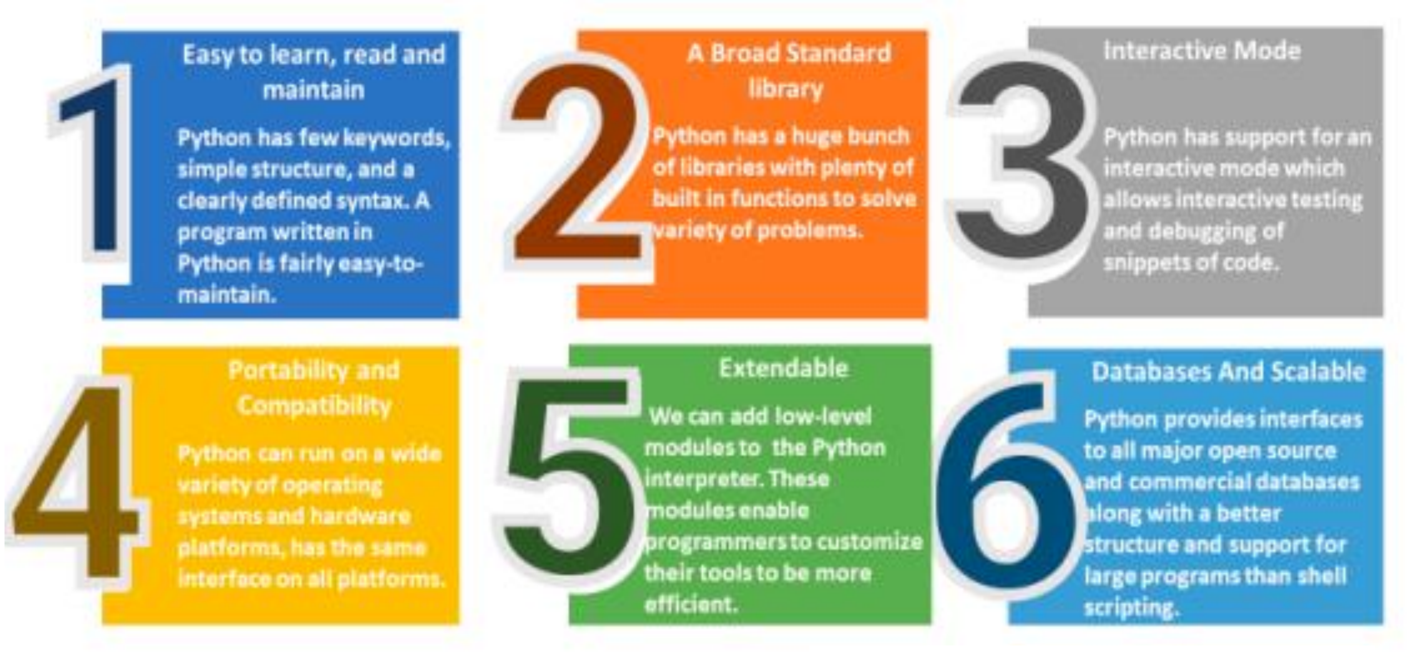
Introduction to Python

A programming language is a formal language that specifies a set of instructions that can be used to produce various kinds of output. **In simple Words, a programming language is a vocabulary and set of grammatical rules for instructing a computer to perform specific tasks**

What is a program? A computer program is a collection of instructions that perform a specific task when executed by a computer. It is usually written by a computer program in a programming language.

Why Python for AI?

Artificial intelligence is the trending technology of the future. You can see so many applications around you. If you as an individual can also develop an AI application, you will require to know a programming language. There are various programming languages like Lisp, Prolog, C++, Java and Python, which can be used for developing applications of AI. Out of these, Python gains a maximum popularity because of the following reasons:



1 Easy to learn, read and maintain

Python has few keywords, simple structure, and a clearly defined syntax. A program written in Python is fairly easy-to-maintain.

2 A Broad Standard library

Python has a huge bunch of libraries with plenty of built in functions to solve variety of problems.

3 Interactive Mode

Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

4 Portability and Compatibility

Python can run on a wide variety of operating systems and hardware platforms, has the same interface on all platforms.

