statinfer

Step by step guide to learn Python

www.statinfer.com

Contents

statinfer

Part1-Introduction to Python

statinfer

Contents

Contents

statinfer

Introduction to Python & History

What is python

- It's a language
- Human-readable syntax and well Documented
- Open Source (Free)
- Powerful scripting language with simple Syntax
- www.python.org
- Used by many data scientists and developers

History

- Python language created by Guido van Rossum (Benevolent Dictator for Life).
- First Python version released in 1991
- Python 2 released in 2000
- Python 3 released in 2008
- Python 3 introduced to overcome future code expanding
- Python 3 is **NOT** fully backwards compatible with Python 2
- Python 2 is frozen and supported until 2020. Good features from Python 3 are back-ported.

Which one to use ? Python 2 or Python 3

- Python 2 is NOT same as Python 3. There are minor changes
- There are some incompatibilities, code in Python 2 may not always run in Python 3 and vice-versa.
- All important packages like NumPy , SciPy and Matplotlib are available for both Python 2 and Python 3
- We are going to use Python 3 in our course

statinfer

Installing Python & Python IDEs

Writing and executing python programs

- Python has many options to write and execute a program
- You can use Text Editors or Command line interfaces or Notebook or an IDE
- We will use Spyder IDE in our course
- Anaconda distribution has all the required software's inbuilt. We just need to download and install it.

statinfer

Installing Python, Anaconda

- Download and install Anaconda3
- It automatically installs
  - Ipython
  - Jupyter notebook
  - Spyder IDE
- Spyder is what we need for our coding

statinfer

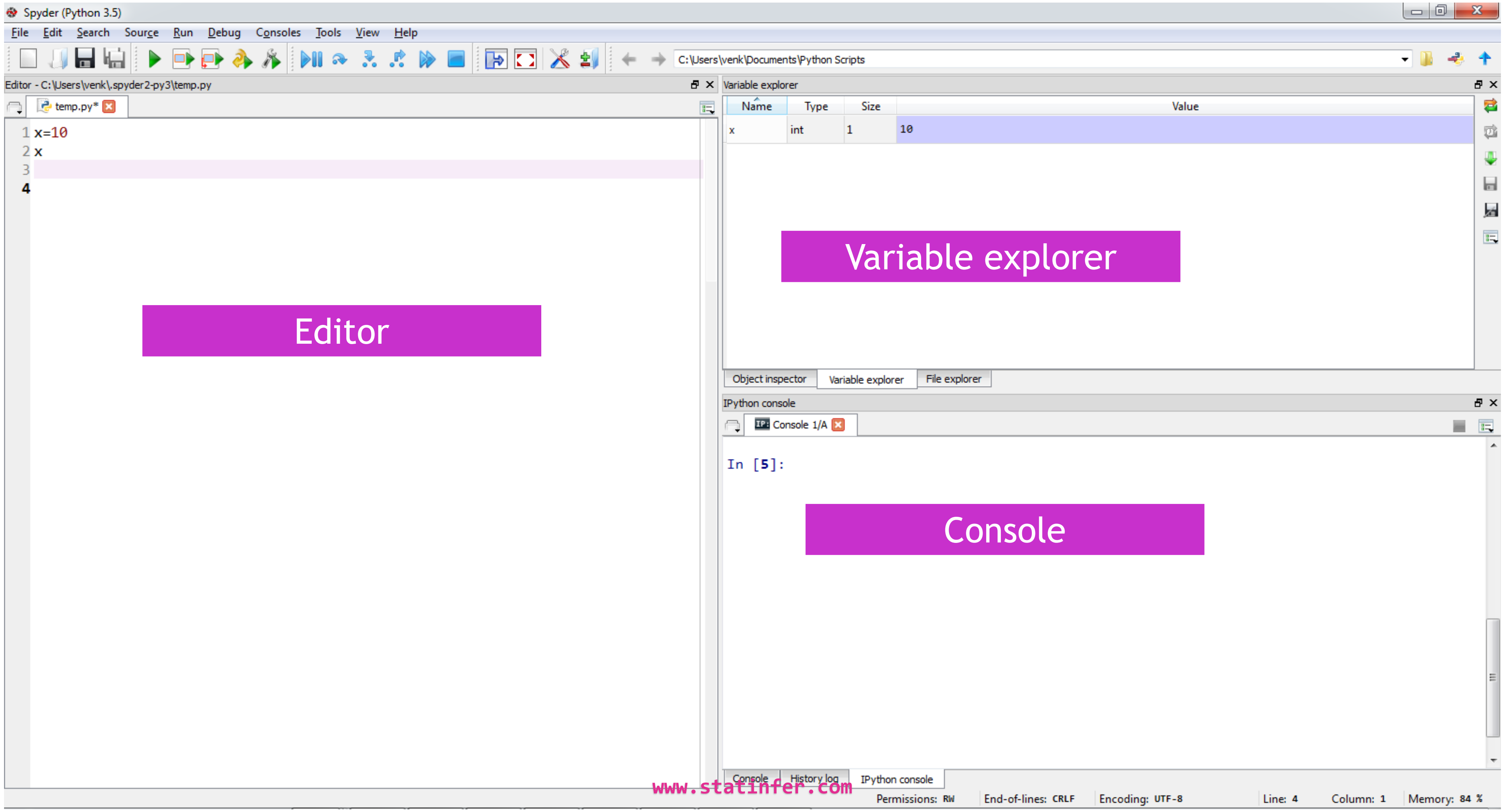
Spyder- Python IDEs

statinfer

Spyder- Python IDEs

- Spyder (formerly Pydee) is an open source cross-platform IDE for the Python language.
- Editor with syntax highlighting and code completion.
- Has an interactive console to execute and check the output of the code
- Testing and debugging is relatively easy
- Best IDE if you are coming from a R-Studio or MATLAB background
- Spyder also provides an object inspector that executes in the context of the console. Any objects created in the console can be examined in the inspector

