Case sensitive,

* Python - High level, object oriented, portable, interactive & interpreted prgm language.

* Python converts the source code into intermediate code called byte code which is then translated to machine lang and is executed.

* Bytecode makes the python a portable language since it can be copied and executed in any o/s.

* Python is developed by Guido Van Rossum & its version 1.0 is released in 1991.

* It is mainly used for web based, scientific & graphic design applications.

* In general python is an interactive language.

* We can run the python in command line or in GUI software (Integrated Development & learning Environment).

* Python IDLE works on different O/S.

* For writing programs we use an editor. The steps to write python program is

1) open an editor
2) Write the program
3) Save the file as filename.py where .py is the extension of Python.
4) You can run in the command line or in the GUI mode.

i) To run in command line type python filename.py.

ii) To run in GUI mode just press run icon in the toolbar.

Literals & Variables.

Literals are also called as ~~Variables~~ Constants. The Values of literals cannot be changed so it is called as Constants.

Eg: 7, 3.9, 'A' and "Hello"

Variables are the memory location to store values.

Variables are also called as Identifiers. Variables are the names given to store something. For naming a variable there are certain rules.

Rules

* First character should be alphabet or underscore
* Rest of the characters may be alphabet, digit or underscore.
* Variables are case sensitive. eg: Num is different from NUM.
* No spl characters except _ is allowed.

Valid Identifiers : my_name, _name, name_1, name, name1.

Datatypes

Datatype is used to identify the type of data that is stored in a variable.

There are 7 basic data types in Python. They are Int, Float, String, Boolean, List, Tuple and Dictionary.

Integer: used to store whole numbers.
Can store both +ve & -ve numbers.

eg: a = 42345

Here a is the integer datatype.

Float: used to store fractional numbers.
It can store both +ve & -ve numbers.

eg: a = 47.132

Here a is float datatype.

String: used to store group of characters.
Strings are represented in double quotes or single quotes.

eg: name = "Dhivya"

Here name is string datatype.

Boolean: used to store either TRUE or FALSE. It is a bpl datatype.

eg: Val = True

Here Val is Boolean Datatype.

List: Collection of items enclosed in square brackets. Each item in the list is seperated by comma. The Items of list can be of different data types. The values stored in list is accessed by index. Index start with 0 and end with n-1 if size of list is n.

eg: a = [11, "East Cross Street", "Salem"]

Here a is a list data type with three items.

a[0] = 11
a[2] = Salem.

Tuple : Tuple is similar to list enclosed within ( ). The only difference between the list and tuple is you can change the values in list where you cannot change in tuple. We can also say that tuple is read-only list. Tuples are immutable

eg: a = ("Arun", 100, 98, 84)

Here a is a tuple data type with four items which cannot be modified.

a[1] = 100
a[2] = 98.

Dictionary: Dictionary is used to store key-value pair. Dictionary is enclosed in { }. key and value is seperated by colon.

eg: Dict = {1: "one", 2: "two"}

Each value is identified by its key.

Dict[1] : one.
Dict[2] = two.

Escape Sequences

It is special characters starting with \. eg:

\n - to go to new line
\t - to print continuous 8 spaces.
\\ - To print \
\' - To print '
\" - To print "

Raw string

It will not handle any escape sequence. It is used to print exactly what we mentioned. Rawstring should prefix with r or R.

print (R," What \'s Your Name")

O/p will be What \'s Your Name.

Indentation

Whitespace at the beginning of line is called Indentation. Indentation is used to group statements as blocks. this means statements inside a block have same indentation

eg:

```
if (a>b):
    print ("Hello")
    print ("A is greater than B")
else print ("B is greater than A")
```

Comments

Comments are non executable statements in a program. Comments make the program more readable and understandable.

Single line comment - # single line

Multiline comment - ''' Multiline Comment '''

Reserved Words

Reserved words are predefined words that has specific meaning. The reserved words cannot be used as identifiers.

eg: and, if, print, while.

~~operators~~ Assigning Values to Variables.

It is not necessary to explicitly declare variables in Python. Declaration is automatically done if we assign value to a variable using = sign.

```
eg: num = 7
    percentage = 92.74
    name = "Kalpana"
```

Multiple Assignment

We can assign single value to more than one variable simultaneously.

```
eg: a = b = c = 0
```

Here value 0 is assigned to variables

a, b and c.

