(x > 3)

print (x[y])
```

Finds out the index where value is greater than 3

Prints the value corresponding to condition

[4. 5. 6. 7. 8.]

Extract: Specifies data and corresponding condition.

```
x = np.arange(9.).reshape(3, 3)
```

```
condition = np.mod(x,2) == 0
print( condition )
print (np.extract(condition, x))
```

Applies above created condition on dataset

Q 19: What is the use of ndenumerate?

A: **ndenumerate** return the co-ordinates and corresponding values in the co-ordinates.

```
A = np.array([[11, 22], [33, 43]])
for index, x in np.ndenumerate(a):
    print(index, x)
```

Output:

```
(0, 0) 11
(0, 1) 22
(1, 0) 33
(1, 1) 43
```

Q 20: Explain how can we draw a histogram using numpy?

A: Consider the following code:

```
import numpy as np  
hist, be = np.histogram([1, 1, 2, 2, 2, 2, 3], bins = range(5))  
print(hist)  
print(be)
```

Generates histogram
based on the values and
ranges given by bins

The output given will be as follows:

Range	Bin	Count
0-1	0	1
1-2	1	2
2-3	2	4
3-4	3	1

The print(be) method in the above prints bin while the hist prints the corresponding count.

Pandas Interview Questions

Q 1: What is Pandas?

A: Pandas is a third party module which helps in easier representation of data in memory to perform analysis. Pandas help in faster representation and processing of data.

Q 2: How does Pandas represent data?

A: Pandas data representation is in similar line to an excel sheet which consists of row and columns.

Columns in Pandas are known as series.

Collection of series is called data frame.

Thus, data frame is the unit of representation of data which is collection of various series.

Shown is the representation of data frame:

Row Id (Index)	Series 1	Series 2
0		
1		
2		

Each column in data frame is also known as series

Row index is also known as index

Q 3: How to create Series?

A:

```
import pandas as pd  
  
data = pd.Series([0.25, 0.5, 0.75, 1.0])  
  
print(data)  
print(data.values)  
print(data.index)
```

Series is created using pd.Series

Prints the values stores in series

0.25, 0.5, 0.75, 1.0

Prints the index associated with each series. 0,1,2,3

Q 4: How to create Data frame?

A: Data frame can be created directly from dictionary or it can also be created by combining various series.

```
import pandas as pd  
  
country_population =  
{'India': 1100000,  
'China': 45679000,  
'USA': 3400000}  
  
population = pd.Series(country_population)  
country_land = {'India': '20000 hectares',  
'China': '40000 hectares',  
'USA': '30000 hectares'}  
area = pd.Series(country_land)  
  
df = pd.DataFrame({'Population': population, 'SpaceOccupied': area})  
print(df)
```

Creates a population series from country_population dictionary

Creates an area series from country_land

Data frame created with this two series

Output:

Population and SpaceOccupied are the series columns

	Population	SpaceOccupied
China	45679000	40000 hectares
India	1100000	20000 hectares
USA	3400000	30000 hectares

Keys from above two series forms row index

Q 5: How are missing values represented in data frame?

A: Consider the following example:

import pandas as pd
missing = pd.DataFrame({('a': 1, 'b': 2), ('b': 3, 'c': 4)})
print(missing)

Data frame generated by using 2 dictionaries with some non common keys resulting in NA

	a	b	c
0	1	2	NaN
1	NAN	3	4

Missing values are represented as NaN

Q 6: Explain the process of creating indexes w.r.t. pandas?

A: Indexes can be created using **pd.Index** function. Indexes support intersection and union.

indA = pd.Index([1, 3, 5, 7, 9])
indB = pd.Index([2, 3, 5, 7, 11])
print(indA & indB)
print(indA | indB)

& stands for intersection

| stands for OR

Q 7: Explain various attributes associated with series.

A:

- axes:** Stands for row.
- dtype:** The data type of the object is given by this attribute.
- empty:** Checks if series is empty.
- ndim:** The dimensions of data are given back.
- size:** Size or number of elements from data are given.
- values:** Gets the values in the form of ndarray.
- head():** The first n rows are returned.
- tail():** The last n rows are returned.

```
import pandas as pd
import numpy as np
s = pd.Series(np.random.randn(4))
print('series',s)
print('axes',s.axes)
print('Empty',s.empty)
print("Dimensions:")
print(s.ndim)
print ("Size of the object:")
print (s.size)
print ("Underlying data is:")
print (s.values)
print ("Top 2 rows:")
print (s.head(2))

print ("Last 2 rows:")
print (s.tail(2))
```

Output:

```
series 0    -0.200094
1      0.923716
2     -0.565540
3     -0.895206
dtype: float64
axes [RangeIndex(start=0, stop=4, step=1)]
```

```
Empty False
Dimensions:
1
Size of the object:
4
Underlying data is:
[-0.20009423 0.92371595 -0.56554023 -0.89520616]
Top 2 rows:
0    -0.200094
1     0.923716
dtype: float64
Last 2 rows:
2    -0.565540
3    -0.895206
dtype: float64
```

Q 8: Explain various statistical measures supported by pandas.

A: **axes**: Print rowindex as well as column index.

sum: Calculates the sum of all series.

mean: Calculates mean of all series.

median: Calculates the median of all series.

std: Calculates standard deviation.

count: Calculates sum of various series.

cumsum: Calculates cumulative sum.

import pandas as pd

import numpy as np

```
d = { 'Name':pd.Series(['Raja', 'Raju', 'Ricky',
'Shiv', 'Vishnu', 'Ramesh', 'Mahender',
'Lious', 'Salman', 'SRK', 'VV', 'MM']),
'Age':pd.Series([25, 26, 25, 23, 30, 29, 23,
34, 40, 30, 51, 46]),
'Rating':pd.Series ([4.23, 3.24, 3.98, 2.56,
3.20, 4.6, 3.8, 3.78, 2.98, 4.80, 4.10, 3.65])}
```

```

df = pd.DataFrame(d)

print(df)
print('axes')
print(df.axes)
print('sum')
print('sum',df.sum())
print('sum of axis 1 ',df.sum(1))

print('mean',df.mean())

print('std',df.std())

print('count',df.count())

print('median',df.median())

print('cumsum',df.cumsum())

```

Calculates sum series wise/columnwise

Calculates sum row wise

Output:

	Age	Name	Rating
0	25	Raja	4.23
1	26	Raju	3.24
2	25	Ricky	3.98
3	23	Shiv	2.56
4	30	Vishnu	3.20
5	29	Ramesh	4.60
6	23	Mahender	3.80
7	34	Lious	3.78
8	40	Salman	2.98
9	30	SRK	4.80
10	51	VV	4.10
11	46	MM	3.65

axes
[RangeIndex(start=0, stop=12, step=1),
Index(['Age', 'Name', 'Rating'], dtype='object')]
sum
sum Age 382