Spyder Environment



```
x=10
x
```

Name	Type	Size	Value
x	int	1	10

```
In [5]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 4 Column: 1 Memory: 84 %

Spyder-Editor

- This is where you write the code
- To execute the code, select and hit Ctrl+Enter
- You can load old code files
- Code written in this editor is saved in .py format
- You can hit the tab button to see the autofill options on objects and function names
- You will be spending most of your time on editor

Spyder-Console

- This is where the code will be executed when you hit Ctrl+Enter in editor
- Helps us in code testing and debugging
- Helps us to avoid errors in the source code at the development phase itself
- Its usual practice to write a chuck of code in editor then execute it and see if it is working well or not
- You can toggle between Console and IPython Console.

Spyder – Variable explorer

- Shows all the variables that are created in the current session
- Helps in physically checking the presence of objects that are created
- Shows a quick summary of type of object, size, length, sample values etc.,
- We can run the code and see the objects getting created, also we can validate the data type and size of the object

statinfer

Basic Commands in Python

Before you code

- Python is case sensitive
- Be careful while using the Variable names and Function names
  - `Sales_data` is not same as `sales_data`
  - `Print()` is not same as `print()`

Basic Commands

```
571+95
19*17
print(57+39)
print(19*17)
print(“Statinfer")

# use hash(#) for comments
#Division example
34/56
```

Basics-What an error looks like?

```
Print(600+900) #used Print() instead of print()
576-'96'
```

LAB: Basic Commands

- Calculate below values
  - 973*75
  - 22/7
- Print the sting “my python file”

statinfer

Assigning and Naming convention

Assignment operator

= is the assignment operator

```
income=12000
income

x=20
x

y=30
z=x*y
z

name="Jack"
name
print(name)
```

```
del x #deletes the variable
```

Printing

```
name="Jack"
name
print(name)
```

Is there a difference between output of `name` and `print(name)`?

```
book_name="Practical business analytics \n using SAS"
book_name
print(book_name)
```

Naming convention

- Must start with a letter (A-Z or a-z)
- Can contain letters, digits (0-9), and/or underscore “_”

```
1x=20 #Doesn't work

x1=20 #works
x1

x.1=20 #Doesn't work
x.1

x_1=20 #works
x_1
```

statinfer

Type of Objects

Type of Objects

- Object to refer to any entity in a python program.
- Python has some standard built in object types
  - Numbers
  - Strings
  - Lists
  - Tuples
  - Dictionaries
- Having a good knowledge on these basic objects is essential to fee comfortable in Python programming

Numbers

- Numbers: integers & floats

```
age=30
age

weight=102.88
weight

x=17
x**2 #Square of x
```

Check the variable types for age and weight in variable explorer

Strings

- Strings are amongst the most popular types in Python. There are a number of methods or built-in string functions

Defining Strings

```
name="Sheldon"
msg="Data Science Classes"
```

Accessing strings

```
print(name[0])
print(name[1])
```

This is as good as substring

```
print(msg[0:9])
```

length of string

```
len(msg)
print(msg[10:len(msg)])
```

statinfer

Strings

Displaying string multiple time

```
msg="Site under Construction"
msg*10
msg*50
```

There is a difference between print and just displaying a variable

```
message="Data Science on R and Data Science on Python \n"
message*10
print(message*10)
```

Strings

```
#String Concatenation
msg1="Site under Construction "
msg2=msg1+"Go to home page \n"
print(msg2)
print(msg2*10)
```

List

- A sequence of related data
- Similar to array
- Lists, in-general are sequences of same kind of elements

Creating a list

```
mylist1=['Sheldon',‘Tommy', ‘Benny’]
```

Accessing list elements

```
mylist1[0] #Python indexing starts from 0
mylist1[1]
mylist1[2]
```

List

Appending to a list

```
mylist2=['L.A','No 173', "CR108877"]
final_list=mylist1+mylist2
final_list
```

Updating list elements

```
final_list[2]=35
final_list
```

Length of list

```
len(final_list)
```

Deleting an element in list

```
del final_list[5]
final_list
```

Tuples

- Also sequence data types
- Crated using parenthesis. Lists were created using square brackets
- Tuples can't be updated – This property is called immutability

```
my_tuple=('Mark','Male', 55)
my_tuple
my_tuple[1]
my_tuple[2]

my_tuple[0]*10
```

Tuples vs Lists

- Lists and Tuple are almost same, Tuple **save** lot of run time
- Immutable objects can give us substantial efficiency and code execution optimization
- Tuples in-general are sequences of different kind of elements
- Lists, in-general are sequences of same kind of elements

Tuples vs Lists

- Difference between tuples and lists

```
#tuple can't be updated

my_list=['Sheldon','Tommy', 'Benny']
my_tuple=('Mark','M', 55)

mylist[2]='Sunny'
my_list

my_tuple[2]=40
```

Tuples vs Lists

```
import time
time.localtime()

list(range(15, 25))
```

1. The first one, a tuple, is a sequence in which position has semantic value. The first position is always a year. This tuple functions as a lightweight record or struct
2. The second one, a list, is a sequence where we may care about order, but where the individual values are functionally equivalent.
3. Adding or removing items from the list without breaking the code that handles it

Dictionaries

- Dictionaries have two major element types key and Value.
- Dictionaries are collection of key value pairs
- Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.
- Keys are unique within a dictionary

```
city={0:"LA", 1:"PA" , 2:"FL"}
city

city[0]
city[1]
city[2]
```

Dictionaries

- In dictionary, keys are similar to indexes. We define our own preferred indexes in dictionaries

```
Make sure that we give the right key index while accessing the elements
in dictionary
names={1:"David", 6:"Bill", 9:"Jim"}
names
names[0] #Doesn't work, why?
names[1]
names[2]
names[6]
names[9]
```

Dictionaries

In the key value pairs, key need not be a number always

```
edu={"David":"Bsc", "Bill":"Msc", "Jim":"Phd"}
edu

edu[0]
edu[1]
edu[David]
edu["David"]
```

Dictionaries

Updating values in dictionary

```
edu
edu["David"]="MSc"
edu
```

Updating keys in dictionary

Delete the key and value element first and then add new element

```
city={0:"LA", 1:"PA" , 2:"FL"}
#How to make 6 as "LA"
del city[0]
city
city[6]="LA"
city
```

Dictionaries

- Fetch all keys and all values separately

```
city.keys()
city.values()

edu.keys()
edu.values()
```

statinfer

Packages

Packages

- A package is collection of python functions. A properly structured and complied code. A package may contain many sub packages.
- Many python functions are only available via “packages” that must be imported.
- For example to find value of log(10) we need to first import match package that has the log function in it

```
log(10)
exp(5)
sqrt(256)

import math
math.log(10)
math.exp(5)
math.sqrt(256)
```

Packages

Most general python coding style

```
import math as mt
mt.log(10)
mt.exp(5)
mt.sqrt(256)
```

Packages

- To be a good data scientist on python, on need to be very comfortable with below packages
  - numpy
  - scipy
  - pandas
  - scikit-Learn
  - matplotlib
  - nltk

Important Packages- NumPy

- NumPy is for fast operations on vectors and matrices, including mathematical, logical, shape manipulation, sorting, selecting.
- It is the foundation on which all higher level tools for scientific Python packages are built

```
import numpy as np

income = np.array([9000, 8500, 9800, 12000, 7900, 6700, 10000])
print(income)
print(income[0])

expenses=income*0.65
print(expenses)

savings=income-expenses
print(savings)
```

statinfer

Important Packages- Pandas

- Data frames and data handling
- Pandas has Data structures and operations for manipulating numerical tables and time series.

