— Guido Van Rassum [1990]

→ Python is general purpose, interpreter and OOP language.
[objectorientedprogramming]

→ It is expressive and has dynamic typing.
↓
no need of declaration

→ It has less code.

→ Automatic memory management

→ 3.6 is the latest version of python.

→ Python is now being developed and maintained by a large team of volunteers and is Available for free from "python Software foundation."

→ It was named after the popular British comedy troupe "Monthly python flying circus".

→ Python is now being used in Googlesearch, youtube, Mozilla and projects in NASA and in transaction processing at the Newyork stock exchange.

Features:-

→ Python incorporates modules, exceptions, classes and dynamic typing.

→ Python is portable (used to run any OS)

→ Python has ~~many~~ interfaces to many systemcalls and libraries

→ The language comes with large standard library.

→ Deals with complex data.

→ It is an opensource. [freely available]

Uses

→ Make games

→ Solve web applications

→ Develop business solutions

→ To make internal tools.

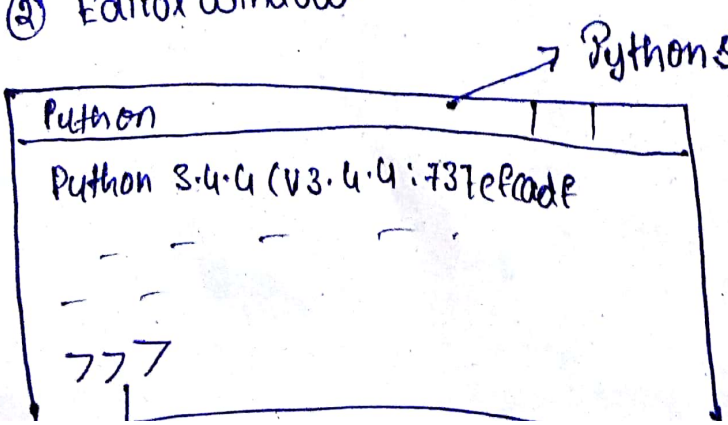
→ Used heavily in scientic fields for academic research & applied work.

IDLE (Interactive development learning Environment)

→ It has mainly two windows

① Shell window

② Editor window.



→ IDLE uses colour code syntax to highlight the code.

Strings → Green
Errors → Red
keywords → Orange
Result → blue

→ with colour coding you can easily determine the use for each kind of text in the application.

→ IDLE also knows about python indentation syntax, which requires code-blocks.

IDLE has 2 windows:

① Shell window
② Editor window.

Shell window:-

→ It is used to test and learn python language.

→ The python shell window (interactive intrepreter) with coloring of code i/p, o/p & error messages.

Editor window:-

→ It is used to create the applications and run the applications.

→ It is possible to have multiple editors, windows simultaneously.

Configure IDE:-

we can configure in both editor & shell window.

→ Select options menu → configure IDLE → open a configuration dialog and change prefers for following....

fonts, Indentation, keybindings, text colour theme, startup

Open

Start → programs → python 34 → idle python 3.4

[we have dotpd versions till 3.6].

To close:

ctrl+d (or) exit

Python as a calculator:

Ex:- 3+6 → 9

10-5 → 7

8*7 → 56

8/2 → 4.0 → float value

6//3 → 2 [we'll get integer when we use //]

5**2 → 25 [** → power]

5**4 (5**2*2) → 625

As a squareroot 625**(1/2) → 25

8**(1/3) → 2.

Python to evaluate the expression we use the following.

PEDMAS → Div, Add, Sub, Mul

↓ ↓

Exponen

Parenthesis

PEDMAS

→ If same operator is there then it considers left → right

Ex:- 5+5*5 → 30

(5+5)*5 → 50

learn and Test

→ learn → test and write the programs.

→ Default Module

↓

builtins - -

① functions and classes

↙

→ It contains arguments

→ Print() func doesn't have any arguments.

→ It returns some value.

Print()

→ It prints an emptyline.

Syntax :-

Print ("strings")

Ex :- Print ("I am a girl")

→ I am a girl.

① It is a sequence of characters.

② Represent strings either in " " (or) ' ' (or) """ """ → It represents long strings.

→ Print ('cat computers')

→ cat computes

Using combination of " " & ' ' :-

Print ("Girl's high school")

Girl's highschool

Print ('He said, "he won the game"')

He said, "he won the game"

single quote

variables :-

→ To store data

→ It is not necessary to declare variables in python. we can use them directly.

(=)

variable = value

Ex:- x = 45

name = "Abhi"

f = 4.35489

→ In python variables are treated as objects.

↳ instance of a class.

type()

→ By using type() we can determine the objects of Particular class. [It can show us the datatype of variables].

