(i) Default constructor.
(ii) Parameterize constructor.
(iii) Non - Parameterize Constructor.

(i) Default Constructor :-

A default Constructor is a constructor that takes no argument. It is used to creat an object with default values for its attribute.

Example :-

```
Class Person:
    def __init__(self):
        Self.name = "John"
        Self.age = 30
Person = Person()
Print (Person.name)
Print (Person.age)
```

Output :-

John.
30.

(ii) Paramesterize constructor :-> A Parameterize constructor is a constructor that takes one or more arguments. It is used to creat an object with custom values for its attribute.

Example :->

```
Class Person:
    def __init__(self, name, age)
        Self. name = name
        Self. age = age
Person = Person ("Alice", 30)
Print (Person . name)
Print (Person . age)
```

Output :->

Alice
30.

(iii) Non- Parameterize constructor :-> A Non-Parameterize constructor is a constructor that does not take any arguments. It is called when we create an instance of a class.

Ex :->

```
Class myclass:
    def __init__(self):
        Self. arg1 = 10
        Self. arg2 = 20
Obj = myclass()
Print(obj.arg1)
Print(obj.arg2)
```

Output :-

10
20

Pygal

* **Pygal :→** Pygal is a python API that enable us to build SVG (scalar vector graphic) graphs and charts in a variety of stiles.

Installation of pygal :⇒

It is available for Python 2.7+, assuming that you have python and pip installed in your system.

- **Pip install :-**

To install using pip, open the terminal and run the following command:

(Pip install pygal)

⇒ **Charts in pygal :→**

There are total 5 types of charts in pygal:-

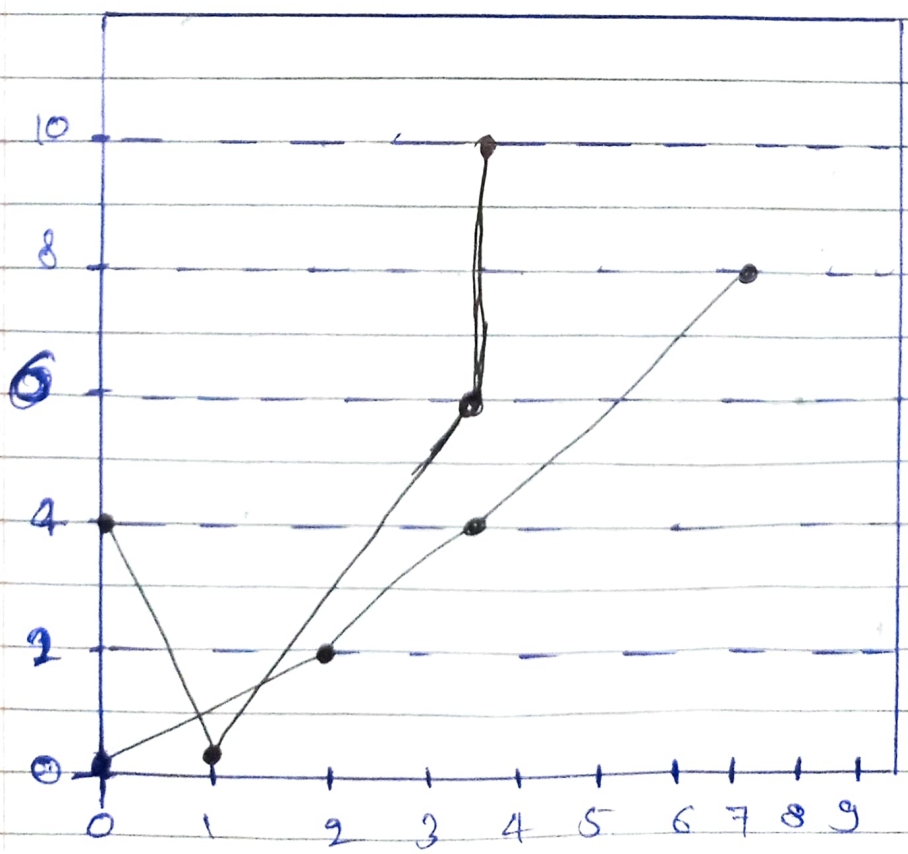
(i) Line graphs.
(ii) Bar chart.
(iii) Pie chart.
(iv) maps.
(v) Customized chart.

(ii) Line graph :-> Line chart is used to represent the data in line formed.

Example :->

```
import pygal
line_chart = pygal.line()
line_chart.title = 'A vs B'.
line_chart.x_labels = map (str, range
                    (0, 10))
line_chart.add ('A', [0, 2, 4, 8])
line_chart.add('B', [4, 1, 6, 10])
line_chart.render_to_file ('basic_
                    line_chart.svg')
```

Output :->

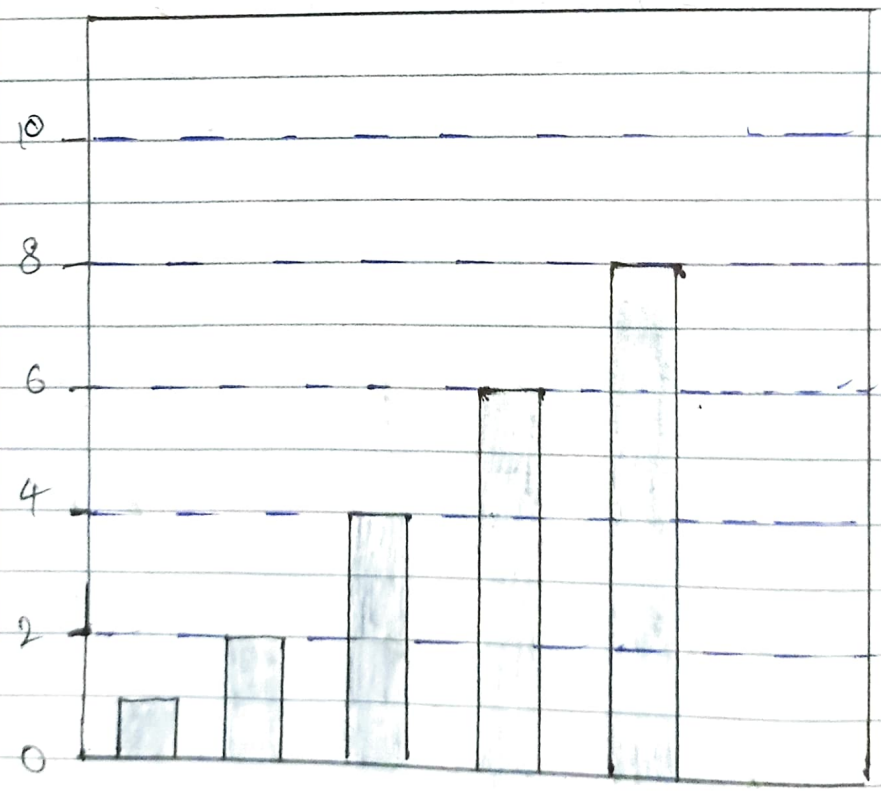


(ii) Bar chart :- It is used to represent the data in bar formed :-

Example :-

```
import pygal
bar_chart = pygal.Bar()
bar_chart.add ('Sequence series',
        [1, 2, 4, 6 8, 10])
bar_chart.render_in_browser()
```

Output :-



Sequence series.

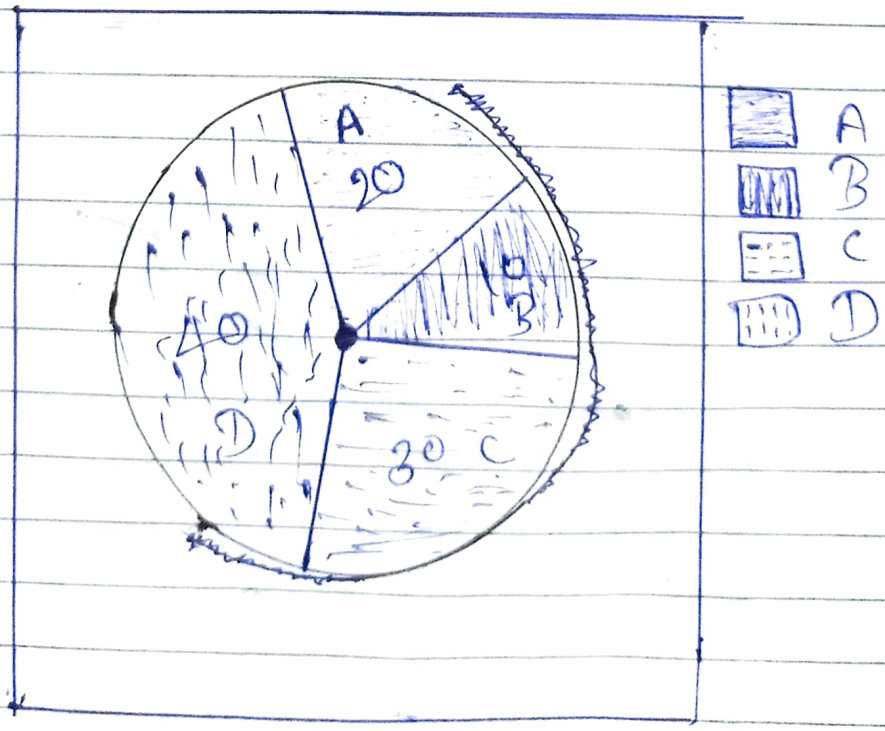
(iii) Pie chart :->

we will now learn how to plot basic pie charts.

Ex :->

```
import pygal.
Pie_chart = pygal.pie()
Pie_chart.title = 'Random data'
Pie_chart.add('A', 20)
Pie_chart.add('B', 10)
Pie_chart.add('C', 30)
Pie_chart.add('D', 40)
Pie_chart.render_to_file('Pie_
        chart.svg')
```

Output :->



(iv) maps :- maps in pygal is used to creat / Plot a map.

installation :-

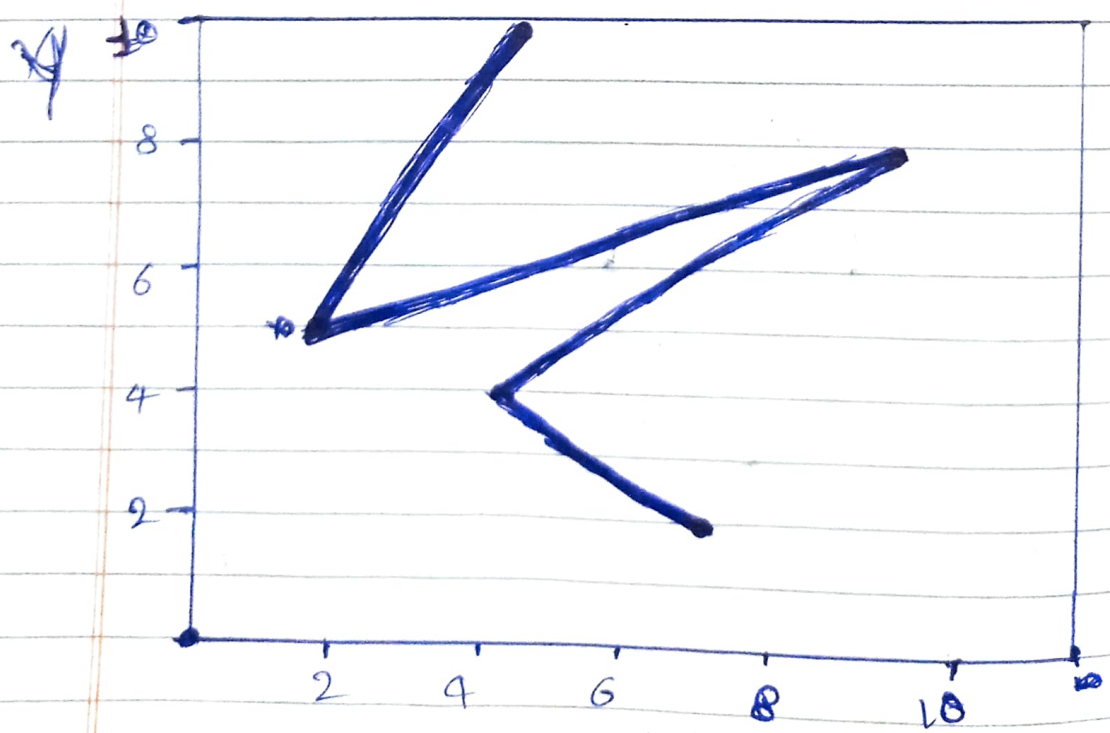
pip install pygal_maps_world.

(v) Customized chart :->

Customized chart in pygal is used to customize the chart.

Properties :->

(i) Plot_background :- background of chart
(ii) background :- background of image.
(iii) font_family :- The main font family



matplotlib

⇒ Matplotlib :->

matplotlib is a multi-platform data visualization library build on Numpy arrays and designed to work with the ~~bend~~ broader scipy stack.

Plots in matplotlib :->

There are 4 types of plots in matplotlib :->

(i) Line plot.
(ii) Bar plot.
(iii) Scatter plot.
(iv) Histogram plot.

(i) Line plot :->

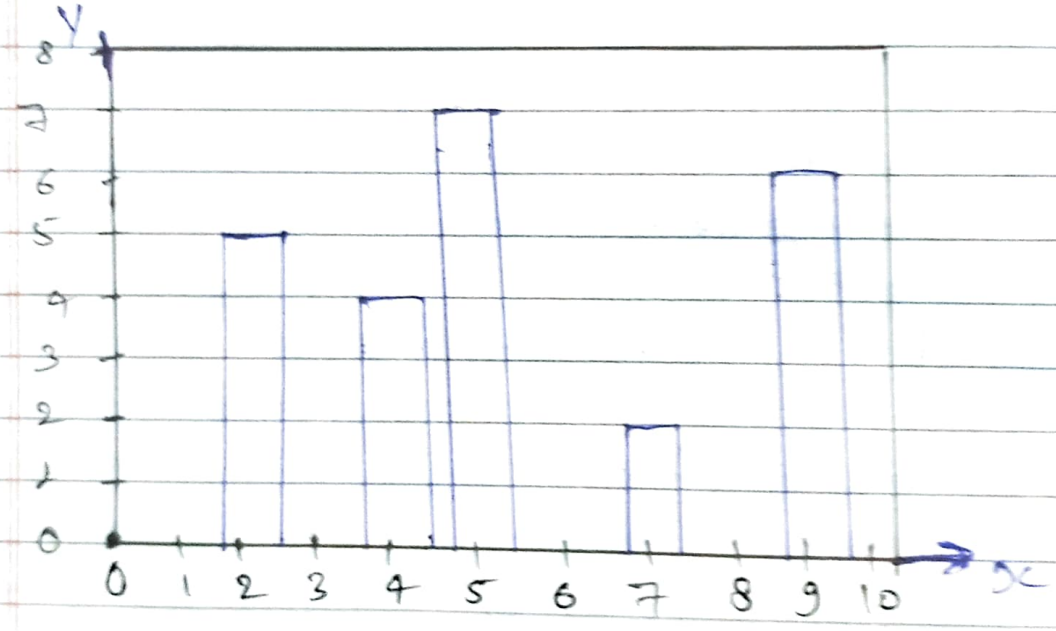
```
from matplotlib import pyplot as plt.
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.plot(x,y)
plt.show()
```

Output

(ii) Bar plot :-

Example :-

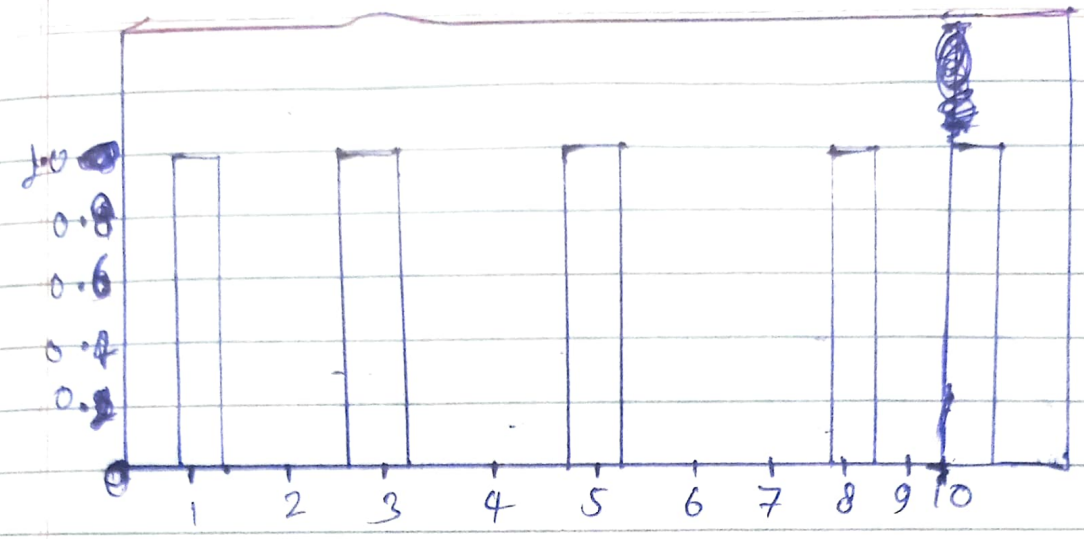
```
from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [7, 5, 6, 4, 2]
plt.bar(x, y)
plt.show()
```



(iii) Histogram :-

Example :->

```
from matplotlib import pyplot as plt
y = [10, 5, 8, 3, 1]
plt.hist(y)
plt.show()
```

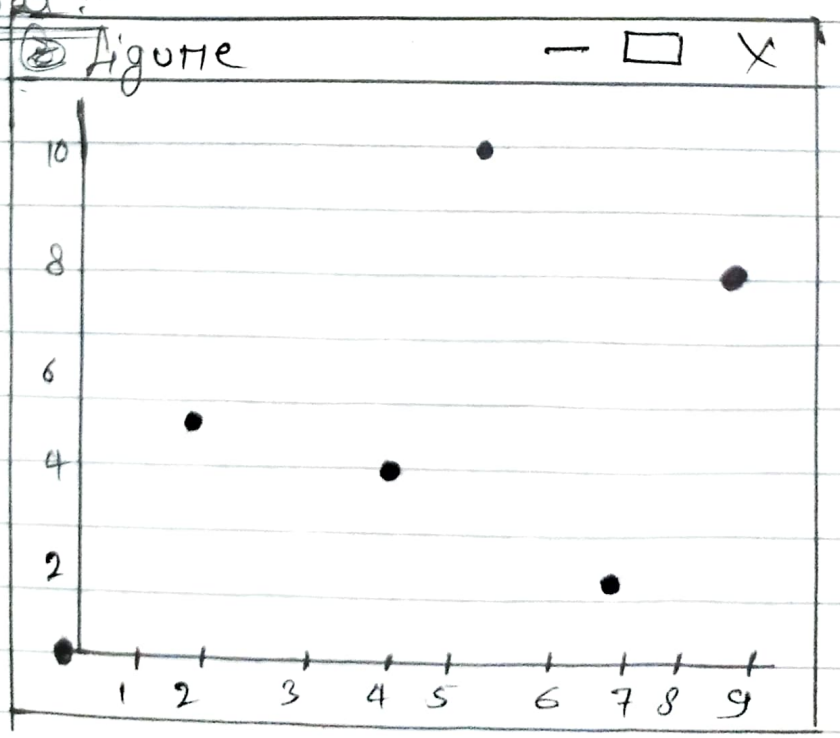


(iv) Scatter Plot :=>

Example :=>

```
from matplotlib import pyplot as plt
x = [5, 2, 9, 4, 7]
y = [10, 5, 8, 4, 2]
plt.scatter(x, y)
plt.show().
```

Output :-

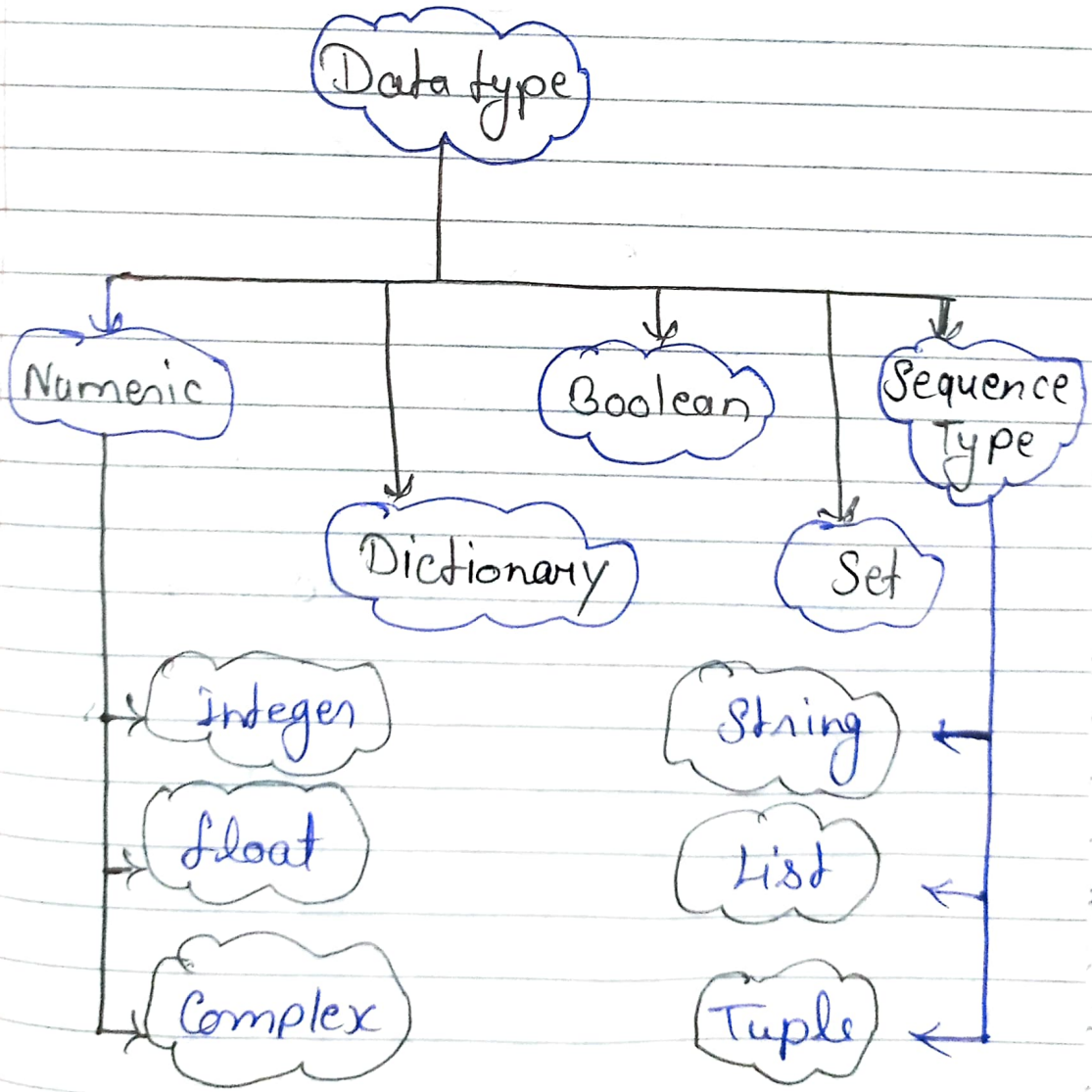


Python :-

Ques ⇒ (1) Write Data Type :-

Ans ⇒ Data type :-

Data type are the classification of data items. It represents kind of value that tells what operation can be performed on particular data.



(fig :- Data type on Python)

⇒ (i) Numeric Data type :-
Numeric Data type represent the data which has numeric value.
Type of Numeric Data type :-
(i) Integer.
(ii) float.
(iii) Complex.

⇒ (ii) Sequance Type :-
Sequance is a ordered collection of similar or different data type.
Type of sequance Data type :-
(i) String :-
A string is a collection of one or more character put in a single qoute, double qoute or triple qoute.

(ii) List :-
List is a collection of data enclosed in ([ ]) square brackets and separated by commas.

(iii) Tuple :- Tuple is an ordered collection of python object. The only defference between tuple and list is, tuple is

immutable, and list is mutable.

(iii) **Boolean** :-

Boolean is one of the built-in data type provided by python, which represents one of the two values i.e. 'True' or 'false'.

(iv) **Set** :- Set is an unordered collection of data, that is mutable and has no duplicate element.

(v) **Dictionary** :- Dictionary in python is an unordered collection of data /value, that used to store data value like :- map.

Ques ② Operators :-

Answer ② Operators :-

Operators are special symbol that Perform operations on variable and values.

Types of operators in python :-

There are 6 type of operators in python :-

(i) Arithmetic Operators.
(ii) Assignment operators.
(iii) Comparison operators.
(iv) Logical Operators.
(v) Bitwise Operators.
(vi) Special Operators.

(i) Arithmetic Operators :-

Arithmetic Operators are used to perform mathematical operators like Addition, Subtraction, multiplication, and Division, etc.

Example to perform Arithmetic operation

```
a = 7
b = 2
Print ('Sum:', a+b)
Print ('Sub:', a-b)
Print ('mul:', a*b)
Print ('div:', a/b)
Print ('mod:', a%b)
Print ('Pow:', a**b)
```

Output

```
Sum: 9.
Sub: 5
mul: 14.
div: 3.5.
mod: 1.
Pow: 49.
```

ii) Assignment Operators :-

Assignment operators are used to assign values to the variables.

Example:-

(=, +=, -=, x=, /=, %=, **=)

(i) Assignment operator (=).
(ii) Addition Assignment (+=).
(iii) Subtraction " (-=).
(iv) multiplication " (*=).
(v) Division " (/=).
(vi) Reminder " (%=).
(vii) Exponent " (**=).

Program:-

```
a = 10
b = 5
a += b
Print(a).
```

Output

=> 15

(iii) Comparison Operators :-

Comparison Operator compare two values / variables and return a boolean Result: 'True / false'.

Ex :->

(i) is equal to (==).
(ii) Not equal to (!=).
(iii) Greater than (>).
(iv) Less than (<)
(v) Grater than or equal to (>=).
(vi) Lessthan or equal to (<=).

Program :->

```
a = 5
b = 2
Print('a==b', a==b)
Print('a!=b', a!=b)
Print('a>b', a>b)
Print('a<b', a<b)
Print('a>=b', a>=b)
Print('a<=b', a<=b)
```

Output :->

```
false
True
True
false
True
false
```

(iv) Logical operators :-

Logical operators used to check whether an expression is 'true' or 'false'.

Ex :-

and, or, Not, et.

Program :-

```
Print (True and True)
Print (True and false)
Print (True or false)
Print (Not true)
```

Output :-

```
True.
false.
True.
false.
```

(v) Bitwise operators :-

Bitwice operater act on operands as if they were string of binary digits.

Ex :-

Bitwise And (&), Bitwise OR (|). Bitwice Not (~), Bitwise XOR (^), Bitwice right shift (>>), Bitwice left shift (<<), etc.

(6) Special operators :→
Python language offer some special types of operators like :- identify operator, membership operator.

Ques→ (3) write type conversion :-

Ans→ Type conversion :—
Type conversion is the conversion of an object from one data type to another data type.

Type of Type conversion :—
(i) Implicit Type conversion.
(ii) Explicity Type conversion.

(i) Implicit Type conversion :—
It is Done automatically by the python interpreter.

(ii) Explicit Type conversion :—
Needs to be done manually by the Programer.

Example of type conversion :-

```
x = 1
y = 2.8
z = 1j
a = float(x)
b = int(y)
c = complex(x).
Print (a)
Print (b)
Print (c)
```

Output :-

```
1.0
2
(1+0j)
```