We can also assign different value to multiple variable simultaneously.

eg: a, b, c = 0, 1, 2

name, percentage = "Arun", 94.27

Default Functions

i) print() - used to print output to user. eg: ① print("Hello") prints Hello in output screen.

② a = 10
print(a) prints value of a in output screen.

ii) input() - used to get input from user. eg: a = input(). it will get the input from user and stores the value in variable a. It accepts input as string.

iii) type() - used to get the datatype of variable

eg: a = "Hello"
print(type(a)) - prints the datatype of a. (ie) it prints class <str> because "Hello" is string

iv) id() - prints or returns the address of variable

eg: a = 10
print(id(a)) - prints 1002 since a is stored in memory 1002.

v) append() - It is used to add or append values to list.

eg: a = [1,2,3]

a.append(4)

print(a)

prints 1,2,3,4 since 4 is appended to a.

vi) Slicing a string

You can extract subset of string in a original string using slice operator ([] and [:]). We should specify the index inside the slice operator.

eg: Str1 = "Python Program"

P	y	t	h	o	n		P	r	o	g	r	a	m
0	1	2	3	4	5	6	7	8	9	10	11	12	13
-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Stmt:

```
Print(Str1)
Print(Str1[2])
print(Str1[4:7])
print(Str1[10:])
print(Str1[:5])
print(Str1[-10])
Print(Str1[::-1])
Print(Str1*2)
Print(Str1 + "is easy")
```

Output:

```
Python Program
t
on P
gram
Python
o
margorP nohtyP
Python ProgramPython Program
Python Program is easy
```

vii) len() - It is a default function used to get length of string.

eg: a = "Hai Hello"

len_a = len(a)

print (len_a)

o/p - 9 (since number of characters in a is 9)

viii) Type Conversion - There are number of default functions available to convert from one datatype to another.

eg: ① a = 10

d = list(a)

Here a is converted to list and is stored in a.

② f = 17.2

i = int(f)

Here f is converted to int and is stored in i.

③ s = "12"

i = int(s)

Here s is converted to int and is stored in i.

Expression

Expression is the combination of operator and operand. Where operators are used to manipulate operands and operands are the values to be manipulated.

eg: 12 + 7 * 2

Here 12, 7, 2 are operands

+, *, - are operators.

Operators are classified into.

* Arithmetic operator
* Comparision / Relational operator
* Assignment operator
* Logical operator
* Unary operator
* Bitwise operator
* Shift operator
* Membership operator
* Identity operator.

Arithmetic operator

It is used to perform basic arithmetic operations.

operator	Description
+ (Addition)	Perform Addition
- (Subtraction)	Perform Subtraction
* (Multiplication)	Perform Multiplication
/ (Division)	Perform Division and return quotient.
// (Floor Division)	Perform Division and retu floor of quotient.
% (Modulus)	Return remainder of division
** (Exponent)	Returns the power Va

example :

```
a = 10
b = 5
```

```
add = a+b
Sub = a-b
p = a**b
mul = a*b
quo = a/b
rem = a%b
f = a//b
print (add, Sub, p, mul, quo, rem, f)
```

O/P : 15 5 100000 50 2 0 2

ii) Comparision / Relational operator

The operators are used to find the relationship between operands. Returns True if the operation is true, it returns false if the comparision is false. The result will be either True or false.

operator	Description.
< (less than)	Return True if operand 1 is less than operand 2 else return false
> (Greater than)	Return True if operand 1 is greater than operand 2 else return false
<= (less than or equal to)	Return True if operand 1 is < operand 2 else return false.
>= (Greater than or equal to)	Return True if operand 1 is > operand 2 else return false.
== (Equal to)	Return True if both operand are equal else return false
!= (Not Equal to)	Return True if both operand are not equal else return false

example: ① a = 10
b = 8

```
if (a < b):
    print ("A is small")
elif (b < a):
    print ("B is small")
```

O/P: B is small.

②
```
a = 12
b = 15
if (a == b):
    print ("Both are equal")
elif (a != b):
    print ("Both are Not equal")
```

O/P: Both are Not equal.

iii) Assignment operator

It is used to assign value to the operand. There are shortcut operators or inplace operators for special assignment.

operator	Description
= (Assignment)	Assign Right side value to Left side Variable
+=, -=, *=, /=, //=, %=, **= (Shortcut Assignment)	Perform the operation and assign value to Variable.

example:

```
a = 10
b = 6
b += 2
b *= 3
b **= 2
print (a, b)
```

O/P: 10 576

iv) Logical operator

There are three logical operators and, or and not. The result of logical operation will be either true or false. Non zero values are considered to be true in logical operation. zero is considered to be false in logical operation.

operator	Description.
and (logical and)	return true if both operands are true. else return false
or (logical or)	return true if either one of the operand is true else return false.
not (logical not)	Return the negation of input.