Ex:- x = 56

type(x)

class 'int'

→ name = 'ricon'

type(name)

class 'string'

class 'str'

→ Pi = 3.145

type(Pi)

class 'float'

→ b = True

type(b)

class 'bool'

```
x = 56
>>> x
56
>>> print(x)
56
>>> print("x value is :", x)
x value is : 56
```

→ closing the shell CTRL+D

Program :-

→ I/p → process → (O/p) Print()

→ used to Represent the comment lines.

```
#program : simple demo program
#input
x=10
y=20
# process
z=x+y
p=x-y
#output
print(z)
print(p)
```

```
#output
print("first no :", x)
print("second no :", y)
Print("Addition of 2 no's is :", z)
Print("Product of 2 no's is :", p)
```

→ Extension .Py

→ RUN the program F5

I/p function:-

input = scanf or cin

x = func( )

↓

It takes value from keyboard.

Syntax:-

x = input ("Prompt")

Eg:-

name = input ("Enter your name")

↓

→ It takes input from keyboard & returns specified variable.

→ It always takes any data in the form of strings.

Ex:-

no = input ("Enter any no")

enter any no 12

type (no)

→ class 'str'

#program : i/p fn demo

#input

name = input("enter your name:")

place = input("enter your city:")

#process

enter your name abhi

enter your city nlr

#output

Print ("your name is", name)

Print("city:", city)

Print ("end")

DIV //
Remain %.
Sauare & Squareroot

14 June

Conversion functions :-

→ Needed to convert one datatype to other.

→ int() eval()
float()
str()

The above following functions does the conversion process.

→ int() → convert str() ⟶ int()
→ float() → convert str() ⟶ float()
→ eval() → It converts and evaluates both int() & float()
→ str() → converts int() → str()

Ex:- x = "120"

→ int(x) + 30
⇒ 150

y = "4.5"
float(y) + 1.2
→ 5.7

· eval("45 + 20")
65
eval("1.2 + 2.3")
3.5

Conversion of str()

```
a=12
print("value of a :"+str(a))
```

value of a : 12

```
#Addition prog
#input
x = input("Enter any no")
y = input("Enter second no")
#process
z = int(x) + int(y)    # str() -> int()
#output
print("x value :", x)
print("y value :", y)
print("Sum of x & y is :", z)
print("End of the program")
```

we can convert while entering the data itself using conv. func

```
x = int(input("Enter first no"))
y = int(input("Enter second no"))
```

or float or eval

Using constants :-

Rules :-

→ constants should be in CAPLOCKS [UPPERCASE]

→ They should be at the top or beginning of the program

find out area of circle & circumference

```
#constant
PI_VALUE
PI_VAL = 3.145
r = eval (input ("Enter radius of circle"))
#process
area = PI_VAL * r * r
cir = 2 x PI_VAL x r
#output
print ("radius :", r)
print ("area:", area)
print ("circumference : ", cir)
print ("End of program")
```

To print sum of n Natural no :

```
n = eval (input ("Enter the value of n : "))
sum = n(n+1)/2
print ("sum of n natural no's = ", sum)
```

To findout area & perimeter of rectangle :

```
l = eval (input ("Enter the val of l "))
b = eval (input ("Enter the val of b"))
Area = l x b
Perimeter = 2(l+b)
print ("Area of rect : ", area)
print ("circumferenc of rect : ", Perimeter)
```

#swap (temp variable)

```
a=eval(input("Enter value of a"))
b=eval(input("Enter value of b"))
c=a
a=b
b=c
print("Before swap print 2 no's")
print("a")
print("b")
print("The vales after swapping of no's")
print("a")
print("b")
```

#swap (without temp)

```
a=a+b
b=a-b
a=a-b
```

These are called Sequential programs.

→ Any sequential Programming follows the following structure.

i.e input → process → output

① Input principleamount, time, rate find SI

$$SI = \frac{P \times T \times R}{100}$$

② Input temp in fahrenheit → celcius

③ Input base, height of Δle, area of Δle

input any 4 digit number and findout sum of 1st and last digit

15th June

we get the last digit by;

let n = 2486

r = n % 10

10) 2486 (248)
2480
(6)

we get the first digit by;

q = n // 1000

1000) 2486 (2)
2000

sum = r + q;

To find out sum of first and last digit

x = eval(input("Enter any 4 digit number"))

process

r = x % 10

q = x // 1000

s = r + q

output

print("sum of first & last digits:", s)

print("End of the program")

input 3 sides of the Δle and findout area of Δle.

To findout the area of Δle Area = $\sqrt{s(s-a)(s-b)(s-c)}$.

x = eval(input("Enter the values of a, b and c"))

process

s = (a+b+c)/2

Area = $(s(s-a)(s-b)(s-c))^{1/2}$

output

print("Area of Δle is", Area)

print("End of the program")

If $ax^2+bx+c=0$ then findout

$$d = b^2 - 4ac$$

$$root1 = \frac{-b \pm \sqrt{d}}{2a}$$

Two points in a plane are specified using co-ordinat (x_1,y_1) (x_2,y_2). WAP to calculate slope of a Line.

$$slope = \frac{y_2-y_1}{x_2-x_1}$$

Distance b/w them

$$\left((x_2-x_1)^2+(y_2-y_1)^2\right)^{**}\left(\frac{1}{2}\right)$$

$ax^2+bx+c=0$

```
n=eval(input("Enter values
  of a,b,c"))
# process
d = b**2-4*a*c
root1 = -b+d**(1/2)
        ------------
            2a
root2 = -b-(d)**1/2
        ------
          2a
#output
print("First real root:",
    root1)
print("second real root:",
    root2)
```

(x_1,y_1) (x_2,y_2)

```
n=eval(input("Enter the values
  of x1,y1,x2,y2"))
#process
slope = (y2-y1)//(x2-x1)
distance = ((x2-x1)**2+(y2-y1)**2)**(1/2)
# Output
print("The slope between them:",
    slope)
print("Distance b/w them:",
    distance)
```

→ Every variable in python is actually an **object**.

→ All variables have a type that can be seen by using the built in **type()** function.

Multi Assignment :-

```
a=b=c=9
```

Here a, b, c get the same value.

→ Several variables are given their values at once

```
x,y,z = 99, 'raj', 4.95
```

Ex: `a,b = eval(input("Enter any 2 nos:"))`

```
Enter any 2 nos: 20,90
a
20
b
90
```

```
>>> x,y,z = 99, 'ram', 7.980
>>> x
>>> 99
>>> y
>>> 'ram'
>>> z
>>> 7.980
```

Complex Numbers :-

→ Python supports complex numbers with the imaginary no with the imaginary letter **'j'**

Ex: `type(2+3j)`

```
class 'complex'
```

→ `y = 4+6j`

```
type(y)
class 'complex'
```

→ we can define complex numbers using complex functions.

```
>>> a: complex(4,6)
  >>> a
  >>> 4+6j
>>> b: complex(8,2)
  >>> b
  >>> 8+2j
```

→ In complex nos we pass the real & imaginary parts.

→ we can add and subtract complex nos in the same way as real nos.

```
>>> a+b
    12+8j
>>> a-b
    -4+4j
```

→ The real and imaginary parts of a complex no can be retrieved real & imag attributes

```
>>> a.real
>>> 4.0
>>> a.imag
>>> 6.0
```

Sequences:-

→ Str
In python we have following sequences with application
① strings , ② lists ③ tubles.

Strings:-

A string is simply a series of characters. It is anything inside quotes.

Ex:- 'Abhi', "2", "5.6"

var = 'RICON'

→ All strings belong to str class. So all stringvariables of object of str class
instances

→ All classes contains attributes and Methods.

Diff blw func & Method:-

→ Method:-
It is function which belongs to class.
→ It can be called along with instance of the class (ie object) using dot (.) operator.

Ex:- s = "ricon technology" (or) "computer".upper()

s.upper()
'RICON TECHNOLOGY'
s1 =
s.lower()
'ricon technology'

st.title() ⟶ first letter will be capital letter
'Ricon technology'

```
s = "silicon technology"
x,y = s.split(" ")
x
'silicon'
y
'technology'
today = "6/15/2018"
month, day, year = today.split("/")
month
'6'
day
'15'
year
'2018'
```

18 June

Format function:-

It substitutes the values

```
x=12
y=20
print("x value:{0}\n y value:{1}".format(x,y))
x value:12
y value:20
```

placeholders in the string

Combining or concatinating strings

Python uses the '+' symbol to combine strings.

Ex: fname : Abhi
Lname : Nandana
name : fname + " " + Lname
print (name)

justifying text with r-just, l-just and centre methods

r-just :-

It is used for right justification

Ex: "ricon".rjust(15)
'ricon'
10 spaces are left

l-just :-

It is used for left justification

Ex: "ricon".ljust(20)
'ricon'

centerjust :-

It is used for center justification

Ex: "ricon".center(10)
' ricon '

With two parameters :-

>>> "ricon".rjust (15,'_')
>>> '----------ricon'
>>> "ricon".ljust(15,'*')
>>> 'ricon**********'
"ricon".centre(20,'=')
>>> '=======ricon========'

Removing white spaces with strip, rstrip and lstrip.

Methods :-

```
s = "cat"
st = "   cat"

s==st
false
s == st.strip()
                 ↑ cuts the spaces
True
```

Sometimes you may want to strip off white spaces characters [space, tab] from the leftside, rightside or both sides of a string. The strip method will return a new string without any whitespace chars at beg or end. The lstrip and rstrip methods will remove whitespace chars from left and right side respectively.

```
s==st.lstrip()
true
```

Replace (Method) :-

```
s2 = "uncle chips"
```

Syntax: s2.replace("~~new~~old name", "~~old~~new name")

```
s2.replace("un", "my")
'mycle chips'
```

count

```
word = "madam"
word.count('a')      Counts the letters
```

join:

```
st : "uncle chips"    word : "madam"
st.join(word)
'muncle chipsauncle chipsduncle chipsauncle chipsn'
word.join(["1", "2", "3"])        # list
'1madam2madam3madam'
```

→ you can give only 1 argument inside so we've used list

#Input any string & find out no:of words

```
#program:to find out no:of words
st : input("Enter any string")
words = st.split(" ")
l: len(words)
print("no:of words:", l)
print("End of the program")
```

Execution

```
>>> cat computer Nellore
>>> 'no:of words': 3
```

Lists :-

→ It is a collection of items in a particular order. You can make a list that includes the letters of the alphabets, digits from (0-9) or the names of all the people.

→ In python [ ] indicates a list and individual elements in list are seperated by commas (,)

→ To access an element in a list, write name

write name

of list followed by index of item enclosed in [ ].

Ex: lst = [90, 80, 70, 60, 45, 23]

fruits = ["apple", "mango", "grapes", "orange"]

mix = [111, "raj", "nir", True, 23.90]

print ("lst[0]") - ~~apple~~ 90

print (lst[-1]) - 23

print (fruits[-2]) - grapes

print (mix[3]) - nir

For accessing
0, 1, 2, 3 → for the first
-1, -2, -3 → from the last

Ex: st = "grapes"

st[0] → 'g'

st[-1] → 's'

st → 'grapes'

st[:] → 'grapes'

st[::-1] → 'separg'

↓

For reversing a string

June 19

lst = [ ]

type(lst)

class 'list'

→ we use append() to add items.

```
lst.append(78)
lst.append(86)
lst.append(120)
lst
[78,86,120]
```

↳ It takes only one value at a time.

→ list is a dynamic datatype.

→ >>> lst[1] = 99 → we can modify the list using index numbers.

```
>>> lst
[78,99,120]
```

→ Append() adds the element only at end position.

Inserting elements into a list :-

You can add new element at any position in your list by using the insert() → method. We do this by specifying index of element and, giving value of that number.

```
lst.insert(0,12)
lst
[12,78,99,120]
lst.insert(3,15)
[12,78,99,15,120]
```

Removing elements from a list :-

By using "del" statement to remove an element. If you know the position of item you wanted to remove you can use "del" statement;

```
del lst[2]
[12,78,15,120]
```

Removing an item using 'pop' method:-

→ Pop() → By default removes last element and gives the value to another variable.

If we give pop(indexno) → it pop's accordingly.

```
x=lst.pop()
pop
x
120
lst
[12,78,15]
```

→ By using pop to remove particular element in any position

```
y=lst.pop(1)
y
78
lst
[12,15]
```

Removing an item by value :-

```
pet=["dog","cat","hen","cat","dove","cat"]
pet.remove("cat")
pet
```

→ It removes one item at one time.

```
['dog','hen','cat','dove','cat']
```

→ when you dont know the position of the value and you wanted to remove from list. If you only know the value of item and you wanted to remove, we use remove().

Membership operator:

For datatypes that are sequences or collections. like strings, lists, tupels. we can test memberships using 'in' operator & non memberships using 'not in' operator.

[Searching]

Ex:- "cat" in pet
True

"tiger" not in pet
True

sort():-

Arranges the list items in Alphabetical order.

```
pet.sort()
pet
['cat', 'cat', 'dog', 'dove', 'hen']
```

Reverse():-

To reverse the original order of a list, you can use reverse(). It doesn't sort backward alphabetically. It simply reverses the order of the list.

```
pet.reverse()
pet
['hen', 'dove', 'dog', 'cat', 'cat']
```

Descending order of a list first sort it then reverse.

length len function:

```
len(pet)
5.
```

Note:

function can be called directly.
Method can be called only with an object.

20/6/18

Copying a list:

To copy list you can make a slice that includes the entire original list by omitting the first index & second index.

→ This tells python to make a slice that starts at the first item and ends with last item producing the copy of entire list.

```
lst=[10,20,30,40,50]
s=lst[:]
s
[10,20,30,40,50]
t=lst[1:3]
t
[20,30]
```

Tupel :- [static]

A tupel looks just like a list except you use paranthesis ( ) instead of [ ] :-

→ An immutable list is called tupel.

→ Once you define a tupel, you can access individual elements by using each item's index just as you would do for a list.

```
tu = (1,5,6)
print (tu[0])
  1
print(tu[1])
  5
print (tu[-1])
  6
```

Dictionary :-

Each item in a dictionary has a key and a corresponding value like a list.

→ A dictionary is a collection of many values.

→ Indices of a dictionary are called keys.

→ A key with its associated value is called a key value.

→ In code, a dictionary is typed with { }.

```
fav_language = {'ram': 'c', 'abhi' : 'c++', 'moni' : 'Java'}
fav_language['abhi']
'c++'
```

To del a value in dictionary:-

To del a value in a dictionary use key.

```
del fav_language['abhi']
fav_language
{'ram': 'c', 'moni': 'Java'}
```

→ To replace a value in a dictionary we also use key.

```
fav_language['abhi'] = 'python'
fav_language
{'ram': 'c', 'abhi': 'python', 'moni': 'Java'}
```

Dictionary methods :-

There are 3 dictionary methods that 'll return list like values.

① keys()
② values()
③ items()

```
fav_language.keys()
dict_keys(['ram', 'moni'])
fav_language.values()
dict_values(['c', 'Java'])
fav_language.items()
dict_items([('ram', 'c'), ('moni', 'Java')])
```

checking whether a key or value exist in a dictionary:

The in and not in operators can check whether a value exist in a list.

```
'oracle' in fav_language.values()
False
'Java' in fav_language.values()
True
'sam' in fav_language.keys()
True
```

A list of dictionaries:

```
d1 = {'color': 'green', 'points': '4'}
d2 = {'color': 'red', 'points': '8'}
d3 = {'color': 'blue', 'points': '12'}
mylist = [d1, d2, d3]
```

A list in a dictionary:

It is sometimes useful to put a list inside a dictionary

```
fav_lang = {'Raj': ['python', 'oracle'],
            'lisa': ['cpp', 'c'],
            'Abhi': ['Java', 'dotnet']}
```

A dictionary in a dictionary:-

You can nest a dictionary inside another dictionary

```
users = {'aeinstein': {'first': 'albert', 'lname': 'einstein',
                       'location': 'UK'}
         'mcurie': {'first': 'Madam', 'lname': 'curie',
                    'location': 'Paris'}
```

Range function :-

To make a list of numbers.

Syntax range (starting no, ending no, step value)
(1) (2) (3) (n-1)

1) range(1) — range(10) → 0-9

range(1,2) - range(1,20) → 1-19

range(1,2,3) - range(1,20,2) - 1 3 5 7 ... 19

list function :-

If you want to make a list of numbers, you can convert the results of a range directly into a list using list().

```
nos = list(range(5))
  nos
[0,1,2,3,4]
nos = list(range(10,20))
nos
[10,11,12,13 . . . ,19]
x = list(range(2,20,2))
x
[2,4,6,8 . . . 18]
```

11/6/18

Conditions

Relational Operators:

We have to specify any condition in the program we use a relational operator.

The relational operators are <, >, >=, <=, !=

The relational operators returns boolean expression i.e either true or false.

Ex: 7 8 7 0
True.

Reserved words or keywords:

These are used only in some special conditions. Here are a list of python keywords.

False	def	global	or
None	del	if	pass
True	elif	import	raise
and	else	in	return
as	except	is	try
assert	finally	lambda	while
break	for	nonlocal	with
class	from	not	yield
continue			

* Simple if statement:

Syntax: if condition:

The condition is a boolean expression that determines whether the body will be executed or not.

A colon (:) must follow the condition.

The block is one or more statements to be executed if the condition is true.

The statements within the block must follow the same no: of spaces from left.

Ex:-

```
#program: dimple if demo.
x=int(input("Enter any no:"))
if x<0:
    print("You entered negative no:", x)
    print("Negative values are not valid")
    print("You try again")
print("EOP")
```

o/p:-

```
Enter any no: -6
You entered negative no
Negative values are not valid
You try again
EOP
```

* if-else statement:

Syntax:

```
if condition:
    block
else:
    block
next statements
```

A colon(:) must follow the else. The else block is executed when if condition is false.

Ex: I/P any two no's. Find out which is biggest number

```
# program: to find biggest no.
x = int(input("enter any no:"))
y = int(input("enter another no:"))
if x>y:
    print("big is :",x)
else:
    print("big is :",y)
    print("EOP")
```

O/P:

```
Enter any no: 9
Enter another no: 10
big is : 10
EOP.
```

Ex:

```
n=input("Enter any
        character")
if n>=chr(65) and n<=chr(
                    chr(9
    print("capital")
else
    print("small")
```

Ex: 2 Input any no & find out whether if it is even no or odd

```
# Program to find evenno or odd no
x=int(input("Enter any no:"))
if x%2==0
    Print("Even no")
else:
    Print("odd no")
Print("EOP")
```

O/P

```
Enter any no: 5
odd no
EOP.
```

Ex:3

I/p any no & findout whether it is +ve or -ve no

Program: to findout positive or negative no

```
x=int(input("Enter any no:"))
if x>0:
    print("Positive no")
else:
    print("Negative no")
print("EOP")
```

O/P

Enter any no: 5
Positive no
EOP

Enter any no: -3
Negative no
EOP

Ex:4

I/p person name & age & findout he is eligible to vote or not.

```
#To findout person eligible to vote or not.
name = input("Enter person name:")
age = int(input("Enter person age:"))
if age>18:
    Print("[0], you are eligible to vote".format(name))
else:
    Print("[0], you are not eligible to vote".format(name))
Print("EOP")
```

o/p:-

Enter person name: Raj

Enter person age : 18

Raj, you are eligible to vote

EOP

*Pass statement:-

Python has a special statements. Passing the statements means doing nothing.

we use this when we wish the program to take no action.

Ex:-

#program: to find & execute how pass works

```
a=int(input("enter person age:"))
if a<0:
    pass
else
    print(a+10)
print("EOP")
```

o/p:

Enter person age :-19

End of pro.

EOP

Enter person age : 18

28

EOP

Compound i/p :-

st1 and st2

e)) compound statements means and/or/not

logical operators

one or more conditions combine within logical operators.

Syntax:

```
if cond1 logicalop cond2 logicalop cond3 :
else    | st1
        | st2
```

AND operator

cond1	cond2	Result
T	T	T
T	F	F
F	T	F
F	F	F

OR operator

cond1	cond2	Result
T	T	T
T	F	T
F	T	T
F	F	F

NOT operator

cond1	Result
T	F
F	T

Eg:

I/p student no & student name and then 3 subj marks Find out total avg & result with following formal O/p (format)

```
enter student no : 111
enter student name : raj
enter sub1 marks : 45
enter sub2 marks : 67
enter sub3 marks : 87
-Ricon Technologies
Nellore
```

student no : 111 studentname : raj

sub1 marks : 45
sub2 marks : 67
sub3 marks : 87

total marks : 199
avg marks : 66.3333

Result : pass

#program: to findout total, avg and result of the student

```
#input
st no : eval(input("enter student no :"))
st name : eval input("enter student name :")
sub1 = eval (input("enter sub1 marks :"))
sub2 = eval (input(" " sub2 " :"))
sub3 = eval (input(" " sub3 " :"))
#processing
  tot = sub1 + sub2 + sub3
  avg = tot/3
if sub1 >= 35 and sub2 >= 35 and sub3 >= 35:
    result = "pass"
else:
  result = "fail"
#output
print ("-" * 75)
print ("\t\t Ricon Technologies")
print ("\t Nellore")
```

```
print ('_' * 75)
print ("student no: ", st no, end = ' '))
print ("\t student name: ", st name)
print ("_" * 75)
print ("\t\t sub1 marks : ", sub1)
print ("\t\t sub2 marks : ", sub2)
print ("\t\t sub3 marks : ", sub3)
print ("_" * 75)
print ("\t\t total marks : ", tot)
print ("\t\t avg marks : ", avg)
print ("_" * 75)
print ("\t\t Result:", result)
print ("_" * 75)
```

2 sline

To findout vowel

```
ch = input ("enter any character")
if ch == 'a' or ch == 'e' or ch == 'i' or ch == 'o'
    print ("vowel")
else:
    print ("consonant")
```

Nested-if:

when one if condition contains another if condi

if condition1 :-

```
if condition2:
    st
else:
    st
```

True block

```
else :
    if condition 3 :
     st
    else :
       st
```

false block

#i/p any 3 no's and findout which is the biggest no

#program : To findout greatest of 3 no's

#input

```
a=eval(input("enter first no :"))
b=eval(input(" enter second no:"))
c=eval(input(" enter third no :"))
```

#process & o/p

```
if a>b:
    if a>c:
        print(" bigno is :", a)
    else:
        print("bigno is :", c)
else:
    if b>c:
        print("bigno is :", b)
    else:
        print("bigno is :", c)
```

ch=='u';

Multiple conditions:

```
if cond1:
    st
elif cond2:
    st
elif cond3:
    st
    :
    :
else:
    last statement
```

To findout whether +ve no or -ve no or zero

```
#input
a=eval(input("Enter any no : "))
#process & o/p
if a==0:
  print("It is equal to zero")
elif a>0:
   print("It is +ve no")
else:
    print("It is -ve no")
```

input any no & print the day name of the week

```
#input
a=eval(input("enter any no :"))
if a==1:
   print("sunday")
elif a==2:
    print("monday")
elif a==3:
     print("tuesday")
elif a==4:
      print("wednesday")
elif a==5:
      print("thursday")
elif a==6:
      print("friday")
elif a==7:
      print("saturday")
else:
     print("invalid")
```

lab

#1 :- i/p any no & print the equalent month name

#2 :- i/p any no & print the equalent character form upto single digit

0 - zero

6 - six

9 - nine

if $ax^2+bx+c=0$, find out the roots

i/p - a, b, c

d

$d>0$, $d=0$, $d<0$ — 3 condition elif

WAP to

find out electricity bill

i/p previous meter reading & current meter reading & no units & charges.

0-50	-1.50
51-100	-2.50
101-200	-3.50
201-300	-4.00
7300	-5.00

Ex: 121

$50 \times 1.50 + 50 \times 2.50 + 25 \times 3.50$

i/p 3 sub marks find out total avg & grades

26 JUNE

Repetative structures :-

LOOPS :-

One or more statements can be executed specified no: of times or) until they satisfy the condition.

→ There are 2 types of Loops in python

① for [specified no: of times]

② while [until they satisfy the condition]

Syntax:

for variable in sequence :

    st 1
    st 2
    Block

→ string [seqence on chaus]
→ list [list of values]

last st

String:

s = "computes" (or) 'Abhi'

List:

lst = [10, 20, 40, 50] (or) using range().

#program for demo

st = "cat computer point"

for i in st:

    print(i)

print()

print("end of the program")

#horizontal printing

st = "cat computer point"

for i in st:

    print(i, end=' ')

print()

print("end of the program")

o/p

c
a
t
c
o
m
p
u
t
e
r
p
o
i
n
t

o/p

cat computer point.

* lst = [10, 20, 30, 40, 50, 60]

```
for i in lst:
    print(i, end = ' ')
print()
print("end of the program")
```

Syntax:

range(arg1) → displays from 0 to (n-1)
10 — 0-9

range(arg1, arg2) → 10, 11, 12 - - - 19.
10, 20

range(arg1, arg2, arg3) → 10, 12, 14, 16, 18
10, 20, 2

Ex:- for i in range(10)

!) program for range

```
for i in range(10):
    print(i)
print()
print("end of the program")
```

```
for i in range(10,20):
    print(i)
print()
print("EOP")
```

```
for i in range(10,20):
    print(i)
print()
print("end of the program")
```

#To print 'n' even no's & their sum

```
n=int(input("Enter any no"))
s=0
for i in range(2,n+1,2)
    s=s+i
    print(i)
print()
print("sum of even no's:",s)
print("EOP")
```

#To print 'n' odd no's & their sum

```
n=int(input("Enter any no"))
s=0
for i in range(1,n+1,2):
    s=s+i
    print(i)
print()
print("sum of odd no's:",s)
print("EOP")
```

#i/p. any no & print factorial value

```
n=int(input("Enter any no"))
f=1
for i in range(1,n+1):
    f=f*i
print()
print("factorial of n is:",f)
print("end of program")
```

$5! = 5\times4\times3\times2\times1$

$= 1\times(1\times2\times3\times4\times5)$

$f \quad i$

Evaluate the ncr

$$ncr = \frac{n!}{(n-r)!\,r!}$$

3 times factorial & mul (n-r)! r! & then divide n! by

27 June

Yp any no & print the equalent multiplication table. [using format()]

Note:
In python all variables are objects.

Note:
In python strings are represented with ' ', " ", ''' ''', """ """.

```
n=int(input("Enter any no"))
for in range(1,11):
    print('{0} x {1} = {2}'.format(n,i,n*i))
print('End of the program')
```

Hilp any no & findout whether it is prime no. or not

```
n=int(input("Enter any no")
count=0
for in range(2,n):
    if(n%.i==0):
        count=count+1.
if(count==0):
    print("prime no")
else
    print("Not a prime no = composite no")
```

continue and break statements :

In loops we use these statements.

break

```
for i in range(1,100):
    if (i==20):
        break
    print(i)
print()
print("End of the program")
```

op
1
2
3
:
19.

continue

```
for i in range(1,100):
    if (i==20):
        continue
    print(i, end=" ")
print()
print("end of the program")
```

o/p
1 2 3 4 ... 19 21 22 ... 99.

To print 100 no's except 9 multiples

```
for i in range(1,100):
    if (i%9==0):
        continue
    print(i)
print()
print("EOP")
```

Loop in reverse order:

```
for i in range(10,0,-1):
    print(i)
```

i/p any string & findout no:of vowels

```
st=input("Enter any string")
count=0
for i in st:
    if i=='a' or i=='e' or i=='i' or i=='o' or i=='u':
        count=count+1
print("No:of vowels :", count)
print("EOP")
```

WAP to find the given no is perfect or not

```
n=int(input("Enter any no:"))
s=0
for i in range(1,n):
    if(n%i==0):
        s=s+i
if(n==s):
    print("It is perfect no")
else:
    print("It is not a perfect no")
print()
print("EOP")
```

Perfect no:
Sum of its factors = same no
Ex:- 6 - (1,2,3)
= 1+2+3

28 June

while loop: [until satisfies the condition]

Syntax:

exp1 ⟶ Initialize the condition

while(exp 2):

```
i=1
while i<=10:
  print(i)
  i=i+1
print()
print("EOP")
```

Print n even no's & their sum

```
n=int(input("Enter any no":))
i=2
s=0
while (i<=n):
    s=s+i
    print(i)
    i=i+2
print()
print("sum is:",s)
print("EOP")
```

Input n' odd no's & print their sum.

```
n=int(input("Enter any no:"))
i=1
s=0
while(i<=n):
    s=s+i
    print(i)
    i=i+2
```

I/p any no & findout their sum of digits

```
r=n%10
s=s+r
n=n//10
```

```
n=int(input("Enter any no:"))
s=0
while n!=0:
    r=n%10
    s=s+r
    n=n//10
print()
print("Sum of digits:", s)
print("EOP")
```

#Reverse of a number

```
r=n%10
s=(s*10)+r
n=n//10

n=int(input("Enter any no:"))
s=0
while n!=0:
    r=n%10
    s=(s*10)+r
    n=n//10
print()
print("Reverse of a no:", s)
print("EOP")
```

```
r=n%10
s=s+1
n=n//10
print()
print("No:of dits:", s)
print("EOP")
```

#To find out no:of digits

```
r=n%.10
s=s+1
n=n//10
```

[equals:]

sum of cubes ~~of~~ given numbers. [Armstrong]

Ex:- $153 = 1^3+5^3+3^3$

$1+125+27$

$=153$.

```
n=int(input("Enter any no"))
  s=0
  t=n
while t!=0:
      r=t%10
      s=s+(r**3)
      t=t//10
if n==s:
   print("given no is Armstrong")
else:
   print("given no is not an Armstrong")
```

Infinity loop:-

Syntax: `while (True):` → keyword [Boolean datatype]
T → should always be CAPITAL

```
st
st
st
if cond:
   break
```

[we use this for many given programs]

29June

Infinity loop:

```
#menu program

while True :
    print ("1.addition")
    print ("2.sub")
    print ("3.multiplication")
    print ("4.Divisron")
    print ("5.exit")
print()
print("-----------------------")
x= eval(input(" Entes uy choice:"))
if x==5:
    break
if x<5:
    a = eval (input( "enter first choice :" ))
    b = eval (input(" enter second choice :"))
if x==1:
    c=a+b
elif x==2:
    c=a-b
elif x==3:
    c= a*b
```

```
elif x==4:
    c=a||b
print("result:",c)
print("-----------------------")
print("EOP")
```

Nested loops:-

```
for i in range(1,5):
    for j in range(1,5):
```

→ Inner loop gets executed fully [throughout it's range] for 1 outer loop

Ex:-

outer	inner
1	4
2	4
3	4
4	4

In Total this loop repeats 4×4 = 16 times.

Usage:-

1) To make patterns
2) To specify range of program - [Ex: 1-100 no's]

Program to print no's in following pattern

(1)

	c1	c2	c3	c4	c5	
r1	1	2	3	4	5	(5×5)
r2	1	2	3	4	5	
r3	1	2	3	4	5	
r4	1	2	3	4	5	
r5	1	2	3	4	5	

program : To print pattern of no's.

```
for i in range(1,6):
    for j in range(1,6):
        print(j, end=" ")
    print()
print("EOP")
```

To print following pattern of stars

(2)

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

program : To print pattern of stars.

```
for i in range(1,6):
    for j in range(1,i+1):
        print("*", end=" ")
    print()
print("EOP")
```

(3)

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

program : To print pattern of no's.

```
for i in range(1,6):
    for j in range(1,i+1):
        print("j", end=" ")
    print()
```

④
```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

#program: To print given pattern of no's

```
for i in range(1,6):
    for j in range(1,i+1):
        print(i, end=" ")
    print()
```

⑤
```
# # # # #
$ $ $ $ $
! ! ! ! !
^ ^ ^ ^ ^
~ ~ ~ ~ ~
```

#program: To print in this particular patter

```
lst=['#','$','!','^','~']
for i in range(1,6):
    for j in range(.
```

```
lst=['#','$','!','^','~']
for i in lst:
    for j in range(1,6):
        print(i, end=" ")
    print()
```

Range Specification programs :-

To print prime no's from 1 to 100

```
for i in range(1,100):
    count=0
    for j in range(2,i):
        if i%j==0
            count=count+1
    if count==0:
        print(i)end=" ")
print()
prin("EOP")
```

To print N Multiplication table

```
n=eval(iput("Enter any no:"))
for i in range(1,n+1):
    for j in range(1,11):
        print("{0}x{1}={2}".format(i,j,i*j))
    print()
print("EOP")
```

Sum of digits upto single digit

```
x=eval(input("Enter any no:"))
while x>10:
    s=0
    while x!=0:
        r=x%.10
        s=s+r
        x=x//10
    x=s
print("sum of digits:",x)
```