URBAN EDGE

Chapter 1 : Modules, Comments & pip

Lets write our very first python program
Create a file called hello.py and paste the below code in it

```
print ("Hello World")
```
→ print is a function (More later)

Execute this file (.py file) by typing python hello.py and you will see Hello World printed on the screen.

Modules

A module is a file containing code written by somebody else (usually) which can be imported and used in our programs

Pip

Pip is the package manager for python
You can use pip to install a module on your system

↳ pip install flask installs flask module.

Types of modules

There are two types of modules in Python

1. Built in modules → Pre installed in python
2. External modules → Need to install using pip.

Some examples of built in modules are os, abc, etc
Some examples of external modules are tensorflow, flask etc

URDAN EDGE

Chapter 1 : Modules, Comments & pip

Lets write our very first python program
Create a file called hello.py and paste the below code in it

```
print ("Hello World")
```
→ print is a function (More later)

Execute this file (.py file) by typing python hello.py and you will see Hello World printed on the screen.

Modules

A module is a file containing code written by somebody else (usually) which can be imported and used in our programs.

Pip

Pip is the package manager for python
You can use pip to install a module on your system

↳ pip install flask installs flask module.

Types of modules

There are two types of modules in Python

1. Built in modules → Pre installed in python
2. External modules → Need to install using pip.

Some examples of built in modules are os, abc, ct
Some examples of external modules are tensorflow, flask ct

Using Python as a Calculator

We can use python as a calculator by typing "python" + ↵ on the terminal
↳ This opens REPL or Read Evaluate Print Loop

Comments

Comments are used to write something which the programmer does not want to execute.
↳ Can be used to mark author name, date etc.

Types of Comments

There are two types of comments in Python

1. Single line Comments → Written using #
2. Multi line Comments → Written using ''' Comment '''

Chapter 2 - Variables and Datatypes

A variable is the name given to a memory location in a program. for example

```
a = 30
b = "Harry"
c = 71.22
```

→ Variables = Containers to store a value.
→ Keywords = Reserved words in Python
Identifiers = class/function/variable name

Data Types

Primarily there are following data types in Python

1. Integers
2. Floating point numbers.
3. Strings → collection of character inside a single '' or "" coty
4. Booleans → It can be define as True / false.
5. None → कुछ नहीं

Python is a fantastic language that automatically identifies the type of data for us.

```
a = 71          ⇒ Identifies a as class<int>
b = 88.44       ⇒ Identifies b as class <float>
name = "Harry"  ⇒ Identifies name as class <str>
```

Rules for defining a Variable name → Also applies to other Identifiers

- A variable name can contain alphabets, digits and underscores.
- A variable name can only start with an alphabet and underscore
- A variable name cant start with a digit
- No white space is allowed to be used inside a variable name.

Examples of a few variable names are :
harry, one8, seven_, _seven etc.

Operators in Python
Following are some common operators in Python:
1. Arithmetic operators => +, -, *, / etc.
2. Assignment operators => =, +=, -= etc.
3. Comparison operators => ==, >, >=, <, != etc.
4. Logical operators => and, or, not

type() function and Typecasting
type function is used to find the data type of a given variable in Python.

```
a = 31
type(a) => class <int>

b = "31"
type(b) => class <str>
```

A number can be converted into a string and vice versa (if possible)
There are many functions to convert one data type into another
Type casting is a way to convert one data type to another.

```
str(31) => "31"      => Integer to String Conversion
int("32") => 32      => String to Integer Conversion
float(32) => 32.0    => Integer to Float Conversion
```

... and so on

Here "31" is a string literal and 31 a numeric literal.

URBAN EDGE

input() function

This function allows the user to take input from the keyboard as a string

a = input("Enter name") ⇒ If a is "harry", the user entered harry

It is important to note that the output of input is always a string (even if the number is entered) ⇒ If a is "34" user entered 34

Chapter 4 - Lists and Tuples

Python Lists are containers to store a set of values of any data type

```
friends = ["Apple", "Akash", "Rohan", 7, False]
           str()                      int()  bool()
```

Can store value of any datatype

List Indexing

A list can be indexed just like a string

```
L1 = [7, 9, "Harry"]

L1[0]  => 7
L1[1]  => 9
L1[70] => Error
L1[0:2] => [7, 9]   => List Slicing
```

List Methods

Consider the following list:

```
L1 = [1, 8, 7, 2, 21, 15]
```

1. L1.sort(): updates the list to [1, 2, 7, 8, 15, 21]
2. L1.reverse(): updates the list to [15, 21, 2, 7, 8, 1]
3. L1.append(8): adds 8 at the end of the list
4. L1.insert(3,8): This will add 8 at 3 index

URBAN EDGE

5) L1.pop(2): Will delete element at index 2 and return its value

6) L1.remove(21): Will remove 21 from the list.

Tuple ⟹ **Tuples in Python**

A tuple is an immutable data type in python.
↳ Cannot change

a = () ⇒ Empty tuple
a = (1,) ⇒ Tuple with only one element needs a comma
a = (1, 7, 2) ⇒ Tuple with more than one element

Once defined a tuple's elements cant be altered or manipulated.

Tuple methods
Consider the following tuple
a = (1, 7, 2)

1) a.count(1): a.count(1) will return number of times 1 occurs in a.

2) a.index(1): a.index(1) will return the index of first occurence of 1 in a.

Chapter 5 : Dictionary & Sets

Dictionary is a collection of key-value pairs

Syntax :

```
a = { "key" : "value",
      "harry" : "code",
      "marks" : "100",
      "list" : [1,2,9] }
```

(keys → "key", "harry", "marks", "list"; values → "value", "code", "100", [1,2,9])

a["key"] ⇒ Prints "value"
a["list"] ⇒ Prints [1,2,9]

Properties of a Python Dictionaries

1. It is unordered
2. It is mutable
3. It is indexed
4. Cannot contain duplicate keys

Dictionary Methodes

Consider the following dictionary

```
a = { "name" : "Harry",
      "from" : "India",
      "marks" : [92,98,96] }
```

1. a.items() : returns a list of (key, value) tuples
2. a.keys() : returns a list containing dictionary's keys
3. a.update({"friend" : "Sam"}) : updates the dictionary with supplied key-value pairs

4: a.get("name"): returns the value of the specified keys (and value is returned eg "Harry" is returned here)

More methods are available on docs.python.org

Sets in Python

Set is a collection of non repetitive elements

```
S = Set()                    => No repetition allowed!
S.add(1)
S.add(2)                     => or Set = {1,2}
```

If you are a programming beginner without much knowledge of mathematical operations on sets, you can simply look at sets in python as data types containing unique values.

Properties of Sets

1. Sets are unordered => Element's order doesn't matter
2. Sets are unindexed => Cannot access elements by index
3. There is no way to change items in sets.
4. Sets cannot contain duplicate values

Operations on Sets

Consider the following Set:

S = {1, 8, 2, 3}

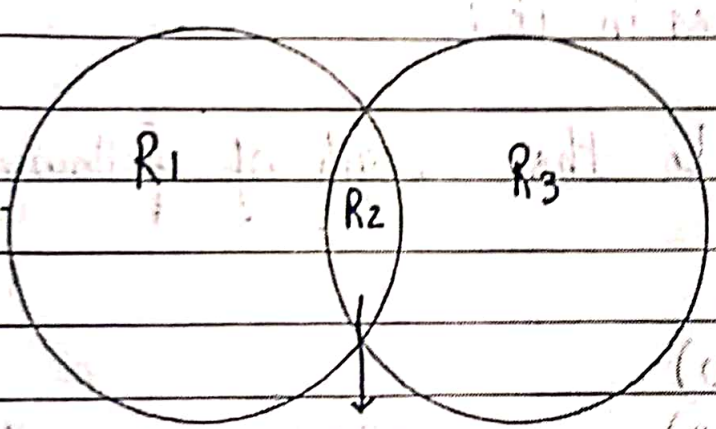
1. len(s): Returns 4, the length of the set
2. S.remove(8): Updates the set S and removes 8 from S.

3. S.pop() : Removes an arbitrary element from the set and returns the element removed

4. S.clear() : Empties the set S

5. S.union({8,11}) : Returns a new set with all items from both sets. ⇒ {1,8,2,3,11}

6. S.intersection({8,11}) : Returns a set which contains only items in both sets. ⇒ {8}



$R_2 \Rightarrow A \cap B$

$R_1 + R_2 + R_3 \Rightarrow A \cup B$

$R_1 + R_3 \Rightarrow A \Delta B$

$R_1 \Rightarrow A - B$

$R_3 \Rightarrow B - A$

Chapter 6 - Conditional Expressions

Sometimes we want to play pubG on our phone if the day is Sunday.
Sometimes we order Icecream online if the day is Sunny.
Sometimes we go hiking if our parents allow.

All these are decisions which depends on a condition being met.
In Python programming too, we must be able to execute instructions on a condition(s) being met. This is what conditionals are for!

If else and elif in Python

If else and elif statements are a multiway decision taken by our program due to certain conditions in our code.

Syntax:

```
if (condition1):                    => if condition 1 is true
    print ("yes")
elif (condition2):                  => if condition 2 is true
    print ("No")
else:                               => otherwise
    print ("May be")
```

Indentation ←

Code example:

```
a = 22
if (a>9):
    print ("Greater")
else:
    print ("lesser")
```

Relational Operators

Relational operators are used to evaluate conditions inside the if statements. Some examples of relational operators are:

`==` → equals
`>=` → greater than/equal to
`<=`, etc.

Logical operators

In python logical operators operate on conditional statements. Example:

and → true if both operands are true else false
or → true if at least one operand is true else false
not → inverts true to false & false to true

elif clause

elif in python means [else if]. An if statement can be chained together with a lot of these elif statements followed by an else statement

```
if (Condition1):
    # Code
elif (Condition 2):
    # Code
elif (Condition 3):
    # Code
.
.
else:
    # Code
```

⇒ This ladder will stop once a condition in an if or elif is met.

Important notes:

1. There can be any number of elif statements.
2. last else is executed only if all the conditions inside elifs fail.

Chapter 7 - Loops in Python

Sometimes we want to repeat a set of statements in our program for instance: Print 1 to 1000

Loops make it easy for a programmer to tell the computer, which set of instructions to repeat and how!

Types of loops in Python
Primarily there are two types of loops in Python

1. While loop
2. For loop

We will look into these one by one!

While loop

The syntax of a while loop looks like this:

```
While Condition:
    #Body of the loop
```

⇒ The block keeps executing until the condition is true

In while loops, the condition is checked first If it evaluates to true, the Body of the loop is executed, otherwise not!
If the loop is entered, the process of [Condition check & execution] is continued until the condition becomes False

Quick Quiz: Write a program to print 1 to 50 using a while loop.

An Example

```
i = 0
while i < 5:
    print("Harry")
    i = i + 1
```

⇒ Prints "Harry" 5 times!

Note: If the condition never becomes false, the loop keeps getting executed.

Quick Quiz: Write a program to print the content of a list using while loops

For loop
A for loop is used to iterate through a sequence like list, tuple or string [iterables]

The syntax of a for loop looks like this:

```
l = [1, 7, 8]
for item in l:
    print(item)
```

→ print 1, 7 and 8

Range function in Python
The range function in python is used to generate a sequence of numbers
We can also specify the start, stop and step-size as follows:

range(start, stop, step_size)

↳ Step size is usually not used with range()

An Example demonstrating range() function

```
for i in range(0, 7):        → range(7) can also be used
    print(i)                 → prints 0 to 6
```

For loop with else
An optional else can be used with a for loop if the code is to be executed when the loop exhausts
Example :

```
l = [1, 7, 8]
for item in l:
    print(item)
else:
    print("Done")    → This is printed when the loop exhausts!
```

Output :

```
1
7
8
Done
```

The break statement
'break' is used to come out of the loop when encountered
It instructs the program to - Exit the loop now

Example :

```
for i in range(0,80):
    print(i)          → This will print 0, 1, 2 and 3
    if i==3:
        break
```

The Continue Statement

'Continue' is used to stop the current iteration of the loop and Continue with the next one
It instructs the program to "Skip this iteration"

Example:

```
for i in range(4):
    print("printing")
    if i == 2:          => if i is 2, the iteration
        Continue           is skipped.
    print(i)
```

pass statement

pass is a null statement in python
It instructs to "Do nothing"

Example:

```
l = [1, 7, 8]
for item in l:
    pass          → Without pass, the program
                    will throw an error
```

Chapter 8 - Functions & Recursions

A function is a group of statements performing a specific task

When a program gets bigger in size and its complexity grows, it gets difficult for a programmer to keep track on which piece of code is doing what!

A function can be reused by the programmer in a given program any number of

Example and Syntax of a function
The syntax of a function looks as follows:

```
def func1():
    print("Hello")
```

This function can be called any number of times, anywhere in the program

Function Call
Whenever we want to call a function, we put the name of the function followed by parenthesis as follows:

```
func1()
```
→ This is called function call

function definition
The part containing the exact set of instructions which are executed during the function call.

Quick Quiz : Write a program to greet a user with "Good Day" using functions

Types of functions in Python
There are two types of functions in Python.
1. Built in functions → Already present in Python
2. User defined functions → Defined by the user

Examples of built in function includes len(), print(), range() etc.

The func1() function we defined is an example of user defined function

Functions with arguments
A function can accept some values it can work with. We can put these values in the parentheses
A function can also return values as shown below:

```
def greet(name):
    gr = "Hello " + name
    return gr

a = greet("Harry")
```

→ "Harry" is passed to greet in name

→ a will now contain "Hello Harry"

Default Parameter Value
We can have a value as default argument in a function.
If we specify name = 'stranger' in the line containing def, this value is used when no argument is passed

URBAN EDGE

for example:

```
def greet (name = "stranger")
        # function body
```

greet() → Name will be "stranger" in function body (default)
greet ("Harry") → Name will be "Harry" in function body (passed)

Recursion

Recursion is a function which calls itself
It is used to directly use a mathematical formula as a function. For example:

factorial (n) = n × factorial (n-1)

This function can be defined as follows:

```
def factorial (n):
    if i==0 or i==1 : → Base condition which doesn't call the function any further
        return 1
    else:
        return n * factorial (n-1) → Function calling itself
```

This works as follows:

Factorial (4)
↓ ↳ [Function called]
4 × factorial (3)
4 × [3 ↓ × factorial (2)]
4 × 3 × [2 ↓ × factorial (1)]
4 × 3 × 2 × [1] [Function returned]

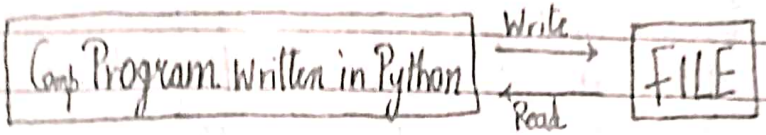
URBAN EDGE

The programmer need to be extremely careful while working with recursion to ensure that the function doesnt Infinitely keep calling itself.
Recursion is sometimes the most direct way to code an algorithm

Chapter 9 - File I/O

The random Access memory is volatile and all its contents are lost once a program terminates
In order to persist the data forever, we use files

A file is data stored in a storage device. A Python program can talk to the file by reading content from it and writing content to it



RAM = Volatile
HDD = Non Volatile

Types of files
There are 2 types of files:
1. Text files (.txt, .c etc)
2. Binary files (.jpg, .dat, etc)

Python has a lot of functions for reading, updating and deleting files.

Opening a file
Python has an open() function for opening files. It takes 2 parameters: filename and mode.

```
open("this.txt", "r")
```

- open is a built-in function
- "this.txt" → filename
- "r" → mode of opening (read mode)

URBAN EDGE

Reading a file in python

```
f = open("thus.txt", "r")   → open the file in read mode
text = f.read()             → Read its contents
print(text)                 → Print its contents
f.close()                   → Close the file
```

We can also specify the number of characters in read() function: f.read(2)
↳ Reads first 2 characters

Other methods to read the file
We can also use f.readline() function to read on full line at a time

f.readline() → Reads one line from the file

Modes of opening a file

r → open for reading
w → open for writing
a → open for appending
+ → open for updating

'rb' will open for read in binary mode
'rt' will open for read in text mode

Writing files in Python

In order to write to a file, we first open it in write or append mode after which, we use the python's f.write() method to write to the file!

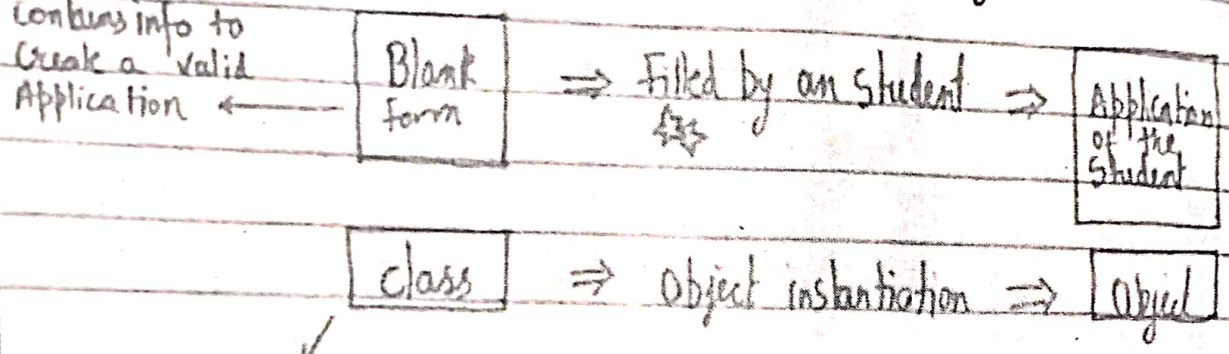
Object Oriented Programming

Solving a problem by creating objects is one of the most popular approaches in programming. This is called Object oriented programming.

This concept focuses on using reusable code.
↳ Implements DRY principle

Class

A class is a blueprint for creating objects.



The syntax of a class looks like this:

The way to define a class is:-

```
Class Employee:
        # methods & variables
```

[Classname is written in PascalCase]

Object

An Object is an instantiation of a class. When class is defined, a template (info) is defined. Memory is allocated only after object instantiation.

Objects of a given class can invoke the methods available to it without revealing the implementation details to the user. → Abstraction & Encapsulation

Modelling a problem in OOPs
We identify the following in our problem

Noun → Class → Employee
Adjective → Attributes → name, age, salary
Verbs → Methods → getSalary(), increment()

Class Attributes
An attribute that belongs to the class rather than a particular object.

Example:

```
Class Employee:
    company = "Google"        → [Specific to each class]

harry = Employee()            → object instantiation
harry.company
Employee.company = "YouTube"  → changing class attribute
```

Instance Attributes
An attribute that belongs to the Instance (object)
Assuming the class from the previous example:

```
harry.name = "Harry"
harry.salary = "30K"      ⇒ Adding instance attributes
```

Note: Instance attributes take preference over class attributes during assignment & retrieval

harry.attributes → ① Is attribute present in object?
② Is attribute present in class?

Self parameter
Self refers to the instance of the class
It is automatically passed with a function call from an object

```
harry.getSalary()   → here self is harry
                    ↳ equivalent to Employee.getSalary(harry)
```

The function getsalary is defined as:

```
class Employee:
    company = "Google"
    def getSalary(self):
        print("Salary is not there")
```

Static method
Sometimes we need a function that doesnt use the self parameter. We can define a static method like this:

```
@staticmethod                → decorator to mark greet
def greet():                   as a static method
    print("Hello user")
```

__init__() Constructor
__init__() is a special method which is first run as soon as the object is created
__init__() method is also known as constructor

It takes self argument and can also take further arguments

For Example

```
Class Employee:
    def __init__(self, name):
        self.name = name

    def getSalary(self):
        ...

harry = Employee("Harry")
```

↳ Object can be instantiated using constructor like this!