```
Name
RajaRajuRickyShivVishnuRameshMahenderLiousSalm...
    Rating           44.92
   dtype: object
sum of axis 1  0      29.23
 1      29.24
 2      28.98
 3      25.56
 4      33.20
 5      33.60
 6      26.80
 7      37.78
 8      42.98
 9      34.80
10      55.10
11      49.65
dtype: float64
mean Age       31.833333
Rating        3.743333
dtype: float64
std Age        9.232682
Rating        0.661628
dtype: float64
count Age       12
Name          12
Rating         12
dtype: int64
median Age     29.50
Rating        3.79
dtype: float64
cumsum      Age           Name  Rating
0  25          Raja  4.23
1  51          RajaRaju  7.47
2  76          RajaRajuRicky 11.45
3  99          RajaRajuRickyShiv 14.01
4 129          RajaRajuRickyShivVishnu 17.21
5 158          RajaRajuRickyShivVishnuRamesh 21.81
6 181          RajaRajuRickyShivVishnuRameshMahender 25.61
7 215          RajaRajuRickyShivVishnuRameshMahenderLious 29.39
```

```
8 255 RajaRajuRickyShivVishnuRameshMahenderLiousSalman 32.37
9 285 RajaRajuRickyShivVishnuRameshMahenderLiousSalm... 37.17
10 336RajaRajuRickyShivVishnuRameshMahenderLiousSalm... 41.27
11 382RajaRajuRickyShivVishnuRameshMahenderLiousSalm... 44.92
```

Most of the measures can be easily made available with the help of describe function.

```
import pandas as pd
import numpy as np
d = {'Name':pd.Series(['Raja', 'Raju', 'Ricky',
'Shiv', 'Vishnu', 'Ramesh', 'Mahender',
'Lious', 'Salman', 'SRK', 'VV', 'MM']),
'Age':pd.Series([25.00, 26, 25, 23, 30, 29,
23, 34, 40, 30, 51, 46.00]),
'Rating':pd.Series ([4.2320, 3.274, 3.98,
2.5628, 3.20, 4.600, 3.800, 3.780, 2.980,
4.8000, 4.1000, 3.65])}
df = pd.DataFrame(d)
print (df.describe())
```

Output:

```
=====
          Age      Rating
count    12.000000  12.000000
mean    31.833333  3.746567
std     9.232682  0.659021
min    23.000000  2.562800
25%    25.000000  3.255500
50%    29.500000  3.790000
75%    35.500000  4.133000
max    51.000000  4.800000
```

Q 9: Explain reindexing.

A: Reindexing allows us to modify the index of one data frame by keeping the other data frame as reference.

It can also be achieved by passing index and corresponding columns.

```

import pandas as pd
import numpy as np

d =
{'Name':pd.Series(['Raja','Raju','Ricky','Shiv','Vishnu','Ramesh','Mahender',
'',
'Licus','Salman','SRK','VV','MM']),
'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),
'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.
65])}

df = pd.DataFrame(d)

df_reindexed = df.reindex(index=[1,0,5], columns=['Age', 'Name', 'Rating'])
print(df_reindexed)

```

Retrieves following
row index data from
original data frame

Retrieves following
columns from
original data frame

Output:

	Age	Name	Rating
1	26	Raju	3.24
0	25	Raja	4.23
5	29	Ramesh	4.60

Q 10: Explain bfill and ffill.

A: While reindexing NaN can be introduced .**bfill** and **ffill** are used to handle NaN.

```

import pandas as pd
import numpy as np

df1 = pd.DataFrame(np.random.randn(4,3),columns=['col1','col2','col3'])
df2 = pd.DataFrame(np.random.randn(2,3),columns=['col1','col2','col3'])

print(df2.reindex_like(df1))

```

Since df2 is re indexed to a higher
value in rows it would causes NaN

Output:

	col1	col2	col3
0	-0.591340	-2.216463	0.305495

```

1    0.264238   0.406872 -0.879538
2        NaN        NaN        NaN
3        NaN        NaN        NaN

```

Bfill:

Fills the value from ahead value into the previous NaN value.

FFill:

Fills the value from behind value into the missing NaN value.

```
print(df2.reindex_like(df1,method='ffill'))
```

0	1.701588	1.142910	0.524430
1	0.629557	-1.137817	-0.763281
2	0.629557	-1.137817	-0.763281
3	0.629557	-1.137817	-0.763281

Q 11: What all type of iterations are provided in pandas data frame?

A:

iteritems():To iterate over the (key,value) pairs.

iterrows():Iterate over the rows as (index,series) pairs.

itertuples():Iterate over the rows as namedtuples.

Consider the following data frame sample:

	Age	Name	Rating
0	25	Raja	4.23
1	26	Raju	3.24
2	25	Ricky	3.98
3	23	Shiv	2.56
4	30	Vishnu	3.20
5	29	Ramesh	4.60
6	23	Mahender	3.80
7	34	Lious	3.78
8	40	Salman	2.98
9	30	SRK	4.80
10	51	VV	4.10
11	46	MM	3.65

for key,value in df.iteritems():

```
print(key,value)
```

Output:

```
Age 0      25
1      26
2      25
3      23
4      30
5      29
6      23
7      34
8      40
9      30
10     51
11     46
Name: Age, dtype: int64
Name 0          Raja
1          Raju
2          Ricky
3          Shiv
4          Vishnu
5          Ramesh
6          Mahender
7          Lious
8          Salman
9          SRK
10         VV
11         MM
Name: Name, dtype: object
Rating 0      4.23
1      3.24
2      3.98
3      2.56
4      3.20
5      4.60
6      3.80
7      3.78
8      2.98
9      4.80
10     4.10
```

for row_index, row in df.iterrows():

```
print (row_index,row)
0    Age           25
     Name        Raja
     Rating      4.23
Name: 0, dtype: object
1    Age           26
     Name       Raju
     Rating      3.24
Name: 1, dtype: object
2    Age           25
     Name       Ricky
     Rating      3.98
Name: 2, dtype: object
3    Age           23
     Name       Shiv
     Rating      2.56
Name: 3, dtype: object
4    Age           30
     Name      Vishnu
     Rating      3.2
Name: 4, dtype: object
5    Age           29
     Name      Ramesh
     Rating      4.6
Name: 5, dtype: object
6    Age           23
     Name    Mahender
     Rating      3.8
Name: 6, dtype: object
7    Age           34
     Name      Lious
     Rating      3.78
Name: 7, dtype: object
8    Age           40
     Name      Salman
     Rating      2.98
Name: 8, dtype: object
9    Age           30
     Name       SRK
```

```
Rating      4.8
Name: 9, dtype: object
10 Age        51
Name       VV
Rating     4.1
Name: 10, dtype: object
11 Age        46
Name       MM
Rating     3.65
Name: 11, dtype: object
```

for row in df.itertuples():

print (row)

Output:

