PYTHON = NOTES

→ BASIC - DATA - TYPES

Int (7,8) — (Any Integer type Value)

Float (9.2, 10.1) — (Any decimal Value)

string (name) — (Its a char Array)

(•) Strings :-

declaration ; = Variable = " string"

Printing ; = Print (" Text") / Print (var)

length of string = len (Var)

To chang case = Variable. upper()
Variable. lower()
(first Alpha ↑) ← Variable. captalize()

To find Position of Alphabet in string :
Variable. find (Alphabet)

Variable. split ('Var) → gives splited list.
Variable. partition (Var) → " " at Var.
which comes first.

(•) LISTS :-

declaration Variable = [Int, float, string ...]

Putting Value at Part Pos : List [Pos] = input.

Put input at end : → List. append (input).

Reketetive counts of Variable. → list. count (Var).

Add list to list with items singly → list extend([ ])

Insert at certain index → list. insert (Index, Var)

Pop " " " → List. pop (index)

(•) Dictonaries :-

\# Declaration : Dictonary { "Key1" : Value , "Key2" : Value }

\# Print (Dictonary [Key]) → Value at thay Key.

\# To get Keys ⇒ Dictonary . Key ( )

\# To get Value ⇒ " . Values ( )

\# To get Both ⇒ " . items ( )

→ ORDERED - DICTONARIES :-

from collection import ordered Dictonari

Dict = ordered Dict{}

\# order will be maintained.

(•) Sets :-

\# Declartion : my Set = set ( )

no rehetetive addition.

\# Add element : → my set . add (var)

(•) Bool :-

\# Comparison Data tyhe.

Eg print(a = = b)

{ if a = b True. if a ≠ b false } output.

→ STRING - SLICING & FORMATING :-

\# Eg string = " Abhishek"

string[0] = A.

string[-] = k.

\# Reverse string :

string [: : -1]

\# Sub-section slicing :-

string [start : Ending]

start ↓ start write

Ending ↓ stop write.

\# steh Slicing : string[Rage : Range : steh]

(•) String - Formatting :-

Used to put variable inside a string.

Eg Var = 10
String = " Hello"

To print both :-

print (" Hello { } ". format (Var))

→ FILE - HANDLING :-

Le Hers - Used :-

w , a , r .

⇒ w is used to create file or replace existing file with same name.

⇒ a is used to attend file or add any thing at last of new file.

⇒ r is to read file which exists.

(•) Basic functions :-

Var = open (" filename.txt ", 'w, a, r")

Var. write (Input)

Var. read (" file name").

→ OPERATORS (comparison)

(•) OR :-

a =< b (or) a > c

↓ syntax

written as c<a<=b

(•) And :-

a =< b (and) a > c

↓ syntax.

(•) not :-

(not) a > b

↓ syntax.

→ STATMENTS :-

(•) DECSION - MAKING :-

```
If condition :
    Execution statment

elif Condition :
    Execution statment

else :
    Execution statment
```

(•) Iterations :-

For Loop :-

⇒ for Var in Datatyp :-
    Execution on Data tyhe.

⇒ for Var in range (start, end):
    Execution

While - Loop :-

```
Declare Var & intialize.
while Condition :-
    Execution.
    inc. / Dec. of variable.
```

⇒ Infinite : while True :

→ FUNCTIONS ON DATA - TYPES :-

Min, Max functions :-

- min (mylist)
- max (mylist)

ZIP function :-

collects 2 < lists

- zip(list 1, list 2)

(•) Random :-

```
from random import randint.
    Var = randint ( start, end ) → range.
```

→ FUNCTIONS :-

(•) Defining :-

```
def function_name (      ):
    Executable
    commands.
```

(•) Calling :-

```
function_name ( )
```

* Use `__init__ == "__main__"` (before execution)

(•) * args :-

Allows to take ∞ no. of arguments in fuctios
stores arguments as list

```
myfunc ( * args)
```

(•) ** kwargs :-

stores arguments as Dictonaries.