Chapter 12 - Advanced Python 1

Exception Handling in Python

There are many built-in exceptions which are raised in Python when something goes wrong. Exceptions in Python can be handled using a try statement. The code that handles the exception is written in the except clause.

```
try:
    # Code                    → Code which might throw Exception
except Exception as e:
    print(e)
```

When the exception is handled, the code flow continues without program interruption.

We can also specify the exceptions to catch like below:

```
try:
    #Code
except ZeroDivisionError:
    # Code
except TypeError:
    # code
except:
    # code          → All other exceptions are handled here.
```

Raising Exceptions

We can raise custom exceptions using the raise keyword in python

Exception Handling in Python

There are many built-in exceptions which are raised in Python when something goes wrong.
Exceptions in Python can be handled using a try statement. The code that handles the exception is written in the except clause.

```
try:
    # Code                          → Code which might throw Exception
except Exception as e:
    print(e)
```

When the exception is handled, the code flow continues without program interruption.

We can also specify the exceptions to catch like below:

```
try:
    #code
except ZeroDivisionError:
    # Code
except TypeError:
    # code
except:
    # code          → All other exceptions are handled here.
```

Raising Exceptions

We can raise custom exceptions using the raise keyword in python.

URBAN EDGE

try with else clause
Sometimes we want to run a piece of code when try was successful.

```
try:
    # Some code
except:
    # Some code
else:
    # Code  → This is executed only if the try was successful
```

try with finally
Python offers a finally clause which ensures execution of a piece of code irrespective of the exception.

```
try:
    # Some code
except:
    # Some code
finally:
    # Some code  → executed regardless of error!
```

if __name__ == '__main__' in Python
__name__ evaluates to the name of the module in Python from where the program is ran

If the module is being run directly from the command line, the __name__ is set to string "__main__"
Thus this behaviour is used to check whether the module is run directly or imported to another file.

The global keyword
global keyword is used to modify the variable outside of the current scope.

Enumerate function in Python
The enumerate function adds counter to an iterable and returns it

```
for i, item in list1:
    print(i, item)
```
↳ Prints the items of list 1 with index!

List Comprehensions
List comprehension is an elegant way to create lists based on existing lists

```
list1 = [1, 7, 12, 11, 22]
list2 = [i for item in list1 if item > 8]
```

Python Program Practice

Swap two Values in python

```
a = 5
b = 6
Temp = a
a = b
b = Temp
print (a)
print (b)
```

```
a = 5
b = 6
a = a + b
b = a - b
a = a - b
print (a)
print (b)
```

Write a python program to print pattern.

Code

①

```
for i in range (4):
    for j in range (4):
        print ("*", end ="")
    print ()
```

```
* * * *
* * * *
* * * *
* * * *
```

②

```
for i in Range (4):
    for j in range (i+1)
        print ("#", end = " ")
    print ()
```

```
#
# #
# # #
# # # #
```

③ Write a python program to print the whether a given no is prime or not

```
num = (int (input) ("Enter a number )
for i in range (2, num):
    If num % i = = 0:
        print("not prime")
        break;
Else
print (prime)
```

Page
Date

Lecture-26 Arrays.

```
Vals = array ('i', [5,9,4,6,8])
print("Vals")
```

Some functions of arrays.

Vals.reverse() → It will reverse the array.

```
for i in range(5):
    print(val[i])
```
→ It will move the array in a Row format one-by-one.