```
Pandas(Index=0, Age=25, Name='Raja',
Rating=4.23)
Pandas(Index=1, Age=26, Name='Raju',
Rating=3.24)
Pandas(Index=2, Age=25, Name='Ricky',
Rating=3.98)
Pandas(Index=3, Age=23, Name='Shiv',
Rating=2.56)
Pandas(Index=4, Age=30, Name='Vishnu',
Rating=3.2)
Pandas(Index=5, Age=29, Name='Ramesh',
Rating=4.6)
Pandas(Index=6, Age=23, Name='Mahender',
Rating=3.8)
Pandas(Index=7, Age=34, Name='Lious',
Rating=3.78)
Pandas(Index=8, Age=40, Name='Salman',
Rating=2.98)
Pandas(Index=9, Age=30, Name='SRK', Rating=4.8)
Pandas(Index=10, Age=51, Name='VV', Rating=4.1)
Pandas(Index=11, Age=46, Name='MM',
Rating=3.65)
```

Q 12: Explain how sorting is supported in pandas?

A: **sort_index**: Allows sorting on index(row) or column wise.

sort_values: Allows sorting on values.

Consider the following data frame:

	col2	col1
1	0.048787	-0.370136
4	0.846598	-0.475094
6	0.626231	-1.028268
2	-2.056041	0.280162
3	-1.398099	-0.425764
5	1.974402	-0.978264
9	1.511170	-0.589333
8	-1.744891	-0.919853
0	-1.273972	0.469276
7	1.794093	1.048517

```
sorted_df = unsorted_df.sort_values(by='col1')  
print(sorted_df)
```

	col2	col1
6	0.626231	-1.028268
5	1.974402	-0.978264
8	-1.744891	-0.919853
9	1.511170	-0.589333
4	0.846598	-0.475094
3	-1.398099	-0.425764
1	0.048787	-0.370136
2	-2.056041	0.280162
0	-1.273972	0.469276
7	1.794093	1.048517

Results are sorted on
col1 in ascending order

```
sorted_df = unsorted_df.sort_index(axis=1)  
print(sorted_df)
```

	col1	col2
1	-0.852017	-0.530043
4	0.422459	1.355061
6	-0.340375	0.226448
2	0.236609	0.074492
3	1.571648	0.770137
5	0.240715	-0.393545
9	-0.247026	-1.105864
8	-0.046523	0.003584
0	-1.024904	-0.761627
7	1.092764	2.268048

Columns are rearranged

```
sorted_df=unsorted_df.sort_index()
print(sorted_df)
```

	col2	col1
0	0.194199	0.019298
1	-0.211437	-0.579378
2	-0.925317	-1.612364
3	-0.968525	-0.399602
4	0.231762	-0.449736
5	-0.444578	-0.670510
6	0.304853	1.758049
7	-0.083144	0.537487
8	-0.301545	0.417964
9	-0.444547	0.010484

Indexes are rearranged
in ascending order

```
sorted_df=unsorted_df.sort_index(ascending=False)
print(sorted_df)
```

	col2	col1
9	-1.003009	-1.151731
8	1.317469	-0.774446
7	-0.170884	0.694429
6	-1.161735	-0.702071

```

5  -1.232564  0.843178
4  -0.550387  2.167216
3  -0.367910  -0.658615
2  -0.297523  0.277371
1   0.551069  1.218894
0   1.320420  0.439020

```

Q 13: How to override default reload option in pandas?

A:

```

pd.set_option("display.max_rows",80)
pd.set_option("display.max_columns",30)

```

Set maximum rows to be displayed

Set maximum columns to be displayed

Q 14: Explain various slicing options available with pandas?

A: Pandas supports the following slicing:

- .loc()** Label based
- .iloc()** Integer based
- .ix()** Both Label and Integer based

Consider the following data frame:

	A	B	C	D
a	-1.53558	0.297237	-0.56182	1.401168
b	0.473623	1.080733	0.095463	0.683253
c	-0.1642	0.823932	-0.66819	1.423232
d	0.943825	-0.40707	-0.04698	0.133415
e	-0.47114	-1.00112	-1.0575	0.843195
f	-0.35179	-1.47116	1.612342	-0.74294
g	-0.63809	0.231685	0.327186	-0.73119
h	-0.60686	-1.59435	-0.24473	-2.33809

```
print (df.loc[:, 'A':'C'])
```

Performs slicing based on label of column only columns A to C would be fetched

	A	B	C
a	-1.53558	0.297237	-0.56182
b	0.473623	1.080733	0.095463
c	-0.1642	0.823932	-0.66819
d	0.943825	-0.40707	-0.04698
e	-0.47114	-1.00112	-1.0575
f	-0.35179	-1.47116	1.612342
g	-0.63809	0.231685	0.327186
h	-0.60686	-1.59435	-0.24473

```
print (df.loc['c':'g', 'A':'C'])
```

Performs slicing on columns A to C

Performs slicing on row c to g

Output:

	A	B	C	D
c	-0.1642	0.823932	-0.66819	1.423232
d	0.943825	-0.40707	-0.04698	0.133415
e	-0.47114	-1.00112	-1.0575	0.843195
f	-0.35179	-1.47116	1.612342	-0.74294
g	-0.63809	0.231685	0.327186	-0.73119
h	-0.60686	-1.59435	-0.24473	-2.33809

```
print (df.iloc[1:5, 2:4])
```

Columns 2 to 4 would be fetched

Slices based on index number. Here row 1 to 5 would be fetched

	A	B	C	D
b	0.473623	1.080733	0.095463	0.683253
c	-0.1642	0.823932	-0.66819	1.423232
d	0.943825	-0.40707	-0.04698	0.133415
e	-0.47114	-1.00112	-1.0575	0.843195

`print (df.ix['c',:4])`

Allows slicing on
indexes as well as labels

	A	B	C	D
c	-0.1642	0.823932	0.66819	1.423232

Q 15: Explain advanced statistics with pandas.

A: Pandas support peak to peak, percentage change, covariance, and correlation on different series in data frame.

```
s1 = pd.Series(np.random.randn(10))
s2 = pd.Series(np.random.randn(10))
frame = pd.DataFrame({'s1':s1,'s2':s2})
print(frame)
print (frame.pct_change())
print (frame.cov())
print (frame.corr())
```

Calculates percentage change

Calculates Covariance

Calculates Correlation

Output

Data frame

	s1	s2
0	-1.448675	-1.045116
1	0.453446	0.558663
2	-0.577190	0.697816
3	0.085386	-0.181952

```

4    0.524836   0.379238
5    1.042358   1.0001935
6    2.558341   0.182684
7    0.517772   -0.358159
8    0.442801   -0.475986
9    1.614731   1.121278

```

Pct change

	s1	s2
0	NaNNaN	
1	-1.313007	-1.534547
2	-2.272896	0.249081
3	-1.147933	-1.260745
4	5.146660	-3.084272
5	0.986064	1.641970
6	1.454378	-0.817669
7	-0.797614	-2.960538
8	-0.144794	0.328977
9	2.646625	-3.355698

Covariance:

	s1	s2
s1	1.211658	0.388197
s2	0.388197	0.484890

Correlation:

	s1	s2
s1	1.000000	0.506455
s2	0.506455	1.000000

Q 16: Explain rolling function.

A: Rolling function allows us to define window size and perform various operations w.r.t. the window.

Expanding function is an alternative to rolling statistics is to use an expanding window, which yields the value of the statistic with all the data available up to that point in time.

Consider the following data frame:

	A
2010-01-01	1.163625
2010-01-02	0.408765
2010-01-03	-0.093993

