1. **fWhat is Python?**

Python is a programming Languages. It is used to create web application.

### Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python is case sensitive language, i.e Variable and variable both are not same.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

1. **How do you identify the Python Version ?**

Python –-version.

## **What do you mean by Python Indentation ?** Indentation refers to the spaces at the beginning of a code line.

1. How do you define the comments?

* Single Line comments can be declared using # ,
* Multiline comments can be declared using “”” .., “”””

1. Why do you define comments?

Comments can be used to explain Python code. Comments can be used to make the code more readable. Comments can be used to prevent execution when testing code.

1. What do you mean by variable ?

Basically variables are containers for storing the data values.

1. What do you mean by Global Variable ?

To create a global variable inside a function, you can use the global keyword.

def myfunc():  
  global x  
  x = "fantastic"  
  
myfunc()  
  
print("Python is " + x)

1. What exactly the obligation of variable?

Variables can store data of different types, and different types can do different things.

## **How do we know the data type of a variable ?**

You can get the data type of any object by using the type() function:

x = 5  
print(type(x))

1. How to set the specific data type of a variable or typecasting

Int(x) or int(y)

1. How to install the Python ?

<https://www.python.org/>

1. Different types of data ?

Primary data types in Python ? **Integer ,Float , Boolean ,String.**

|  |  |
| --- | --- |
| Text Type: | Str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | Dict |
| Set Types: | set, frozenset |
| Boolean Type: | Bool |
| Binary Types: | bytes, bytearray, memoryview |

1. What is the list and what are the different kind of operations on List ?

List is ordered and changeable object.

Creating a list :

L1 = [1,2,3,4] 🡪 Square breackets.

How to access a single element from a list ? l1[0]

How to access a multiple elements from a list ?

l1[0],l1[1]

Accessing the sub list from beginning and End

L1[2:5] 🡪 2 included, 5 excluded. Index starts with zero.

L1[-5,-2] 🡪 Index starts with -1 from the end. -5 to -3 , -2 element excluded.

Check whether element is exists or not ?

thislist = ["apple", "banana", "cherry"]  
if "apple" in thislist:  
  print("Yes, 'apple' is in the fruits list")

thisdict = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
if "model" in thisdict:  
  print("Yes, 'model' is one of the keys in the thisdict dictionary")

txt = "The rain in Spain stays mainly in the plain"  
x = "ain" not in txt  
print(x)

Change the list elements

L1[0 ] = ‘Banana’

Add the elements in a list :

* append
* Insert

Append will add elements at the end of the string.

Insert will add at specified index.

Remove the elements:

Pop ,

del

Del l1[0] .. Delete the first element

Del l1[0] .. Delete the first element ., Any option two delete multiple elements.

Del l1 🡪 Delete complete list.

Pop : By default it will delete the last element.

1. Different Kinds of operators ?

* Arithmetic operators
* Assignment operators
* Comparison operators
* Logical operators
* Identity operators
* Membership operators
* Bitwise operators

16.Casting :

In order to the casting and change the data type we have constructors in Python

As shown below List() , int(),float(),complex(),….

1. We have three types of numbers:
2. Int , float , complex (3j .Imaginary Type)
3. String is nothing but sequence of characters , The strip() method removes any whitespace from the beginning or the end:
4. quantity = 3  
   itemno = 567  
   price = 49.95  
   myorder = "I want {} pieces of item {} for {} dollars."  
   print(myorder.format(quantity, itemno, price))
5. Dict : unordered and change

This is so called Mapping type.

Remove options:

Del d; 🡪 delete dictionary.

Pop(“name”) 🡪 remove the specific key.

Popitem() 🡪 remove the lastitem.

Add the element d[“age”:”28”]

Dictionary: Mixed Data types of data will store in key, value pair.

defining a Dictionary:

Dict1 = {'Priya':40,'Siva':55}

Dict2 = {'Priya':[40,78],'Siva':55}

accessing a element from Dictionary using keys:

d= {“Name”:”Priya”}

Accessing the Dict : d[“name”]

accessing an element from Dictionary using multiple keys (or) SubList.

Accessing all keys from Dictionary:

for x in thisdict.values():  
  print(x)

Accessing all keys,values from Dictionary:

for x, y in thisdict.items():  
  print(x, y)

How to change the value of Dictionary ?

Adding single element:

Dict1["Anusha"] = 22

Adding multiple elements in Dictionary:

Dict1({['abc':30,'xyz':67]}) = 22

Different ways to add a element using Dictionary?

How to delete the elements from a Dictionary ?

Deleting the single element -> del Dict1["Anusha"].

Deleting multiple or sub list of Keys:

Deleting complete list :

Pop(“key3”) –> deleting the particular item.

PopItem –> deleting the last item.

Del Dict1[0] 🡪 deleting the first element.

del Dict1. 🡪 delete the complete dictionary

clear -> clear the dictionary.

Copy the Dictionary:

Dict1 = Dict1.Copy()

Copy the Dictionary ?

1. Set: Unordered and unindexed

Different ways to create a set:

S = set{4,5} 🡪 using set constructor.

Adding the elements can be done using **Insert and Update methods.**

We can add single elements using add method and using update method we can add multiple elements to set.

Pop: it will delete the element , but we don’t know which element it will delete.

Remove() 🡪 remove the element if the element does not exist it throw the error.

Discard() 🡪 remove the element if the element does not exist it does not throw error.

Set Constructor :

1. Tuple: Ordered and unchanged

How to declare a tuple ?

How to access elements using tuple ?

How to change the elements in a list ?

if you want to change the i.e add or modify the values in list please convert the tuple to list and add the elements to it.

1. What is Lambda expression ? What is the use of it ?

Can take multiple arguments and it has one expression , x = lambda a, b: a \* b print(x(5, 6))

1. File handling in Python

F = open(‘sample.txt’,r) -- read

f.write(“hello”,’w’) – write

f.write(“hello”,a) -- append

f.close() – close,

27. datetime

Datetime.datetime.now()

Strfformat

%a %A .., %b %B

28.PIP ..python package manager

29.Import re .. is nothing but regular expression.

30.f = open("demofile.txt", "r")  
print(f.readline())  
f.close()

32. The try block lets you test a block of code for errors.

The except block lets you handle the error.

The finally block lets you execute code, regardless of the result of the try- and except blocks.

