Page: 1
Date: / /

What is programming?

Just like we use English, Hindi or other Language to communicate with each other, we use the programming language like python to communicate with the computer, programming is the way to instruct the computer to perform various tasks.

What is Python?

Python is a simple and easy to understand language which feels like reading simple English. This pseudo code nature of Python make it easy to learn and understandable by beginners.

it can be used for

Console Applications

Window/Desktop Applications

Web Applications

Machine Learning

Apart from the types of Applications mentioned above, it can be used many more types of applications Python was Created by Guido Rossum in 1989.

The idea of Python started in the early 1980s but the real implementation started in 1989 and it was finally published in 1991 (Feb 27, 1991)

By the way, the language is named after BBC show "Monty Python's Flying Circus". Guido Van Rossum worked that time in a project at the Centrum Wiskunde & Informatica, called Amoeba, a distributed operating system.

Features of Python

Python is simple and easy to learn, read and write.

Python language is freely available and the source code is also available. Therefore, it is an open source programming language.

Python language is more expressive than most of the language, means it is more understandable and readable.

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is portable language

High level language.

Know Program

Python Support procedure-oriented programming as well as object-oriented programming.

Python has a large and broad library which provides a rich set of module and functions for rapid application development.

Installation of Python

Python can be easily installed by python.org

when you click on the download button, python can be installed right after you complete the set-up by executing the file for your platform.

Just install it like a game!

First Program

Lets write our very first python program. Create a file called hello.py and paste the code which is given below.

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

here print is a function which we will study later.

Know Program

Execution 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 someone else which can 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.

Types of Modules

There are two types of modules in Python.

1. Build in Modules – Pre-Installed in Python
2. External Modules – Need to Install using pip

Comments

Comments are used to write something the programmer does not want to execute.
i.e can be mask author name, date, etc.

Know Program

Types of Comments.

There are two types of comments in Python

1. Single line comments - written using #Comment

2. Multi line comments - written using """Comment"""

Know Program

Variable

- A Variable is the name give to a memory location in a program.
- A Variable is a Container to store values.

for example

a) a=234 , b="name" , C=24.53

where, a, b and c are the Variables that store respective data.

Identifiers

An identifiers can denotes various entities like Variable type, Subroutines, labels, functions, packag and so on.

for example

a = "Deepak Kumar"

where, a is an identifier or a Variable.

Keyword

Reserved words in Python are known as keywords. They Can't be used as Variable names as their meaning is already reserved.

Example to see all the Keywords

```
Import keyword          #we learn about import functi
                         later.
print(keyword.kwlist)
```

Know Program

Data Types

Primarily there are the following data types in python.

1. Integers
2. Floating point numbers
3. String
4. Boolean
5. None

Python is the Dynamic Language that automatically identifies the type of data for us.

- **Integers & Floating point numbers**

Integers and floating points are Separated by the presence or absence of a decimal point. for instance, 5 is an integer whereas 5.0 is a floating point number.

- **Boolean**

It is used to Store two Values i.e. True or false. it is used to test whether the result of an expression is True or False.

- **None**

It is used to define a null Variable.

for example

a = 234	- Identifies a as class <int>
name = "Deepak Kumar"	- Identifies name as class<string>
c = 24.53	- Identifies c as class<float>
d = True or false	- Identifies d as class<bool>
e = None	- Identifies e as class<none>

Know Program

Rules of Defining a Variable name:-

- A Variable name can contain alphabet, digits, underscores
- A Variable name can only start with an alphabet and underscore.
- A Variable name can't start with a digit.
- No white space is allowed to be used - inside a Variable name.

Examples of few Variable names are:-
deepak, one8, Seven, _Seven, etc.

Operators in Python

Operator is a Symbol that performs certain operations.

e.g.-
```
a = 1
b = 2
my_sum = a + b
```
Here 'a' and 'b' are the operands and '+' is the operators.

There are Seven operators in Python:-

1. Airthmetic Operators

+	Unary plus
-	Unary Subtract
*	Multiple two operands.

/	Divide Left oprand with the right and the result is in float. e.g. 6/3 = 2.0
**	Left operand raised to the Power of right. e.g. 2**3 = 8
%	Remainder of the division of left operand by the right. e.g. 5%2 = 1
//	Division that results into whole number adjusted to the left in the number line e.g.- 7//3 = 2

Note Floor division can perform both floating point and integral arithemetic. if arguments are int type then result is int type. if atleast an one argument is float type then result is float type.

2. Assignment operators

The equal sign (=) is used to assign a Value to a Variable.

e.g.
```
a=20
b= 5*9
print(a*b)
```
Output-
900

Know Program

Compound Assignment Operators

```
x = 5
```

1. '/=':

```
x /= 5
print(x)
1.0
```

2. '%=':

```
x %= 5
print(x)
0
```

3. '//=':

```
x //= 2
print(x)
2
```

4. '**=':

```
x **= 5
print(x)
3125
```

3. Comparision operators

==	is equal to
>	is greater than
<	is less than
!=	is not equal to
>=	is greater than or equal to
<=	is less than or equal to

Know Program

Comparision operators always return the Boolean Value (True / False).

e.g.
```
a = 10
b = 20
print("a>b is", a>b)
print("a<=b is", a<=b)
```
Output-
False
True

4. Logical operators

Logical and
'and' is used to get two Boolean and check whether they are both True. if either or both are not True, then the resultant expression is also false.

Logical Or
'or' is used to get the second Value of the first one is false. if the first one is True, it gets the first one.

Logical Not
'Not' is used to invert the Boolean. Gets the opposite Value.

Know Program

5 Bitwise operators

&	if both bits are 1 then only results is 1 otherwise result is 0.
|	if atleast one bit is 1 then result is 1 otherwise result is 0.
^	if bits are different then only result is 1 otherwise result is 0.
~	bitwise Complement operator i.e. 1 means 0 and 0 means 1.
>>	Bitwise left shift operator
<<	Bitwise right shift operator

e.g. print(4&5)
1
print(10.5&5.6)
TypeError

Note:- In the case of Left shift operator we do multiplication where as in case of right shift operator we do division.

Know Program

6. Identity operators

These operators are used to check if two values (or variable) are located on the same part of the memory. Two variables that are equal does not imple that they are identical.

'is' : True if the operands are identical

'is not' : True if the operands are not identical.

e.g.
```
x1 = "hello"
x2 = 1234
y1 = "hello"
y2 = 1234
print (x1 is y1)
print (x1 is not y1)
print (x1 is x2)
```
Output:-
```
True
false
false
```

Note:- '==' operators compare the value where as 'is' operator compares the addresses.

7. Membership Operators:-

We can use membership operators to check whether the given object present in the given collection.

Know Program

'in': Returns True if the given object present in the specified collection.

'not in': Returns True if the given object not present in the given specified collection.

e.g. x = "Hello learning Python is easy"
print ('l' in x) >> True
print ('d' in x) >> false
print ('d' not in x) >> True
print('Python' in x) >> True

Operator Precedence

if multiple operators present then which operator will be evaluated first is decided by operator precedence.

The following list describes the operators precedence in Python.

Priority Order

1. ()
2. **
3. ~, -
4. *, /, %, //
5. +, -
6. <<, >>
7. &
8. ^

Know Program

9. |
10. >, >=, <, <=, ==, !=
11. =, +=, -=, *= ...
12. is, not is
13. in, not in
14. not
15. and
16. or

We have learned about Variable with their data types and operators. Now, we should learn about type casting which is convert one data type into another one.