```
2010-01-04 -1.206746  
2010-01-05 -0.862096
```

```
print(df.rolling(window=3).mean())
```

A

2010-01-01 1.163625
2010-01-02 0.408765
2010-01-03 -0.093993
2010-01-04 -1.206746
2010-01-05 -0.862096

Calculates rolling mean
using the window size as 3

The mean of first three
values will be calculated
and placed inside 3 location.
First 2 values cannot have
mean since they do not
satisfy the window of 3

```
print(df.expanding(min_periods=3).mean())
```

2010-01-01 1.163625
2010-01-02 0.408765
2010-01-03 -0.093993
2010-01-04 -1.206746
2010-01-05 -0.862096

Expanding makes all the
values available till then for
calculation. min_periods
condition needs to be satisfied

Q 17: How can we handle NA in pandas?

A: We can handle NA in three ways:

1. dropna: Removes the data where NA is present.
2. pad
3. backfill

Consider the following dataframe:

	one	Two	Three
a	0.121671	-0.90002	0.11258
b	NaN	NaN	NaN
c	-1.15264	-0.70297	0.76683
d	NaN	NaN	NaN
e	-0.03259	-0.75593	-0.3015
f	0.968174	0.107281	1.42007
g	NaN	NaN	NaN
h	-0.06344	2.590979	0.59222

dropna output

```
print (df.dropna())
```

	one	Two	three
a	0.121671	-0.90002	0.11258
c	-1.15264	-0.70297	0.76683
e	-0.03259	-0.75593	-0.3015
f	0.968174	0.107281	1.42007
h	-0.06344	2.590979	0.59222

Pad output

```
print (df.fillna(method='pad'))
```

	one	two	three
a	0.121671	-0.90002	0.11258
b	0.121671	-0.90002	0.11258
c	-1.15264	-0.70297	0.76683
d	-1.15264	-0.70297	0.76683
e	-0.03259	-0.75593	-0.3015
f	0.968174	0.107281	1.42007
g	0.968174	0.107281	1.42007
h	-0.06344	2.590979	0.59222

NaN are replaced
by previous non
NaN values

Backfill:

```
print (df.fillna(method='backfill'))
```

	one	two	three
a	0.121671	-0.90002	-0.11258
b	-1.15264	-0.70297	-0.76683
c	-1.15264	-0.70297	-0.76683
d	-0.03259	-0.75593	-0.3015
e	-0.03259	-0.75593	-0.3015
f	0.968174	0.107281	-1.42007
g	-0.06344	2.590979	-0.59222
h	-0.06344	2.590979	-0.59222

Nan is replaced
with ahead values

Q 18: Explain group by function.

A: **group_by** allows us to group data based on single or multiple columns.

It is equivalent to *Group by* clause supported in **Structured Query language (SQL)**.

Consider the following dataframe:

	Points	Rank	Team	Year
0	876	1	R	2014
1	789	2	R	2015
2	863	2	D	2014
3	673	3	D	2015
4	741	3	K	2014
5	812	4	K	2015
6	756	1	K	2016
7	788	1	K	2017
8	694	2	R	2016
9	701	4	RR	2014
10	804	1	RR	2015
11	690	2	R	2017

Will group the
above data based
on year

```
grouped = df.groupby('Year')
```

	Points	Rank	Team	Year
0	876	1	R	2014
2	863	2	D	2014
4	741	3	K	2014
9	701	4	RR	2014

Year wise
grouped

2015

	Points	Rank	Team	Year
1	789	2	R	2015
3	673	3	D	2015
5	812	4	K	2015
10	804	1	RR	2015

2016

	Points	Rank	Team	Year
6	756	1	K	2016
8	694	2	R	2016

2017

	Points	Rank	Team	Year
7	788	1	K	2017
11	690	2	R	2017

Q 19: Explain merge functions w.r.t data frame.

A: Data frame in pandas support merge operations in which two related data from diverse data frames can be brought in single view.

Consider the following left and right data frame:

Left

	FirstName	Id	subject_id
0	Vishwa	11	sub1
1	Mahesh	2	sub2
2	Kallu	33	sub4
3	Ballu	44	sub6
4	Mallu	55	sub5

Right

	LastName	Id	subject_id
0	Nara	11	sub1
1	Kal	2	sub4
2	Challu	3	sub3
3	Shana	4	sub6

Records matching by id
on both the data frames
would be returned

```
print ('id merge',pd.merge(left,right,on='id'))
```

Id	merge	FirstName	id2	subject_id_x	LastName	subject_id_y
0	Vishwa		11	sub1	Nara	sub1
1	Mahesh		2	sub2	Kal	sub4

Left merge:

In left merge, all data from left side will come and only those matching from right would come.

Left indicates everything from left

NaN can be introduced

```
print ('left',pd.merge(left, right, on='id', how='left'))
```

left	FirstName	id	subject_id_x	LastName	subject_id_y
0	Vishwa	11	sub1	Nara	sub1
1	Mahesh	2	sub2	Kal	sub4
2	Kallu	33	sub4	NaN	NaN
3	Ballu	44	sub6	NaN	NaN
4	Mallu	55	sub5	NaN	NaN

Right merge:

In right merge everything from right side comes and only matching in left would come else it would come as NaN.

```
print ('right',pd.merge(left, right, on='id', how='right'))
```

right	FirstName	id	subject_id_x	LastName	subject_id_y
0	Vishwa	11	sub1	Nara	sub1
1	Mahesh	2	sub2	Kal	sub4
2	NaN	3	NaN	Challu	sub3
3	NaN	4	NaN	Shana	sub6

Outer merge:

Data from both left and right would come.

```
print ('outer',pd.merge(left, right, on='subject_id', how='outer'))
```

outer	FirstName	id_x	subject_id	LastName	id_y
0	Vishwa	11	sub1	Nara	11
1	Mahesh	2	sub2	NaN	NaN
2	Kallu	33	sub4	Kal	2
3	Ballu	44	sub6	Shana	4
4	Mallu	55	sub5	NaN	NaN
5	NaN	NaN	sub3	Challu	3

Q 20: Explain concat method.

A: **Concat** method allows combining two different data frames either at row level or at column level.

Consider the following left and right data frame:

Left

	Name	Id	subject_id
1	Vishwa	11	sub1
2	Mahesh	2	sub2
3	Kallu	33	sub4
4	Ballu	44	sub6
5	Mallu	55	sub5

Right

	Name	Id	subject_id
1	Nara	11	sub1
2	Kal	2	sub4
3	Challu	3	sub3
4	Shana	4	sub6

```
print (pd.concat([left,right]))
```

	Name	Id	subject_id
0	Vishwa	11	sub1
1	Mahesh	2	sub2
2	Kallu	33	sub4
3	Ballu	44	sub6
4	Mallu	55	sub5
0	Nara	11	sub1
1	Kal	2	sub4
2	Challu	3	sub3
3	Shana	4	sub6

```
print (pd.concat([left,right],axis=1))
```

	Name	id	subject_id	Name	id	subject_id
0	Vishwa	11	sub1	Nara	11	sub1
1	Mahesh	2	sub2	Kal	2	sub4
2	Kallu	33	sub4	Challu	3	sub3
3	Ballu	44	sub6	Shana	4	sub6
4	Mallu	55	sub5	NaN	NaN	NaN

Q 21: Explain how time related range can be generated in pandas.

A: Pandas support generating date range with the help of two functions **date_range** and **bdate_range** (business date range).

Periods and frequency can be specified as attribute to the ranges.