```
import pandas as pd
bank= pd.read_csv('C:\\Users\\venk\\Google
Drive\\Training\\Datasets\\Bank Tele
Marketing\\bank_market.csv')

print(bank)
```

Important Packages- Matplotlib

Plotting library similar to MATLAB plots

```
import matplotlib as mp
import numpy as np

X = np.random.normal(0,1,1000)
Y = np.random.normal(0,1,1000)

mp.pyplot.scatter(X,Y)
```

Important Packages- Scikit-Learn

- Machine learning algorithms made easy

```
import sklearn as sk
import pandas as pd

air = pd.read_csv("D:\\Google
Drive\\Training\\Datasets\\AirPassengers\\AirPassengers.csv")
air

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(air[["Promotion_Budget"]], air[["Passengers"]])

#Coefficients
print(lr.coef_)
print(lr.intercept_)
```

statinfer

If-Then-Else statement

If Condition

```
age=60
if age<50:
    print("Group1")
print("Done with If")

age=60
if age<50:
    print("Group1")
    print("Done with If")

age=40
if age<50:
    print("Group1")
print("Done with If")
```

If-else statement

```
age=60
if age<50:
    print("Group1")
else:
    print("Group2")
print("Done with If else")
```

Multiple else conditions in if

```
marks=75

if(marks<30):
    print("fail")
elif(marks<60):
    print("Second Class")
elif(marks<80):
     print("First Class")
elif(marks<100):
     print("Distinction")
else:
    print("Error in Marks")
```

Multiple else conditions in if

If condition for checking whether a candidate secured First class/ second class or failed in an exam.

```
marks=20

if(marks<30):
    print("fail")
elif(marks<60):
    print("Second Class")
elif(marks<80):
     print("First Class")
elif(marks<100):
     print("Distinction")
else:
    print("Error in Marks")
```

Nested if

- If condition for checking whether a number is less than 30 or less than 40 or less than 50 or greater than 50

```
x=45

if(x<50):
    print("Number is less than 50")
    if(x<40):
         print ("Number is less than 40")
         if(x<30):
             print("Number is less than 30")
         else:
             print("Number is greater than 30")
    else:
        print("Number is greater than 40")
else:
    print("Number is greater than or equal to 50")
```

Nested if

```
x=35

if(x<50):
    print("Number is less than 50")
    if(x<40):
         print ("Number is less than 40")
         if(x<30):
             print("Number is less than 30")
         else:
             print("Number is greater than 30")
    else:
        print("Number is greater than 40")
else:
    print("Number is greater than or equal to 50")
```

statinfer

For loop

For loop

Print first 20 values

```
#Example-1
my_num=1

for i in range(1,20):
    my_num=my_num+1
    print("my num value is", my_num)

for i in range(1,20,2):
    my_num=my_num+2
    print("my num value is", my_num)
```

LAB : For loop

Print first 20 values cumulative running sum

```
#Example-2
sumx = 0
x=1

for x in range(1,20):
     sumx = sumx + x
     print(sumx)
```

Break Statement in for loop

- To stop execution of a loop
- Stopping the loop in midway using a condition

- Print cumulative sum and stop when sum reaches 500

```
sumx = 0
x=1

for x in range(1,200):
     sumx = sumx + x
     if(sumx>500):
         break
     print(sumx)
```

statinfer

General notes

General notes

- Variable is lost after restarting shell
- Using the same object name overwrites the old object
- Customize the color coding and highlighting of coding syntax, it makes it easy to read
- Make use of variable explorer for physical verification of created variables

statinfer

Conclusion

Conclusion

- In this session we got basic introduction to Python. We tried some basic commands in Python
- In later sessions we will see data handling and basic descriptive statistics

statinfer

Part2-Data Handling in Python

Contents

Contents

- Data importing
- Working with datasets
- Manipulating the datasets
- Creating new variables
- Exporting the datasets into external files
- Data Merging
- Conclusion

Data import from CSV files

- Need to use the function read.csv
- Need to use “/” or “\\” in the path. The windows style of path “\” doesn’t work

Importing from CSV files

```
import pandas as pd     # importing library pandas

Sales = pd.read_csv("C:\\Users\\Datasets\\Superstore Sales 
Data\\Sales_sample.csv")

print(Sales)


Sales1 = pd.read_csv(r"C:\Datasets\Superstore Sales Data\Sales_sample.csv")


print(Sales1)
```

Data import from Excel files

- Need to use pandas again

Data import from Excel files

```
import pandas as pd

wb_data = pd.read_excel("C:\\Users\\venk\\Google
Drive\\Training\\Datasets\\World Bank Data\\World Bank Indicators.xlsx" ,
"Data by country”)

print(wb_data)
```

Basic Commands on Datasets

- Is the data imported correctly? Are the variables imported in right format? Did we import all the rows?
- Once the dataset is inside Python, we would like to do some basic checks to get an idea on the dataset.
- Just printing the data is not a good option, always.
- Is a good practice to check the number of rows, columns, quick look at the variable structures, a summary and data snapshot

Check list after Import

Data: Superstore Sales Data\\Sales_sample.csv

Code	Description
`Sales.shape`	To check the number of rows and columns
`Sales.columns.values`	What are the column names?, Sometimes import doesn't consider column names while importing
`Sales.head(10)`	First few observations of data
`Sales.tail(10)`	Last few observations of the data
`Sales.dtypes`	Data types of all variables

Quick Summary

Code	Description
`Sales.describe()`	Summary of all variables
`Sales[‘unitsSold'].describe()`	Summary of a variable
`Sales.salesChannel.value_counts()`	Get frequency table for a given variable
`sum(Sales.custId.isnull())`	Missing value count in a variable
`Sales.sample(n=10)`	Take a random sample of size 10

Lab: Printing the data and meta info

- Import “Superstore Sales Data\\Sales_by_country_v1.csv” data
- How many rows and columns are there in this dataset?
- Print only column names in the dataset
- Print first 10 observations
- Print the last 5 observations
- Get the summary of the dataset
- Print the structure of the data
- Describe the field unitsSold
- Describe the field custCountry
- Create a new dataset by taking first 30 observations from this data
- Print the resultant data
- Remove(delete) the new dataset

Sub setting the data

- Dataset: “./World Bank Data/GDP.csv“