33. import random  
  
print(random.randrange(1,10))

34. How to raise an exception ?

x = -1  
  
if x < 0:  
  raise Exception("Sorry, no numbers below zero")

**33. import** pyodbc  
**import** pandas **as** pd  
**import** numpy **as** np  
  
conn = pyodbc.connect(**'Driver={SQL Server};'**,**'Server=CANATHSTDEVDB12\SQL5;'**,**'Database=MDM\_QA;'**)

currentTimestamp = datetime.datetime.strftime(datetime.datetime.now(),**"%Y%m%d%H%M%S"**)

print(currentTimestamp)

34.

con = ibm\_db.connect(**"DATABASE=bigsql;HOSTNAME=CANATBIMQA004;PORT=32051;PROTOCOL=TCPIP;UID=CA\_SC\_EDH\_HDPQA;PWD=YD0uW@nt2Bother;"**, **""**, **""**)

conn = ibm\_db\_dbi.Connection(con)

41. BIRTH\_DTupdate = pandas.io.sql.execute(BIRTH\_DTQuery,conn)  
BacthIDupdateQuery = **"update  siebel.s\_contact set batch\_id = "** + currentTimestamp+ **"  where 1=1"** *# Updating all records  
#BacthIDupdateQuery = "update  siebel.s\_contact set batch\_id = " + currentTimestamp+ "  where row\_id = " +"'" +scontactrowid + "'"*BacthIDupdate= pandas.io.sql.execute(BacthIDupdateQuery,conn)  
sql1 = **"select row\_id, NICK\_NAME,batch\_id from siebel.s\_contact where  row\_id   = "** +**"'"** +scontactrowid + **"'"**SelectQuery = pandas.io.sql.read\_sql(sql1,conn)

35. How to declare a variables ?

* variable can not start with special characarters other than @
* variable can not start with numbers..

36. Data type:

We have 4 types of data

Integer

Float

Boolean

String.

We are not declaring types in Python code.. Python implemently consider this..

To know the its data type we should provide its type as

int a

type(a)

37. What are functions in Python ?

Functions are nothing but reusable piece of code.

Created for solving specfic pattern.

38. Problem with existing Data Types -- int , float , bool , str

Problems with Existing Data Types

1. Data in a variable can be stored in a single format i.e either integrsers , decimals

2. Large memory overhead by creating large number of variables

3.unfit storing large structured data.

Two DS for sloving these problem

Lists {1,Python,2}

Dictionary {'Ramesh':150},'Suresh':10 }

39. What are Operators ?

Three types of operators

Arthimetic operator:: + , - ,% ,/

Comparsion Operator :: > ,<, >= ,<= , == , =!

Logical Operator :: and , or , not

difference between / , //

which opertaor gives the remainder.. ?

40.

import random  
  
print(random.randrange(1,10))

41. targetjobexcel= pd.read\_excel(**"targetjob.xlsx"**)  
targetjob = pd.DataFrame(targetjobexcel,columns=[**'mdmID'**,**'AccountID'**,**'AccountName'**,**'JobTitle'**])  
print(targetjob.shape) *#17868*print(targetjob.columns)  
  
MissingJobContacts = pd.merge(left=sourcejob,right=targetjob,how=**"left"**,left\_on=**"mdmID"**,right\_on=**"mdmID"**)  
MissingJobContacts.to\_excel(**"Res.xlsx"**)  
print(MissingJobContacts.shape) *#18207*MissingJobContacts[**'JobTitle?'**] = np.where(MissingJobContacts[**'Jobtitle\_ May31'**] == MissingJobContacts[**'JobTitle'**], **'True'**, **'False'**)  
MissingJobContacts[**'Jobtitle\_ May31\_len'**] = MissingJobContacts[**'Jobtitle\_ May31'**].astype(str).map(len)  
MissingJobContacts[**'JobTitle\_len'**] = MissingJobContacts[**'JobTitle'**].astype(str).map(len)  
MissingJobContacts[**'Difference'**] = MissingJobContacts[**'Jobtitle\_ May31\_len'**] - MissingJobContacts[**'JobTitle\_len'**]  
MissingJobContacts.to\_excel(**"Res.xlsx"**)

42. Apply colors

**import** StyleFrame  
**import** pandas **as** pd  
**import** jinja2  
**import** numpy **as** np

df[**'Countdifference'**] = df[**'SourceCount'**] == df[**'TargetCount'**]  
*#df['difference'] = np.where(df['SourceCount'] == df['Target count'],0,df['SourceCount'] - df['Target count'])*df.to\_excel(**'styled.xlsx'**)  
  
styled = (df.style  
 .applymap(**lambda** v: **'background-color: %s'** % **'red' if** v == **False else ''**))  
styled.to\_excel(**'styled.xlsx'**, engine=**'openpyxl'**)

43. df1 = pd.DataFrame(dfexcel1)  
l1 = df1[**'SourceCount'**].values.tolist()  
l2 = df1[**'TargetCount'**].values.tolist()  
**for** i **in** range(0,10):  
 **with** self.subTest(i=i):  
 self.assertEqual(l1[i],l2[i], **"test\_function\_one."**)

43. **if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 testReport = run\_all\_test\_generate\_xml\_report()  
 print(**"Test Report Name"** + testReport)

**class** Execution(unittest.TestCase):

44.

GroupJobExcel = pd.read\_excel(**"JobTitlesResults.xlsx"**)  
GroupJob = pd.DataFrame(GroupJobExcel)  
GroupJob = GroupJob[GroupJob[**'ReasonNumber'**]==1] -- Filter  
mdmids = GroupJob[**'mdmID'**].values.tolist()  
listofIDs = str(mdmids)  
listofIDs = listofIDs.replace(**'['**,**' '**)  
listofIDs = listofIDs.replace(**']'**,**' '**)  
print(listofIDs)

45.

df.append(df2, ignore\_index=**True**,sort=**False**)

46. @unittest.skip(**"This is a skipped test."**)  
**def** test\_skip(self):  
 *""" This test should be skipped. """* **pass**

47.

path = **r"Left.xlsx"**x3 = np.random.randn(100, 2)  
df3 = pd.DataFrame(x3)  
  
x4 = np.random.randn(100, 2)  
df4 = pd.DataFrame(x4)  
  
writer = pd.ExcelWriter(path, engine = **'xlsxwriter'**)  
df3.to\_excel(writer, sheet\_name = **'x3'**)  
df4.to\_excel(writer, sheet\_name = **'x4'**)  
writer.save()  
writer.close()

48.

df1[**'pricesMatch?'**] = np.where(df1.Price1 == df2.Price2, **'True'**, **'False'**) *#create new column in df1 to check if prices match*df1[**'priceDiff?'**] = np.where(df1.Price1 == df2.Price2, **'0'**, df1.Price1 - df2.Price2) *#create new column in df1 for price diff*print (df1)

49.

suite = unittest.TestSuite()  
suite.addTest(MyTestExample(**'test\_function\_one'**))  
suite.addTest(MyTestExample(**'test\_function\_two'**))

50.

**import** unittest  
**from** selenium **import** webdriver  
  
**class** SearchText(unittest.TestCase):  
 **def** setUp(self):  
 *# create a new Firefox session* self.driver = webdriver.Firefox()  
 self.driver.implicitly\_wait(30)  
 self.driver.maximize\_window()  
 *# navigate to the application home page* self.driver.get(**"http://www.google.com/"**)

51.

print(**"Single Filter"**)  
dffilter = df1[df1.Cars == **'Nissan'**]  
print(dffilter)  
  
print(**"Multiple Filters"**)  
dffilter1 = df1.loc[(df1.Cars == **'Hero Honda'**) & (df1.Price == 100)]  
print(dffilter1)

52.

SELECT

REGEXP\_SUBSTR('500 Oracle Parkway, Redwood Shores, CA',

',[^,]+,') "REGEXPR\_SUBSTR"

FROM DUAL;

REGEXPR\_SUBSTR

-----------------

, Redwood Shores,

53.

**from** selenium **import** webdriver  
**import** selenium  
**from** sphinx.testing.path **import** path  
**from** webdriver\_manager.chrome **import** ChromeDriverManager  
  
*#webdriver.Chrome()  
#driver = webdriver.Chrome(ChromeDriverManager().install(),path="C:/Users/kbharathi/Desktop/MDM/Cards/")  
#driver = webdriver.Chrome("C:/Program Files (x86)/Google/Chrome/Application/chrome.exe")*driver = webdriver.Chrome(**"C:/Users/kbharathi/Desktop/MDM/Cards/chromedriver/78.0.3904.70/win32/chromedriver.exe"**)  
*#driver.get("www.google.com")*driver.get(**"https://my.dnet.deloitte.com/"**)  
driver.implicitly\_wait(20)  
driver.find\_element\_by\_xpath(**'//\*[@id="userNameInput"]'**).send\_keys(**"kbharathi@deloitte.com"**)  
driver.implicitly\_wait(20)  
driver.find\_element\_by\_xpath(**'//\*[@id="passwordInput"]'**).send\_keys(**"Contestant@24"**)  
driver.find\_element\_by\_xpath(**'//\*[@id="isLaptop"]'**).is\_selected()  
driver.implicitly\_wait(20)

54. print(**'Values in ACT only: '**)  
**for** val\_a **in** act\_vals:  
 **if** (val\_a **not in** sat\_vals):  
 print(val\_a)

55. **def** fix\_participation(column):  
 **return** column.apply(**lambda** cells: cells.strip(**'%'**))

56. **import** pandas **as** pd  
  
  
TestCases = {  
 **'Req'**: [**'1'**],  
 **'Test Case#'**: [**'TC001'**],  
 **'Description'**: [**'Verify whether Adhoc merge has happened as expected or not'**],  
 **'Step Description'**:[**''**],  
 **'Expected Result'**: [**'Job should run scussessfully with fail.'**],  
 **'Actual Result'**: [**'Pass'**],  
 **'Pass/Fail'**: [**'Pass'**],  
 **'Comments'**: [**'All is well'**],  
 **'Created By'**: [**'priya'**],  
 **'Executed By'**: [**'priya'**],  
 **'Created Date'**: [**'priya'**]  
 }  
  
df = pd.DataFrame(TestCases)  
  
*#df = pd.DataFrame({'Ticket': ['a','b','c','d'],  
# 'Category': [2,1,4,3]})*

*H/15 \* \* \* \**

writer = pd.ExcelWriter('Trial Version.xlsx', engine='xlsxwriter')  
df.to\_excel(writer, sheet\_name=**'Sheet1'**, index=**False**)  
workbook=writer.book  
worksheet = writer.sheets[**'Sheet1'**]  
  
format = workbook.add\_format({**'text\_wrap'**: **True**})  
*# Setting the format but not setting the column width.*worksheet.set\_column(**'C:C'**, 15, format)  
worksheet.set\_column(**'D:D'**, 15, format)  
worksheet.set\_column(**'E:E'**, 15, format)  
  
  
  
*# Add a header format.*header\_format = workbook.add\_format({  
 **'bold'**: **True**,  
 **'text\_wrap'**: **True**,  
 **'valign'**: **'top'**,  
 **'fg\_color'**: **'#D7E4BC'**,  
 **'border'**: 1,  
}  
)  
  
*# Write the column headers with the defined format.***for** col\_num, value **in** enumerate(df.columns.values):  
 worksheet.write(0, col\_num + 1, value, header\_format)  
  
*# Close the Pandas Excel writer and output the Excel file.*writer.save()

57.

cursor = conn.cursor()  
cursor = cursor.execute(**"update  admin.process\_table set process\_flag = "**+**"'"** +processflag + **"'"**+ **"where source ="** +**"'"**+ sourcecode+**"'"**)  
cursor = cursor.execute(**"update  admin.process\_stage set status = "**+**"'"** +processflag + **"'"**+**" where 1=1"**)  
conn.commit()  
  
sqlQuery = pd.read\_sql\_query(**"select source,processid from admin.process\_table where 1=1"**,conn)  
df = pd.DataFrame(sqlQuery)  
print(df)

58.

**for** i,j **in** ColumnNames.iteritems():

    SourceColumns = ColumnNames[i]

    Targetdata = ColumnNames[**"Targetdata"**] + **' '** + ColumnNames[**"TargetColumnnames"**]

59.

currentTimestamp = datetime.datetime.strftime(datetime.datetime.now(),**"%Y%m%d%H%M%S"**)

print(currentTimestamp + **"InsertintoEloqua"**)

print(df)

60.data = pd.DataFrame.drop\_duplicates(df)

61.

MaxProcesss = conn.cursor()

MaxProcesssID = MaxProcesss.execute(**'select max(processid) from admin.process\_table'**)

**for** MaxProcesssID **in** MaxProcesss:

    print(MaxProcesssID[0])

sqlquery1 = pd.read\_sql\_query(**"select "** + SQLContactQuery + **" from crm.S\_CONTACT WHERE processid = '"**+secondmaxprocessid + **"'"**,conn)

HistoryRecord= pd.DataFrame(sqlquery1)

HistoryRecord = HistoryRecord.reset\_index(drop=**True**)

print(HistoryRecord.shape)

HistoryRecord = HistoryRecord.fillna(**"null"**)

LatestRecord = LatestRecord.fillna(**"null"**)

62. df3 = pd.DataFrame()

**for** i,j **in** LatestRecord.iteritems():

*#print('Colunm Name : ', LatestRecord[i])*

df3[**"Latest\_"**+ i] = LatestRecord[i]

   df3[**"History\_"** + i] = HistoryRecord[i]

   df3[i+**"\_Change"** ] = np.where((LatestRecord[i]) == HistoryRecord[i], **'True'**, **'False'**)

print(df3)

63. export\_excel = df3.to\_excel(**'export\_dataframe.xlsx'**, index = **None**, header=**True**)

**for** index,row **in** grMatchLinksData.iterrows():

  print(sqlQueryPart1,coulunnames)

print(index,row)

64.

MatchContactMainOld = MatchContactMainOld[MatchContactMainOld[**'MDMID'**]==**'INS-2851258130980'**]

MatchContactMainOld.reset\_index(inplace = **True**)

65. df.sort\_values(by=[‘Brand’],inplace=True)

66. How to read the data from various files in Python ?

Read\_csv

read\_excel

read\_sql.

67. Display the list of columns in Python ?

df.columns

68.

Types -- df.Types

69. How to display the single column in Python ?

print(carData["sex"])

70. How to display the multiple columns in Python ?

print(carData[["sex","population"]])

-- We must use the double square brackets.

69. encoding="utf-8" , While reading excel or CSV it is always preferable to have the encoding option.

encoding="latin-1")

carData = pd.DataFrame(carDa

70. What is Python Pandas ?

It is a data manipulation and data analysis Library.

71. What is Python Data Frames..

It is 2 dimenionsal hetrogenous array .. which has rows and columns.

and It is mutuable.

72. Shape and ndim .., 2 dimenional array , we should print 2 dimeionsal graph.

73. head and Tail -- First and Last 5 rows..

74.os.getcwd -- Get current working directory

75.desired\_width=320

pd.set\_option('display.width', desired\_width)

pd.set\_option('display.max\_columns',10)

pd.display.options.format.float\_Format = ${0.02f}

76.Lambda Expression --- lambda x: x \* 2 .., double(x) .. Lambda (Anonyms function).

77. What is join Operator in Python ?

ListofColumnNames = ','.join(ListofColumnNames)

78. What is Python spilt, join , find, tolower ,toupper methods ?

Spilt ---

str -- "I am learning Python" ','.spilt , Output: -- "I","am","Learning","Python".

Join.. ','.join(str) Output:

79. What is Python contains.. how to check whether string exists or not ?

for "processid" in columnnames[i]:

Column exits ..print

80. cursor = con.conn.cursor()

for index,row in grMatchLinksData.iterrows():

     print(index,row)

Display 1st 1 column

till the number of rows..

Itertaing to display all the rows in Pandas Data Frame.

81. How to install Python ?

There are two steps to install it.

(i).Install Python 3.7

(ii). Install PyCharm or Jupiter Notebook.

82.Data Types in Python ?

83. declaring Variables in Python. ?

84.Responsbility of Data Analytics.

(i). Load the Raw data into excel.

(ii). Do manipulation and analysis..

(iii). Data Visulaixzation

(iv). Finding Pattrnes

85. What is Python ?

It is Open source.

It has large set of libaratiries to support Data Manipulations and Analysis.

Syntax as simple as English.

86.Operators

Three types of operators

Arthimetic operator:: + , - ,% ,/

Comparsion Operator :: > ,<, >= ,<= , == , =!

Logical Operator :: and , or , not

87. difference between / , //

88.which opertaor gives the remainder.. ?

89.How to give Input ?

90. One Dimensional -- series , Two dimensional --Pandas

91.apply

93. Dictionary -- Key Value pairs.. d = {"StudentName":['Priya','Ash'] "Age":['45','40']}

94. How to convert DS to Pandas DF ?

df = pd.dataframe(l1) --List

df = pd.dataframe(d1) --dictionary

df = pd.dataframe(series) --dictionary

95.

Data type

We have 4 types of data

Integrer

float

boolean

String.

96.Convert

astype(float)

97.a="10" .. type(a) -- To know the type of the varible.

98.

Variables

a=10

b=5

c=a

c=4

different locations..

we need to deal with Constants plus variables..

In computer programming, a variable or scalar is a storage location paired with an associated symbolic name,

which contains some known or unknown quantity of information referred to as a value.

99.

-- Select Query

Step1: import pyoodbc as py

Step2: pyodbc.connect

s3: sqlResult = pd.io.sql.read\_sql("sqlquery1",con)

s4: pd.DataFrame(sqlResult)

-- Insert\update\delete

-- Insert data into SQL row by Row

sqlquery: insert into TN(col1,col2) values(?,?)

con.cursor() #crosscheck..

for row,index in df.Iterrows():

con.execute(row["column1"],..);

con.close()

99.slice a string

str = "abcd,"

str = str[0:len(str)-1]

100. len(l1) length of the list 4

101. .. range(len(l1).., display as a (0,4). It will be used in for loop.

102. How to loop elements in a dictionary?

dict1 = {"StdudentName":['Priya','Ashok'],

"Age":[2,3]}

for key,value in dict1.items():

print(key,value)

103.relation operators

104.logical oper: and or , !

105.

Python ..

It is a enviorment for statistical analysis

data analysis software

programming language

open source

huge online community.

106.frequency distributions..

Number of times we have automated cars

Number of times we have gear cars

107. R is a staisitical tool for analysis software. We can integrate R with tableau.

108. Conditional and for loops.

109. How to rename a column?

carData.rename(columns={'Area Abbreviation':'Area1234','Area Code':'AreaCode'},inplace=True)

print(carData.head())

110. How to remove a column in a dataframe ?

del carData['Y1961']

111. How to change the column from string to float.

112.Remove all the punctuations and letters from the ISBN code and make sure its data type is still a string.

display the data ..

113. Visualizations

finding patterns

decesion makes faster

Green color -- perform well.

114. Identify the relationship between the variables.

115. What is DV

Reperesentation of data in Vsialaus

116. How to acheive DV ?

Integrate Different DataSets

Analyze

Visualize

117.Why BI ?

Transforms the Business data into Information or Knowledge

118. How to remove duplicates in DF

119. How to enter key from Keyboard ?

120.

Both inplace= true and inplace = False are used to do some operation on the data but: When inplace = True is used, it performs operation on data and nothing is returned. When inplace=False is used, it performs operation on data and returns a new copy of data

121.mean -- avg

median -- middle value

mode -- the most frequently repeated value.

range -- largest value - smallest value.

devitaion means.. How far from the normal

standard deviation means..

How spread out numbers are.

(or)

Square root of the variance.

VARIANCE..

Claucauate Mean.

Then for each number: subtract the Mean and square the result

Then work out the average of those squared differences.

Square root of Varinace is called SD.

https://www.guru99.com/data-mining-tutorial.html

Descripotive Statictics..

Describe the charactseristics of data.

measures of Central tendency

mean -- avg

mode -- most frequently repeated number

median --

representation of the entire or a sample of a

measures of variability (spread).

how the data is distributed within the set

variance.. Spread ., how the data has spread across the data set.

standard deviation

min

max

Descriptive statistics summarizes or describes characteristics of a data set.

Descriptive statistics consists of two basic categories of measures: measures of central tendency and measures of variability or spread.

Measures of central tendency describe the center of a data set.

Measures of variability or spread describe the dispersion of data within the set.

Statistics is a form of mathematical analysis that uses quantified models, representations and synopses for a given set of experimental data or real-life studies

Statistics studies methodologies to gather, review, analyze, and draw conclusions from data.

There are many different types of statistics pertaining to which situation you need to analyze.

Statistics are used to make better-informed business decisions.

Types of Statistics

including the arithmetic mean, which shows how well a specific commodity performs over time

Variance

Variance is a measurement of the span of numbers in a data set. The variance measures the distance each number in the set is from the mean.

probability : the extent to which an event is likely to occur, measured by the ratio of the favourable cases to the whole number of cases possible.

the quality or state of being probable; the extent to which something is likely to happen or be the case.

"the rain will make the probability of a postponement even greater"

Standard deviation is a number used to tell how measurements for a group are spread out from the average (mean), or expected value. A low standard deviation means that most of the numbers are close to the average. A high standard deviation means that the numbers are more spread out

https://www.shanelynn.ie/using-pandas-dataframe-creating-editing-viewing-data-in-python/

https://www.geeksforgeeks.org/how-to-select-multiple-columns-in-a-pandas-dataframe/

https://www.geeksforgeeks.org/reset-index-in-pandas-dataframe/

Categorical variable

hat is nominal variables

https://towardsdatascience.com/how-to-filter-rows-of-a-pandas-dataframe-by-column-value-51996ea621f8

https://cmdlinetips.com/2018/02/how-to-subset-pandas-dataframe-based-on-values-of-a-column/

We genearelly analyze sample or subset of data.

declare @MonthsSale table

(

MonthNumber int,

MonthName varchar(9),

MonthSale decimal(18,2)

)

insert into @MonthsSale

select

1, 'January', 100.00

union select

2, 'February', 200.00

union select

3, 'March', 300.00

union select

4, 'April', 400.00

union select

5, 'May', 500.00

union select

6, 'June', 600.00

union select

7, 'July', 700.00

union select

8, 'August', 800.00

union select

9, 'September', 900.00

union select

10, 'October', 1000.00

union select

11, 'November', 1100.00

union select

12, 'December', 1200.00

select \* from @MonthsSale

select SUM(MonthSale) as [TotalSales] from @MonthsSale

IBM Data Replication portfolio provides log based change data capture with transactional integrity to support big data integration and consolidation, warehousing and analytics initiatives at scale

120. What is assertGreaterEqual(a, b) : a >= b?

It is a>=b.

2. reset\_index

3. panadas.io.readsql -- reading the SQL query from SQL Server

and place the select query in Pandas

and performa all the required data manipulation ioerartions

4.pandas.io.execute -- it executes particulr update,delete statements in SQL.

5. handling multiline quiereis

str1 = ("select \n"

" abc ")

6.

When we look at the smaller dataframe, it might still carry the row index of the original dataframe.

If the original index are numbers, now we have indexes that are not continuous. Well, pandas has reset\_index() function.

So to reset the index to the default integer index beginning at 0, We can simply use the reset\_index() function.

Pandas Library

The huge libraries which are available in Pandas

provides high performance and effectively perform the data analysis.

The Data structures which are provided by Python Easy to use

Key Features of Pandas

• We can load the data into Data Frame from different File formats.

• We can perform various operations like

• Grouping

• Filtering

• Joining

• Merging

• Pivot

• Handling the missing data

How to install the Pandas

Pip install pandas

Pandas provide three kind of DS which are built on top of Numpy which means they are fast and efficient.

• Series

• Data Frames

• Panel

Series: One dimensional array , immutable , Homogenous .

Data Frames - Two dimensional array , Heterogenous , mutable.

Panel -- Three dimensional array , Heterogenous, mutable.

1. How to create an empty Series , Data Frame, Panel ?

2. How to get the data type of column ?

3. How to get the rows and columns of a Data Frame ?

4. How to get the Top most records in Data Frame ?

5. How to get the last few records from the table ?

6. In what ways can we create the series ?

Series

ND Array-- With Index , Without Index

Array -- With Index , Without Index

Dict -- With Index , Without Index

Scalar -- Index is must

7. In what ways can we create the DataFrame ?

List -- With Index , Without Index

Dict -- With Index , Without Index

Series

Numpy Array

Another Data Frames

8. In what ways can we create the Panel ?

From ndarrays

From dict of DataFrames

9. How to access the first 3 elements in Series/DF/Panel ?

Series -- First 3 elements

DF -- First 3 rows

First 3 columns , First 3 rows

10. How to access the last 3 elements in Series/DF/Panel ?

Series -- last 3 elements

DF -- last 3 rows

last 3 columns , last 3 rows

11. How to access the elements from 2, 5 in Series/DF/Panel ?

Series -- elements from 2,5

DF -- elements from 2,5

elements from 2,5 , columns 2to 5

12. How to access the first element ?

Series -- 1 st

DF -- 1st column

13. How to access the last element ?

Series -- 2 st

DF -- last column

14. How to access the required elements in DF \Series ?

Series -- 2 st

DF -- last column

15. What are the basic operations that can be performed on the series ?

axes ,

dtype ,

empty ,

ndim ,

size ,

values ,

head ,

tail.

16. What are the basic operations that can be performed on the series ?

T ,

axes ,

dtypes ,

empty ,

ndim ,

head ,

tail ,

size ,

shape ,

values.

17. Descibe ?

Sum , Max , min , mean , std ,variance ,

1 count() Number of non-null observations

2 sum() Sum of values

3 mean() Mean of Values

4 median() Median of Values

5 mode() Mode of values

6 std() Standard Deviation of the Values

7 min() Minimum Value

8 max() Maximum Value

9 abs() Absolute Value

10 prod() Product of Values

11 cumsum() Cumulative Sum

12 cumprod() Cumulative Product

18.

Table wise Function Application: pipe()

Row or Column Wise Function Application: apply()

Element wise Function Application: applymap()

19.

Reindexing changes the row labels and column labels of a DataFrame. To reindex means to conform the data to match a given set of labels along a particular axis.

Multiple operations can be accomplished through indexing like -

Reorder the existing data to match a new set of labels.

Insert missing value (NA) markers in label locations where no data for the label existed.

20.

Python Pandas - Iteration

21. Sorting

22. Built in functions.

Sr.No Function & Description

1

lower()

Converts strings in the Series/Index to lower case.

2

upper()

Converts strings in the Series/Index to upper case.

3

len()

Computes String length().

4

strip()

Helps strip whitespace(including newline) from each string in the Series/index from both the sides.

5

split(' ')

Splits each string with the given pattern.

6

cat(sep=' ')

Concatenates the series/index elements with given separator.

7

get\_dummies()

Returns the DataFrame with One-Hot Encoded values.

8

contains(pattern)

Returns a Boolean value True for each element if the substring contains in the element, else False.

9

replace(a,b)

Replaces the value a with the value b.

10

repeat(value)

Repeats each element with specified number of times.

11

count(pattern)

Returns count of appearance of pattern in each element.

12

startswith(pattern)

Returns true if the element in the Series/Index starts with the pattern.

13

endswith(pattern)

Returns true if the element in the Series/Index ends with the pattern.

14

find(pattern)

Returns the first position of the first occurrence of the pattern.

15

findall(pattern)

Returns a list of all occurrence of the pattern.

16

swapcase

Swaps the case lower/upper.

17

islower()

Checks whether all characters in each string in the Series/Index in lower case or not. Returns Boolean

18

isupper()

Checks whether all characters in each string in the Series/Index in upper case or not. Returns Boolean.

19

isnumeric()

Checks whether all characters in each string in the Series/Index are numeric. Returns Boolean

23.

Pandas provide API to customize some aspects of its behavior, display is being mostly used.

The API is composed of five relevant functions. They are -

get\_option()

set\_option()

reset\_option()

describe\_option()

option\_context()

Let us now understand how the functions operate.

24.Python Pandas - Indexing and Selecting Data

loc , iloc ,

25.Python Pandas - Statistical Functions

26.Python Pandas - Window Functions

27.aggregations, group by

28. merging

29. Joins

30. missing data.

31. Concenation

32. timedelta

33.Python Pandas - Categorical Data

34. Date functionaility

35. Catgoryical data

36.Python Pandas - IO Tools

37. Python Pandas - Visualization

38. Python Pandas - Sparse data

39.Python Pandas - Caveats & Gotchas

40. Comparison with SQLs.

* 1. <https://pandas.pydata.org/pandas-docs/stable/user_guide/style.html>
  2. Df.style.apply – row or column or table level
  3. Df.style.applymap – element wise data.

Apply or Applymap takes argument as a function.

l1 = {1,2.0,**"priya"**}  
print(l1)

**NumPy** is a **Python** package which stands for 'Numerical **Python**'. It is the core **library** for scientific computing, which contains a powerful n-dimensional array object,

str = (df.apply(**lambda** row: **"'"**+str(row[**"Name"**]) + **"','"** + str(row[**"Percentage"**])+**"'"**, axis = 1))

df[“Name”].map(str) +” , ” +df[“age”].map(str)

url =**"[h\_ttps://deloittecanada.ca1.qualtrics.com/API/v3/surveys/](_https:/deloittecanada.ca1.qualtrics.com/API/v3/surveys/%20)"**payload = {}  
headers = {**'X-API-TOKEN'**: Token}  
response = requests.get(url, headers=headers, data = payload)  
Parsed\_data = json.loads(response.content)  
Parsed\_data1 = json.loads(response.text)  
*print(Parsed\_data1)*

1. SELECT ID,

abc = STUFF(

(SELECT ',' + name FROM temp1 FOR XML PATH ('')), 1, 1, ''

)

FROM temp1 GROUP BY id

2. <https://towardsdatascience.com/exploratory-data-analysis-tutorial-in-python-15602b417445>

## 3. 10.1  Tuples are immutable

4. df.iloc[:, : 50]

5. Data Frames are nothing but Rows and Columns it is like table. We can perform select ,delete ,update, insert methods in it.

Select singlecolumn , multiple columns.

Delete Dataframe.drop(‘col1,col2’,inplace = TRUE)

Add columns

# making data frame from csv file

data = pd.read\_csv("nba.csv", index\_col ="Name" , names = ‘’,’’,’’)

# dropping passed columns

data.drop(["Team", "Weight"], axis = 1, inplace = True)

df.sort\_values(by=['Brand'], inplace=True)

Row Selection:

Pandas provide a unique method to retrieve rows from a Data frame.[DataFrame.loc[]](https://www.geeksforgeeks.org/python-pandas-extracting-rows-using-loc/) method is used to retrieve rows from Pandas DataFrame

df[df['row\_id'].isin(l1)]

new\_row = pd.DataFrame({'Name':'Geeks', 'Team':'Boston', 'Number':3,

                        'Position':'PG', 'Age':33, 'Height':'6-2',

                        'Weight':189, 'College':'MIT', 'Salary':99999},

                                                            index =[0])

# simply concatenate both dataframes

df = pd.concat([new\_row, df]).reset\_index(drop = True)

# dropping passed values

data.drop(["Avery Bradley", "John Holland", "R.J. Hunter",

                            "R.J. Hunter"], inplace = True)

**for**i,j **in**SourceComparisonData.iteritems():  
    print(SourceComparisonData[i])  
    print(TargetComparisonData[i])  
    df4[**"df1"**+ i] = SourceComparisonData[i]  
    df4[**"df2"**+ i] = TargetComparisonData[i]  
    df4[i + **"Result"**] =  SourceComparisonData[i] == TargetComparisonData[i]  
   *# df4[i+"Result" ] = np.where((SourceComparisonData[i]) == TargetComparisonData[i], 'True', 'False')*df4.to\_excel(**"Res1.xlsx"**)

df[df['row\_id'].isin(l1)]

# **Getting frequency counts of a columns in Pandas DataFrame**

Value\_counts,GroupBy

# Check if the updated price is available or not

if 'Updated Price' in df.columns:

    df['Final cost'] = df['Updated Price'] - (df['Updated Price']\*0.

# Check if the updated price is available or not

if {'Updated Price', 'Discount'}.issubset(df.columns):

# **Get unique values from a column in Pandas DataFrame**

# Get number of unique values in column 'C'

df.C.nunique(dropna = True)

# **How to lowercase column names in Pandas dataframe**

# df['A'] = df['A'].str.lower()

df.columns = df.columns.str.lower()

# **Apply uppercase to a column in Pandas dataframe**

One can use apply() function in order to apply function to every row in given dataframe.

 df['add'] = df.apply(np.sum, axis = 1)

Lower or Upper case applying to a column:

df.columns = df.columns.str.lower()

Lower or Upper case applying to a column Name:

**import** pandas **as** pd  
**import** numpy **as** np  
  
sourcejobexcel= pd.read\_csv(**'currentprocessID.csv'**,low\_memory = **'False'**,dtype=**'unicode'**)  
sourcejob  = pd.DataFrame(sourcejobexcel)  
sourcejob = sourcejob.iloc[:80, : 30]  
sourcejob.fillna(0,inplace=**True**)  
*#sourcejob = sourcejob.iloc(:0,50)*print(sourcejob.shape) *#15237  
#print(sourcejob.columns)*SourceRowIDs = sourcejob[**'ROW\_ID'**].values.tolist()  
SourceColumns= sourcejob.columns.values.tolist()  
  
  
targetjobexcel= pd.read\_csv(**'previousprocessID.csv'**,low\_memory = **'False'**,dtype=**'unicode'**)  
targetjob  = pd.DataFrame(targetjobexcel)  
targetjob = targetjob.iloc[:80, : 30]  
targetjob.fillna(0,inplace=**True**)  
print(targetjob.shape)  *#17868  
#print(targetjob.columns)*TargetRowIDs = targetjob[**'ROW\_ID'**].values.tolist()  
  
*#CommonContacts = pd.concat(sourcejob,targetjob)  
#CommonContacts = pd.merge(left=sourcejob,right=targetjob,how="inner",left\_on="ROW\_ID",right\_on="ROW\_ID")  
#CommonContacts.to\_excel("Res.xlsx")  
#print(CommonContacts.shape) #18207*i = 0  
l1 = []  
*#print(SourceRowIDs)  
#print(TargetRowIDs)*df3 = pd.DataFrame()  
**for** val\_a **in** SourceRowIDs:  
    **if** (val\_a  **in** TargetRowIDs):  
        i = i + 1  
       l1.append(val\_a)  
  
  
  
print(**"Number of contacts which exists in TargetRowIDs"**+ str(i))  
  
  
commonSource = pd.DataFrame()  
commonTarget = pd.DataFrame()  
SourceComparisonData = sourcejob[sourcejob[**'ROW\_ID'**].isin(l1)]  
*#SourceComparisonData = SourceComparisonData.columns.str.lower()*SourceComparisonData = SourceComparisonData.reset\_index(drop=**True**)  
SourceComparisonData = SourceComparisonData.sort\_values(by=[**'ROW\_ID'**],ascending=**True**)  
SourceComparisonData = SourceComparisonData.reset\_index(drop=**True**)  
print(SourceComparisonData)  
SourceComparisonData.to\_excel(**"Res7.xlsx"**)  
  
TargetComparisonData = targetjob[targetjob[**'ROW\_ID'**].isin(l1)]  
TargetComparisonData = TargetComparisonData.reset\_index(drop=**True**)  
*# TargetComparisonData.sort\_values(by=['ROW\_ID'],ascending=True, inplace=True)*TargetComparisonData = TargetComparisonData.sort\_values(by=[**'ROW\_ID'**],ascending=**True**)  
  
  
print(TargetComparisonData)  
TargetComparisonData.to\_excel(**"Res8.xlsx"**)  
  
*#df3 = pd.merge(SourceComparisonData,TargetComparisonData, on=['ROW\_ID'],how='inner')  
#df3['df3diff'] = df3['FST\_NAME'] - df3['FST\_NAME']  
#print(df3.shape)*df4 = pd.DataFrame()  
  
i = 1  
**for** i,j **in** SourceComparisonData.iteritems():  
   *# print(SourceComparisonData['ROW\_ID'],TargetComparisonData['ROW\_ID'])  
    #print(TargetComparisonData[i])* df4[**"df1"**+ i] = SourceComparisonData[i]  
    df4[**"df2"** + i] = TargetComparisonData[i]  
    df4[i + **"Result"**] =  SourceComparisonData[i] == TargetComparisonData[i]  
    df4[i+**"Result"** ] = np.where((SourceComparisonData[i]) == TargetComparisonData[i], **'True'**, **'False'**)  
df4.to\_excel(**"Res1.xlsx"**)  
  
*#df.to\_excel('MDMCountResult.xlsx')  
#styled = (df.style  
            #.applymap(lambda v: 'background-color: %s' % 'Green' if v == False else ''))  
#styled.to\_excel('MDMCountResult.xlsx', sheet\_name = 'CRMData')*

Q. How to update Same excel with different sheets ?

Excel = pd.ExcelWriter(**'MDMBusinessTables.xlsx'**)

df.to\_excel(Excel, sheet\_name=**'Main'**)

df1.to\_excel(Excel, sheet\_name=**'Main\_Multiple'**)  
Excel.save()

Q. # Remove column name 'A'

df.drop(['A'], axis = 1)

Q. MaxProcesss = conn.cursor()

MaxProcesssID = MaxProcesss.execute('select max(processid) from admin.process\_table')

for MaxProcesssID in MaxProcesss:

    print(MaxProcesssID[0])

Q. **import** SQLConnection **as** con  
**import** pandas **as** pd  
**import** pandas.io.sql **as** SQL  
  
excelWriter = pd.ExcelWriter(**'MDMBusinessData.xlsx'**)  
  
TablesData = pd.read\_excel(**'MDMBackupTables.xlsx'**,sheet\_name=**'MDMBusinessTables'**)  
TablesData = pd.DataFrame(TablesData)  
ListofTableNames = TablesData[**'TableNames'**].values.tolist()  
CountofTables = len(TablesData[**'SQLQuery'**].values.tolist())  
  
  
ListofIDs = pd.read\_excel(**'MDMBackupTables.xlsx'**,sheet\_name=**'ListofIDs'**)  
ListofIDs = pd.DataFrame(ListofIDs)  
ListofIDs = ListofIDs[**'ID'**].values.tolist()  
ListofIDs = str(ListofIDs)  
ListofIDs = ListofIDs.replace(**'['**,**''**)  
ListofIDs = ListofIDs.replace(**']'**,**''**)  
  
print(CountofTables)  
*#i= 0***for** i **in** range(CountofTables):  
    print(i)  
    sql =TablesData[**'SQLQuery'**][i] + **' and grid in ('**+ListofIDs+**')'** *#sql = TablesData[i]+' and grid in ('+ListofIDs+')'* print(sql)  
    f= open(**'sql.txt'**,**'a'**)  
    f.write(str(sql))  
    dfMainExcel = SQL.read\_sql(sql,con.conn)  
    dfMain = pd.DataFrame(dfMainExcel)  
    print(dfMain)  
    dfMain.to\_excel(excelWriter, sheet\_name=ListofTableNames[i])  
  
f.close()  
excelWriter.save()

df = df[**'Percentage'**].apply(**lambda** num : num + 5)  
print(df)

df[**'neagtivevalues'**] = df[**'Percentage'**].apply(**lambda** num : num < 5)print(df)

Name Age Stream Percentage neagtivevalues

0 Ankit 21 Math 88 False

1 Amit 19 Commerce 92 False

2 Aishwarya 20 Arts -4 True

3 Priyanka 18 Biology 70 False

def color(val):

if val < datetime.now():

color = 'green'

elif val > datetime.now():

color = 'yellow'

elif val > (datetime.now() + timedelta(days=60)):

color = 'red'

return 'background-color: %s' % color.

**import** pandas **as** pd  
**def** color\_negative\_red(val):  
 color = **'red' if** val < 0 **else 'black'  
 return 'color: %s'** % color  
  
  
df = pd.DataFrame(dict(col\_1=[1.53,-2.5,3.53],  
 col\_2=[-4.1,5.9,0]))  
  
df = df.style.applymap(color\_negative\_red,subset=[**'col\_1'**,**'col\_2'**])  
print(df)  
df.to\_excel(**"example.xlsx"**)

<https://mode.com/example-gallery/python_dataframe_styling/>

PIG Latin Simple Compact 10 Java -1 PIG , PIG Compiler - Jar File - JVM background is MapReduce Job.

pig -x local .. it uses local file system. it provides grunt shell.

Pig … it will start the local mode.

It gives the grunt shell.

DAC (Directed Acyclic Graph)

Lazy way of evaluation.

describe pig\_hdfs ..provides metadata information



grp\_pig = group pig\_hdfs by c1

(2,{(2,2,2)(3,3,3)})

Bag – every tuple represents one row of HDFS.

Every Item of this tuple basically referred as a Item.

Dfs , hdfs

Dfs .. storage part on top of that MAP REDUCE Frame.. YARN .. one top that .. PIG Script ,Hive Script .. Oozie workflow.