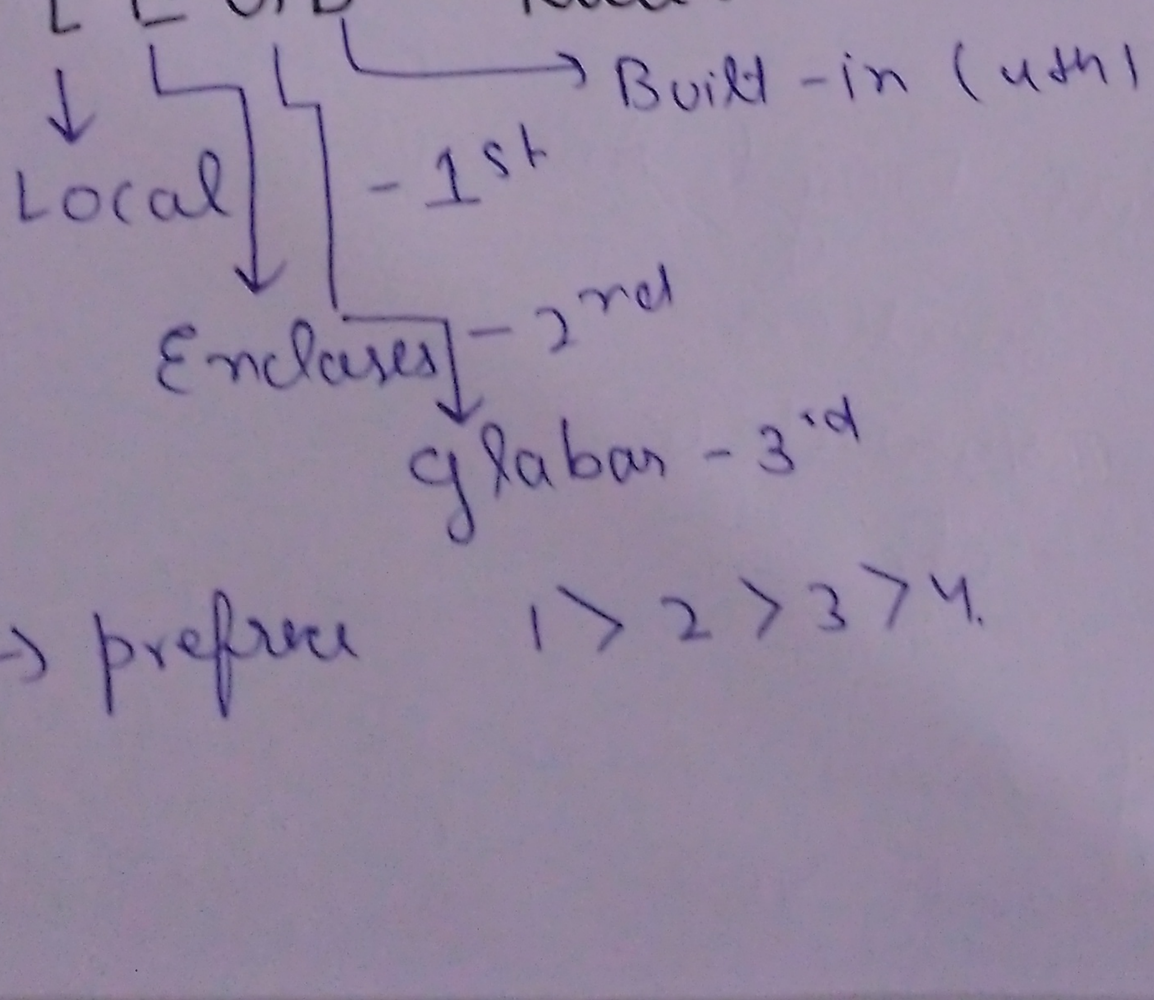
```
myfuc ( ** kwargs)
```

(•) Mah fuction :-

```
mah ( function, list)
```

* helps apply fuction on whole - list.

→ LEGB - Rule :-



→ prefree 1 > 2 > 3 > 4.

→ OOP :-

(•) Declaration :-

```
Class Name_var( ):
        def--init--( self, Attributes(n))
providing Value → Self. Attribute(n) = Attribute(n)
  to instance
```

⇒ Example :-

```
Class Dog ( ):
        def --inil -- (self, breed, name)
                self. name = "Jorry"
                self. breed = " Mong."
```

∴ Dog. breed will give "Mong".

(•) Functions inside Class :-

```
def -- my. fuction -- ( vars )
```

⇒ calling Var. --myfunction — (Args)

(•) Inheritance :-

To draw properties of one class in another class.

Eg class Dog (Animal) → Parent Class

functions
variables
etc...

(•) Polymorphism :-

* Use fuctions of same name in diff. classes.
* Use same name vars in functions.

→ EXCEPTION - HANDLING :-

It avoids errors on runtime by executing excpt at error

→ Syntax :-

```
try :-
    execution commands
except :
    execution when error comes
finally :  → (Alway Run)
    statment
```

→ Check Code - Marks :-

pip install pylint.

Pylint. file.Py.

→ Python - DECORATORS :-

Puts fuction return in another fuction. Directly.

→ syntax :-

```
@ fuct - 1 ( )
def fue - 2 ( )
        Executable
        comands
        return a
```

⇒ ∴ return of fun - c will be added in fuct - 1 & it will be executed.