```
import pandas as pd
#The below line may throw some error
gdp1=pd.read_csv("C:\\Users\\venk\\Google Drive\\Training\\Datasets\\World Bank 
Data\\GDP.csv")

#Include encoding = "ISO-8859-1" or encoding = "utf8" to tackle the error
gdp=pd.read_csv("C:\\Users\\venk\\Google Drive\\Training\\Datasets\\World Bank 
Data\\GDP.csv",encoding = "ISO-8859-1")

gdp.shape
gdp.columns.values
```

- New dataset with selected rows

```
gdp1 = gdp.head(10)
gdp2=gdp.iloc[[2,9,15,25]]
```

Sub setting the data

- New dataset by keeping selected columns

```
gdp3 = gdp[["Country", "Rank"]]
gdp3
```

Sub setting the data

- New dataset with selected rows and columns
  - `gdp4 = gdp[["Country", "GDP"]][0:10]`
  - `gdp4`

- New data by excluding columns
  - `gdp5=gdp.drop(["Country_code"])`
  - `#This code doesn't work; You will get axis related error`

  - `Use axis =1 ; 0 for rows and 1 for column; Default value is 0`
  - `gdp5=gdp.drop(["Country_code"], axis=1)`
  - `gdp5`

statinfer

All the subset combinations with Index

```
rows_to_keep=list(range(50, 81))
rows_to_drop=list(range(2, 41))
col_to_keep=["Country", "GDP"]
col_to_drop=["Country", "Country_code"]

#Keeping selected rows and keeping selected cols
gdp9 = gdp[col_to_keep].iloc[rows_to_keep]
print(gdp9.head())

#Keeping selected rows and dropping selected cols
gdp10 = gdp.drop(col_to_drop, axis=1).iloc[rows_to_keep]
print(gdp10.head())
```

statinfer

All the subset combinations with Index

```
#Dropping selected rows and keeping selected cols
gdp11 = gdp[col_to_keep].drop(rows_to_drop, axis=0)
print(gdp11.head())

#Dropping selected rows and dropping selected cols
gdp12 = gdp.drop(col_to_drop, axis=1).drop(rows_to_drop, axis=0)
print(gdp12.head())
```

Lab: Sub setting the data

- Data : "./Bank Tele Marketing/bank_market.csv"
- Create separate datasets for each of the below tasks
  - Select first 1000 rows only
  - Select only four columns "Cust_num" "age" "default" and "balance"
  - Select 20,000 to 40,000 observations along with four variables "Cust_num" "job" "marital" and "education"
  - Select 5000 to 6000 observations drop "poutcome" and "y"

Subset with variable filter conditions

- Selection with a condition on variables
  - For example, selection of customers with age>40.
  - `bank_subset=bank_data[bank_data['age']>40]`
- And condition & filters

```
bank_subset1=bank_data[(bank_data['age']>40) &
(bank_data['loan']=="no")]
```

- OR condition & filters
  - `bank_subset2=bank_data[(bank_data['age']>40) | (bank_data['loan']=="no")]`

statinfer

Subset with variable filter conditions

- AND, OR condition Numeric and Character filters
  - `bank_subset3= bank_data[(bank_data['age']>40) & (bank_data['loan']=="no") | (bank_data['marital']=="single" )]`
  - `bank_subset3`

statinfer

Lab: Subset with variable filter conditions

- Data : "./Automobile Data Set/AutoDataset.csv"
- Create a new dataset for exclusively Toyota cars
- Create a new dataset for all cars with city.mpg greater than 30 and engine size is less than 120.
- Create a new dataset by taking only sedan cars. Keep only four variables(Make, body style, fuel type, price) in the final dataset.
- Create a new dataset by taking Audi, BMW or Porsche company makes.

```
list(dataset.coloumn.values)
auto[auto["make"].isin(["Audi","BMW","Porsche"])]
```

Calculated Fields

- Calculate and Assign it to new variable

```
auto_data['area']=(auto_data[' length'])*(auto_data[' width'])*(auto_data['
height'])

auto_data['area']
```

Sorting the data

- Its ascending by default

```
Online_Retail_sort=Online_Retail.sort_values(by='UnitPrice')
Online_Retail_sort.head(20)
```

- Use ascending=False for descending sort

```
Online_Retail_sort=Online_Retail.sort_values(by='UnitPrice',ascending=False)
Online_Retail_sort.head(20)
```

- Sorting with two cols

```
Online_Retail_sort2=Online_Retail.sort_values(by=['Country','UnitPrice'],
ascending=[True, False])
Online_Retail_sort2.head(5)
```

Dataset “Online Retail Sales Data\Online Retail.csv”

LAB: Sorting the data

- AutoDataset
- Sort the dataset based on length
- Sort the dataset based on length descending

statinfer

Identifying & Removing Duplicates

Datasets: Telecom Data Analysis\Bill.csv

```
#Identify duplicates records in the data
dupes=bill_data.duplicated()
sum(dupes)

#Removing Duplicates
bill_data_uniq=bill_data.drop_duplicates()
```

Identifying & Duplicates based on Key

- What if we are not interested in overall level records
- Sometimes we may name the records as duplicates even if a key variable is repeated.
- Instead of using duplicated function on full data, we use it on one variable

```
#Identify duplicates in complaints data based on cust_id
dupe_id=bill_data.duplicated(['cust_id'])