Type() FUNCTION AND TYPECASTING

#function will be explained further

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) => clas<str>
```

A number can be converted into a string and Vice-Versa (if Possible)

Know Program

There are many functions to convert one data type into another.

str(31) => "31" => integar to string conversion
int("31") => 31.0 => string to integar conversion
float(32) => 32.0 => integar to float conversion ...& so on

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

Input() Function:

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

Inside parenthesis we will

```
a = input ("Enter name")
```

=> if a is "Deepak", the user entered Deepak

Note:- it is important to note that the output of input is always a string (even if the numbers is entered.)

e.g.
```
X = input ("Enter some data:")
print (X)
```

Output of Program:

Enter some data: 123 # Here, we entered 123 numeric value.
"123"

Conditional Expression

Sometimes we want to play PubG on our phone if the day is SUNDAY

Sometimes we order ice-cream online if the day is SUNDAY

Know Program

Sometimes we ~~order~~ go hiking if our parent allow. All these are decision which depends on a condition being met.
In Python programming too, we must be able to extent instruction on a condition(s) being act.
This is what conditionals are for!

if else and elif in Python
if else and elif statement are a multi-way decision taken by our program due to certain condition in our code.

Indentation
It is refers to the spaces at the beginning of a code line.
Python uses indentation to indicate a block of code.

SYntax:

```
if (condition1):                  => if condition 1 is True.
    print("Yes")                  # Here we use Tab key for
                                    indent the code block
elif(condition2):                 => if condition 2 is True.
    print("No")                   # Here we use Tab key for
                                    indent the code block
else:                             => otherwise
    print("May be")               # Here we use Tab key for
                                    indent the code block.
```

Know Program

Code example

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

Output

Greater

elif Clause

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

Syntax:

```
if(Condition 1):
    # Here we write Our Code

elif (Condition 2):
    # Here we write Our Code

elif (Condition 3):
    # Here we write Our Code

else:
    # Here we write our Code.
```

=> This ladder will stop once a Condition in an if or elif is met.

Code Example

```
num = int(input("Please Enter A Number"))

if (num>0):
    print("%d is a positive number"%num)

elif (num<0)
    print("%d is Negative number"%Num)

else:
    print("%d is zero"%num)
```

Note: Essentially an if statement has a Condition, and then any number of elif, and finally an else. However, the elif and else are optional.

- Last else is executed only if all the Condition inside elif fail.

Loops

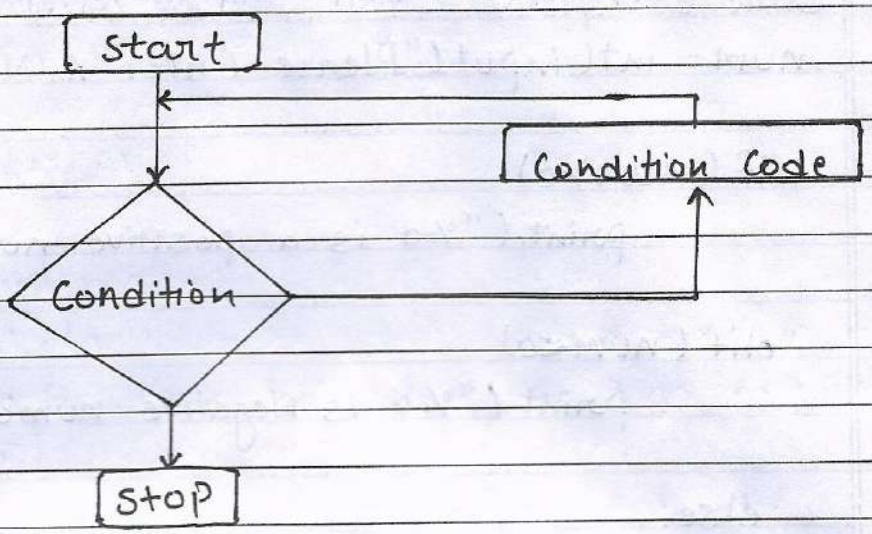
- Sometimes we want to repeat a set of statement in our program for instance: print 1 to 1000.
- Loops make it easy for a programmer to tell the Computer, which set of instruction to repeat and how!

Types of Loops

Primarily there are two types of loops in Python

1. while loop
2. For loop

Know Program



Advantages of loops

- Loop allows the execution statement or a group of statement multiple times
- In order to enter the loop there are certain conditions defined in the beginning.
- Once the condition becomes false the loop stops and the control moves out of the loop.

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!

Know Program

if the loop is entered, the process of (Condition check & execution) is continued until the condition become false.

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

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

For loop

A for loop is used to iterate through a sequence like list, tuple or string (which we learn about later).

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:

The range function in Python is used to generate a sequence of numbers.
we can also specify the start, stop and step-size as follow:

range(start, stop, step-size)

↓

Step-size is usually not used with range()

Know Program

An Example demostrating range() function

```
for i in range(0,7):
    print(i)
```

for loop with else

An optional else can be used with a for loop if the code is to be executed when the loop exhaust.

e.g.
```
l=[1,7,8]
for item in l:
    print(item)
else:
    print("Done")      -> This is printed when the loop exhaust.
```

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 pop now.

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

Know Program

Pass statement

'Pass' is a null statement in Python.
It instructs to "Do nothing".

e.g.
```
l = [1, 7, 8]
For item in l:
    pass
```

Data Structures

There are two types of data structure.

- Mutable: Lists, Dictionaries, sets
- IMMUtable: Numbers, Strings, Tuples

Strings

String is a data type in Python.
String is a sequence of characters enclosed in quotes.
we can primarily, write a string in these three ways

- Single quoted strings → a = 'Abhinav'
- Double quoted strings → b = "Abhinav"
- Triple quoted strings → c = """Abhinav"""

String Slicing

A string in Python can be sliced for getting a part of the string.

name = "SURAJ" ⇒ length = 5

	S	U	R	A	J
	0	1	2	3	4
	(-5)	(-4)	(-3)	(-2)	(-1)

Know Program

Consider the following string:

name = "RAJAT"

0	1	2	3	4
R	A	J	A	T
-5	-4	-3	-2	-1

=> length = 5

The index in a string starts from 0 to (length-1) in Python. In order to slice a string, we use the following syntax.

sl = name[ind_start : ind_end]

ind_start → first index included
ind_end → last index is not included

sl = [0:3] returns "Raj" → character from 0 to 3
sl = [1:3] returns "aj" → character from 1 to 3

- Negative Indices: Negative Indices can also be used as shown in the figure above -1 corresponds to the (length-1) index, -2 to (length-2)

- Slicing with skip Value
We can provide a skip value as a part of our slice like this.

word = "amazing"
word = [1:6:2] → 'mzn'

Other advanced slicing techniques

word = "amazing"

word = [:7] → word[0:7] → 'amazing'
word = [0:] → word[0:7] → 'amazing'

Know Program

String Functions

Some of the mostly used functions to perform operations on or manipulate strings are:-

- len() function → This function returns the length of the String

  len("Deepak") → returns 6

- string.endswith("rry") → This function tells whether the Variable strings ends with the string "rry" or for "rry" since Harry ends with rry

- string.count("c") → Counts the total number of occurance of any character

- String.Capitalize() → This function Capitalizes the first character of a given String.

- String.find(word) → This function finds a word and returns the index of first Occurance of that word in the string.

- String.replace(old word, new word) → This function replaces the oldword with new word in the entire String.

Know Program

Few string operations in tabular form.

Syntax	operations
print(len(string_name))	String length
print(string_name.index("char"))	Locate a character in String
print(string_name[start:stop])	Slicing
print(string_name.upper())	Convert the letter in a String to upper case

- Escape Sequence characters
Sequence of character after backslash '\'

↓

Escape Sequence char.

Escape Sequence character comprises of more than one characters but represents one character when used within the String.

e.g. Examples are :- \n → newline
\t → Tab
\' → Single quote
\\ → backslash.

Know Program

List

⇒ Python lists are Container to store a set of values of any data type.

⇒ A List is like an array of the objects it stores the elements an ordered Sequentially in slots, lists are dynamic in Python.

HOW to SPot A LIST in CODE:-

Lists are tuple methods always enclosed in Square brackets, and the objects Contained within the list are always Separated by a Comma.

Creating List:-

- Creates an empty list by assigning [] to a Variable Called prices