```
start = pd.datetime(2018, 1, 1)
```

```
end = pd.datetime(2020, 1, 5)
```

```
print (pd.bdate_range(start, end))
```

Output:

```
DatetimeIndex(['2018-01-01', '2018-01-02',
 '2018-01-03', '2018-01-04',
 '2018-01-05', '2018-01-08', '2018-01-09',
 '2018-01-10',
 '2018-01-11', '2018-01-12',
 ...
 '2019-12-23', '2019-12-24', '2019-
```

```

12-25', '2019-12-26',
              '2019-12-27', '2019-12-30', '2019-
12-31', '2020-01-01',
              '2020-01-02', '2020-01-03'],
dtype='datetime64 [ns]', length=525,
freq='B')

print(pd.date_range('1/1/2011', periods=5,freq='M'))

```

Output:

```
DatetimeIndex(['2011-01-31', '2011-02-28', '2011-03-31', '2011-
04-30',
               '2011-05-31'],
              dtype='datetime64[ns]', freq='M')
```

Q 22: Explain which all data sources can pandas retrieve values.

A: Pandas are capable of reading and writing from variety of sources.

Following are the sources and the corresponding read or write methods:

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq
Remote data sources			

Format Type	Data Description	Reader	Writer
	Reads from remote data sources like <ul style="list-style-type: none"> • Google Finance • Morningstar • IEX • Robinhood • Enigma • Quandl • St.Louis FED (FRED) • Kenneth French's data library • World Bank • OECD • Eurostat • Thrift Savings Plan • Nasdaq Trader symbol definitions • Stooq • MOEX 	web.DataReader	

Q 23: How to print a histogram using pandas?

A: Using `series.value_counts()` functions we can easily generate the value and its corresponding count.

```
import pandas as pd
import numpy as np
s = pd.Series([1,1,100,120,30,40,60,60])
print(s)
print(s.value_counts())
```

Output:

S	
0	1
1	1
2	100

3	120
4	30
5	40
6	60
7	60

Histogram

Value count

60	2
1	2
30	1
40	1
100	1
120	1

Scipy and its Applications

Q 1: Explain Scipy library.

A: Scipy is used for scientific computation. It is a collection of components and provides eco system for scientific computing.

Data and computation:

- pandas, provides ease of use and very high performing data
- SymPy, for symbolic mathematics and computer algebra.
- scikit-image used for image processing.
- scikit-learn is used for machine learning.
- h5py and PyTables can both access data stored in the HDF5 format.

Q 2: Explain how can we perform Normality Tests.

A: Normality tests are the ones which test for data distribution in the form of Gaussian.

Each of this has following assumptions on input and output:

Input assumptions:

- Observations in each sample are independent and identically distributed.

Output assumptions:

- H0: the sample has a Gaussian distribution
- H1: the sample does not have a Gaussian distribution

There are three types of test within.

All the three types are supported as a part of scipy.

```
import scipy.stats as ss
Shapiro-Wilk Test
stat, p = ss.shapiro(data)
D'Agostino's K^2 Test
```

```
stat, p = ss.normaltest(data)
Anderson-Darling Test
result =ss.anderson(data)
```

Conclusion: If calculate p value < alpha, we can conclude null hypothesis rejected else it cannot be rejected.

Q 3: Explain how can we perform correlation test?

A: **Pearson's Correlation Coefficient**

Tests have a linear relationship that exist between two data set.

Input Assumptions:

All these are made w.r.t Observations

- They are independent and identically distributed
- They are normally distributed
- They have the same variance

Output assumptions:

- H0: the two samples are independent.
- H1: two samples have dependency.

```
import scipy.stats
corr, p = scipy.stats.pearsonr(data1, data2)
```

Based on value of **corr** and **p** we can conclude whether their relationship between both columns.

Spearman's Rank Correlation and Kendall's Rank Correlation.

Both of them have same input and output assumptions.

Tests whether monotonic relationship exists between data set.

Input assumptions:

All assumptions are made w.r.t. observations:

- They are independent and identically distributed
- They can be ranked

Output assumptions:

- H0: The two samples are independent
- H1: There is a dependency between the samples

Spearman's Rank Correlation

```
import scipy.stats
corr, p = scipy.stats
```

```
.spearmanr(data1, data2)
```

Kendall's Rank Correlation

```
import scipy.stats
corr, p = scipy.stats.kendalltau(data1, data2)
```

Based on value of **corr** and **p** we can conclude whether their relationship between both columns.

Chi-Squared Test

Tests independence between two categorical variables:

Input assumptions:

- Observations are independent.
- 25 or more examples in each cell are required w.r.t. the contingency table.

Output assumptions:

- H0: the two samples are independent.
- H1: there is a dependency between the samples.

```
import scipy.stats
stat, p, dof, expected = scipy.stats.chi2_contingency(table)
```

Q 4: Explain tests pertaining to Parametric Statistical Hypothesis Tests.

A: Comparison of data is done by the following test:

Student's t-test

It tests whether the means of two independent samples are significantly different.

Input assumptions:

All assumptions are made w.r.t observation.

- They are independent and identically distributed (iid).
- They are normally distributed.
- They have the same variance.

Output assumptions:

- H0: the means of the samples are equal
 - H1: the means of the samples are unequal
- ```
import scipy.stats as ss
stat, p = ss.ttest_ind(data1, data2)
```

### Paired Student's t-test

Tests whether the significant difference exists between the means of two paired samples.

#### Input assumptions:

- Observations in each sample are independent and identically distributed
- Observations in each sample are normally distributed
- Observations in each sample have the same variance
- Observations across each sample are paired

#### Output assumptions:

- H0: the means of the samples are equal
- H1: the means of the samples are unequal

```
import scipy.stats
stat, p = scipy.stats.ttest_rel(data1, data2)
```

### Analysis of Variance Test (ANOVA)

It tests whether the means of two or more independent samples are significantly different.

#### Input assumptions:

All assumptions are made w.r.t observation:

- They are independent and identically distributed (iid).
- They are normally distributed.
- They have the same variance.

#### Output assumptions:

- H0: the means of the samples are equal.
- H1: one or more of the means of the samples are unequal.

```
import scipy.stats
stat, p = scipy.stats.f_oneway(data1, data2,
...)
```

Q 5: Explain how to test Nonparametric Statistical Hypothesis Tests.

A:

Mann-Whitney U Test:

It tests whether the two independent samples are equal or not.

#### Input Assumptions:

All assumptions are made w.r.t observation:

- They are independent and identically distributed.
- They can be ranked.

### Output Assumptions:

- H0: the distributions of both samples are equal.
- H1: the distributions of both samples are not equal.