* Array values from user in python

```
from array import
arr = Array ('i' [])
n = int (input("Enter the array"))
for i in range(n)
    x= int (input("Enter the value"))
    arr.append (x)
    print (arr)
```

Different ways for creating an array are -

There are mostly six ways in python for creating an array

1. array()
2. LinSpace() → arr = linspace (0,16,10)
3. LogSpace()
4. arrange()
5. Zeroes()
6. Ones()

How to Copy an array

```
arr1 = array ([2, 5, 4, 6, 8])
arr2 = arr1.view → In view you use a shallow copy.
print ("arr1")
Print ("arr2")
print (id (arr1))
print (id (arr2))
```

* Working with Matrix in Python :-

```
arr1 = array ([[1, 2, 3], [4, 5, 6]])
print (arr1.ndim) → It tells the dimension of an array.

arr2 = arr1.flatten() → It should combined both the array.
printarr2()

m = matrix ('1, 2 3; 4 5 6; 7 8 9;')
print m
```

* Keyword variable length arguments :-

```
def person (name, * data):
    print (Name)
    print (data)
person ('navin', age = 28, city = 'Mumbai' Mob-no)
```

Page :
Date :

fibbonaci Series in Python code :-

```
def fibonacci series ( )
    a = 0
    b = 1
print (a)
print (b)
for i in range (2, n):
        c = a + b
        a = b
        b = c
        print c
fibbonacci (10)
```

Q. Program of a factorial number in Python

```
diffact (n):
    f = 1
    for i in range (1, n+1);
        f = f * i
    return f
x = 4
result = fact (x)
print (result)
```

* Lambda function Example

```
nums = [3, 4, 5, 6, 7, 8, 9]
even = list(filter(lambda n : n % 2 == 0, nums))
print even.

doubles = List(Map(lambda n : n * 2 evens.
print (doubles):

Sum = ( Reduce (Map (lambda a,b : a + b doubles)
print Sum
```

In this way we will use Map, Reduce, filter function in Lambda.

Tutorial Decorator In Python :-

Using decorator we can use various feature in a function.

⇒ By using Decorator we can change the Behaviour of the existing function

```
def div (a, b)
    print (a/b)
def smart div (func)

    def inner (a,b)
        if a<b:
            a,b = b,a
        return func (a,b)
div = Smart_div (div)
div (2,4)
```

Special variable in module.

demo.py

```
def name ()
    print ("welcome to python")

print (name)

name ()
```

calcu.py

```
def calc ()
    print ("result of python")
    print ("modules in python")

calc()
```

⇒

```
if name == "main":
    main ()
```

* Instance Variable and class variable in OOPS

```
class Car:
    def __init__(self):
        self.mil = 50
        self.com = BMW

C1 = Car()
C2 = Car()  ---> Car.wheels = 5
print (C1.com, C1.mil, C1.wheels)
print (C2.com, C2.mil, C2.wheels)
```

*

Page :
Date :

* Python Inner class.

When a classe · defined inside a class than it is known as innerclasses.

```
class student :
    def_init_ (Self , name, rollno)
        Self · name = name
        sdf · rollno = rollno
        Self · dap = Self · Laptop ( ) We have creat the object
    def show (Self):          of · to intone the class
        print ("Self · name , sdf · rollno
```

```
class Laptop
    def_init_ (Self):
        Self · brand = hp
        Self · cp = 'i5'
        Self · ram = 8
    dg show(Self) :
        print (Sdf · brand , sdf · cpu . sdf · ram .
```

```
S1 = student ('navin' ; 8 )
S2 = student ('Sachin , 9)
```

* Inheritence — the property of
* When a class inherit other class is known as the inheritence
ano

```
class A:
    def feature1(Self):
        print (" feature 1 working)

    def feature (2):
        print (" feature 2 is working)
```

```
a = A()
a1.feature
class B (A)
    def feature 3 (sdf):
        print ("featur 3 working")
    def feature 4 (sdF)
        print("Feature 4 is working)

a1 = a()
a1.feature 1()
a2.featur 2()
B1 = b()
```

Python Program to Print the Lincar Search

```
Pos = -1
def search (list, n):
    i = 0
    while i < len (list):
        if list[i] == n:
            print ("found here") or return True
        i = i+1
    return false

list = [5,8, 4, 6, 2,9]
n = 10
if Search (list, n):
        print ("found at", pos)
    else:
    print (not found)
```