```
prices = []
```

- List of temperatures in degrees fahrenheit

```
temps = [32.0, 212.0, 0.0, 81.6, 100.0, 45.3]
```

- A list of String objects

```
words = ['hello', 'world']
```

Know Program

- A List of objects of different type:

Car_details = ['Honda City', 'UP-80', 2.2, 5939]

List Indexing

A List can be indexed just like a String.

L1 = [7, 9, 'Deepak']

L1[0] = 7

L2[1] = 9

L3[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(): update the list to [1, 2, 7, 8, 15, 21]
2. L1.reverse(): update the list to [15, 21, 2, 7, 8, 1]
3. L1.append(8): adds 8 at the end of the list.

Know Program

4. L1.insert(3,8): This will add 8 at 2 index.

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

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

extend() Function:
The extend method takes a second list and adds each of its objects to an existing list. This method is very useful for combining two lists into one.

append() Function:
append() adds its argument as a single element to the end of a list. The length of the list itself will increase by one.

insert() Function:
The insert method inserts an object into an existing list before a specified index value.

Know Program

How to Copy a Data Structure

To solve this problem, lists come with a copy() method, which does the right thing. Take a look at how copy works.

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

b = a.copy()

print(b)
```

Output:
[1, 2, 3, 4]

Note:- Don't use assignment operator to copy list. Use the copy method instead.

Know Program

Tuples

⇒ A tuple is an immutable (Can't change) data type in Python.

⇒ Tuples are like a lists, except once created they can't change. Tuples are constant lists.

```
a = ()          => Empty tuple
a = (1,)        => Tuple with only one element needs a Comma.
a = (1, 7, 2)   => Tuple more than one element
```

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 occurrence of 1 in a.

Know Program

Day: 32
Date: / /

Dictionary:

⇒ Dictionary is a Collection of Key-Value pairs

⇒ Insertion Order is NOT Maintained in the Case of Dictionaries. So, use keys to access data in a dictionary.

Syntax:

```
a={"key":"Value", "Deepak":"Code", "marks":100, "list":[1,2,9]}
```

a["key"] ⇒ print "Value"

a["list"] ⇒ print [1,2,9]

Properties of Python Dictionaries.

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

Dictionary Methods

Consider the following dictionary

```
a={"name":"yogesh", "from":"india", "marks":[92,93,90]}.
```

1. a.items(): returns a list of (key, value) tuples.

Know Program

2. a.key(): 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 e.g. "yogesh" is returned here).

Dictionary Methods Table:

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromskeys()	Returns a dictionary with the specified keys and value
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
Setdefault()	Returns the value of the specified key. If the key doesn't exist: insert the key with the specified value.
Values()	Returns a list of all the values in the Dictionary

Know Program

Sets

Set is a collection non repetitive element.

```
S = set()
S.add(1)
S.add(2)
```

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

Properties of sets

- Sets are unordered.
- Sets are indexed.
- There is no way to change items in sets.
- Sets cannot contain duplicate values.

Operation 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.

Know Program

4. s.clear(): Emplies 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 contain only items in both sets.
   => {8}

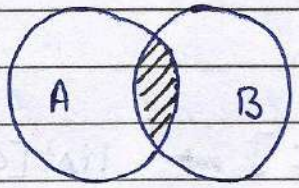
Commonly used set operation

Union: Union of A and B is a set of all the element from both sets. Union is performed using | operator.

```
1  A = {1,2,3,4}
2  B = {3,4,5,6}
3  print (A|B)
```

Output = {1,2,3,4,5,6}

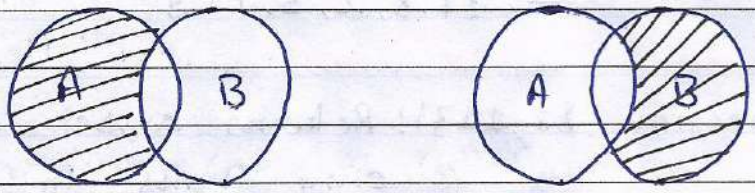
Intersection: Intersection of A and B is a set of element that are common in both sets. The intersection is performed using & operator.



intersection of A and B

Know Program

Difference: Difference of A and B (A-B) is a set of elements that are only in A but not in B. Similarly, (B- A) is a set of an element in B but not in A.



(A-B) (B-A)

Sequence Operations:

- Concatenation

['1','b', 2.5] + ['d'] → [1,'b',2.5,'d']

- Repetition

['a','b',2.5] * 2 → ['a','b',2.5,'a','b',2.5]

- Slicing

list = ['a','b','c','d'] → list[1:3] → ('b','c')

- Indexing

list = ['a','b','c'] → list[0] → 'a'

Know Program

Function

⇒ A Function is a group of Statement performing a specific task.

⇒ when a program gets bigger in size and its complexity grows, it gets difficult for a programm-er to keep track on which piece of code is doing what!

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

Advantage of Functions

⇒ They break large computing into smaller ones. it means with the help of function we modularize the program / problem.

⇒ Code Reusability / Code optimization / increase Reusability, By Creating functions in Python, you can call it many times. So don't need to write much code. it means By using the function avoid code repetitions.

Know Program

Example and Syntax of a function

The Syntax of a function looks as follows.

```
def fun1():
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:

fun1() -> This is called function call

Function definition:
The part containing the exact set of instruction which are executed during the function call.

Types of Functions in Python

There are two types of function in Python.

1. Build in Functions
2. User defined functions

Know Program

➤ Example of build in Function includes len(), print() rangel), etc.

The fun1() 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 parenthesis. A Function Can also return Values as Shown below:

```
Def greet(name):
 gr = "Hello" + name
 return gr "Deepak" is passed to greet in name
 a= greet("Deepak")
 a will now Contain "Hello Deepak"
```

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.

e.g.

```
def greet(name="Stranger"):
# function body
greet -> Name will be "Stranger" in function body (default)
greet("Deepak") -> Name will be "Deepak" in function
                   body (passed)
```

Know Program

Ways to Define A FUNCTION

There are four ways to define function

1. Takes Nothing, Returns Nothing
2. Takes Something, Returns Nothing
3. Takes Nothing, Return Something
4. Takes Something, Returns Something

1. Take Nothing, Returns Nothing.

