# PYTHON

## DAY 1

Getting started with Python

IPython – python interpretor

print(5*2)

pri + click tab [Tab completion]
becomes print

abs(-10.5)
10.5

round ? # shows the documentation of round.

round (2.48)
O/P: 2.0

round (2.484, 2)
O/P: 2.48 ↑ upto 2 decimal digits

cancel the command ← Ctrl + C.

Using plot command interactively

% pylab
↑
provides plotting functionality

Page No.
Date

Linspace

↑

returns evenly spaced nos between start & stop

eg: linspace (1, 10, 5)

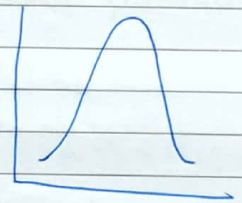
array ([1. , 3.25, 5.5, 7.75, 10. ])

t = linspace (-pi, pi, 100)

len (t)

O/P : 100

plot (t, cos(t))



clf () → clears the plots

helps to avoid overlapping of plots.

Embellish a plot

plot (x, sin(x), 'r')

↑ colour

plot (x, cos(x), linewidth = 2)

plot (x, cos(x), '.')

title (r" Parabolic function $-x^2+4x-5$ ")

treats it as a raw string

xlabel ("x axis")

ylabel ("y axis")

(-2,1) → local maxima
annotate ("local maxima", xy=(2,-1))

xlim() ← returns x axis limits
ylim() ← -11- y axis limits

To set limits
xlim(-4,5)
ylim(-15,2)

Saving Plots.

savefig('sine.png')
savefig('sine.eps')
pwd ← current directory

Multiple Plots

legend(['sin(x)', 'sin(y*)'])

SubPlots

subplot(2,1,1)
- 2 → no of rows of subplot to be created
- 1 → no of columns of subplot to be created
- 1 → serial no of subplot

Suppose you want to plot 2 diff plots in same fig

→ subplot

subplot (2, 1, 1)

subplot (2, 1, 2)

Exercise $P \propto \frac{1}{V}$ $P \propto T$

$P = \frac{nRT}{V}$ $P = \frac{nR}{V} T$

Additional features of IPython

% history

↑

to check full history of commands.

title ("$x and x^2 $")

O/P : x and $x^2$

% history 5 # 5th command

% history -n 5-10 # Prints 5th to 10th cmds.

↑ numbers them

% Save plot_script.py 2 4-7

% run -i plot_script.py

Page No.
Date

# DAY 2.

## Loading data from files

```
cat primes.txt
prime = loadtxt("primes.txt")
print(prime)
```

O/P: [ ] ← All data in prime

```
L, T = loadtxt("primes.txt", unpack=True)
```

↑ Divides columns into separate sequences.

```
L, T = loadtxt("pend.txt", unpack=True,
               delimiter=";")
print(L)
print(T)
```

## Plotting Data

```
l = [. . . .]
t = [. . . .]
tsquare = square(t)
plot(L, tsquare, 'o')
errorbar(tsquare, xerror=delta-l, yerr=delta-t,
         fmt='bo')
```

↑ to represent error.

To plot with red plus → 'r+'

Page No.
Date

# Other Types of Plots

## Scatter plot

```
year, profit = loadtxt('company.txt', unpack=True)
```

Syntax: scatter(x, y)

```
scatter(year, profit)
scatter(year, profit, marker='d')
```

'd' → diamond shaped

## Log-log graph

```
loglog(x, y)
```

$y = 5x^3$

$x = 1, 20$

## Plotting charts

### Pie chart

```
pie(values, labels = labels)
```

values → data to be plotted

labels → label for each wedge.

```
year, profit = loadtxt(r'C\...\...txt',
                       unpack=True)
```

For colors

```
pie(values, labels = lab, colors = ('b', 'c', ...))
```

Page No.
Date

Bar chart
bar(x,y)

bar(year,profit)

To not fill colour
bar(year,profit,fill=False, hatch='/')

↑ fills bar with this

matplotlib info
matplotlib.sourceforge.net/contents.html

Getting started with Lists

type ← returns type of variable

x = [ ] // empty list
x = [1,2,3]

x[0]
O/P: 1

x[-1]
↑ gives last element.

list = [[1],[3,4]]
list[1][1]
O/P: 4
list[1]
O/P [3,4]

len() → length of list

append() → adds element to end of list
var.append(ele)
↑ list name

Removing elements
① del keyword
del list[index]
② remove
list.remove[element]

Getting started with for

for <var> in sequence:
<stmt 1>
:
<stmt n>

eg: list=[4,9,16,25]
for i in list:
print(~~num~~ i ** 0.5)

O/P: 2
3
4
5

to use range:
range(start, stop, step value)
eg: range(1,20,2)
range(20)

Page No.
Date

# Getting started with strings

'String'
"Str"
"""Strs"""
} all are strings

```
x = "Hi"
y = "There"
z = x + " " + y + "!"
z
O/P : Hi There !

x * 5
"HiHiHiHiHi
```

```
a = "Hello"
a[0] → 'H'
a[3] → 'l'
a[-1] → 'o'

a[0] = "B"  # you get error
```

Why?
Strings cannot be modified after creation

Join: str.join(sequence)
str.split(" ")

```
s = "-"
seq = ('a', 'b', 'c')

s.join(seq)
a-b-c
```

Page No.
Date

Getting started with files

Open a file

file_obj = open (filename, mode)

use location here

r — read mode
w — write mode
a — append mode
r+ → read + write

what mode file will open in

```
f = open ("c:\...\...txt")
pend = f.read() // to read all values of the file opened

lst = pend.splitlines() // get values in separate lines

filename.close()

for i in filename:
    print(i)

To store in list:
for i in filename:
    lst.append(i)
```

Page No.
Date

Parsing Data

↑

reading data in text form

split() → divides string

Syntax : str.split( arg) ← string tokenizing

```
x = "a ; bc ; d"
x.split(" ; ")
O/P : ['a', 'bc', 'd']
```

strip() → removes all leading & trailing whitespaces

```
s = "   H  I   "
s.strip()
O/P : "H I"
```

```
m = 5.5
m1 = float(m)
```

Convert to int

```
m = "5.5"
f = float(m)
5.55
n = int(f)
5
```

Page No.
Date

# Statistics

Install numpy // numerical python
import numpy as np

To get data as array → loadtxt()

```
L = loadtxt("A.txt", usecols = (3,4,5,6,7),
                delimiter = ';')
```

↑ loads the text

↑ cols to be used from the text

L.shape // gives you (row, col)
↑ tuple

To find mean

```
L[0]
total = sum(L[0])
total / len(L[0])  // avg
```

OR

```
np.mean(L[0])
np.mean(L,0) ← mean of marks
np.median(L,0)

np.std(L[:,0])
```

| A | B | C |
|---|---|---|
| a | 1 | 1 |
| b | 2 | 1 |
| c | 3 | 1 |

→ sum = np.sum(L, 0)
mean = np.mean(L,0)
std(L,0)

Page No.
Date

Getting started with arrays.

```
s1 = np.array([1,2,3])
s1
o/p: array([1,2,3,4])

2d array
s = ([[1,2,3],[4,5,6]])
s

array([[1,2,3],
       [4,5,6]])

arange (to arrange elements in array)
ar = np.arange(1,9)
ar
array([1,2,3,4,5,6,7,8])

reshape
ar.reshape(2,4)
ar
array([[1,2,3,4],
       [5,6,7,8]])

ar.shape
(8,)

np.identity(2)
([[1,0],
  [0,1]])
```

tuple → ordered list of elements

```
np.zeros ((4,5))

array ([[0 0 0 0 0],
        [0 0 0 0 0],
        [0 0 0 0 0],
        [0 0 0 0 0]])

np.zeros_like ((2,2))
  array ([0,0])

np.ones ((2,2))
  array ([[1,1,]
          [1,1]])

np.ones_like ((2,2))
  array ([1,1])

  a1 = ([1,2,3])
  a1 * 2
array([2,4,6])
  a1 + 2
array ([3,4,5])

  a1 += 2
  a1
  array ([3,4,5])

  np.array
a = ([1,2,3])
a1 = ([2,2,2])
a + a1 = ([3,4,5])
a * a1 = ([2,4,6])
```

Page No.
Date

## Accessing parts of arrays.

A[2]
O/P: 1

For 2d arrays
C[1][3]
O/P: 9

Change value in array
A[1] = 14

Summary for 2d array
C[1][3] = 144.

In 2D array
To change complete row
C[2] = 5

```
([1 2 3 4 5],
 [6 7 8 9 10],
 [5 5 5 5 5] ])
```

Striding

To get odd rows only
C[0:5:2, 0:5, 2]
start, stop, steps

Page No.
Date

Image manipulation using Arrays

75
3
225

imread

```
img = plt.imread("s.png")
plt.imshow(img)
```

img → shows array

```
img.shape  //(300, 300)

plt.show()
```

Basic Matrix operations.

```
from numpy import matrix
m1 = matrix([1,2,3,4])
m1.shape
   (1,4)
from numpy import asmatrix, arange
m = asmatrix(arange(1,9).reshape(2,4))

m
([[1,2,3,4],
  [5,6,7,8]])
m3 = ([[5,6,7,8],
       [9,10,12,12]])
m+m3
([[6,8,10,12],
  [14,16,18,20]])
m-m3
([[-4 -4 -4 -4],
  [4  4  4  4]])
```

m * 6.5
(all nos multiplied by 6.5)

m4 = as matrix (arange (1,9). reshape (4,2))
(m2 * m4)
2x2

Transpose
print (m4.T)

Determinant
numpy. ~~to~~ linalg
det ( )

eg: det ([[1,2,3],[4,5,6],[7,8,9]])

Inverse
m = inv (- X. )
m1 = matrix ([x])
Identity matrix : eye ( )

from numpy import eye, allclose

allclose (m * m1 ), eye(3))
① ②
True

This checks whether the ① & ② are same.

Page No.
Date

Eigen values
eig (A)[0] ← eigen values
eig (A)[1] ← " vector
eigvals (A) ← eigen values

numpy.linalg → eig, eigvals

m = asmatrix (diag ((1,2,3)))

val = eig (m)[0]
→ array ([1, 2, 3])

eig (m)[1]
matrix ([[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])

## Advanced Matrix Operations

flatten() → returns 1-D array from multi-D array

eg: a = asmatrix (arange (1,10). reshape (3,3))
a.flatten()
matrix ([1,2,3,4,5,6,7,8,9])

Frobenius : Square root of the sum of the absolute squares of its elements.

norm(mm) ← numpy.linalg

Page No.
Date

Infinity norm
Max value of sum of the absolute value of elements in each row.

```
im
from numpy import inf
norm (mm, ord = inf).
```

★ Singular Value Decomposition
— Factorization of real or complex matrix

```
from numpy.linalg import svd
m1
U, sigma, V_conjugate = svd(m1)
```

↑ only diag elements of matrix

allclose → U * sigma * V_conjugate = m1

Page No.
Date

## Least Square fit

$T^2 = m * L + c$

slope of line — intercept of line

lstq ( ).

tsq. = A * p

(1-d array) in — nx2 — 1-d array size 2

$p = \begin{bmatrix} m \\ c \end{bmatrix}$

1st step:

inter = array ((L, ones_like(L)))

↓

2nd inter.T (transpose)

result = lstq (A, tsq, rcond = None)

p = result [0]

m, c = p

fit = m * L + c

plt.plot (L, tsq, 'bo')

plt.plot (L, fit, 'r')

Page No.
Date

# Basic datatypes & operators

Datatypes
Numbers, Boolean

Operators: { +, -, *, /, %.
Boolean

Numbers
int, float, complex

c.real  c.img
c = 3.4 + 4.6j
abs(c) → 5.72...

type(c)
complex.

t = True
f = Not t
False

t or t → True
f and t → False

Paranthesis matters

% → gives remainder

7**8 → $7^8$

a = 73
a** = 34
a = 2482

Page No.
Date

Sequence datatypes → elements are kept in a sequential order

list ⟺ list to tuple
string ⟺ string to list
tuple

list → [1, 2, 3] //can mutate.

To make ' " ' appear in string, use triple quotes
eg: m = """ 'Hello there. I'm Aadi.' """
'Hello there. I'm Aadi.'

tuple → (1, 2, 3)

tup[1] = 2 //cannot mutate

L = [1, 2, 3] L1 = [4, 5]
L + L1
[1, 2, 3, 4, 5]

t = ("2g",)
↑ necessary to make it a tuple.

To check for element in list/tuple [Ans: True/False]
3 in l
True
4 in tup
False

max(l) → 3 //MAX VALUE IN LIST
min(tup) = 1 //MIN -11- -11- TUPLE

String s = "Hi" // IMMUTABLE

t = tuple(s) // String to tuple
('H', 'i')

Tuple to list
ll = list(tuple_list)

String to list
s = "q,w,e,r"
l = s.split(',')
['q', 'w', 'e', 'r']

','.join(l)
'q,w,e,r,t,y'

Input - Output

a = "Hi"
a
'Hi'
print(a)
Hi

b = "A line \n new"
b
'A line \n new'
print(b)
A line
new

%. → format
%.d → int format
%.s → string format
%.f → float format

print("Hi %3.4f" %.x)
↑ 4 decimal places.

```
print ("Hello", end=' ')
print ("W")
  Hello W

ip = input()
ghg
ip
'ghg'
```

## Conditional statements

```
if
if i == 1 :
    print ("True")
else:
    print ("False")

if <condn> :
    stmt
elif <condn> :
    stmt
else:
    stmt
```

Ternary operator

```
a = 1
b = 2
num = a if a > b else b
op num
  o/p : 2
```

Page No.
Date

Pass statement
→ nothing happens when it executes

eg: a=11
```
if a %.2==0:
    print("E")
else:
    pass
```

## Loops

while loop

```
while <condn>:
    -
    -
```

```
for n in range(1,10,2):
    print(n)
while (i<10):
    print(i)
    i+=2
```

break

```
for i in 'python':
    if i=='h':
        break
    else
        print(i)
```

p
y
t

continue

```
for i in 'python':
    if i=='h':
        continue
    else
        print(i)
```

p
y
t
o
n

Page No.
Date

## Manipulating lists

Slicing: p[start:stop]

↑ This element isn't included.

Striding

ll = [1,2,3,4,5,6]

Odd nos:

ll[1:6:2]

O/P : [1,3,5]

ll[:3]

[1,2,3]

ll[2:]

[3,4,5,6]

ll.sort()

sorted()

b = sorted(a)

r = [1,2,3,4]

r.reverse() / r[::-1]

r

[4,3,2,1]

Sort & reverse at same time

sorted(marks, reverse = True)

Page No.
Date

## Manipulating strings

### Slicing

name [start : stop]
or
name [start : stop : step]

### Reverse string

s[: : -1]

```
s = 'dad'
s1 = s.upper() ⟶ DAD
s1.lower() ⟶ dad
```

### Replace

```
em = "amarica"
em.replace("a","e")
→ emerice
```

```
ee = ['a', 'c', 'e']
ee = ','.join(e)
ee
→ 'a,c,e'
```

### Getting started with tuples

```
a=5
b=7
swap
```

```
a, b = b, a
a= 7
b = 5
```

Page No.
Date

**Dictionaries : { }**

— unordered collection of items . has key-value pairs

key: unique; value: may be unique / may not be

```
s = {'a': 1, 'b': 2, 'c': 3}
```

o/p

```
s['a'] = 1
```

To add key:value

```
s['d'] = 4
```

```
s
```

o/p : {'a': 1, . . . 'd': 4}

To delete

```
del s['c']
```

o/p : s

{'a': 1, 'b': 2, 'd': 4}

'a' in s

True

```
keys() // list of keys
values() // list of values
```

```
s.keys()
dict_keys (['a', 'b', 'c', 'd'])
s.values()
dict_values ([1, 2, 3, 4])
```

s.items()

dict_items([('a', 1), ....])

## Sets in Python

unordered collection of unique elements

empty set = set()

a1 = set(a) // a1 = [1,2,3,4]

a = [1,2,3,4,1]

f.add(13)

p.remove(13)

Union { f | p
f.union(p)

Inter-section { f & p
f.intersection(p)

f - p
f.difference(p)

f ^ p
f.symmetric_difference(p) // f|p - (f&p)

a = set([1,2,3])

b = set([1,2])

b.issubset(a) True

a.issuperset(b)
True

len(a) // a

Slicing & striding not valid in sets

→ Print duplicates

```
m = [20,23,20,21,22,23]
m1 = set(m)
for n in m1:
    m.remove(n)

dup = set(m)
print(dup)
```

Getting started with functions

Function

→ helps reuse of code, code redundancy

```
def func_name(parameters):
    """doc"""
    stmts
    return <exp>
```

NOT necessary to have return.

```
def square(x):
    y = x ** x
    return y

square(3)
O/P: 9
```

```
def g():
    print("Hi")
g()
O/P: Hi
```

# Advanced features of functions

## Default arguments

eg
```
def welcome (greet, name = "World")
    return (greet, name)
```

```
welcome ("Hi", "Aadi")
Hi Aadi

welcome ("Hi")
Hi World
```

default args should come at the end

## Keyword Arguments

```
marks (f = 11, b = 22, t = 33)
11
22
33
```

```
def f(*names):
    print (names)
```

*names → arbitary no of positional arguments

```
def f(**attributes):
    print (attributes)
```

**attributes → arbitary no of keyword arguments

Page No.
Date

Visit : docs.python.org/3/library/functions.html.

abs()
any()
dir()
help()

## Using Python Modules

modules contain Python defns & stmts

python3 hello.py  ./run hello.py

only works on Linux

Import functions
Should use names than just *

Math: math, random
Internet access: urllib2, smtplib
System & Cmd line args : sys
OS interface : os
Regular expression : re
Compression : gzip, zipfile, tarfile

## Writing Python scripts

For Windows, to run python:
./run "fullpath\x.py"

importsys
sys.path

You can import directly if path is the same
import gcd_script

how to call?
god_script.func_name(args)

Test code should only be executed when we run the python script independently.

To execute test code when module is imported from other scripts, we use __name__

Every python file can be run in 2 ways.

Script  module

## Testing and Debugging

Software testing → evaluate functionality of a software or program.

Meaningful names are important

1. Space indentation
2. 79 word limit on a limit
3. Use docstring
4. Use whitespaces

## Handling errors & exceptions

Python < Syntax errors
Exception

1/0

ZeroDivisionError: division by zero

Exception ← It is reported by the programming language

Page No.
Date

```
try:
    a = input ("Enter no.")
    num=int(a)
    print(num)
except ValueError:
    print("Wrong input")
```

a Enter no: ac

o/p : Wrong input

∴ debug ← Shows where the problem is

Page No.
Date

# Linear Regression using OLSM in python

Linear Regression is a linear model.
multiple I/P variable & 1 o/p variable
$(x_1, x_2 .. x_n)$ $(y)$

$$Y = \beta_0 + \beta_1 x$$

$\beta_0$ — Intercept, $\beta_1$ — Slope

Approaches to find coefficients
- OLSM
- Gradient Descent

## OLSM

$$\beta_1 = \frac{\Sigma(x_i - \bar{x})(y_i - \bar{y})}{\Sigma(x_i - \bar{x})^2}$$

$$\beta_0 = \bar{y} - \beta_1 * x$$

Accuracy of the model:
Root mean squared error

$$= \sqrt{\frac{1}{n}\Sigma(y_i - \hat{y}_i)^2}$$

$y_i$ — actual value, $\hat{y}_i$ — predicted value

$R^2$ → accuracy of model $= \frac{\Sigma(\hat{y}_i - \bar{y})^2}{\Sigma(y_i - \bar{y})^2} = \frac{SSR}{SST}$

SSR — Sum of squared residuals
SST — Total sum of sq. (*)

Page No.
Date

Dependencies needed
—numpy —pandas —matplotlib

Simple Linear Regression Steps :
1. Load data using pandas
2. Calculate $\beta_0$ & $\beta_1$
3. Calculate $Y_{predicted} = \beta_0 + \beta_1 * x$
4. Calc. $R^2$
5. Plot original values of y vs x & estimated values of $Y_{predicted}$ vs x

Multivariable LR model using gradient descent

Multiple independent x related to one dependent variable y.

Model Representation

$$f(x_1, x_2 \cdots x_n) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

$\theta_0$ : bias coefficient
$\theta_1, \theta_2, \ldots, \theta_n$ : coefficients

Cost func$^n$ : quantifies diff. bet$^n$ predicted & expect value.

$$J(\theta) = \frac{1}{2m} \sum_{i}^{m} (h_\theta(x_i) - y_i)^2$$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i}^{m} (h_\theta(x_i) - y_i)\, x_{ij}$$

$x_{ij}$ : $j^{th}$ feature of $i^{th}$ observation