→ job / career plan

a. 1st year invest grasp
b. 2nd year certifications [at least 1 certificate in 1.5 year]
c. 3rd year jump

→ Exception handling

→ Error vs. Exception vs. Exception handling

a. Error: some issue in logic or input or hardware that cannot allow normal flow anymore
b. Exception: report of error
c. Exception handling: Process of either continuing alternate flow or safe close in presence of error

→ blocks in Error handling

a. try: [mandatory block] code where error is possible
b. except Exception: [mandatory block]
c. else: to keep the code which should run only if no error is reported
d. finally: to keep code that should execute in any situation,

→ Normal flow: try + else + finally
→ Error flow: try(partially) + exception + finally
→ mandatory blocks are try & except
→ to trap which error may occur, one may write
`except Exception as arg:`
→ To create custom exception, we would use super class '`Exception`', code '`__init__(self)`' method to accept data, and '`__str__(self)`' method to return message, and to raise exception use keyword '`raise`'
→ while creating exceptions, first specify specific exceptions, then specify general exceptions

→ Pandas

a. Python library used for working with data sets
b. Has functions for analyzing, cleaning, exploring & manipulating data
c. Pandas refer to “Panel data” & “Python Data Analysis”
d. Created by Wes McKinney in 2008
e. Used to clean data sets,

→ series: 1D data, list/serial data; data frames:2D data, ; panel: 3d
→ Database is collection of tables
→ pandas support three basic data structures:

a. Series: 1D data
b. Data frames: 2D data, multiple fields and its data
c. Panel: 3D data, data frames with time or some other dimension

→ in order to work with pandas, you’ve to first import pandas using
`import pandas as pd`
→ series:

a. series is created from a list using `s=pd.Series(list[])`
b. Series can be accessed by index `s[3]`
c. But series does not support negative index
d. Multiple general operations can be performed on series as `len(s), sum(s) , max(s) , min(s)`
e. Series also support slicing as `s[0:4]`

→ data-frame:

a. To upload csv file to colab you to import io library using
   `import io`
b. And then create data frame using
   `df=pd.read_csv(io.BytesIO(uploaded['mycsv.csv']))`
c. Then print data-frame using
   `print(df)`
d.
e. sa

→ series supports scalar operations, addition, multiplication, division
→ series supports only positive index as a list, but does not support negative index
→ CSV file: it is alternate to a structured table, that takes data as a record kept in a one record per line, separated by commas
→ in order to access records from dataframe index wise,we use ‘`df.loc[4]`’
→ slicing can be directly applied on data frame
→ one can use ‘`df.sort_values()`’
→ all deletion can be performed using ‘`drop`’ command
→ one can also delete specific records by specifying key-value to ‘`drop`’ command
→ common methods in pandas:

a. df.describe()
b. df.isnull()
c. df.info()

d. df.shape()
e. df.unique()
f. df.sample()
g. df.sum()
h. df.fillna()
i. df.value_counts()
j. df.loc()
k. df.iloc()
l. df.concat()
m. df.count()

→ Data Visualization

→ data visualization types:

a. Comparison
b. Distribution
c. Composition
d. Relationship

→ for Plotting, we'll use

```
import matplotlib.pyplot as plt
```

→ for plotting a graph, we need to set 'x' & 'y' values, and pass it to

```
plt.plot(x, y)
```

→ in order to set label and title for x, we use

```
plt.title("graph title")
plt.xlabel("x wala")
plt.ylabel("y wala")
```

→ `plt.his()` for histogram

→ `plt.bar()` for bar-graph

→ plotting a graph from using data-frame

a. We'll need '`pandas`' and '`matplotlib.pyplot`' both
b. We need to provide columns (x, y) from data-frame to graph plot

→ Numpy

a. Numpy is specially used for matrix manipulation
b. To use numpy, we'll use
   `import numpy as np`
c. In 1D , it is taken as 'n' rows
d. `.ndim` returns dimension of given array, If square or rectangular in case of 1D, 2D, 3D , else only 1D
e. If square or rectangular, `m.size` returns number of elements otherwise returns number of rows
f. By default `len(m)` returns number of rows
g. when given with row index, it'll return no of columns, e.g. `len(m[0])` will return no of columns

→ Common methods in numpy:

a. `import numpy as np`
b. `m=np.array([[10, 20, 30], [12, 23, 34, 45]])`
c. `m.shape` # prints rows, cols only if square or rectangular manner, else only number of rows; prints only number of cols in case of 1D matrix
d. `m.size` # prints number of elements in numpy matrix/array only if square/rectangle manner otherwise only no of rows
e.

→ randomization

a. To use random, one must import random class by writing
   `from numpy import random`
b. This class is known for generating integer & fractional random numbers
c. It has three basic formats:
   i. For random one integer value between 0..n
      `random.randint(6)` # generates only one number but up-to '6'
   ii. For a list of a number of random integer values between 0..n
      `random.randint(6, size=6)` # generates six values
      `[2 4 0 5 2 0]`
   iii. For a matrix(a, b) of random integer values between 0..n
      `random.randint(100, size(6, 6)`

```
[[83 46 17 20 44 81]
 [56 15 33 27 91 39]
 [47 71  3  5  6 87]
 [63 84 30 75 40  8]
 [36 73 93  7 41 54]
 [82 38 19  2 59 31]]
```

d. `np.sort()` will sort contents of array row-wise
e. `np.sort(m1, axis=0)` will sort contents of array row-wise, for column-wise, put axis=1
f.

Module End exam-plan

1. Section 1 (2 questions - 10 marks)
   For, while, if..elif..else etc and other basics
   Without using inbuilt functions
2. Section 2 (1 question - 15 marks)
   functions
3. Section 3 (1 question - 15 marks)
   Pandas / numpy

Note: CSV file will be provided

MCQ exam– 30 min 0.5hr – 40 marks
Main Exam– 90 min 1.5hr – 20 marks

Slot 1 1400-1600
Slot 2 1610-1810