

Data Science and Artificial Intelligence



# Python for Data Science

## CLASSES AND MODULES



Lecture No. 02



By- Kashif Sir

Python

3 hrs

Data Structure

1 hr

Linear Algebra

3 hrs

- 1) Half an hour before every class - Revision of last class
- 2) On Sunday, revise everything whatever u have studied in the complete week.
- 3) Make proper Notes (one single note for each subject)

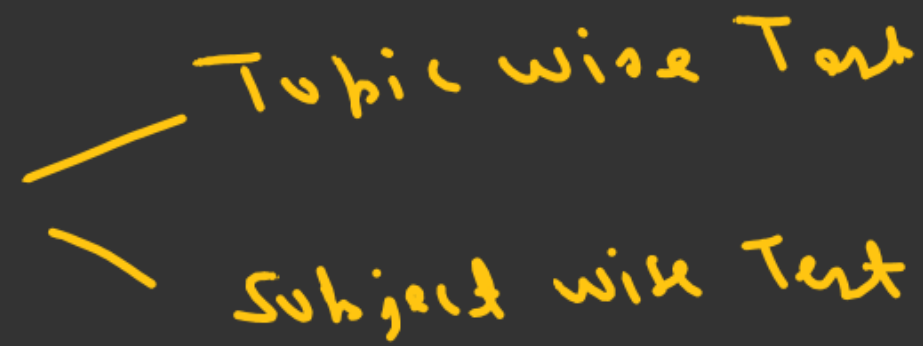
4) GATE PYQs (Topic wise)

College Time  $\Rightarrow$  (Revision, Practice)



5) When the new subject is going on: 2 days in a week  
PYQs of completed subject