5 Extendable

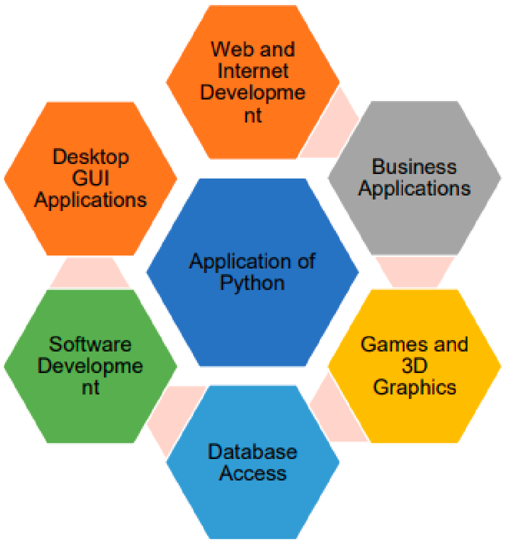
We can add low-level modules to the Python interpreter. These modules enable programmers to customize their tools to be more efficient.

6 Databases And Scalable

Python provides interfaces to all major open source and commercial databases along with a better structure and support for large programs than shell scripting.

Applications of Python

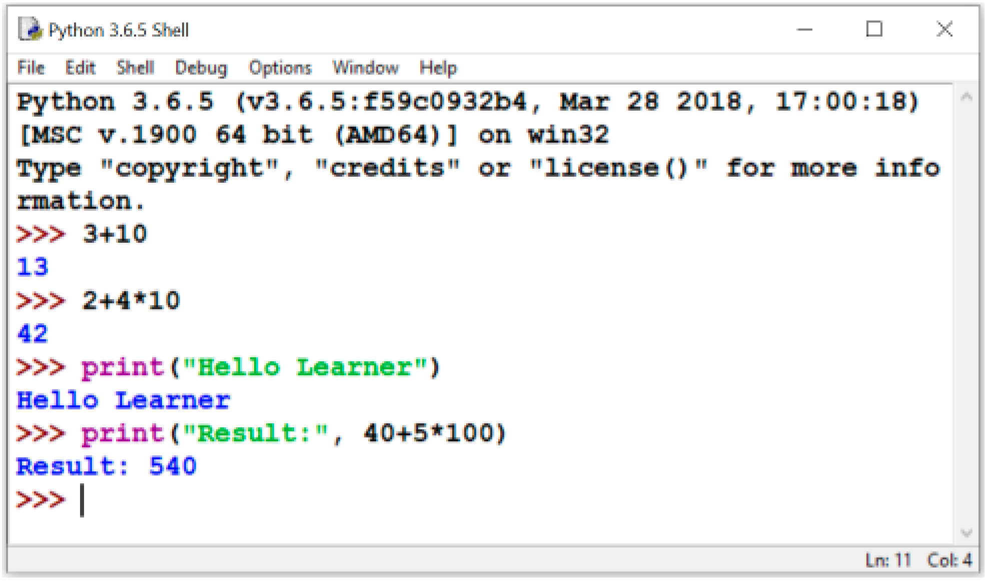
Python is used for a large number of applications. Some of them are mentioned here:



In Detail………………

When we install Python, an IDE named IDLE is also installed. We can use it to run Python on our computer. IDLE (GUI integrated) is the standard, most popular Python development environment. IDLE is an acronym of Integrated Development Environment. It lets one edit, run, browse and debug Python Programs from a single interface. This environment makes it easy to write programs. Python shell can be used in two ways, viz., interactive mode and script mode. Where Interactive Mode, as the name suggests, allows us to interact with OS; script mode lets us create and edit Python source file.