```
import scipy.stats
stat, p = scipy.stats.mannwhitneyu(data1, data2)
```

Q 6: Implement logistic regression in python?

A: **sklearn.linear\_model** contains **LogisticRegression** which can be used to represent logistic regression.

```
import sklearn.linear_model as sklm
logmodel = sklm.LogisticRegression()
logmodel.fit(feautures,label)
y_pred= logmodel.predict(X_test)
```

Predict using the above created and enhanced model with test data

Make the model with the help of train data. features and label represents data and corresponding labels used for training the model

Q 7: Explain how to implement decision tree in python.

A: **sklearn.tree** module contains **DecisionTreeClassifier** which can be used for classification.

```
import sklearn.tree as slt
model = slt.DecisionTreeClassifier()
model.fit(feautures,label)
y_pred = model.predict(X_test)
```

Fit the model using features and label where features represents data and labels represents labels generated used for training

Q 8: How to implement Random forest in python?

A: **sklearn.ensemble** module contains **RandomForestClassifier** which can be used for Random forest generation.

```
import sklearn.ensemble as ske
model=ske.RandomForestClassifier(n_estimators=50)

model.fit(features,label)

y_pred=clf.predict(X_test)
```

n\_estimators  
indicates number of  
trees in random  
forest

For regression type we will use **RandomForestRegressor** instead of **RandomForestClassifier**.

Q 9: How to implement support vector machine in python?

A:

**sklearn.svm** module contains SVC which is used for svm.

**SVM for linear model code:**

```
import sklearn.svm as svm
model = svm.SVC(kernel='linear')
model.fit(features,label)
y_pred=clf.predict(X_test)
```

Indicates linearly  
separable  
approach

**SVM for non-linear/polynomial mode code**

```
import sklearn.svm as svm
model = svm.SVC(kernel='poly', degree=9)
model.fit(features, label)
y_pred=clf.predict(X_test)
```

The value of kernel to poly  
and degree indicates that this  
svm supports polynomial  
approach

Q 10: Which all kernels are supported by svm in python?

A: Linear, poly, sigmoid (which returns 0 or 1 kernel='sigmoid') and gaussian (to support gaussian distribution kernel='rbf')

Q 11: Implement KNN algorithm using python.

A: **sklearn.neighbors** module contains **KNeighborsClassifier** which can be used for classification:

```
import sklearn.neighbors as sn
model = sn.KNeighborsClassifier(n_neighbors=3)
model.fit(features,label)
predicted= model.predict(X_test)
```

n\_neighbors  
stands for number  
of neighbors used  
for classification

Q 12: How to select k in KNN algorithm?

A: Finding an optimal value of **k** is very important in order to ensure success of KNN algorithm.

The value of **k** is heavily dependent on the data set.

A very small value of **k** implies noise can have a bigger influence on the result.

A large value of **k** implies more calculations which can be time and memory consuming.

Mostly Odd classes are preferred and **k= sqrt(noOfElements)** is generally taken as an approach to find **k**.

Alternatively, Elbow method can also be used. In the case of Elbow method. In Elbow method we calculate the sum of squared errors (SSE) for different values of **k** and plot the same.

The one which looks like an elbow is selected as the optimal.

Q 13: How to implement K means in Python?

A: We can implement K means with the help of **sklearn.cluster** module which contains **KMeans** .

```
import sklearn.cluster as skc
kmeans = skc.KMeans(n_clusters=20)
model = kmeans.fit(X)
labels = model.predict(X)
centroids = model.cluster_centers_
```

n\_clusters stands  
for number of  
clusters

Cluster centers  
can also be  
determined

Q 14: How can accuracy of any model be calculated?

A: Following approaches can be used to determine accuracy:

1.

Gives a tabular representation of expected vs predicted

```
import sklearn.metrics as skm
skm.confusion_matrix(labels_train, pred)
```

2.

```
from sklearn.metrics import accuracy_score
accuracy_score(labels_test, pred)
```

Accuracy score is a good indicator of how the algorithm has performed

3.

```
from sklearn.metrics import classification_report
report = classification_report(Y_test, predicted)
```

displays the precision, recall, f1-score and support for each class

4.

```
from sklearn.metrics import recall_score
recall_score(label,predicted)
```

Recall /sensitivity is the fraction of positives events predicted correctly

5.

```
from sklearn.metrics import precision_score
precision_score(label,predicted)
```

is the fraction of predicted positives events that are actually positive

6.

```
from sklearn.metrics import f1_score
f1_score(label,predicted)
```

harmonic mean of recall and precision. High value is preferred

Q 15: Explain regression metrics.

A: Regression metrics gives an indication of how the regression algorithm performed.

Following are the important ones:

| Metric name                           | Definition                                                             | Code in Python                                                                                                                                                     |
|---------------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Mean Absolute Error (MAE)</b>      | Sum of the absolute differences between predictions and actual values. | <pre>import sklearn as sk result = sk.model_selection.cross_val_ score(model, X, Y, scoring='neg_ mean_absolute_error') print(results.mean(), results.std())</pre> |
| <b>Root Mean Squared Error (RMSE)</b> | It is the standard deviation of the prediction errors.                 | <pre>from sklearn.metrics import mean_ squared_error from math import sqrt rmse = sqrt(mean_squared_error(y_ actual, y_predicted))</pre>                           |



# Matplotlib Samples to Remember

Q 1: Explain how to draw bar plot.

A:

```
from matplotlib import pyplot as plt

a1= [5,8,10]

b1= [12,16,6]

a2 = [6,9,11]

b2 = [6,15,7]

plt.bar(a1, b1, align = 'center')

plt.bar(a2, b2, color = 'g', align = 'center')

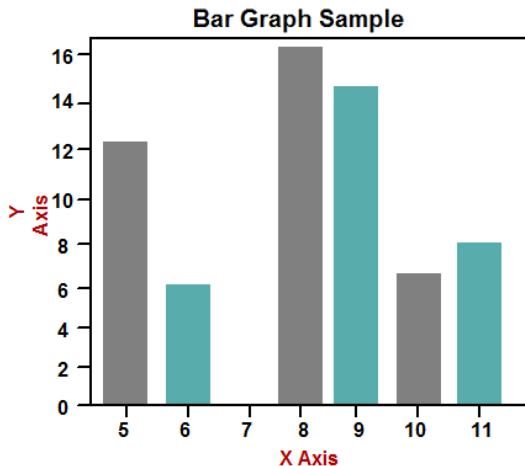
plt.title('Bar graph sample')

plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

Generates bar using a2 and b2

Name given to x and y co-ordinate in graph

Display plot generated



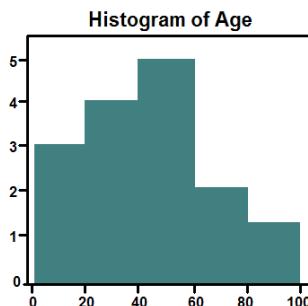
Q 2: How to draw histogram?