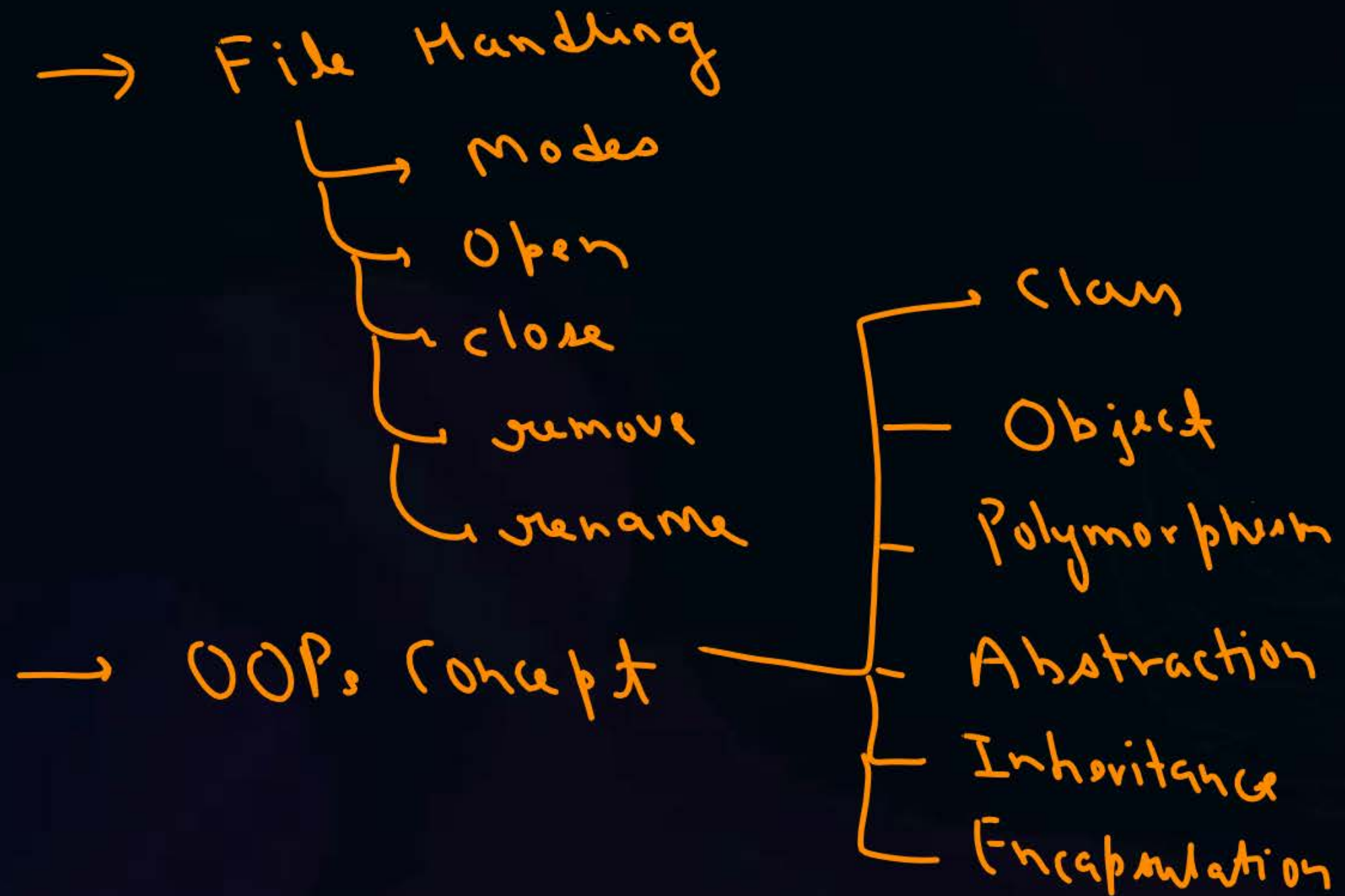
6) After finishing PYQs

7) Test Series 

- Topic wise Test
- Subject wise Test



# RECAP





# AGENDA



- Comments
- Types of Errors
- Exception Handling
- Bytecode

```
class BankAccount:  
    bankname = "NDFC"  
    def __init__(self, amount):  
        self.amount = amount  
  
    def deposit():  
  
    def withdrawal():
```





# COMMENTS



Comments are the notes that python ignores while executing the program

```
def summation(a, b):  
    return a+b
```

→ Single line comment

```
# This function returns sum.
```

→ Multiple line comments

① 

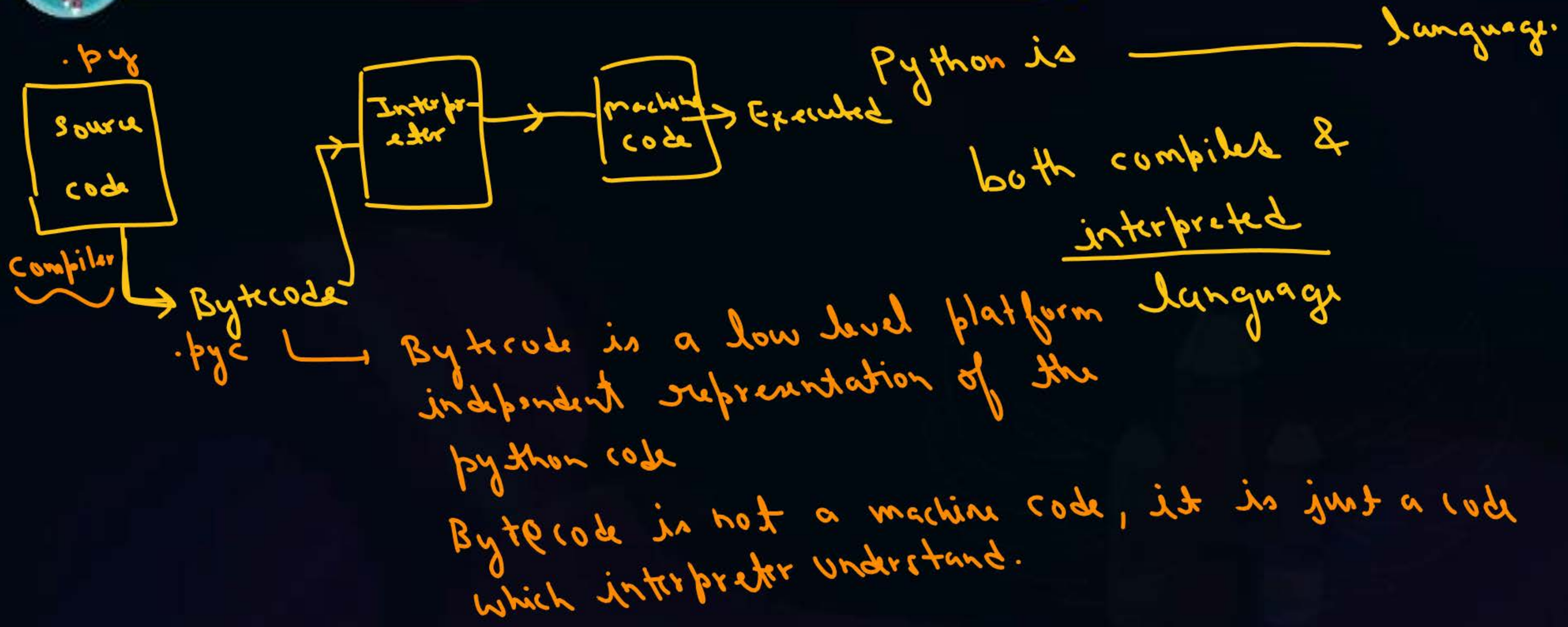
```
# The function return  
# the sum of two no's
```