Interactive Mode

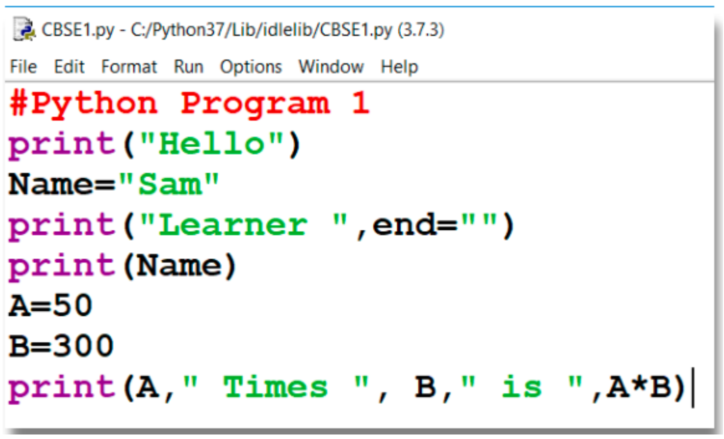


```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18)
[MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more info
rmation.
>>> 3+10
13
>>> 2+4*10
42
>>> print("Hello Learner")
Hello Learner
>>> print("Result:", 40+5*100)
Result: 540
>>> 
```

Script Mode

In script mode, we type Python program in a file and then use the interpreter to execute the content from the file. Working in interactive mode is convenient for beginners and for testing small pieces of code, as we can test them immediately. But for coding more than few lines, we should always save our code so that we may modify and reuse the code.

Python Script/Program: Python statements written in a particular sequence to solve a problem is known as Python Script/Program.


CBSE1.py - C:/Python37/Lib/idlelib/CBSE1.py (3.7.3)
File Edit Format Run Options Window Help

```
#Python Program 1
print("Hello")
Name="Sam"
print("Learner ",end="")
print(Name)
A=50
B=300
print(A," Times ", B," is ",A*B)
```

Python Statement and Comments

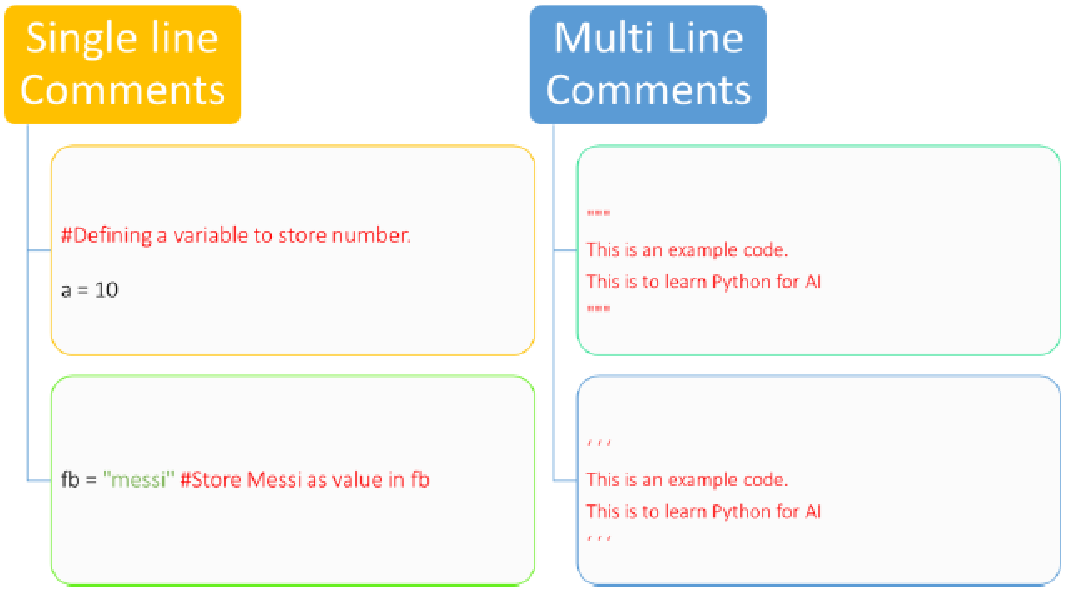
In this section we will learn about Python statements, why indentation is important and how to use comments in programming.

Python Statement

Instructions written in the source code for execution are called statements. There are different types of statements in the Python programming language like Assignment statement, Conditional statement, Looping statements etc. These help the user to get the required output.
For example, n = 50 is an assignment statement.

Python Comments

A **comment** is text that doesn't affect the outcome of a code, it is just a piece of text to let someone know what you have done in a program or what is being done in a block of code.
In Python, we use the hash (#) symbol to start writing a comment.



```
#Defining a variable to store number.
a = 10
```

```
fb = "messi" #Store Messi as value in fb
```

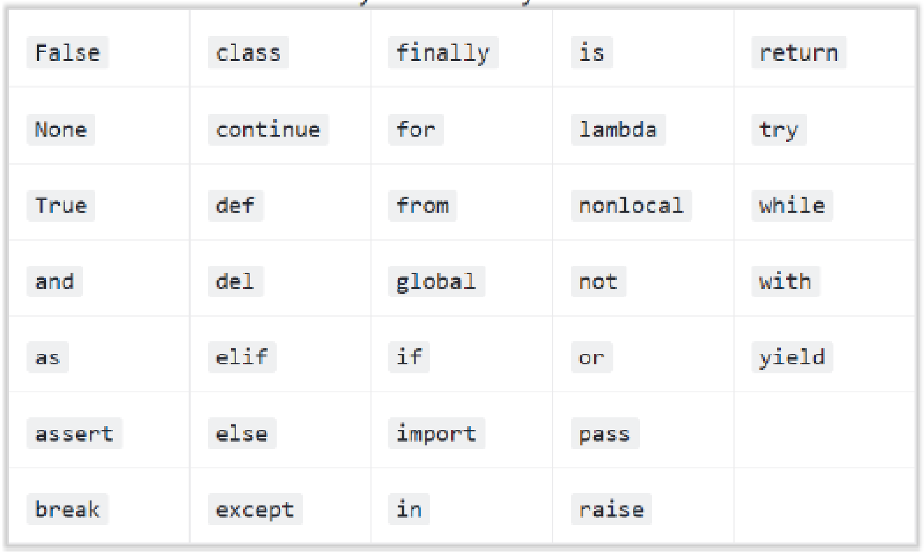
```
"""
This is an example code.
This is to learn Python for AI
"""
```

```
'''
This is an example code.
This is to learn Python for AI
'''
```

Python Keywords and Identifiers

Keywords are the reserved words in Python used by Python interpreter to recognize the structure of the program.

Keywords in Python



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

An **Identifier** is a name given to entities like class, functions, variables etc. It help to differentiate one entity from another.

Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore _.

An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.

Keywords cannot be used as identifiers.

We cannot use special symbols like !, @, #, $, % etc. in our identifier.

Identifier can be of any length.

Python is a case-sensitive language. This means, `Variable` and `variable` are not the same. Always name identifiers that make sense.

While, `c = 10` is valid. Writing `count = 10` would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.

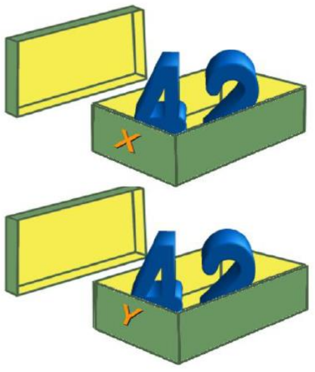
Multiple words can be separated using an underscore, for example **this_is_a_long_variable.**

Variables and Datatypes

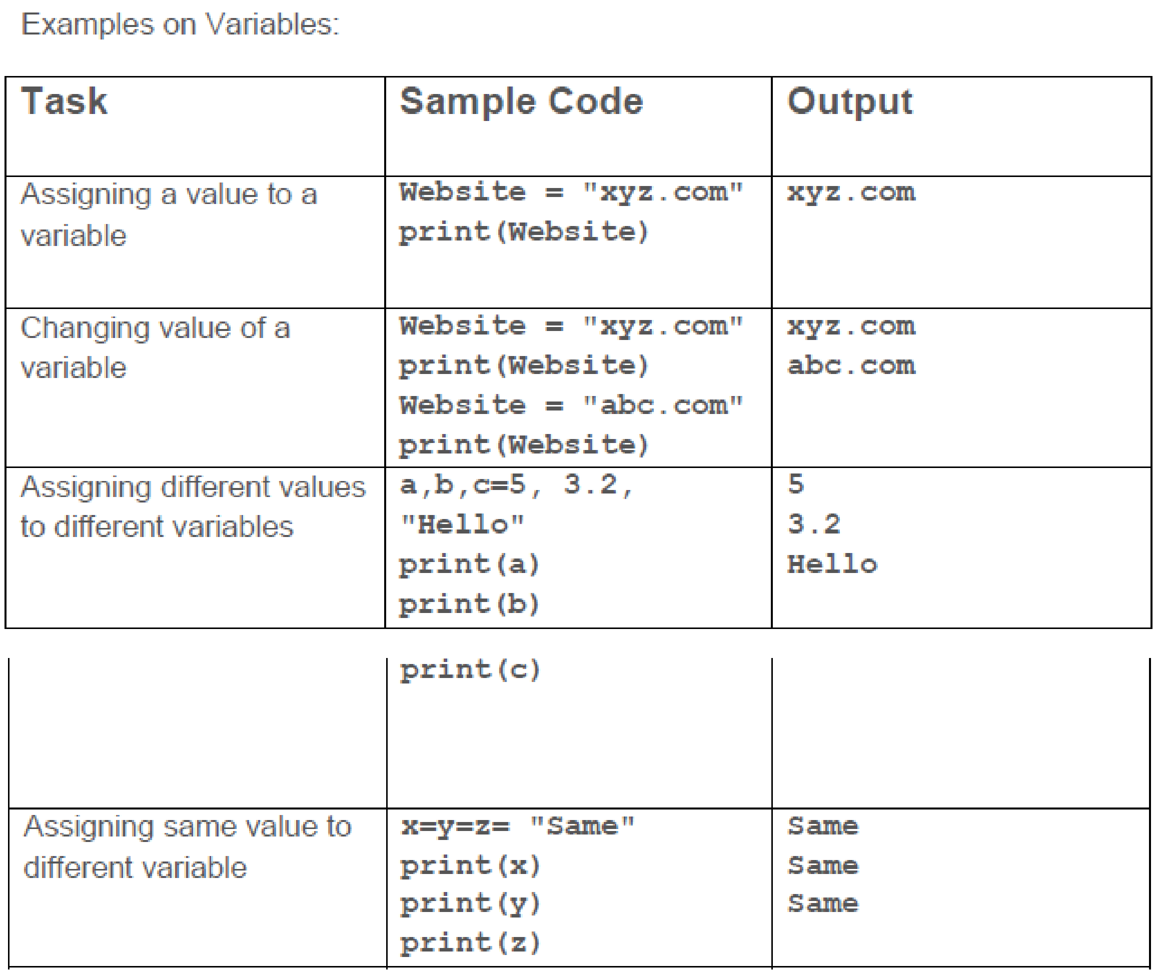
Variables

A variable is a named location used to store data in the memory. It is helpful to think of variables as a container that holds data which can be changed later throughout programming. For example,

```
x = 42
y = 42
```



These declarations make sure that the program reserves memory for two variables with the names x and y. The variable names stand for the memory location. It's like the two shoeboxes, which you can see in the picture. These shoeboxes are labelled with x and y and the corresponding values are stored in the shoeboxes. Like the two shoeboxes, the memory is empty as well at the beginning.



Examples on Variables:

Task	Sample Code	Output

Assigning a value to a variable	`Website = "xyz.com"`
	`print(Website)`
		`xyz.com`

Changing value of a variable	`Website = "xyz.com"`
	`print(Website)`
	`Website = "abc.com"`
	`print(Website)`
		`xyz.com`
		`abc.com`

Assigning different values to different variables	`a,b,c=5, 3.2, "Hello"`
	`print(a)`
	`print(b)`
	`print(c)`
		`5`
		`3.2`
		`Hello`

Assigning same value to different variable	`x=y=z= "Same"`
	`print(x)`
	`print(y)`
	`print(z)`
		`Same`
		`Same`
		`Same`

Constants:
A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.
Non technically, you can think of constant as a shoe box with a fixed size of shoe kept inside which cannot be changed after that.

Assigning Value to a constant in Python
In Python, constants are usually declared and assigned on a module. Here, the module means a new file containing variables, functions etc. which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words.

Example : Declaring and assigning value to a constant

- • **Create a info.py**

```
NAME = "Ajay"
AGE = 24
```

- **Create a main.py**

```
import info
print(info.NAME)
print(info.AGE)
```

- • **When you run the program the output will be,**

```
Ajay
24
```

In the above program, we create a constant.py module file. Then, we assign the constant value to PI and GRAVITY. After that, we create a main.py file and import the constant module. Finally, we print the constant value.

Note: In reality, we don't use constants in Python. The global or constants module is used throughout the Python programs.

Create a name that makes sense. Suppose, vowel makes more sense than v.

Use camelCase notation to declare a variable. It starts with lowercase letter. For example: myName

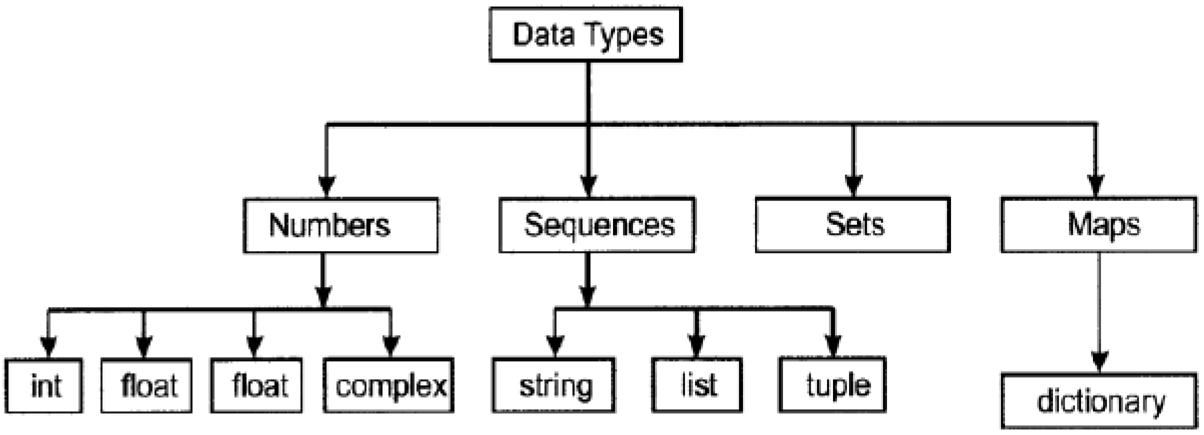
Use capital letters where possible to declare a constant. For example: PI

Never use special symbols like !, @, #, $, %, etc.

Constant and variable names should have combination of letters in lowercase or uppercase or digits or an underscore (_).

Datatypes

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes. There are various data types in Python. Some of the important types are mentioned below in the image.



Python Data types

1. Python Numbers

Number data type stores Numerical Values. These are of three different types:

a) Integer & Long
b) Float / floating point
c) complex

Integer & Long Integer
Range of an integer in Python can be from -2147483648 to 2147483647, and long integer has unlimited range subject to available memory.
Integers are the whole numbers consisting of + or – sign with decimal digits like 100000, -99, 0, 17. While writing a large integer value, don’t use commas to separate digits. Also, integers should not have leading zeros.
Floating Point:
Numbers with fractions or decimal point are called floating point numbers.
A floating-point number will consist of sign (+,-) sequence of decimals digits and a dot such as 0.0, -21.9, 0.98333328, 15.2963.
Complex:
The Complex Data Type in Python is the data type that is used for representing complex numbers. As mentioned above a complex number is represented in the form of “a + bi'. In Python, we use the j or J suffix instead of i, for denoting the imaginary part of the complex number.
For example, 3 + 3j.

2. Sequence

A sequence is an ordered collection of items, indexed by positive integers. It is a combination of mutable and non-mutable data types. Three types of sequence data type available in Python are:

a) Strings
b) Lists
c) Tuples

String
String is an ordered sequence of letters/characters. They are enclosed in single quotes (‘ ‘) or double (“ “). The quotes are not part of string. They only tell the computer where the string constant begins and ends. They can have any character or sign, including space in them.
Example: **name = “AJAY”**
Long_sent=”This is a string”
Month=’January’

Lists
List is also a sequence of values of any type. Values in the list are called elements / items. These are indexed/ordered. List is enclosed in square brackets.
Example: **dob = [19,"January",1990]**

Tuples:
Tuples are a sequence of values of any type, and are indexed by integers. They are immutable. Tuples are enclosed in ().
Example: **t = (5,'program',2.5)**

Python Operators

Operators are special symbols which represent computation. They are applied on operand(s), which can be values or variables. Same operators can behave differently on different data types. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment. Value and variables when used with operator are known as operands.

Arithmetic Operators

Operator	Meaning	Expression	Result
+	Addition	10 + 20	30
-	Subtraction	30 - 10	20
*	Multiplication	30 * 100	300
/	Division	30 / 10	20.0
		1 / 2	0.5
//	Integer Division	25 // 10	2
		1 // 2	0
%	Remainder	25 % 10	5
**	Raised to power	3 ** 2	9

Comparison operators

Comparison operators are used to compare values. It either returns True or False according to the condition**.**

Operator	Meaning	Expression	Result
>	Greater Than	20 > 10	True
		15 > 25	False
<	Less Than	20 < 45	True
		20 < 10	False
==	Equal To	5 == 5	True
		5 == 6	False
!=	Not Equal to	67 != 45	True
		35 != 35	False
>=	Greater than or Equal to	45 >= 45	True
		23 >= 34	False
<=	Less than or equal to	13 <= 24	True
		13 <= 12	False

Logical operators

Logical operators are the and, or, not operators.

Operator	Meaning	Expression	Result
And	And operator	True and True	True
		True and False	False
Or	Or operator	True or False	True
		False or False	False
Not	Not Operator	not False	True
		not True	False

Assignment operators

Operator	Expression	Equivalent to
=	X=5	X = 5
+=	X +=5	X = X + 5
-=	X -= 5	X = X - 5
*=	X *= 5	X = X * 5
/=	X /= 5	X = X / 5

Type Conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion. Python has two types of type conversion. 1. Implicit Type Conversion 2. Explicit Type Conversion

Implicit Type Conversion

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

Example:

```
# Code to calculate the Simple Interest

principle_amount = 2000
roi = 4.5
time = 10

simple_interest = (principle_amount * roi * time)/100

print("datatype of principle amount : ", type(principle_amount))
print("datatype of rate of interest : ", type(roi))


print("value of simple interest : ", simple_interest)
print("datatype of simple interest : ", type(simple_interest))
```

When we run the above program, the output will be

```
datatype of principle amount : <class 'int'>
datatype of rate of interest : <class 'float'>

value of simple interest :  900
datatype of simple interest : <class 'float'>
```

In the above program,

• We calculate the simple interest by using the variable priniciple_amount and roi with time divide by 100

• We will look at the data type of all the objects respectively

• In the output we can see the datatype of principle_amount is an integer, datatype of roi is a float.

• Also, we can see the simple_interest has float data type because Python always converts smaller data type to larger data type to avoid the loss of data.

Explicit Type Conversion

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like int(), float(), str(), etc to perform explicit type conversion.

This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

Example Code	Sample Output	Explanation

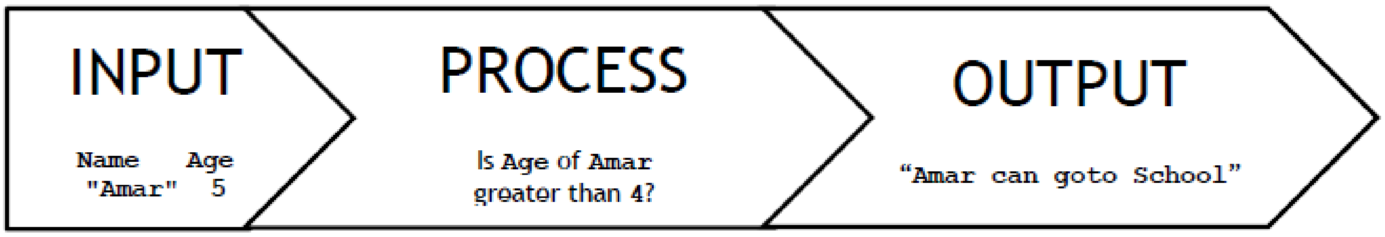
`a = 20`
`b = "Apples"`
`print(str(a)+ b)`
	20 Apples	Writing str(a) will convert integer a into a string and then will add to the string b.

`x = 20.3`
`y = 10`
`print(int(x) + y)`
	30	Writing int(x) will convert a float number to integer by just considering the integer part of the number and then perform the operation.

Python Input and Output

Python Output

Using print() function We use the print() function to output data to the standard output device (screen). We can also output data to a file.



An example is given below.

Example Code	Sample Output

`a = 20`
`b = 10`
`print(a + b)`
	`30`

`print(15 + 35)`	`50`
`print("My name is Kabir")`	`My name is Kabir`

`a = "tarun"`
`print("My name is :",a)`
	`My name is : tarun`

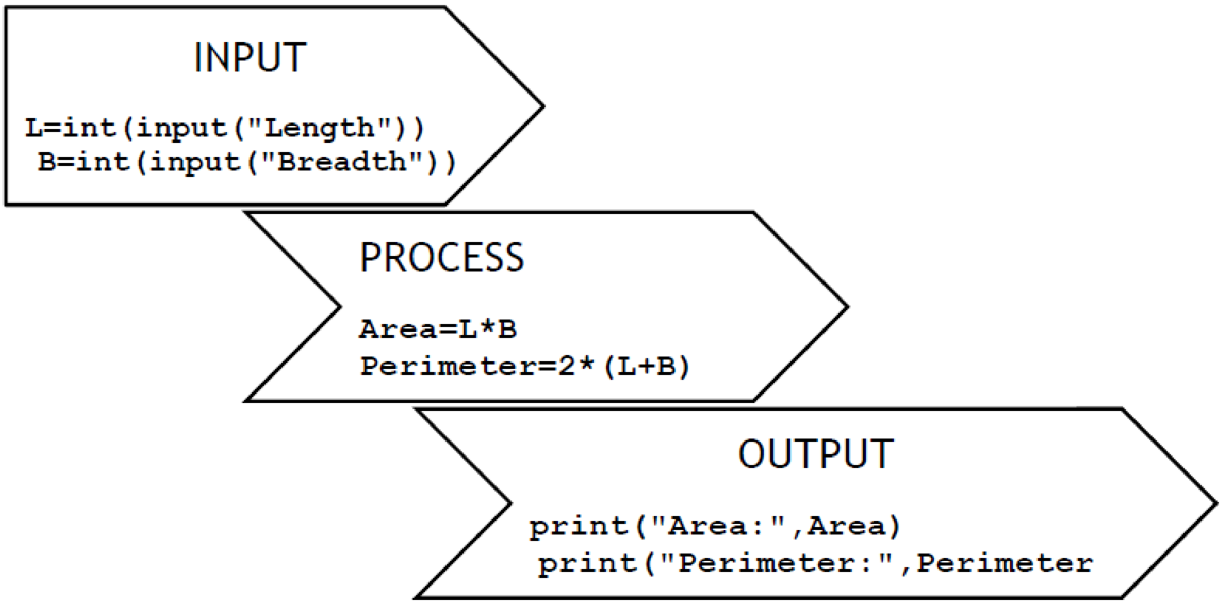
`x = 1.3`
`print("x = /n", x)`
	`x =`
	`1.3`

`m = 6`
`print(" I have %d apples",m)`
	`I have 6 apples`

User input

. In python, input() function is used for the same purpose.

Syntax	Meaning
`<String Variable>=input(<String>)`	For string input
`<integer Variable>=int(input(<String>))`	For integer input
`<float Variable>=float(input(<String>))`	For float (Real no.) input



```
L=int(input("Length"))
 B=int(input("Breadth"))
```

```
Area=L*B
Perimeter=2*(L+B)
```

```
print("Area:",Area)
 print("Perimeter:",Perimeter
```