```
def Sum():
print ("Please Enter Two number")
    Var 1 = int (input())
    Var2 = int (input())
    result = Var 1 + Var2
print ("The Result is", result)
    Sum()
```

2. Take Something Return Nothing

```
def Sum ( Var1 , Var2):

 result = Var 1 + Var 2

 print ("The Result is ", result)

 print ("Please Enter Two number")

 num 1 = int ( input ())

 num 2 = Int ( input ())

  Sum (num 1 , num 2)
```

3. Take Nothing Return Something

```
def Sum ():

print ("Please Enter Two number")

 Var 1 = int (input ())

 Var 2 = int (input ())

 result = Var 1 + Var 2

 return result

  Sum ( )
```

Know Program

4. Take Something Return Something

```
def Sum(Var1, Var2)

result = Var1 + Var2

return result

print ("Please Enter Two number")

num1 = int (input())

num2 = int (input())

Sum (num1, num2)
```

Note:- When function returns nothing that time it returns None.

Default Argument

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

```
# This demo is for Default Arguments

def printInfo (name, age=35):
    print ("Name: ", name)
    print ("Age ", age)
```

Know Program

Python Keyword Arguments

When we call a function with some values, these values get assigned to the arguments according to their position. Python allows functions to be called using keyword argument. when we call function in this way, the order (position) of the argument can be changed.

```
# This demo is for keyword arguments

def greet(name, msg):
print("Hello", name + ', ' + msg)
greet("Dev", "Good morning!")
greet(name = "Dev", msg = "How do you do?")
greet(msg = "How do you do?", name = "Dev")
```

Variable-length Arguments

You may need to process a function for more arguments than you specified while defining the function. These arguments are called Variable-length arguments and are not named in the function definition, unlike required and default arguments.

Know Program

Syntax :-

```
def functionname ([formal_args,] * var_args_tuple):
    function_suite
    return [expression]
```

An asterisk (*) is placed before the variable name that holds the values of all non-keyword variable arguments. This tuple remains empty if no additional are specified during the function call.

```
def printValue (arg1, *vartule):
    print (arg1)
    for v in vartule:
    # print (v)
    print (v, end = '\t')
    #printValue (1)
    #printValue (1, 2, 3, 4)
    printValue (1, 2, "Three", 4.0)
```

Know Program

Recursion:

Recursion is a function which calls itself. it is used to directly use a mathematical formula as a function.

e.g.

factorial = n* factorial (n-1)

This function can be defined as follows:

```
def function (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.
```

The work as follows:

Factorial (4)
[Funtion Called]
4* Factorial (3)
4* [3* Factorial (2)]
4* 3* [2* Factorial (1)]
4* 3* 2* [1] [function returned

Note:- • The programmer needs to be extremely carefully while working with recursion to ensure that the Function doesn't infinitely keep calling itself.

• Recursion is sometimes the most direct way to code an algorithm.

Know Program

Anonymous Functions / Lambda Functions

These functions are called anonymous because they are not declared in the standard manner by using the def keyword. You can use the lambda keyword to create small anonymous functions.

Syntax :-

```
lambda [arg1 [,arg2, ... argn]]: expression
```

```
# This demo is for Anonymous Function

Sum = lambda arg1, arg2 : arg1 + arg2
```

bytes() Function:

The bytes() Function returns a bytes object.

it can convert objects into bytes objects, or create empty bytes object of the specified size.

The difference between bytes() and bytearray() is that bytes() returns an object that cannot be modified, and bytearray() returns an object that can be modified.

Syntax:

```
bytes (x, encoding, error)
```

Know Program

Date: / /

bytearray() Function:

The bytearray() function returns a bytecorray object.
it can convert objects into bytearray objects, or create empty bytearray object of the specified size.

Syntax:

```
bytearray (x, encoding, errar)
```

Parameter	Value
x	A source to use when creating the bytes object. if it is integer, an empty bytes object of the specific size will be created. if it is a string, make sure you specify the encoding of the source.
Encoding	The encoding of the string.
Errar	specified what to do if the encoding fails.

Know Program

First class Functions:

First class objects in a language are handled uniformly throughout. They may be stored in data structures, passed as arguments, or used in control structures. A programming language is said to support first-class functions if it treats functions as first-class objects. Python supports the concept of first class functions.

Properties of first-class functions

- A function is an instance of the object type.
- You can store the function in a variable.
- You can pass the function as a parameter to another function.
- You can return the function from a function
- You can store them in data structures such as hash tables, lists etc.

Function is objects

Python functions are first-class objects.

Know Program

Date: / /

for example

```
def shout(text):
return text.upper()
print(shout('Hello'))
yell = shout
Print(yell('Hello'))
```

Function can be passed as arguments to other function

Because Functions are objects we can pass them as arguments to other functions. Functions that can accept other functions as arguments are also called higher-order functions.

e.g.

```
def shout(text):
return text.upper()
def whisper(text):
return greet(func):
    # Staring = Func("Hi, I am created by a
                function passed as an argument.")
greeting = func("Hi, I am created by a function
            passed as an argument.")

print greeting
greet(shout)
greet(whisper)
```

Know Program

Functions can return another function
Because functions are objects we can return a function from another function.

e.g.
```
def create_adder(x):
    def adder(Y):
    return x+y
    return adder
    add_15 = create_adder(15)
```

Modules

- Modules refer to a file containing Python statements and definitions.
- A file containing Python code, for e.g. GetSum.py is called a module and its module name would be GetSum.
- we use modules to break down large programs into smaller files.

e.g. GetSum.py

```
def add(a,b):
    result = a+b
    return result
```

Know Program

Date: / /

How to Import Module in Python

We use the import keyword to do this. To import our previously defined module GetSum we type the following in the Python in the Python prompt.

Ex. (Tester.py)

```
Import GetSum
    GetSum.add(4, 5.5)
            # This program use GetSum Module
```

```
Import GetSum
Var1 = int(input("Please Enter first Number"))
Var2 = int(input("Please Enter second Number"))
    Receive = GetSum.add(Var1, Var2)
    print("The Result is, "Receive")
```

Variable in Module:

The module can contain function, as already described, but also variables of all types (arrays, dictionaries, objects, etc):

e.g.

```
Person1 = {
        "name": "John",
        "age": 36,
        "country": "Nurway"
}
```

Save this code in the file mymodule.py

Know Program

Example:-
Import the module named mymodule, access the person1, dictionary

```
Import mymodule

a = mymodule.person1["age"]
print(a)
```

Python Dates
A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.

e.g. Import the datetime module and display the current date:

```
import datetime

x = datetime.datetime.now()

print(x)
```

Data Output
When we execute the code from the example above the result will be

```
2020-12-13 18:19:35.869279
```

Know Program

Date.

e.g.
```
Import datetime

x = datetime.datetime.now()

Print(x.year)
Print(x.strftime("%A"))
```

Creating Data objects:
To create a date, we can use the datetime() class (constructor) of the datetime module.
The datetime() class requires three parameters to create a date: year, month, day.

e.g.
```
import datetime

x = datetime.datetime(2020,5,17)

print(x)
```

Calendar Module In Python
Calendar module in Python has the Calendar class that allows the calculations for various task based on date, time, month and year.

e.g.
```
import Calendar
a1 = Calendar.year(2020, 12)
print(a1)
```

Know Program

Output:-

```
    December 2020
Mo Tu We Th Fr Sa Su
          1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

Ques Difference between import and from in Python?

Ans Python's "import" loads a Python module into its own namespace, So that you have to add the module name followed by a dot in front of references to any names form the imported module that you refer to.

e.g.

```
import math
print("The Value of pi is", math.pi)
```

Output:

The Value of pi is 3.141592653589793

"from" loads a Python module into the current namespace, So that you can refer to it without the need to mention the module name again.

```
from math import pi                    #import only pi from
print("The Value of pi is", pi)          math module
```

Know Program

Package in Python

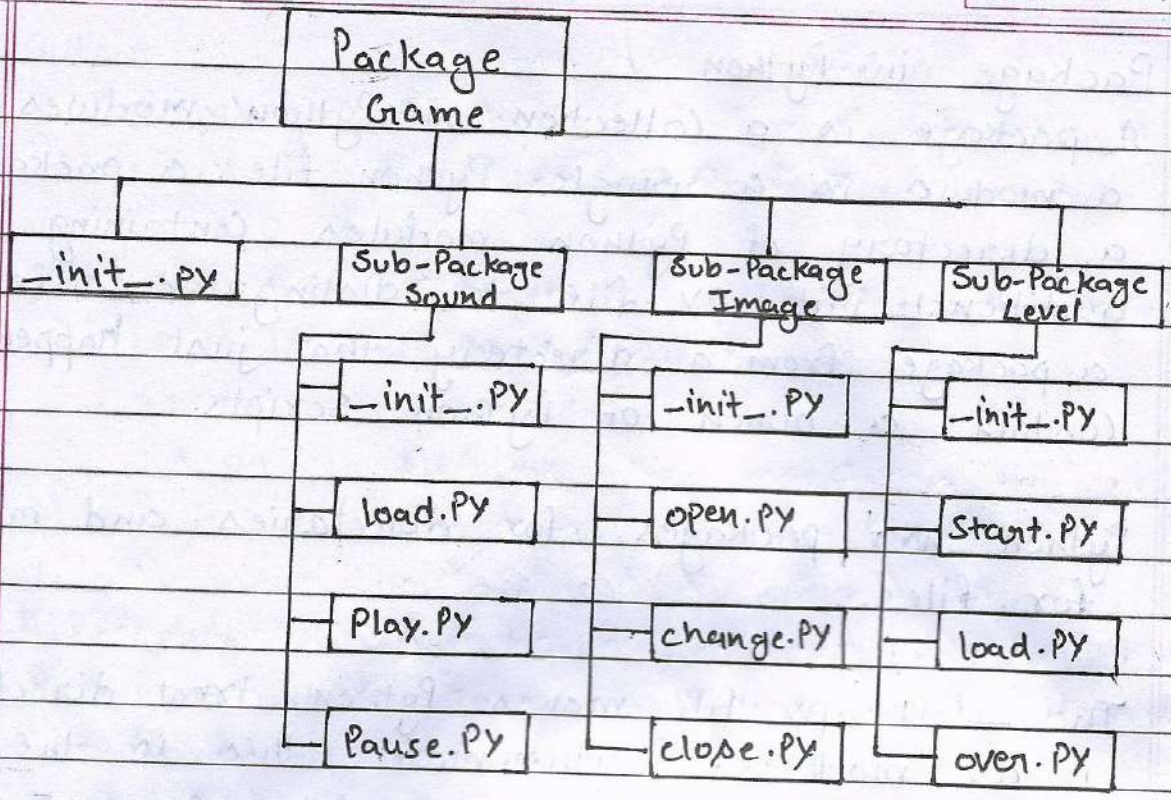
A package is a collection of Python modules while a module is a single Python file, a package is a directory of Python modules containing an additional __init__.py file, to distinguish/differentiate a package from a directory that just happens to contain a bunch of Python scripts.

Python has packages for directories and modules for files.

Note:- The __init__.py file makes Python treat directories it as modules. Furthermore, this is the first fide to be loaded in a module, so you can use it to execute code that you want to run each time a module is loaded, or specify the Submodules to be exported.

A package is basically a directory with Python files and a file with the name __init__.py. Similar, as a directory can contain sub-directories and files, a python package can have sub-package and modules.

Know Program



Importing module from a package:
We can import modules from packages using the dot (.) operator.

e.g. if we want to import the Start module
Then
import Game.Level.Start

Now if this module contains a function named Select_difficulty(),
we must use the full name to reference it.

Game.Level.Start.Select_difficulty(2)

Know Program

Math Module

The math module is a standard module in Python and is always available. To use mathematical functions under this module, you have to import the module using import math.

it gives access to the underlying C library functions

e.g.
```
#Square root Calculation
import math
math.Sqrt(4)
```

Note:- This module does not support complex data types. The cmath module is the complex counterpart.

List of Functions in Python Math Module

Function	Description
ceil(x)	Returns the smallest integer greater than or equal to x.
copysign(x,y)	Returns x with the sign of y
fabs(x)	Returns the absolute value of x
factorial(x)	Returns the factorial of x

Know Program

function	Description
floor(x)	Returns the largest integer less than or equal to x.
fmod(x, y)	Returns the remainder when x is divided by y
frexp(x)	Returns the manitissa and exponent of x as the pair (m, e)
fsum(iterable)	Returns an accurate floating point sum of values in the iterable.
isfinite(x)	Returns True if x is neither an infinity nor a NaN (Not a Number)
isinf(x)	Returns True if x is a positive or negative infinity
isnan(x)	Returns True if x is a NaN
ldexp(x, i)	Returns x * (2 ** i)
modf(x)	Returns the fractional and integer parts of x
trunc(x)	Returns the truncated integer value of x

Know Program

exp(x)	Returns e**x
expm1(x)	Returns e**x - 1
log(x[,b])	Returns the logarithm of x to the base b (default to e)
log1p(x)	Returns the natural logarithm of 1+x
log2(x)	Returns the base-2 logarithm of x
log10(x)	Returns the base-10 logarithm of x
Pow(x,y)	Returns x raised to the power y
Sqrt(x)	Returns the square root of x
acos(x)	Returns the arc cosine of x
asin(x)	Returns the arc sine of x
atan(x)	Returns the arc tangent of x
atan2(y,x)	Returns atan(y/x)
cos(x)	Returns the cosine of x
hypot(x,y)	Returns the Euclidean norm, Sqrt(x*x + y*y)

Know Program

Sin(x)	Returns the Sine of x
tan(x)	Returns the tangent of x
degrees(x)	Converts angle x from radians to degrees.

Python JSON :-

JSON (JavaScript object Notation) is a popular data format used for representing structure data. it's common to transmit and receive data between a server and web applications in JSON format.

In Python, JSON exist as a string.
for example

p= "{"name", Bob", "languages":["Python","Java"]}"

it is also common to store a JSON object in a file.

Import JSON module :-

To work with JSON (String, or file containing JSON object), you can use Python's json module. You need to import the module before you can use it.

Know Program

Import json

e.g.-

Python JSON to dict:-

```
import json

person = '{"name":"Bob", "Language": ["English", "Fench"]}'

person_dict = json.loads(person)

print(person_dict)

print(person_dict["Languages"])
```

Output:-

```
{'name':'Bob','Language': ['English', 'Fench']}

['English', 'Fench']
```

We can parse a JSON string using json.loads() method. The method returns a dictionary.

Here person is a JSON string, and person_dict is a dictionary.

Python Converts to JSON string:- You can Convert a dictionary to JSON string using json.dump() method.

Know Program

e.g.
```
import json

Person_dict = {"name": "Bob", "age": 12" children": None}

Person_json = json.dumps (person_dict)

print (person_json)
```

Output:-

```
{"name": "Bob", "age": 12, "children": null}
```

Here is a table showing the Python objects and their equivalent conversion to JSON:-

PYTHON	JSON EQUIVALENT
Dict	object
List, tuple	array
Str	string
Int, float, int	number
True	true
False	false
None	null

Know Program

Writing JSON to a file

```
import json

person_dict = {"name": "Bob", "Language": ["English", "French"],
               "married": True, "age": 32}

with open("person.txt", "w") as json_file:
    json.dump(person_dict, json_file)
```

In the above program, we have opened a file named person.txt in writing mode using "w". If the file doesn't already exist, it will be created. Then, json.dump() transforms person_dict to a JSON string which will be saved in the person.txt file.

When you run the program, the person.txt file will be created. The file has following text inside it

```
{"name": "Bob", "Languages": ["English", "French"],
 "married": true, "age": 32}
```

Know Program

Regular Expressions in Python:-

- Regular Expressions is a special Text string for Describing a Search Pattern.

Regular expressions is a language. Not a programming language.

There's a specific syntax and keywords that you can use in regular expressions to express what you want.

we use regular expressions to specify patterns in text.

e.g.1 a1 a4 a3 a56 here is a pattern of 'a' followed by a number, followed by a space.

2. Abhinav, Rajat, Suraj
here is a pattern which is set of characters, followed by a comma, followed by a space.

Advantages of using Regular Expressions:-

1. A regular expression is a method / approach used in programming for pattern matching.

2. Regular expressions provide a flexible and concise means to match strings to text.

Know Program

for example, a regular expression could be used to search through large volumes of text and change all occurences of "cat" to "dog".

Applications of Regular Expressions:-

1. Validations - To develop registration from validation noramlly we use RE.

2. Pattern Matching Applications - CTRL+F in windows os and grep command in Unix.

3. Translator like compilers, interpreters, assemblers uses regular expression.

Important function of regular expressions:-

Function compile():

Regular Expressions are compiled into pattern objects, which have methods for various operations such as searching for pattern matches or performing string susbstitutions.

e.g.

```
import re

Pattern = re.compile("Python")

print(type(pattern))
```

Know Program

Output:-
<class 're.Pattern'>

Finditer() Function:-
Re.finditer(pattern, string, flags=0)

Return an iterator yielding MatchObject instances over all non-overlapping matches for the RE pattern in string.

The String is Scanned left-to-right, and matches are returned in the order found.

Matches = pattern.finditer("Learning Python is very Easy")

1. Start(): Start index of the match.
2. end(): end+1 index of the match.
3. group(): Returns matched string.

e.g.

```
import re

Count=0

Pattern=re.Compile("is")
```

Know Program


```
matcher = pattern.finditer("mississippi")

    for match in matcher:

        count += 1

print("Match is available at start index:", match.start
                                                    ())

    print("The number of Occurences:", count)
```

Output:-

Match is available at start index: 1

Match is available at start index: 4

The number of Occurences: 2

Meta characters:-

- \ Used to drop the special meaning of character
- [] Represent a character class
- ^ matches the begining
- $ Matches the end
- . Matches any character except newline

Know Program

- ? matches zero or one occurence.
- | means OR (Matches with any of the characters separated by it.
- * Any number of occurences (including 0 occurrences)
- + One are more occurrences
- {} Indicate number of occurrences of a preceding RE to match.
- () Enclose a group of Res

The four most important Component of regex:-

- *'.';
- *'+';
- *'*';
- *'?'

⇒ The '.' means "anything"; Such as a letter, number, symbol, space, etc... * but not newline characters *.

Know Program

=> So '.+' means "one or more of anything".

=> '.*' means "zero or more of anything".

=> '.?' means "zero or one of anything".

We can use character classes to search a gp of character

1. [abc] -> Either a or b or c
2. [^abc] -> Except a and b and c
3. [a-z] -> Any lower case alphabet symbol
4. [0] -> Any digit from 0 to 9.

e.g.

```
import re
matcher = re.finditer("[^abc]", "a7b@K9z")
for m in matcher:
    print(m.start(), "...", m.group())
```

Output:-

```
1 ...7
3 ...@
4 ...K
```

Know Program

5 9

6 ....z

Predefined character classes:

\s → space character

\d → any digit

\D → except digit

\w → Any word character (alpha number character)
[a-zA-Z0-9]

. → Every character

e.g.
```
import re
matcher = re.finditer("\d", "a7b@K9z")
for m in matcher:
    print(m.start(), "...", m.group())
```

Output:-

1. ...7

5 9

Know Program

Quantifier

Quantifiers can be used to specify the number of occurrences to match

a -> Exactly one "a"

a* -> atleast one 'a'

a(m,n) -> Minimum m number of a's and Maximum n number of a's

e.g.

```
import re

matcher = re.finditer("a{2,3}", "abaabbaaa")

for m in matcher:

    print(m.start(), '...', m.group())
```

Output:-

2 ....aa

6 ....aaa

Know Program

Python Library

After Modules and Python Packages, we shift our discussion to Python Libraries. This Python Library Tutorial, we will discuss Python Standard Library and different libraries offered by Python Programming Languge : Matplotlib, scipy, numpy, etc.

What is Python Library?

A Python library is a reusable chunk of code that you may want to include in your program / projects. Compared to languages like c++ or c, Python libraries do not pertain to any specific context in Python. Here, a 'library' loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like rubygems or npm.

Python Standard Library

The Python Standard Library is a collection of exact syntax, token, and semantics of Python. it comes bundled with core Python distribution. we mentioned this when we began with an introduction.

Know Program

it is written in C, and handles functionally like I/O and other core modules. All this functionally together makes Python the language it is. More than 200 core modules sit at the heart of the standard library. This library ships with Python. But in addition to this library, you can also access a growing collection of several thousand components from the Python Package Index (PyPI).

Important Library in Python

1. Mathplotlib:
Matplolib helps with data analyzing, and is a numerical plotting library. we talked about it in Python for Data Science.

2. Pandas:
It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, build on top of the Python programming language.

3. Requests:
Requests is a Python Library that lets you Send HTTP/1.1 requests, add headers, form data, multipart files, and parameters with simple Python dictionaries.

Know Program

4. Numpy:
It has advanced math functions and a rudimentary scientific computing package.

5. SQLAlchemy:
SQLAlchemy is a library with well-known enterprise-level patterns. it was designed for efficient and high-performing database-access.

6. Beautiful Soup:
It may be a bit slow, Beautiful Soup has an excellent XML- and HTML-parsing library for beginners.

7. Pyglet:
Pyglet is an excellent choice for an object--oriented programming interface in developing games.

8 h. Scipy:
It has a number of user-friendly and efficient numerical routiners.

Know Program

Object - oriented Programming

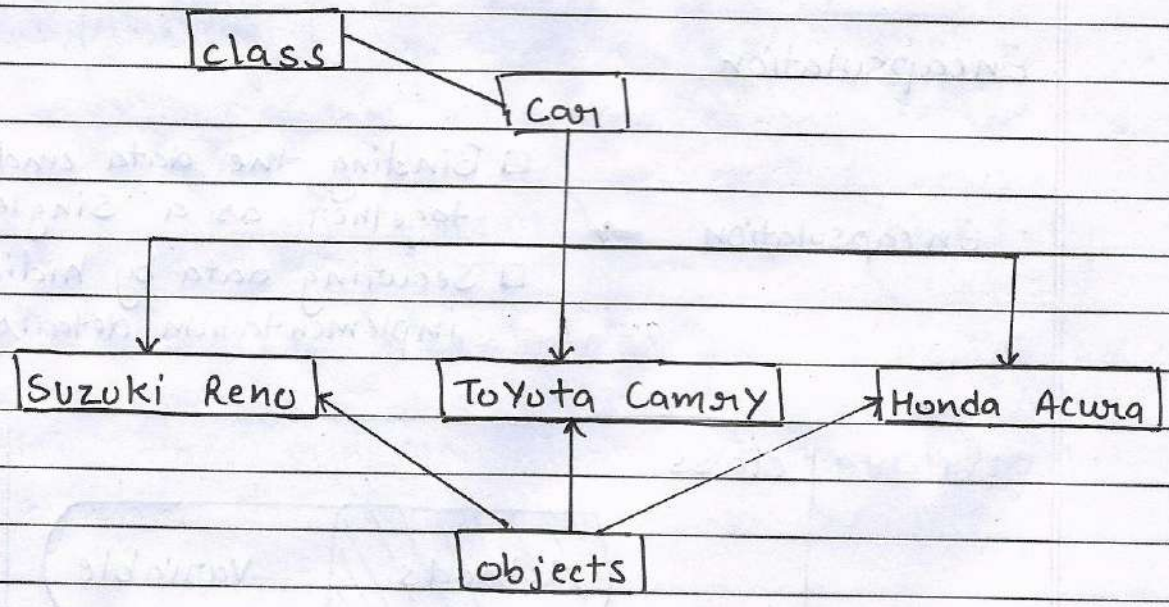
Solving a problem by Creating objects is one of the most popular approches in programming.. This is called object oriented Programming.

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.

class Employee: [class name is written in Pascal case]

#methods & Variable

Know Program

Object

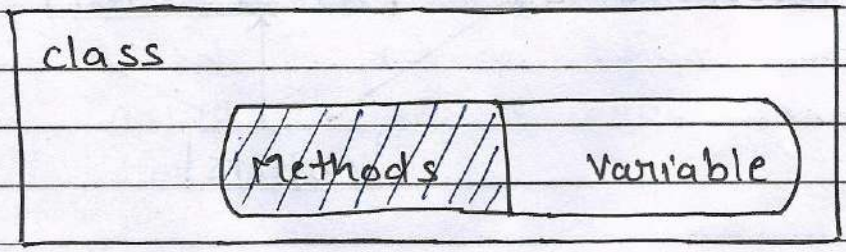
An object is an instantiation of a class. When class is defined, a template (into) is defined. memory is allocated only after object instantiation.

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

Abstraction & Encapsulation

Encapsulation

Encapsulation →
- Binding the data and code together as a single unit.
- Securing data by hiding the implementation details to user.



→ Encapsulation involves packaging one or more components together.

→ Encapsulation is a process wrapping data and related behaviour together into a single unit.

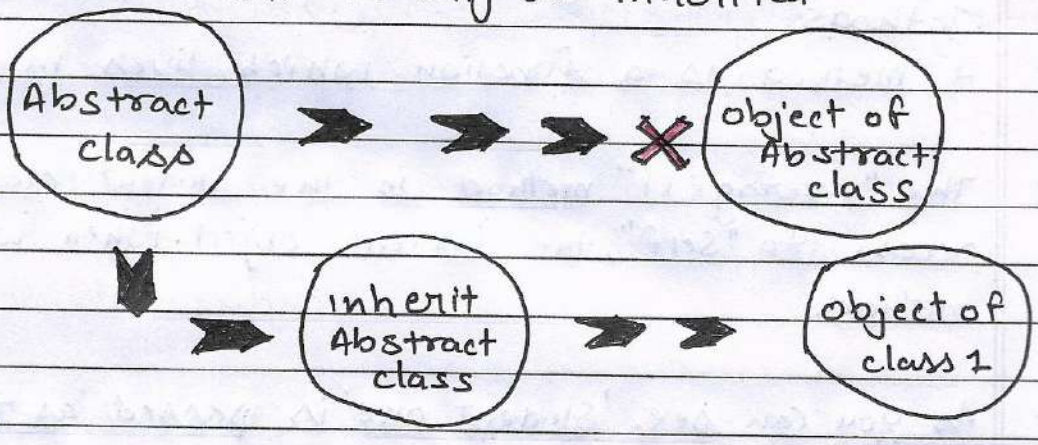
Know Program

e.g. Capsule that is mixed of several medicines.

→ Encapsulation is defined as the process of enclosing one or more items within a physical or logical package.

Abstraction

- Abstract class cannot be instantiated
- it can only be inherited



- Hides the implementation details and only provides the functionality to the use
- You can achieve abstraction using Abstract classes and interfaces.

Need of class object

we've looked at dictionaries as able to represent what_something_is.

Know Program

e.g.
```
{
'name': 'Dev Singh'
'grades': [70, 88, 90, 99]
}
def averge_grade (student):
return sum(student['grades'])/len (student['grades'])
result = averge_grade (my_student)
print (result)
```

Methods

- A method is a function which lives in a class.
- The "average()" method in the Student class also has access to "self", the current object. When we call the method.
- As you can see, 'student_one' is passed as the first argument (and that is what 'self' is in the method definition).

e.g.
```
"""
def averge(self):
return sum (self.grades)/len (self.grades)
```
So again, because 'self' is 'student_one', 'self.grades' is 'student_one.grades'.
Thus:
* The sum of 'self.grades' is the sum of '[70, 88, 90, 99]: 347

Know Program

* The length of 'self.grades' is 4.
The result will be '86.75'.

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
A attributes that belong to the class rather than a particular object.

e.g.

```
class Employee:

    Company = "Google"      [Specific to each class]

abhinav = Employee()        object instantiation

abhinav.Company

Employee.Company = "innovation"   changing class attribute
```

Know Program

Instance Attributes

An attributes that belongs to the Instance (object)
Assuming the class from the previous example!

```
abhinav.name = "abhinav"

abhinav.salary = "30K"
```

Note: Instance attributes take preference over class attributes during assignment & retrievals.

```
abhinav.attribute1 = 1. Is attributes1 present in object?

                        2. Is attributes1 present in class?
```

Self Parameter

Self refers to the Instance of the class. it is automatically passed with a function call from an object.

```
abhinav.getSalary()
```

The function getsalary is defined as:

```
class Employee.

    Company = "Google"

    def getSalary(self):

        print("Salary is not there")
```

Know Program

Static Method

Sometimes we need a function that doesn't use the self-parameter. we can define a static method like this

```
@staticmethod
def greet():
    Print("Hello user")
```

__init__() Constructor

__init__() is 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

e.g.

```
class Employee

    def __init__(self, name):
        self.name = name

    def getsalary(self):

abhinav = Employee("abhinav")
```

Know Program

Inheritance

Inheritance is way of creating a new class from an existing class.

Syntax:

```
class Employee:
    #code                    Base class
    ...

class programmer (Employee):
    #code                    Derived or child class
```

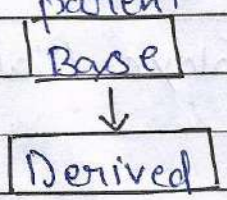
we can use the methods and attributes of Employee in Programmer object.
Also, we can overwrite or add new attribute and methods in Programmer class.

Type of Inheritance

1. Single Inheritance
2. Multiple Inheritance
3. Multilevel Inheritance

Single Inheritance

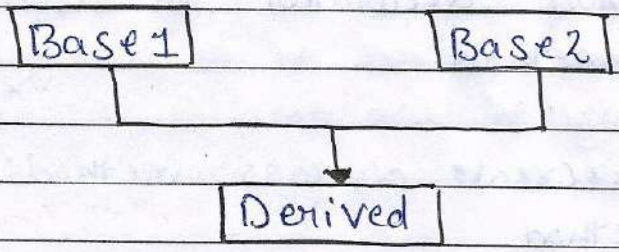
Single Inheritance occurs when child class inherits only a single parent class



Know Program

Multiple Inheritance

Multiple Inheritance occurs when the child class inherits from more than one parent class.



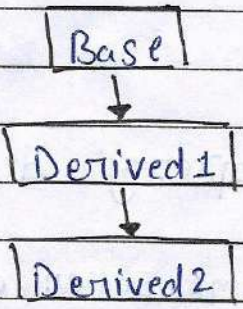
Multilevel Inheritance

when a child class becomes a parent for another child class.

Super() Method

Super() method is used to access the methods of a Super class in the derived class.

Super().__init__() Calls constructor of the base class



Know Program

Class Methods

A class Method is a method which is bound to the class and not the object of the class.
@classmethod decorator is used to create a class method.

Syntax to create a class method:

```
@classmethod
def (cl, p1, p2):
    ...
```

@property decorators

Consider the following class

```
class Employee
    @property
    def name (self):
        return self.ename
```

if e = Employee() is an object of class employee, we can print (e.name) to print the ename / call name() function.

@.getters and @.setters

The method name with @property decorder is called getter method.
we can define a function + @name.setter decorator like below:

```
@name.setter
def name (self, value):
    self.ename = value
```

Know Program

operator overloading in Python

operator is python can be overloaded using dunder methods

These methods are called when a given operator is used on the objects.

operator is python can be overloaded using the following methods:

p1 + p2 => p1.__add__(p2)

p1 - p2 => p1.__sub__(p2)

p1 * p2 => p1.__mul__(p2)

p1 / p2 => p1.__truediv__(p2)

p1 // p2 => p1.__floor__(p2)

Other dunder / magic methods in Python

__str__() => used to get what gets display upon calling str(obj)

__len__() => used to set what gets displayed upon calling __len__() or len(obj).

Know Program

File Input/Output:

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.

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 parameter. filename and mode.

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

- open is a built-in function
- filename
- mode of opening (read mode)

Know Program

Reading a file in Python

```
f= open ("this.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 specific the number of characters in read() function: f.read(2)
(Read 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() → Read 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

Know Program

'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!

```
f = open ("this.txt", "w")

f.write("This is nice") → Can be multiple times

f.close()
```

with Statement

The best way to open and close the file automatica-lly is the with statement

```
with open ("this.txt") as f:

        f.read()
```

Don't need to write f.close() as it is done automatically.

Exception Handling

There are many build-in exceptions which are raised in Python when Something goes wrong. Exception in Python can be handled using a try Statement. The code that handles the exception is written in the except clause.

Know Program

e.g.
```
try:
    #code
except Exception as e:
    print(e)
```

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

we can also specify the exception 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.

try and else clause

Sometimes we want to run a piece of code when try was Succesful.

e.g.
```
try:
    #Some code
except:
    #Some code
```

Know Program


```
else:
    #Some code
```

try with finally

Python offers a finally clause which ensures executions of a piece of code inspective of the exception.

eg.
```
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 run. if the module is being run directly from the command line, the __name__ is set to string "__main__"

Thus this behavior is used to check whether the module is run directly or imported to another file.

Know Program

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)
```

↓

Print the items of list1 with index!

List Comprehensions

List Comprehension is an elegant way to create list based on existing lists

```
list1 = [1, 7, 12, 11, 22]

list2 = [i for item in list1 if item > 8]
```

Vintual Environment

An environment which is some as the system interpreter but is isolated from the other python environment on the System.

Know Program

Installation

To use virtual environments, we write

pip install virtualenv → install the package

we create a new environment using:

virtualenv myprojectenv → Create a new venv

The next step after creating the virtual environment is activating it.

we can now use this virtual environment as a separate python installation.

PiP freeze Command

pip freeze returns all the package installed in a given python environment along with the versions

"pip freeze > requirements.txt"

The above commond creates a file named requirements.txt in the same directory containing the output of pip freeze

we can distribute this file to other users and they can recreate the same environment using:

pip install -v requirement.txt

bin method (strings)

Create a string from iterable objects

Know Program


```
l=["apple","mango","banana"]
",and,".join(l)
```

The line will return "apple, and, mango, and, banana"

format method (String)

formats the values inside the string into a desired output

```
template.format(p1,p2...)
                ↳(arguments)
```

Syntax for format looks like:

```
"{} is good{}".format("Abhinav","boy")    1
"{1} is good{0}".format("Abhinav","boy")  2
```

Output for 1

Abhinav is a good boy

Output for 2

boy is a good abhinav

Map, filter & Reduce

Map applies a function to all the items in an input_list.

Know Program

Syntax:

map(function, input_list)

filter (create a list of items for which the function returns true.

list (filter (function))

Can be a lamba function

Reduce applies a rolling computation to Sequential pair of elements. from functools import reduce

Val = reduce (function, list1)

Can be a lamba function

if the function computer Sum of two numbers and the list [1, 2, 3, 4]

1 2 3 4

3 3 4 ⇒ Sequential Computation

6 4

10

Know Program

Date:

Directory

- if there is a large number of file to handle in your program, you can arrange your code within different directories to make things more manageable.
- A directory or folder is a collection of files and sub directories
- Python has the os module, which provides us with many useful methods to work with directories (and files as well)

Logging

- Logging is a means of tracking events that happen when some software runs
- Logging is important for software developing, debugging and running.
- if you don't have any logging record and your program crashes, there are very little chances that you detect the cause of the problem
- And if you detect the cause, it will consume a lot of time.
- with logging, you can leave a trail of breadcrumbs so that if something goes wrong, we can determine the cause of the problem.

Know Program

Get Current Directory:
we can get the present working directory using the getcwd() method.

e.g.
```
import os
print(os.getcwd())
```

Changing Directory:-
we can change the current working directory using the chdir() method.

e.g.
```
import os os.chdir('e:/Demos')
print(os:getcwd())
```

Level of log message:

Debug:
These are used to give detailed information, typically of interest only when diagnosing problems.

info:
There are used to confirm that things are working as expected.

Warning
These are used an indication that something unexpected happened, or indicative of some problem in the near future.

Know Program

Error:
This tells that due to a more serious problem, the Software has not been able to perform some function.

Critical:
This tells serious error, indicating that the program itself may be unable to continue running.