#Removing duplicates based on a variable
bill_data_cust_uniq=bill_data.drop_duplicates(['cust_id'])
```

LAB: Handling Duplicates

- DataSet: "./Telecom Data Analysis/Complaints.csv"
- Identify overall duplicates in complaints data
- Create a new dataset by removing overall duplicates in Complaints data
- Identify duplicates in complaints data based on cust_id
- Create a new dataset by removing duplicates based on cust_id in Complaints data

Data sets merging and Joining

- Datasets: TV Commercial Slots Analysis/orders.csv & TV Commercial Slots Analysis/slots.csv

```
orders1=orders.drop_duplicates(['Unique_id'])
slots1=slots.drop_duplicates(['Unique_id'])

##Inner Join
inner_data=pd.merge(orders1, slots1, on='Unique_id', how='inner')
###Outer Join
outer_data=pd.merge(orders1, slots1, on='Unique_id', how='outer')
##Left outer Join
L_outer_data=pd.merge(orders1, slots1, on='Unique_id', how='left')
###Right outer Join
R_outer_data=pd.merge(orders1, slots1, on='Unique_id', how='right’)

####Other options
left_on : a column or a list of columns
right_on : a column or a list of columns
```

LAB: Data Joins

- Datasets
  - "./Telecom Data Analysis/Bill.csv"
  - "./Telecom Data Analysis/Complaints.csv"
- Import the data and remove duplicates based on cust_id
- Create a dataset for each of these requirements
  - All the customers who appear either in bill data or complaints data
  - All the customers who appear both in bill data and complaints data
  - All the customers from bill data: Customers who have bill data along with their complaints
  - All the customers from complaints data: Customers who have Complaints data along with their bill info

Conclusion

- In this session we started with Data imploring from various sources
- We saw some basic commands to work with data
- We also learnt manipulating the datasets and creating new variables
- Sorting the datasets and handling duplicates
- Joining the datasets is also an important concept
- There are many more topics to discuss in data handling, these topics in the session are essential for any data scientist

statinfer

Part3-Basic Statistics, Graphs and Reports

statinfer.com

Contents

- Taking a random sample from data
- Descriptive statistics
  - Central Tendency
  - Variance
- Quartiles, Percentiles
- Box Plots
- Graphs

statinfer

Sampling in Python

Sampling in Python

- We need to use sample() function

```
Online_Retail=pd.read_csv("C:\\Users\\venk\\Google
Drive\\Training\\Datasets\\Online Retail Sales Data\\Online Retail.csv",
encoding = "ISO-8859-1")
Online_Retail.shape


sample_data=Online_Retail.sample(n=1000)
sample_data.shape
sample_data.head(10)
```

Sample with seed

```
#Regenerating same sample again
sample_data1=Online_Retail.sample(n=1000 , random_state=10)
sample_data1.shape
print(sample_data1.head())
```

LAB: Sampling in Python

- Import “Census Income Data/Income_data.csv”
- Create a new dataset by taking a random sample of 5000 records
- Take a random sample of 5000 records with seed

Code: Sampling in Python

```
#Import “Census Income Data/Income_data.csv”
Income=pd.read_csv("D:\\Google Drive\\Training\\Datasets\\Census Income
Data\\Income_data.csv")

Income.shape
Income.head()
Income.tail(3)

#Sample size 5000
Sample_income=Income.sample(n=5000)
Sample_income.shape
```

statinfer

Descriptive statistics

Descriptive statistics

- The basic descriptive statistics to give us an idea on the variables and their distributions
- Permit the analyst to describe many pieces of data with a few indices
- Central tendencies
  - Mean
  - Median
- Dispersion
  - Range
  - Variance
  - Standard deviation

statinfer

Central tendencies: Mean and Median

Central tendencies

- Mean
  - The arithmetic mean
  - Sum of values/ Count of values
  - Gives a quick idea on average of a variable

Mean in Python

```
gain_mean=Income["capital-gain"].mean()
gain_mean
```

Median

- Mean is not a good measure in presence of outliers
- For example Consider below data vector
  - 1.5,1.7,1.9,0.8,0.8,1.2,1.9,1.4, 9 , 0.7 , 1.1
- 90% of the above values are less than 2, but the mean of above vector is 2
- There is an unusual value in the above data vector i.e 9
- It is also known as outlier.
- Mean is not the true middle value in presence of outliers. Mean is very much effected by the outliers.
- We use median, the true middle value in such cases
- Sort the data either in ascending or descending order

Median

1.5
1.7
1.9
0.8
0.8
1.2
1.9
1.4
9
0.7
1.1

0.7
0.8
0.8
1.1
1.2
1.4
1.5
1.7
1.9
1.9
9

- Mean of the data is 2
- Median of the data is 1.4
- Even if we have the outlier as 90, we will have the same median
- Median is a positional measure, it doesn’t really depend on outliers
- When there are no outliers then mean and median will be nearly equal
- When mean is not equal to median it gives us an idea on presence of outliers in the data

Mean and Median

Import “Census Income Data/Income_data.csv”

```
#Mean and Median on python
gain_mean=Income["capital-gain"].mean()
gain_mean

gain_median=Income["capital-gain"].median()
gain_median
```

Mean is far away from median. Looks like there are outliers, we need to look at percentiles and box plot.

LAB: Mean and Median

- Dataset: "./Online Retail Sales Data/Online Retail.csv”
- What is the mean of “UnitPrice”
- What is the median of “UnitPrice”
- Is mean equal to median? Do you suspect the presence of outliers in the data?
- What is the mean of “Quantity”
- What is the median of “Quantity”
- Is mean equal to median? Do you suspect the presence of outliers in the data?

Code: Mean and Median

```
Online_Retail=pd.read_csv("D:\\Google
Drive\\Training\\Datasets\\Online_Retail_Sales_Data\\Online Retail.csv", encoding = "ISO-8859-
1")
Online_Retail.shape
Online_Retail.columns.values

#Mean and median of 'UnitPrice' in Online Retail data
up_mean=Online_Retail['UnitPrice'].mean()
up_mean

up_median=Online_Retail['UnitPrice'].median()
up_median

#Mean of "Quantity" in Online Retail data
Quantity_mean=Online_Retail['Quantity'].mean()
Quantity_mean