A:

```
from matplotlib import pyplot as plt
import numpy as np

data = np.array([22,87,5,43,56,73,55,54,11,20,51,5,79,31,27])
plt.hist(data, bins = [0,20,40,60,80,100])
plt.title("Histogram of age")
plt.show()
```

Histogram generated using data and groups represented by bins



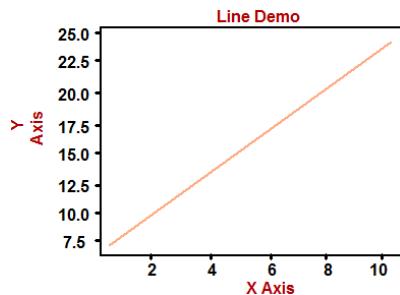
Q 3: How to draw line chart?

A:

```
import numpy as np
```

```
from matplotlib import pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
```

```
x = np.arange(1,11)
y = 2 * x + 5
f = plt.figure()
plt.title("Line demo")
plt.xlabel("X Axis")
plt.ylabel("Y axis")
plt.plot(x,y)
plt.show()
```



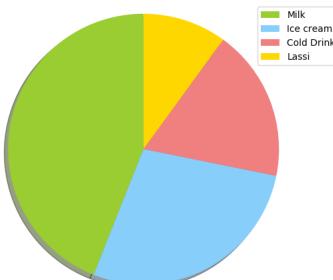
Line chart which plots the x data on x co-ordinate and y data on y co-ordinate

Q 4: Draw Pie chart.

A:

```
import matplotlib.pyplot as plt

labels = ['Milk', 'Ice cream', 'Cold Drink', 'Lassi']
sizes = [48.4, 30.6, 20, 11]
colors = ['yellowgreen', 'lightskyblue', 'lightcoral', 'gold']
patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
```



Legend created with labels

Data passed as sizes with colors and extra attributes

Q 5: Draw scatter plot.