Core-site.xml.

Hdfs\*.xml

yarn\*.xml

Scoop Import 🡪 Local to HDFS

Export 🡪 HDFS to local

Pseudo dstritubed mode

Pseudo dm:

3 modes to run the job.

Pseudo

Local..

**Change the color of cell:**

**import** pandas **as** pd  
**def** color\_negative\_red(val):  
 color = **'red' if** val < 0 **else 'black'  
 return 'color: %s'** % color  
  
  
df = pd.DataFrame(dict(col\_1=[1.53,-2.5,3.53],  
 col\_2=[-4.1,5.9,0]))  
  
df = df.style.applymap(color\_negative\_red,subset=[**'col\_1'**,**'col\_2'**])  
print(df)  
df.to\_excel(**"example.xlsx"**)

The purpose of **Web Service Testing** is to verify that all of the Application Programming Interfaces (APIs) exposed by your application operate as expected

Web Services is the mechanism or the medium of communication through which two applications / machines will exchange the data irrespective of their underline architecture and the technology

This WebService can be called by a Software Application using SOAP or HTTP protocol.

Web Services can be implemented in different ways, but the following two are the popular implementations approaches.

1. SOAP (Simple Object Access Protocol)
2. REST (Representational State Transfer architecture)

## SOAP

SOAP is a standard protocol defined by the W3C Standard for sending and receiving web service requests and responses.

SOAP uses the **XML format to send and receive the request** and hence the data is platform independent data

WebService Testing involves following steps -

1. **Understand the WSDL file**
2. **Determine the operations that particular web service provides**
3. **Determine the XML request format which we need to send**
4. **Determine the response XML format**
5. **Using a tool or writing code to send request and validate the response**

## Summary

* Software Applications communicate and exchange data with each other using a WebService
* SOAP and REST are 2 popular protocols to create a WebService
* SOAP supports XML based data exchange
* REST support XML, Json or exchange of data in simple URL.
* WSDL is XML based language which will be used to describe the services offered by a web service. SOAP is defined using WSDL.
* To test WebService you can
  + Create your own code. For instance use Axis2 API for Java
  + Use WebService Test Automation tools like SoapUI
* Automation Tools like SoapUI will jumpstart your testing efforts, will require less coding effort compared to creating your own code using Axis2 API

**PART 2) Using SoapUI to Test the WebService**

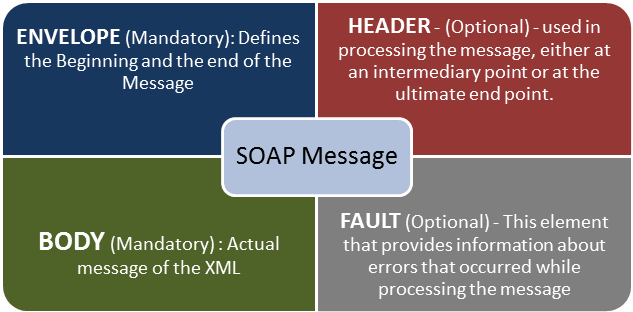
In SoapUI

1. Go to File > New Soap Project
2. Enter the project Name and the WSDL URI location
3. Click OK

## **What is SOAP UI?**

* SOAP UI is the leading open source cross-platform API[Testing](https://www.guru99.com/software-testing.html)tool
* SOAPUI allows testers to execute automated functional, regression, compliance, and load tests on different Web API.

SoapUI ca functional,Regression,Compliance and Load test your API.



Test Suite

Testcase

New Test step description.

### What Is an Assertion?

Assertion means act of affirming or stating something

1. Property Content
2. Compliance Status Standard
3. Script
4. SLA
5. JMS
6. Security

**How to find hidden files in current directory?**  
  
   $ ls -a**2) How to find current running processes in Unix server?**  
  
  $ ps -ef  
  
and if we want to find specific process we can use 'grep' with pipe  
  $ ps -ef | grep -i 'application'**3) How to find process which is taking maximum memory in server?**  
  $ top  
top command tell us about cpu usage , process id and other details.

**4) How to find Exception in log files available in current directory and how to find number of occurrence?**  
      
   $ grep 'Exception' log1.txt | wc -l**5) find all files in current and subdirectories which contains 'log' name?**  
  
   $ find . -name 'log'**6)** **How do you access command line arguments from within a shell script?**

Arguments passed from the command line to a shell script can be accessed within the shell script by using a $ (dollar sign) immediately followed with the argument's numeric position on the command line.

**7) How to tails last 200 lines of any log fine?**  
  
    $ tail -200f filename.txt

**8) How to find remaining disk space in unix/linux server?**

    $ df -kl

**9) How to make any script file executable?**

    $chmod 755 \*.sh  
 **10) How to kill process in unix server?**  
    $ kill -9 #pid  
these #pid can be found using ps -ef command.

**11) Command to run a job say job.sh in background?**

.job.sh &

**12) How to check a process named Informatica is running or not?**

Ps –ef|grep Informatica

**13) Command to force any process to stop execution?**

Kill -9 pid      (process id)

**14) Command to get the present directory you are working?**

Pwd

**15) Ipconfig equivalent in unix or command which gives all the network related details?**

Ifconfig

We need to identify tools ..

* Design validation tests ,
* set test environment ,
* run database migration process ,
* run validation tests,
* analyze results and report defects.

If the migration is to the same type of Database, then,

* Verify if the queries executed in the new database yield same results as in the older one
* Verify if the number of records in the old database and new database is the same. Here use appropriate automation tool :: Count
* Verify that there are no redundancies and new database works exactly as the older one
* Verify if the schema, relationships, table structures are unaltered or set back to match the old database image :: Metadata validation
* Verify whether the changes made in application updates new database with correct values and type
* Verify if after the new database connection is provided to all the components of the application. Application, server, interfaces, firewall, network connectivity etc.

:: Dependent system validation.

* Verify the query performance (time-taken to execute complex queries) of the new database is not more than earlier performance ::Performance testing.

**II)** If the migration is a different type of Database, then along with above validation points, few or more has to be taken care:

* Verify data handling for all the fields. Major challenges will be handling data for calendar dates, floating numbers, hexadecimal etc.

Hi.. not uploaded yet..

Multiple joining conditions..

select m1.mdmid, m1.FirstName,m1.LastName,e1.emailaddress,e1.EmailType,p1.PhoneType,p1.PhoneNumber,

a1.AddressType,a1.AddressName,a1.City,a1.Country,a1.PostalCode

from map.Contact\_Main m1 left outer join map.Contact\_Email e1

on  m1.mdmid = e1.mdmid

left outer join map.Contact\_Phone p1

on  e1.mdmid = p1.mdmid

left outer join map.Contact\_Address a1

on p1.mdmid = a1.mdmid

where 1=1 and m1.mdmid in ('ELQ-EL4457874','ELQ-EL4457876','CRM-1-3CY-2474','ELQ-CDMSL000000104675')