Quantity_median=Online_Retail['Quantity'].median()
Quantity_median
```

statinfer

Dispersion Measures : Variance and Standard Deviation

Dispersion

- Just knowing the central tendency is not enough.
- Two variables might have same mean, but they might be very different.
- Look at these two variables. Profit details of two companies A & B for last 14 Quarters in MMs

															Mean
Company A	43	44	0	25	20	35	-8	13	-10	-8	32	11	-8	21	15
Company B	17	15	12	17	15	18	12	15	12	13	18	18	14	14	15

- Though the average profit is 15 in both the cases
- Company B has performed consistently than company A.
- There was even loses for company A
- Measures of dispersion become very vital in such cases

Variance and Standard deviation

- Dispersion is the quantification of deviation of each point from the mean value.
- Variance is average of squared distances of each point from the mean
- Variance is a fairly good measure of dispersion.
- Variance in profit for company A is 352 and Company B is 4.9

Value	Value-Mean	(Value-Mean)^2
43	28	784
44	29	841
0	-15	225
25	10	100
20	5	25
35	20	400
-8	-23	529
13	-2	4
-10	-25	625
-8	-23	529
32	17	289
11	-4	16
-8	-23	529
21	6	36
15.0		352

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n}$$

Value	Value-Mean	(Value-Mean)^2
17	2	4
15	0	0
12	-3	9
17	2	4
15	0	0
18	3	9
12	-3	9
15	0	0
12	-3	9
13	-2	4
18	3	9
18	3	9
14	-1	1
14	-1	1
15.0		4.9

Standard Deviation

- Standard deviation is just the square root of variance
- Variance gives a good idea on dispersion, but it is of the order of squares.
- Its very clear from the formula, variance unites are squared than that of original data.
- Standard deviation is the variance measure that is in the same units as the original data

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n}}$$

Code-Variance and Standard deviation

- Divide the Income data into two sets. USA vs Others
- Find the variance of “education.num” in those two sets. Which one has higher variance?

```
usa_income=Income[Income["native-country"]==' United-States']
usa_income.shape

other_income=Income[Income["native-country"]!=' United-States']
other_income.shape

#USA
var_usa=usa_income["education-num"].var()
var_usa

std_usa=usa_income["education-num"].std()
std_usa
```

Code-Variance and Standard deviation

```
#Others
var_other=other_income["education-num"].var()
var_other

std_other=other_income["education-num"].std()
std_other
```

LAB: Variance and Standard deviation

- Dataset: "./Online Retail Sales Data/Online Retail.csv”
- What is the variance and s.d of “UnitPrice”
- What is the variance and s.d of “Quantity”
- Which one these two variables is consistent?

LAB: Variance and Standard deviation

```
#var and sd UnitPrice
var_UnitPrice=Online_Retail['UnitPrice'].var()
var_UnitPrice

std_UnitPrice=Online_Retail['UnitPrice'].std()
std_UnitPrice

#variance and sd of Quantity
var_UnitPrice=Online_Retail['Quantity'].var()
var_UnitPrice

std_UnitPrice=Online_Retail['Quantity'].std()
std_UnitPrice
```

statinfer

Percentiles & Quartiles

Percentiles

- A student attended an exam along with 1000 others.
  - He got 68% marks? How good or bad he performed in the exam?
  - What will be his rank overall?
  - What will be his rank if there were 100 students overall?
- For example, with 68 marks, he stood at 90th position. There are 910 students who got less than 68, only 89 students got more marks than him
- He is standing at 91 percentile.
- Instead of stating 68 marks, 91% gives a good idea on his performance
- Percentiles make the data easy to read

statinfer

Percentiles

- p^{th} percentile: p percent of observations below it, (100 - p)% above it.
- Marks are 40 but percentile is 80%, what does this mean?
- 80% of CAT exam percentile means
  - 20% are above & 80% are below
- Percentiles help us in getting an idea on outliers.
- For example the highest income value is 400,000 but 95th percentile is 20,000 only. That means 95% of the values are less than 20,000. So the values near 400,000 are clearly outliers

Quartiles

- Percentiles divide the whole population into 100 groups where as quartiles divide the population into 4 groups
- p = 25: First Quartile or Lower quartile (LQ)
- p = 50: second quartile or Median
- p = 75: Third Quartile or Upper quartile (UQ)

Percentiles & Quartiles

- By default summary gives 4 quartiles

```
Income['capital-gain'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
Income['capital-loss'].quantile([0, 0.1, 0.2, 0.3,0.4,0.5,0.6,0.7,0.8,0.9,1])
Income['hours-per-week'].quantile([0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1])
```

- Looks like some people are working 90 hours perweek.

LAB: Percentiles & Quartiles

- Dataset: “./Bank Tele Marketing/bank_market.csv”
- Get the summary of the balance variable
- Do you suspect any outliers in balance ?
- Get relevant percentiles and see their distribution.
- Are there really some outliers present?
- Get the summary of the age variable
- Do you suspect any outliers in age?
- Get relevant percentiles and see their distribution.
- Are there really some outliers present?

Code: Percentiles & Quartiles

```
#Get the summary of the balance variable
#we can find the summary of the balance variable by using .describe()
summary_bala=bank["balance"].describe()
summary_bala

#Get relevant percentiles and see their distribution.
bank['balance'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])

#Get the summary of the age variable
summary_age=bank['age'].describe()
summary_age

#Get relevant percentiles and see their distribution
bank['age'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])
```

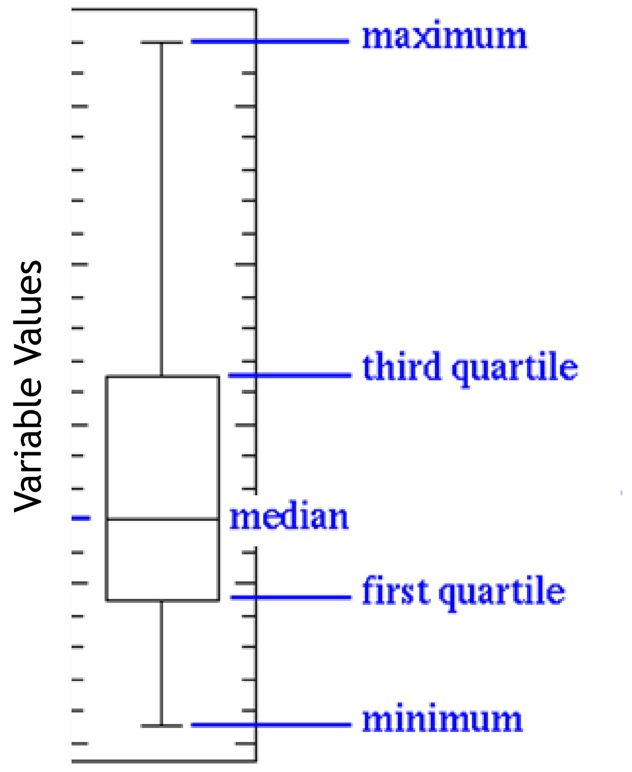
statinfer

Part-7: Box plots and outlier detection

statinfer

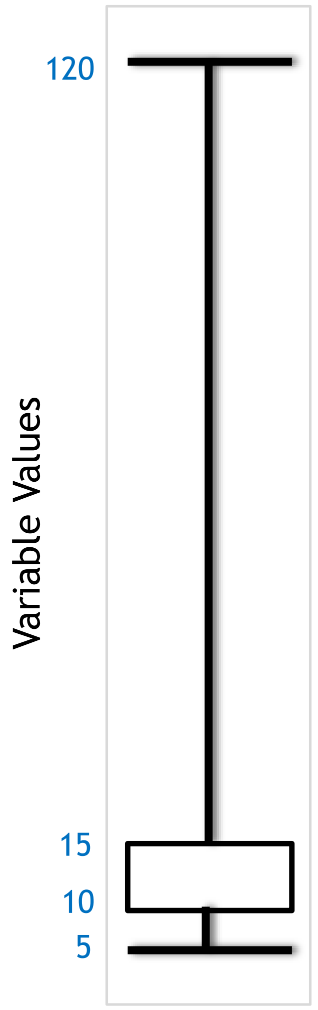
Box plots and outlier detection

- Box plots have box from LQ to UQ, with median marked.
- They portray a five-number graphical summary of the data Minimum, LQ, Median, UQ, Maximum
- Helps us to get an idea on the data distribution
- Helps us to identify the outliers easily
- 25% of the population is below first quartile,
- 75% of the population is below third quartile
- If the box is pushed to one side and some values are far away from the box then it’s a clear indication of outliers

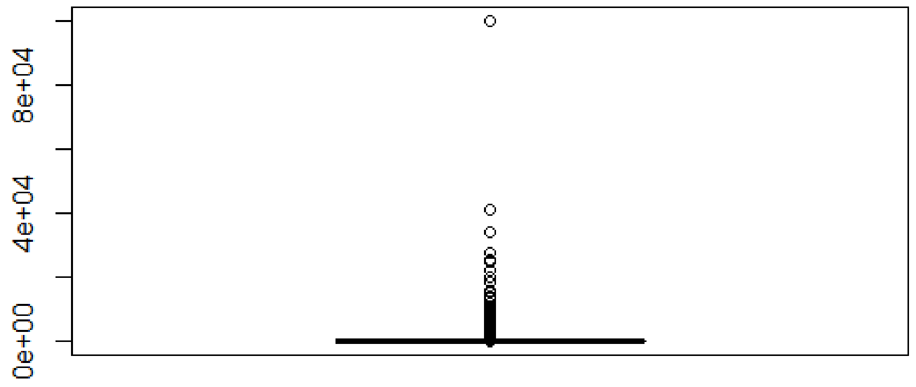


statinfer

Box plots and outlier detection



- Some set of values far away from box, is gives us a clear indication of outliers.
- In this example the minimum is 5, maximum is 120, and 75% of the values are less than 15
- Still there are some records reaching 120. Hence a clear indication of outliers
- Sometimes the outliers are so evident that, the box appear to be a horizontal line in box plot.



Box plots and outlier detection

```
import matplotlib.pyplot as plt
plt.boxplot(bank.balance)
```

statinfer

LAB: Box plots and outlier detection

- Dataset: “./Bank Marketing/bank_market.csv”
- Draw a box plot for balance variable
- Do you suspect any outliers in balance ?
- Get relevant percentiles and see their distribution.
- Draw a box plot for age variable
- Do you suspect any outliers in age?
- Get relevant percentiles and see their distribution.

statinfer

LAB: Box plots and outlier detection

```
import matplotlib.pyplot as plt
#Basic plot of boxplot by importing the matplot.pyplot as plt ("plt.boxplot())
plt.boxplot(bank.balance)

#Get relevant percentiles and see their distribution
bank['balance'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,0.95, 1])
# outlier are present in balance variable

#Do you suspect any outliers in age
#detect the outliers in age variable by plt.boxplot()
plt.boxplot(bank.age)
#No outliers are present

#Get relevant percentiles and see their distribution
bank['age'].quantile([0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95,1])
#outliers are not present in age variable
```

statinfer

Creating Graphs

Creating Graphs

- Scatter Plot:
  - Scatter plots give us an indication on the relation between the two chosen variables.
- Example:

```
cars=pd.read_csv("C:\\Users\\venk\\Google Drive\\Training\\Datasets\\Cars Data\\Cars.csv",encoding = "ISO-8859-1")
cars.shape
cars.columns.values


import matplotlib.pyplot as plt
plt.scatter(cars.Horsepower,cars.MPG_City)
```

LAB: Creating Graphs

- Dataset: "./Sporting_goods_sales/Sporting_goods_sales.csv"
- Draw a scatter plot between Average_Income and Sales. Is there any relation between two variables?
- Draw a scatter plot between Under35_Population_pect and Sales. Is there any relation between two?

```
import matplotlib.pyplot as plt
plt.scatter(cars.Horsepower,cars.MPG_City)
```

Code: Creating Graphs

```
import matplotlib.pyplot as plt

#Draw a scatter plot between Average_Income and Sales. Is there any
relation between two variables
plt.scatter(sports_data.Average_Income,sports_data.Sales)

import numpy as np
np.corrcoef(sports_data.Average_Income,sports_data.Sales)

#Draw a scatter plot between Under35_Population_pect and Sales. Is
there any relation between two
plt.scatter(sports_data.Under35_Population_pect,sports_data.Sales,colo
r="red")
np.corrcoef(sports_data.Under35_Population_pect,sports_data.Sales)
```

Bar Chart

- Bar charts used to summarize the categorical variables

```
import matplotlib.pyplot as plt

freq=cars.Cylinders.value_counts()
print(freq)
freq.values
freq.index

plt.bar(freq.index,freq.values)
```

LAB: Bar Chart

- Dataset: "./Sporting_goods_sales/Sporting_goods_sales.csv"
- Create a bar chart summarizing the information on family size.

```
freq=sports_data.Avg_family_size.value_counts()
freq.values
freq.index

import matplotlib.pyplot as plt
plt.bar(freq.index,freq.values)
plt.bar(freq.index,freq.values, align="center")
plt.bar(freq.index,freq.values, align="center",tick_label=freq.index)
```

Code: Bar Chart

```
freq=sports_data.Avg_family_size.value_counts()
freq.values
freq.index

import matplotlib.pyplot as plt
plt.bar(freq.index,freq.values)
```

Trend chart

- Trend chart is used for time series datasets

```
AirPassengers=pd.read_csv("D:\\Datasets\\Air Travel Data\\Air_travel.csv",
encoding = "ISO-8859-1")
AirPassengers.head()
AirPassengers.columns.values

import matplotlib.pyplot as plt
plt.plot(AirPassengers.AIR)

#X axis lable
#Format the date to DD-MM-YYYY before importing
AirPassengers['new_time']=pd.to_datetime(AirPassengers['DATE'],format='%d-%m-%Y')
plt.plot(AirPassengers.new_time,AirPassengers.AIR)
```

statinfer

User Defined functions in Python

Writing Function in

```
def my_function_name(param1, param2, param3):
   code lines
   code lines
   code lines
    return;
```

Distance Calculation function

- Distance Calculation function

```
def mydistance(x1,y1,x2,y2):
    import math
    dist=math.sqrt(pow((x1-x2),2)+pow((y1-y2),2))
    print(dist)
    return;

mydistance(0,0,2,2)
mydistance(4,6,1,2)
```

Lab: User Defined Functions

- Create a function that calculates the Absolute percentage difference between two input values. Take second value as reference
  - Test it with (30,80)
- Create a function that takes a tuple as input and gives the sum of squares of input tuple values as output

```
def abspe(x,y):
    abpe=abs((x-y)/y)
    print(abpe)
    return;

abspe(5,9)
abspe(10,100)
```

Code: User Defined Functions

```
def sumsquares(*inputnums):
     s = 0
     for n in inputnums:
           s =s + pow(n,2)
           print(s)
     return s;


sumsquares (1,1,1,1,1)
sumsquares (1,2,5,8,-1,-7,9,12,32,4)
```

User Defined Function for Var name and Mean

Function input => Dataset
Function Output => Variable name and Mean

```
import pandas as pd
column_names = ["Name","Mean”]
summary_df=pd.DataFrame(columns=column_names)

def allsummary(df):
    i=1
    for f in df.columns.values:
        summary_df.set_value(i,"Name",f)
        summary_df.set_value(i, "Mean",df[f].mean())
         i=i+1;
   print(summary_df)

credit_risk=pd.read_csv("D:Datasets\\Give me some Credit\\cs-training.csv",encoding
="ISO-8859-1")
allsummary(credit_risk)
```

Using at[] instead of set_values

```
import pandas as pd
column_names = ["Name","Mean”]
summary_df=pd.DataFrame(columns=column_names)

def allsummary(df):
    i=1
    for f in df.columns.values:
         summary_df.at[i,"Name"]=f
         summary_df.at[i,"Mean"]=df[f].mean()
         i=i+1;
    print(summary_df)

credit_risk=pd.read_csv("D:Datasets\\Give me some Credit\\cs-training.csv",encoding
="ISO-8859-1")
allsummary(credit_risk)
```

User Defined Function for Var name and Mean

- Function input => Dataset
  Function Output => Variable name, Mean, Median and variance

LAB: User Defined Function for Quantiles

- Create a function that takes dataset as input and gives below summary metrics for every variable
  - Variable name, Mean, Median and variance
  - Minimum value, Maximum value
  - 10th percentile
  - 25th percentile
  - 50th percentile
  - 75th percentile
  - 90th percentile
  - Count and percentage of missing values
- Test the above function on credit risk data

Hint: summary_df.set_value(i, "p5",df[f].dropna(axis=0).quantile(0.05))

Number of rows= df.shape[0]

Code: User Defined Function for Quantiles

```
import pandas as pd
column_names = ["Name","Mean", "Median", "Variance","S.D"]
summary_df=pd.DataFrame(columns=column_names)

def allsummary(df):
    i=1
    for f in df.columns.values:
        summary_df.set_value(i,"Name",f)
        summary_df.set_value(i, "Mean",df[f].mean())
        summary_df.set_value(i, "Median",df[f].median())
        summary_df.set_value(i, "Variance",df[f].var())
        summary_df.set_value(i, "S.D",df[f].std())
        summary_df.set_value(i, "p95",df[f].dropna(axis=0).quantile(0.95))
        i=i+1;
    print(summary_df)

credit_risk=pd.read_csv("D:\\cs-training.csv", encoding = "ISO-8859-1")
allsummary(credit_risk)
```

```
column_names = ["Name","Mean", "Median", "Variance","S.D", "p5", "p10", "p20", "p25", "p30", "p50", "p75", "p80", "p90", "p95",
"p97", "p99"]
summary_df=pd.DataFrame(columns=column_names)

def allsummary(df):
    i=1
    for f in df.columns.values:
        summary_df.set_value(i,"Name",f)
        summary_df.set_value(i, "Mean",df[f].mean())
        summary_df.set_value(i, "Median",df[f].median())
        summary_df.set_value(i, "Variance",df[f].var())
        summary_df.set_value(i, "S.D",df[f].std())
        summary_df.set_value(i, "p5",df[f].dropna(axis=0).quantile(0.05))
        summary_df.set_value(i, "p10",df[f].dropna(axis=0).quantile(0.1))
        summary_df.set_value(i, "p20",df[f].dropna(axis=0).quantile(0.2))
        summary_df.set_value(i, "p25",df[f].dropna(axis=0).quantile(0.25))
        summary_df.set_value(i, "p30",df[f].dropna(axis=0).quantile(0.3))
        summary_df.set_value(i, "p50",df[f].dropna(axis=0).quantile(0.5))
        summary_df.set_value(i, "p75",df[f].dropna(axis=0).quantile(0.75))
summary_df.set_value(i, "p99",df[f].dropna(axis=0).quantile(0.99))
        i=i+1;
    print(summary_df)
```

statinfer

Part-9: Conclusion

Conclusion

- In this session we discussed some basic data reporting and graph
- Studying descriptive statistics is essential before we start our advanced modeling. It gives us an idea on variable distribution
- We also discussed drawing graphs using some useful packages in Python