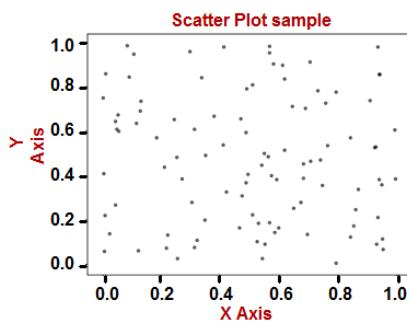
A:

```
import numpy as np
import matplotlib.pyplot as plt

N = 100

x = np.random.rand(N)
y = np.random.rand(N)
colors = (0,100,100)
area = np.pi*4

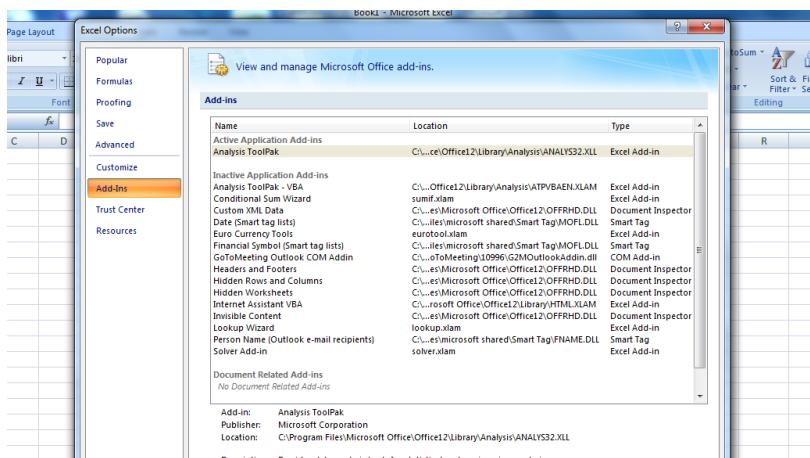
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.title('Scatter plot sample')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



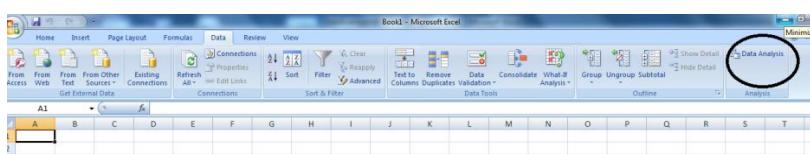
# Statistics with Excel Sheet

Q 1: Does Excel has any support for statistics?

A: Yes, Excel sheet do provide a lot of support for statistics with few add-ins like *Analysis Tool pack*. In order to enable this analysis tool pack, select excel options and the go to Add-ins option and enable *AnalysisToolPak*.



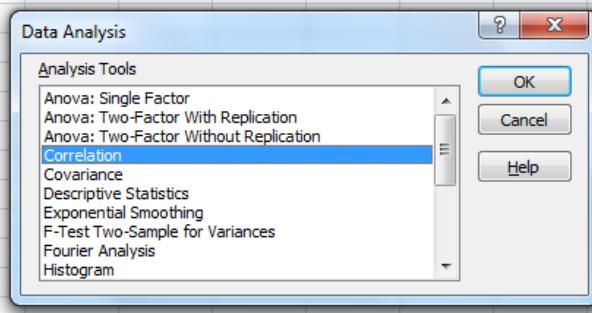
Once done, the data analysis option is enabled in Data tab of excel, it can be seen under the **Data** tab as shown in the image down below:



Q 2: Find correlation using Excel.

A: Correlation can be easily found with the help of **CORREL** function of Analysis Toolpak.

Select **Data Analysis** → Select **correlation** as an option.



After selecting **Correlation** from Data Analysis select the corresponding input field and set output values to be displayed on the excel sheet by setting **Output Range** available under **Output options**,

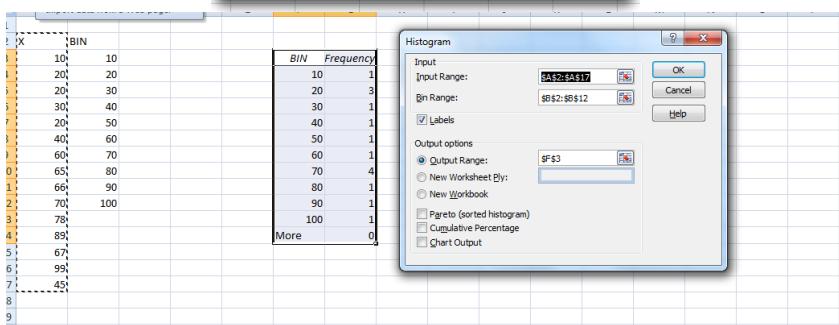
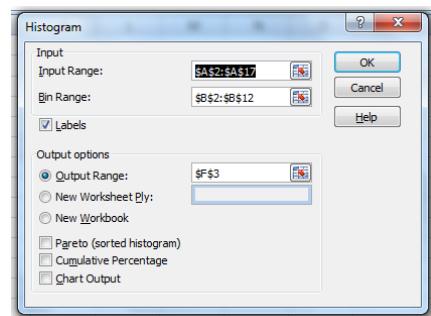
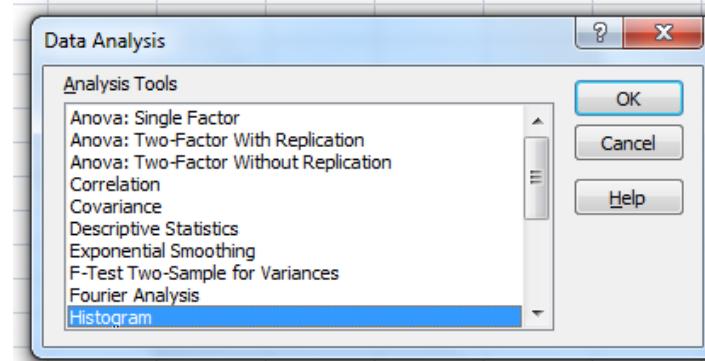
|    | X | Y |
|----|---|---|
| 1  |   |   |
| 2  | 1 | 4 |
| 3  | 2 | 8 |
| 4  | 3 | 3 |
| 5  | 4 | 4 |
| 6  |   |   |
| 7  |   |   |
| 8  |   |   |
| 9  |   |   |
| 10 |   |   |
| 11 |   |   |
| 12 |   |   |
| 13 |   |   |
| 14 |   |   |
| 15 |   |   |
| 16 |   |   |
| 17 |   |   |

and after **OK** is clicked, the result for same of Covariance can be found on excel sheet.

Q 3: How to get Histogram in excel?

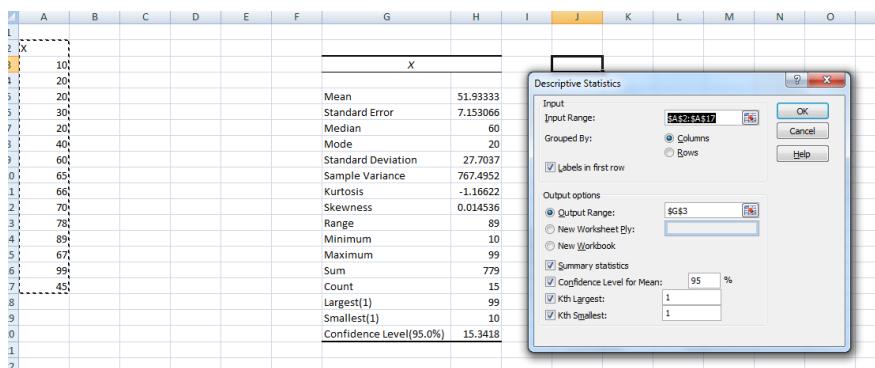
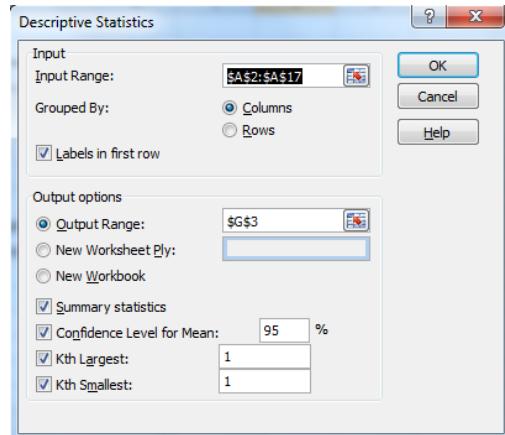
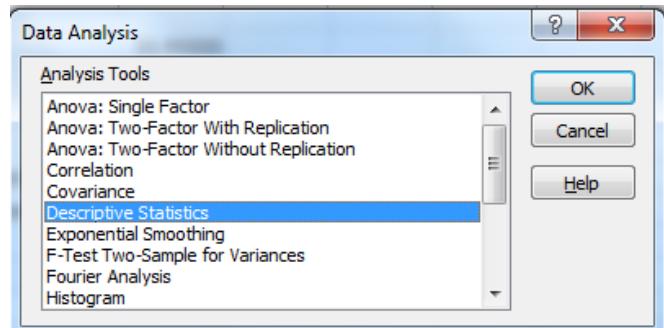
A: To get Histogram in excel we generally require values and corresponding bins. Bins represent the range in which frequency is calculated.

From **Data Analysis** → select **Histogram** function → we can select input range and bins range



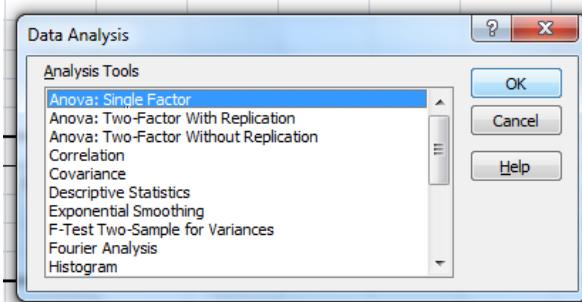
Q 4: Explain how to get Descriptive Statistics using Excel.

A: To get **Descriptive Statistics** select **Data Analysis** => Select **Descriptive Analysis**=> then select different statistics you prefer.

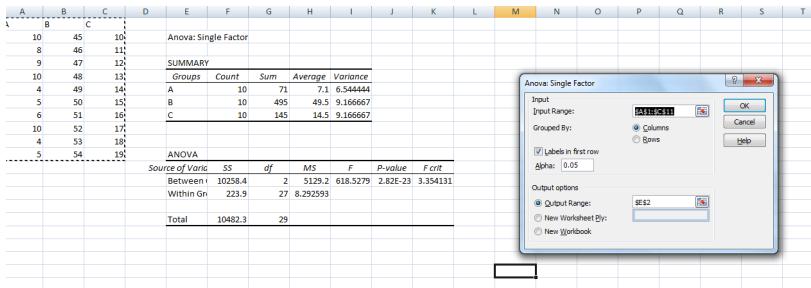
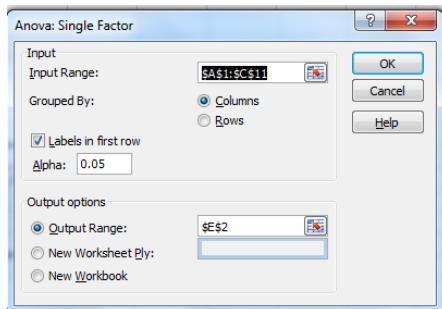


Q 5: Explain how to perform Anova in excel?

A: To perform Anova select Data Analysis =>Anova: Single factor



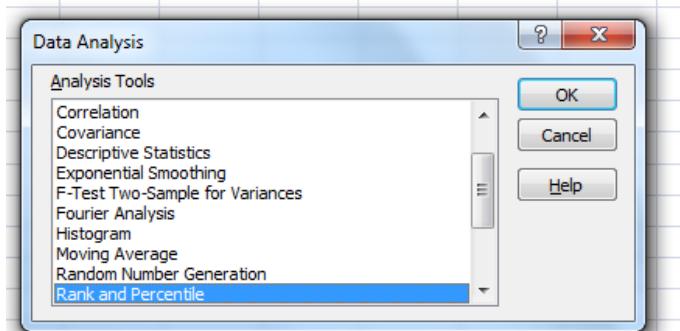
Select corresponding input and output values.



Based upon the value of F and F critical, we can conclude whether null hypothesis is rejected or accepted.

Q 6: Explain how to perform Rank and Percentile in excel.

A: From Data Analysis → select Rank and Percentile



Select the input and output range to get the answer: