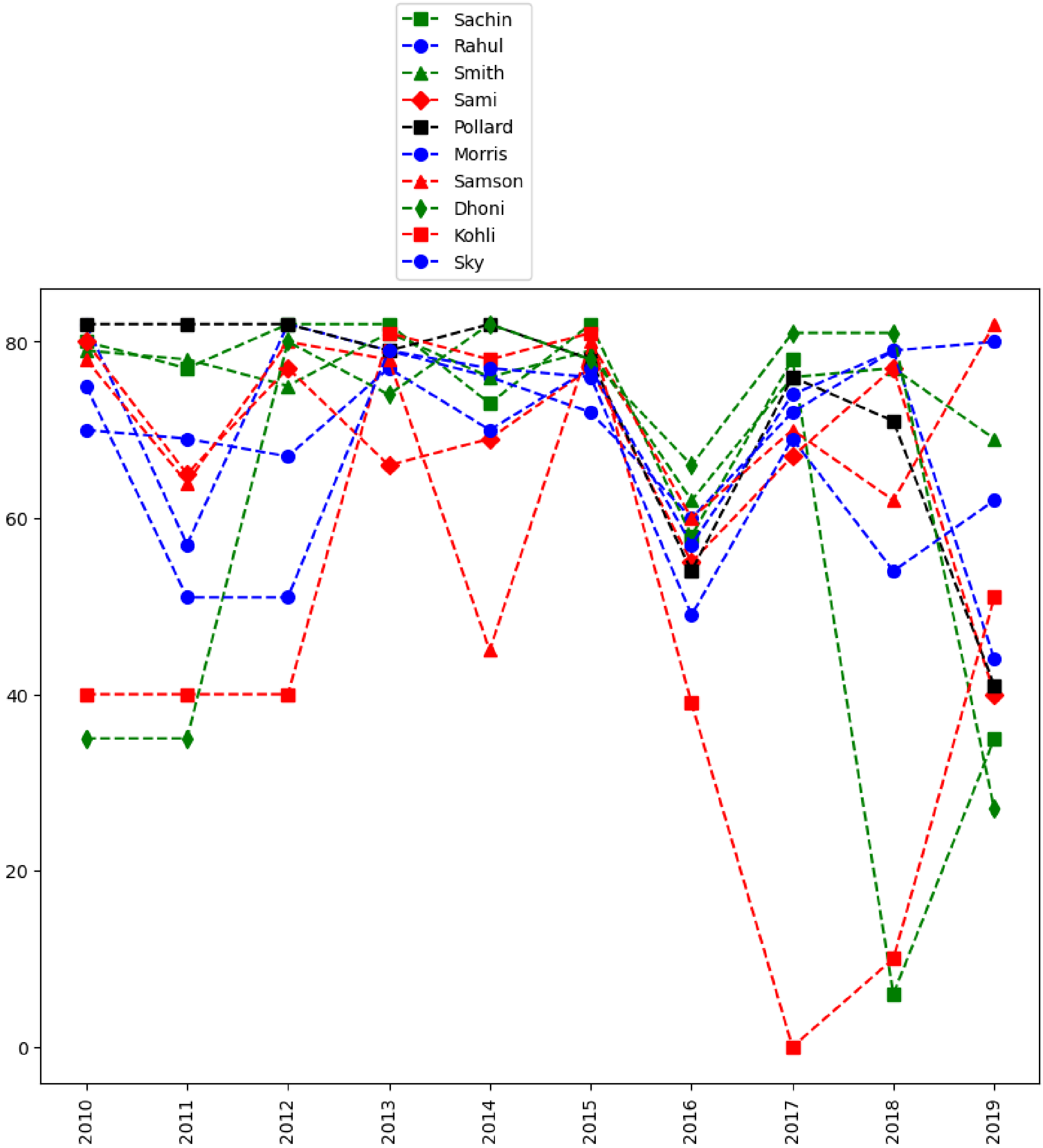

In this section we learned - 1>Matrices 2>Building matrices - np.reshape 3>Dictionaried in python (order doesnot mater) (keys & values) 4>visualizaing using pyplot 5>Basket ball analysis

Basic PYTHON PROGRAMMING LANGUAGE

In [ ]:

```
Basic python today -- i will share these file

1- using third variable
2- using operate
3- using binary number system
4- using xor operator
5- a,b = b, a ( try to use this one for swap concept)

bitwise operator -->
---
```

```
complement operator (~)
and &
or |
xor ^
left shift << ( gain the bit )
right shift >> (loose the bit)

https://docs.python.org/3.11/tutorial/
# comment inline

python has no constatn only variable

what is the benefit of import math as m

input() -- alwasy valuew as string
a + b = ab
5 + 6 = 56

basic python we are completed todaYs
```

Python Became the Best Programming Language & fastest programming language. Python is used in Machine Learning, Data Science, Big Data, Web Development, Scripting. we will learn pyton from start to end || basic to expert. if you are not done programm then th0at is totally fine. I will explain from starting from scratch. python software - pycharm || vs code || jupyter || spyder

PYTHON INTERPRETTER

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

**PYTHON INTERPRETER --> What is Python interpreter? A python interpreter is a computer program that converts each high-level program statement into machine code. An interpreter translates the command that you write out into code that the computer can understand

**PYTHON INTERPRETER EXAMPLE --> You write your Python code in a text file with a name like hello.py . How does that code Run? There is program installed on your computer named "python3" or "python", and its job is looking at and running your Python code. This type of program is called an "interpreter".

**IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) =>

- using IDE - one can write code, run the code, debug the code
- IDE takes care of interpreting the Python code, running python scripts, building executables, and debugging the applications.
- An IDE enables programmers to combine the different aspects of writing a computer program.
- if you wnated to be python developer only then you need to install (IDE -- PYCHARM)

PYTHON INTERPRETER & COMPILER

Both compilers and interpreters are used to convert a program written in a high-level language into machine code understood by computers. Interpreter -->

- Translates program one statement at a time
- Interpreter run every line item
- Execut the single, partial line of code
- Easy for programming

Compiler -->

- Scans the entire program and translates it as a whole into machine code.
- No execution if an error occurs
- you can not fix the bug (debug) li

*Is Python an interpreter or compiler? Python is an interpreted language, which means the source code of a Python program is converted into bytecode that is then executed by the Python virtual machine. Python is different from major compiled languages, such as C and C + +, as Python code is not required to be built and linked like code for these languages.

**How to create python environment variable 1- cmd - python (if it not works) 2- find the location where the python is installed -- > C:\Users\kdata\AppData\Local\Programs\Python\Python311\Scripts 3- system -- env - environment variable screen will pop up 4- select on system variable - click on path - create New 5- C:\Users\kdata\AppData\Local\Programs\Python\Python311 6- env - sys variable - path - new - C:\Users\kdata\AppData\Local\Programs\Python\Python311\Scripts 7- cmd - type python -version 8- successfully python install in cmd
by line

ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

```
In [5]:  1 + 1 # ADDITION
```

```
Out[5]:  2
```

```
In [7]:  2-1
```

```
Out[7]:  1
```

```
In [9]:  3*4
```

```
Out[9]:  12

In [11]: 8 / 4 # Division

Out[11]: 2.0

In [13]: 8 / 5 #float division

Out[13]: 1.6

In [15]: 8/4 ## float division

Out[15]: 2.0

In [17]: 8 // 4 #integer divisio

Out[17]: 2

In [19]: 8 + 9 - 7

Out[19]: 10

In [21]: 8 + 9 - 7

Out[21]: 10

In [23]: 5 + 5 * 5

Out[23]: 30

In [25]: (5 + 5) * 5 # BODMAS (Bracket || Oders || Divide || Multiply || Add || Substact)

Out[25]: 50

In [27]: 2 * 2 * 2 * 2 * 2 # exponentaion

Out[27]: 32

In [29]: 2 ** 5

Out[29]: 32

In [31]: 15 / 3

Out[31]: 5.0

In [33]: 10 // 3

Out[33]: 3

In [35]: 14 % 2 # Modulus

Out[35]: 0

In [37]: 15 %% 2
```

```
  Cell In[37], line 1
    15 %% 2
        ^
SyntaxError: invalid syntax
```

In [39]:

```
a,b,c,d,e = 15, 7.8, 'nit', 8+9j, True

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
15
7.8
nit
(8+9j)
True
```

In [41]:

```
print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'complex'>
<class 'bool'>
```

In [43]:

```
type(c)
```

Out[43]:

```
str
```

In [45]:

```
- So far we code with numbers(integer)
- Lets work with string
```

```
  Cell In[45], line 1
    - So far we code with numbers(integer)
         ^
SyntaxError: invalid syntax
```

In [47]:

```
'Naresh IT'
```

Out[47]:

```
'Naresh IT'
```

python inbuild function - print & you need to pass the parameter in print()

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

In [50]:

```
print('naresh it')
```

```
naresh it
```

In [52]:

```
"max it technology"
```

Out[52]:

```
'max it technology'
```

In [54]:
```
s1 = 'naresh it technology'
s1
```

Out[54]:
```
'naresh it technology'
```

In [56]:
```
a = 2
b = 3
a + b
```

Out[56]:
```
5
```

In [58]:
```
c = a + b
c
```

Out[58]:
```
5
```

In [60]:
```
a = 3
b = 'hi'

type(b)
```

Out[60]:
```
str
```

In [62]:
```
a + b
```

```
-----------------------------------------------------------------
TypeError                               Traceback (most recent call last)
Cell In[62], line 1
----> 1 a + b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [64]:
```
print('naresh it's 'Technology')
```

```
  Cell In[64], line 1
    print('naresh it's 'Technology')
          ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

In [66]:
```
print('naresh it\'s"Technology"') #\ has some special meaning to ignore the erro
```

```
naresh it's"Technology"
```

In [68]:
```
print('naresh it',  'Technology')
```

```
naresh it Technology
```

In [70]:
```
print("naresh it',  'Technology")
```

```
naresh it',  'Technology
```

In [72]:
```
# print the nit 2 times
'nit' + ' nit'
```

Out[72]:
```
'nit nit'
```

In [74]:
```
'nit'  ' nit'
```

Out[74]:
```
'nit nit'
```

In [76]:
```
#5 time print
5 * 'nit'
```

Out[76]:
```
'nitnitnitnitnit'
```

In [78]:
```
5 *' nit ' # soace between words
```

Out[78]:
```
' nit  nit  nit  nit  nit '
```

In [80]:
```
print('c:\nit') #\n -- new line
```

```
c:
it
```

In [82]:
```
print(r'c:\nit') #raw string
```

```
c:\nit
```

variable || identifier || object

In [85]:
```
2
```

Out[85]:
```
2
```

In [87]:
```
x = 2 #x is variable/identifier/objec, 2 is the value
x
```

Out[87]:
```
2
```

In [89]:
```
x + 3
```

Out[89]:
```
5
```

In [91]:
```
y = 3
y
```

Out[91]:
```
3
```

In [93]:
```
x + y
```

Out[93]:
```
5
```

In [95]:
```
x = 9
x
```

Out[95]:
```
9
```

In [97]:
```
x + y
```

Out[97]:
```
12
```

In [99]:
```
x + 10
```

Out[99]:
```
19
```

```
In [101…  y
Out[101…  3

In [ ]:

In [ ]:

In [ ]:

In [111…  # string variable
          name = 'mit'

In [113…  name
Out[113…  'mit'

In [115…  name + 'technology'
Out[115…  'mittechnology'

In [117…  name + ' technology'
Out[117…  'mit technology'

In [119…  name 'technology'

  Cell In[119], line 1
    name 'technology'
         ^
SyntaxError: invalid syntax

In [121…  name
Out[121…  'mit'

In [123…  len(name)
Out[123…  3

In [125…  name[0] #python index begins with 0
Out[125…  'm'

In [127…  name[5]

----------------------------------------------------------------
IndexError                               Traceback (most recent call last)
Cell In[127], line 1
----> 1 name[5]

IndexError: string index out of range

In [129…  name[7]
```

```
---------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[129], line 1
----> 1 name[7]

IndexError: string index out of range
```

In [131…
```
name[-1]
```

Out[131…
```
't'
```

In [133…
```
name[-2]
```

Out[133…
```
'i'
```

In [135…
```
name[-6]
```

```
---------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[135], line 1
----> 1 name[-6]

IndexError: string index out of range
```

slicing

In [137…
```
name
```

Out[137…
```
'mit'
```

In [139…
```
name[0:1] #to print 2 character
```

Out[139…
```
'm'
```

In [141…
```
name[0:2]
```

Out[141…
```
'mi'
```

In [143…
```
name[1:4]
```

Out[143…
```
'it'
```

In [145…
```
name[1:]
```

Out[145…
```
'it'
```

In [147…
```
name[:4]
```

Out[147…
```
'mit'
```

In [149…
```
name[3:9]
```

Out[149…
```
''
```

In [151…
```
name1 = 'fine'
```

```
name1
```

Out[151…

```
'fine'
```

In [153…

```
name1[0:1]
```

Out[153…

```
'f'
```

In [155…

```
name1[0:1] = 'd' # i want to change 1st character of naresh (n) - t
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[155], line 1
----> 1 name1[0:1] = 'd'

TypeError: 'str' object does not support item assignment
```

In [157…

```
name1
```

Out[157…

```
'fine'
```

In [159…

```
name1[0] = 'd' #strings in python are immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[159], line 1
----> 1 name1[0] = 'd'

TypeError: 'str' object does not support item assignment
```

In [161…

```
name1[1:]
```

Out[161…

```
'ine'
```

In [163…

```
'd' + name1[1:] #i want to change fine to dine
```

Out[163…

```
'dine'
```

In [165…

```
len(name1) #python inbuild function
```

Out[165…

```
4
```

List

In [167…

```
# LIST LIST LIST
nums = [10,20,30]
nums
```

Out[167…

```
[10, 20, 30]
```

In [169…

```
nums[0]
```

Out[169…

```
10
```

In [171…

```
nums[-1]
```

```
Out[171…  30

In [173…  nums[1:]

Out[173…  [20, 30]

In [175…  nums[:1]

Out[175…  [10]

In [177…  num1 = ['hi', 'hallo']

In [179…  num1

Out[179…  ['hi', 'hallo']

In [181…  num2 = ['hi', 8.9, 34] # we can assign multiple variable
          num2

Out[181…  ['hi', 8.9, 34]

In [183…  # can we have 2 list together
          num3 = [nums, num1]

In [185…  num3

Out[185…  [[10, 20, 30], ['hi', 'hallo']]

In [187…  num4 = [nums, num1, num2]

In [189…  num4

Out[189…  [[10, 20, 30], ['hi', 'hallo'], ['hi', 8.9, 34]]

In [191…  nums

Out[191…  [10, 20, 30]

In [193…  nums.append(45)

In [195…  nums

Out[195…  [10, 20, 30, 45]

In [197…  nums.remove(45)

In [201…  nums

Out[201…  [10, 30]

In [199…  nums.pop(1)

Out[199…  20

In [203…  nums
```

Out[203…

```
[10, 30]
```

In [205…

```
nums.pop() #if you dont assign the index element then it will consider by defaul
```

Out[205…

```
30
```

In [207…

```
nums
```

Out[207…

```
[10]
```

In [209…

```
num1
```

Out[209…

```
['hi', 'hallo']
```

In [211…

```
num1.insert(2,'nit') #insert the value as per index values i.e 2nd index we are
```

In [213…

```
num1
```

Out[213…

```
['hi', 'hallo', 'nit']
```

In [215…

```
num1.insert(0, 1)
```

In [217…

```
num1
```

Out[217…

```
[1, 'hi', 'hallo', 'nit']
```

In [219…

```
#if you want to delate multiple value
num2
```

Out[219…

```
['hi', 8.9, 34]
```

In [221…

```
del num2[2:]
```

In [223…

```
num2
```

Out[223…

```
['hi', 8.9]
```

In [225…

```
# if you need to add multiple values
num2.extend([29,15,20])
```

In [227…

```
num2
```

Out[227…

```
['hi', 8.9, 29, 15, 20]
```

In [229…

```
num3
```

Out[229…

```
[[10], [1, 'hi', 'hallo', 'nit']]
```

In [231…

```
num3.extend(['a', 5, 6.7])
```

In [233…

```
num3
```

Out[233…

```
[[10], [1, 'hi', 'hallo', 'nit'], 'a', 5, 6.7]
```

In [235…

```
nums
```

Out[235…
```
[10]
```

In [237…
```
min(nums) #inbuild function
```

Out[237…
```
10
```

In [239…
```
max(nums) #inbuild function
```

Out[239…
```
10
```

In [241…
```
max(nums) #inbuild function
```

Out[241…
```
10
```

In [243…
```
num1
```

Out[243…
```
[1, 'hi', 'hallo', 'nit']
```

In [245…
```
min(num1)
```

```
---------------------------------------------------------------
TypeError                             Traceback (most recent call last)
Cell In[245], line 1
----> 1 min(num1)

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [247…
```
sum(nums) #inbuild function
```

Out[247…
```
10
```

In [249…
```
nums.sort() #sort method
```

In [251…
```
nums
```

Out[251…
```
[10]
```

Tuple

In [253…
```
# TUPLE TUPLE TUPLE
tup = (15,25, 35)
tup
```

Out[253…
```
(15, 25, 35)
```

In [255…
```
tup[0]
```

Out[255…
```
15
```

In [257…
```
tup[0] = 10
```

```
---------------------------------------------------------------
TypeError                             Traceback (most recent call last)
Cell In[257], line 1
----> 1 tup[0] = 10

TypeError: 'tuple' object does not support item assignment
```

as we are unable to change any value or parameter in tuple so iteration very faster in tuple compare to list

Set

In [259…

```
# SET SET SET
S = {}
```

In [261…

```
s1 = {21,6,34,58,5}
```

In [263…

```
s1
```

Out[263…

```
{5, 6, 21, 34, 58}
```

In [265…

```
s3= {50,35,53,'nit', 53}
```

In [267…

```
s3
```

Out[267…

```
{35, 50, 53, 'nit'}
```

In [269…

```
s1[1] #as we dont have proper sequencing thats why indexing not subscriptable
```

```
---------------------------------------------------------------
TypeError                             Traceback (most recent call last)
Cell In[269], line 1
----> 1 s1[1]

TypeError: 'set' object is not subscriptable
```

DICTIONARY

In [271…

```
# DICTIONARY DICTIONARY DICTIONARY
data = {1:'apple', 2:'banana',4:'orange'}
data
```

Out[271…

```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

In [273…

```
data[4]
```

Out[273…

```
'orange'
```

In [275…

```
data[3]
```

```
---------------------------------------------------------------
KeyError                              Traceback (most recent call last)
Cell In[275], line 1
----> 1 data[3]

KeyError: 3
```

In [277…

```
data.get(2)
```

Out[277…

```
'banana'
```

In [279…

```
data.get(2)
```

Out[279…

```
'banana'
```

In [281…

```
data.get(3)
```

In [283…

```
print(data.get(3))
```

```
None
```

In [285…

```
data.get(1,'Not Fount')
```

Out[285…

```
'apple'
```

In [287…

```
data.get(3,'Not Found')
```

Out[287…

```
'Not Found'
```

In [289…

```
data[5] = 'five'
```

In [291…

```
data
```

Out[291…

```
{1: 'apple', 2: 'banana', 4: 'orange', 5: 'five'}
```

In [293…

```
del data [5]
```

In [295…

```
data
```

Out[295…

```
{1: 'apple', 2: 'banana', 4: 'orange'}
```

In [297…

```
#list in the dictionary
prog = {'python':['vscode', 'pycharm'], 'machine learning' : 'sklearn', 'datasci
```

In [299…

```
prog
```

Out[299…

```
{'python': ['vscode', 'pycharm'],
 'machine learning': 'sklearn',
 'datascience': ['jupyter', 'spyder']}
```

In [301…

```
prog['python']
```

Out[301…

```
['vscode', 'pycharm']
```

In [303…

```
prog['machine learning']
```