② 

```
'''  
_____  
_____  
_____  
'''
```



# BYTECODE





--name--

if --name-- == "--main--":

a.py

```
print("Hello")
```

inbuilt variable

```
if --name-- == "--main--":  
    print("world")
```

directly execute  
a.py file

a.py

```
print("Hello")
```

```
def sum(a,b):  
    return a+b
```

```
print(sum(2,3))
```

Hello  
5



a.py → print("Hello")

True → if \_\_name\_\_ == "\_\_main\_\_":  
body { print("World") }

b.py  
import a.  
print("ABC")



# TYPE OF ERRORS



✓ 1) **Syntax Error** : This error occurs whenever there is a issue in the syntax of the code.

For ex: missing colon, imbalance of paranthesis

`print("Hello")`  $\Rightarrow$  Syntax error

`while i > 0`  $\Rightarrow$  Syntax error

==  
==  
==

✓ 2) **Indentation Error** : This error occurs when the indenting in the code is incorrect.

`if a > b:`

`print("Hello")`  $\Rightarrow$  Error

Incorrect usage of tabs and spaces.





# TYPE OF ERRORS



3) Type Error: This error occurs when value is not of expected type. For example when trying to use string as int.

`a = "5"`

`b = 3`

`print(a+b)`  $\Rightarrow$  Type Error

4) Value Error: This error occurs when the value is not in the expected range or format. For example trying to convert string to integer.

`a, b = '1,2,3,4'`

$\rightarrow$  Value error

`s = 'abcd'`

`print(int(s))`  $\Rightarrow$  Value error

`l = [10, 20, 30, 40, 50]`

`l.index(100)`

$=$  Value error





# TYPE OF ERRORS

5) Zero division error  
Zero division error occurs when you try to divide by 0.

$a = 10$

$b = 0$

$c = a / b \Rightarrow$  Zero division error.

6) Name Error: When a variable is not defined & you are using it, then it gives Name error.

`print(a)`  $\Rightarrow$  Name error

7) Attribute error: This type of error occurs when trying to access an attribute that doesn't exist.

Attribute  
error

`s = "hello"`

$\Leftarrow$  `s.reverse()`



2) Index error  
Whenever you try to access out of range index  
then it gives Index error.

$l = [1, 2, 3, 4, 5]$   
0 1 2 3 4

`print(l[7])`  $\Rightarrow$  Index error

$a = 10$   
 $b = "5"$   
 $\underbrace{a + b}$



# EXCEPTION HANDLING

```
a = 10 ✓  
b = 0 ✓  
print(a+b) → 10  
print(a/b) → Error  
print(a*b)
```

It is used to gracefully handle the error and exception that may occur in the code.

try, except, finally, else





# EXCEPTION HANDLING

**try :** The try block is used to enclose the code in which error may<sup>occur</sup>. It is like a safety net that is used to catch the error.

**except :** The except block is used to handle the exception raised in try block. You can specify the type of exception you want to catch.



# EXCEPTION HANDLING



```
try :  
    a = int(input("Enter first number"))  
    b = int(input("Enter second number"))
```

```
    → c = a / b  
except Exception as e:  
    print("Exception is:", e)
```

```
{  
    print(a + b)  
    print(a * b)  
}
```

```
a = 10  
b = 0  
Exception is ZeroDivisionError  
10  
0
```



try:

l = [10, 20, 30, 40, 50]

number = input("Enter the number")

⇒ i = l.index(number)  
print(i)

⇒ print(number / i)

— ⇒ except ZeroDivisionError:  
print("Zero Division Error")

— except ValueError:  
print("Value Error")

except Exception as e:  
print("Unknown error:", e)

number = 70  
✓ Value Error

number = 10  
0  
Zero Division Error

finally: The finally block code is executed regardless whether the exception has come in try block or not.

try:

number = 2  
5.0

Out of Try-except

number = int(input("Enter the number"))

c = 10 / number  
print(c)

except Exception as e:

print("Error:", e)

number = 0

Error: Zero Division Error

Out of Try-Except

finally:

→ print("Out of Try-except")



else : This block gets executed when there is no exception in try block.

number = 2  
5.0  
No exception  
Finally

number = 0  
Error: Zero Division Error  
Finally

try:

number = int(input("Enter a number"))  
c = 10 / number  
print(c)

except Exception as e:  
print("Error:", e)

else:

print("No exception")

finally:

print("Finally")

print("Hello")  
print("World")  
print("HATE")  
Syntax error





## Summary



→ Types of error

→ `--name--`

→ Byte code

→ Exception Handling

- try
- except
- finally
- else

→ Comments

**THANK - YOU**