Truth Table

and

I_1	I_2	R
T	T	T
T	F	F
F	T	F
F	F	F

or

I_1	I_2	R
T	T	T
T	F	T
F	T	T
F	F	F

not

I	R
T	F
F	T

example: ①
```
a = 10
b = -4
c = 0
if ( a and b ):
    print (" a and b is True ")
elif ( b and c ):
    print (" b and c is True ")
```

O/P : a and b is True.

②
```
a = 10
b = -5
c = 0
if ( a or c ):
    print (" either a or c is
                        True ")
else :
    print (" Both a and c is
                        False ")
```

O/P either a or c is True

③
```
a = 0
c = not c
if (c):
    print (" c is True ")
else:
    print (" c is False ")
```

O/P c is True.

④
```
a = 10
b = 20
c = 15
if ( a < b) and ( a < c):
    print (" a is small ")
else:
    print (" a is not small ")
```

O/P : a is small.

v) unary operators

Unary operand acts on single operand. When an operand is preceded by - then it is called unary minus, else if it is preceded by + then it is called unary plus. If an operand has no sign then it is implicitly understood that it is +.

operator	Description
+ (unary plus)	used to represent +ve value.
- (unary minus)	used to represent -ve value.

eg: a = 5
b = -10
print (a, b)

O/P : 5 -10

vi) Bitwise operator

It perform bitwise operations on binary bits. The input is converted to binary and operation is carried out and result is converted back to decimal

operator	Description
Bitwise & (and)	If Both inputs are 1 then output is 1 else output is 0.
| (Bitwise or)	If any one of the input is 1 then output will be 1 else output is 0.

^ (Bitwise xor) — If both inputs are different then output is 1 else output is 0.

~ (Bitwise Not) — If input is 1 then output is 0. if input is 0 then output is 1.

Truth Table

(Bitwise and)

I_1	I_2	R
1	1	1
1	0	0
0	1	0
0	0	0

(Bitwise or)

I_1	I_2	R
1	1	1
1	0	1
0	1	1
0	0	0

(Bitwise xor)

I_1	I_2	R
1	1	0
1	0	1
0	1	1
0	0	0

(Bitwise Not)

I	R
0	1
1	0

Example:

```
a = 10
b = 13
res1 = a & b
res2 = a | b
res3 = a ^ b
res4 = ~ b
print (res1, res2,
       res3, res4)
```

O/P: 8 15 7 2

Working

a&b

```
a - 1010
b - 1101
-------
a&b - 1000
```

a|b

```
a - 1010
b - 1101
-------
a|b - 1111
```

a^b

```
a - 1010
b - 1101
-------
a^b - 0111
```

~b

```
b - 1101
-------
  0010
```

a=2.

0000 0010

~a = 1111 1101

1's = 0000 0010

0000 0011 => 3

1-3

vii) <u>Shift operator</u>

It also operates on bits of operand. First operand is value that we want to shift and second operand will be the number of times we want to shift

operator	Description
<< (left shift)	Shifts the bits of operand to left.
>> (Right shift)	Shifts the bits of operand to right.

Example

```
a = 5
b = 2
res1 = a << b
res2 = a >> b
print (res1, res2)
```

o/p : 20 1

Working

a << b

a - 0000 0101

0000 1010

0001 0100

a >> b

a - 0000 0101

0000 0010

0000 0001

(iii) Membership operator

There are two operators namely. in and not in. It is used to check whether left operand is present or not present in Right operand.

operator	Meaning
in	return true if value is present in Right operand
not in	return true if value is not present in right operand.

example

```
a = 5
b = 15
li = [1,2,3,4,5]
if (a in li):
    print (" a is present in list ")
else:
    print (" a is not present in list ")
```

```
if (b is not in li):
    print (" b is not in list ")
else:
    print (" b is present in list ")
```

O/P: a is present in list
b is not in list.

ix) Identity operator

There are two identity operator namely is & not is. It is used to check whether address of two variables are equal or not.

operator	Description
is	check and return true if both operands have same address
not is	check and return true if both operands dont have same address.

example:

```
a = 10
b = 10
c = 20
if (a is b):
    print (" a and b has same address ")
else:
    print (" a and b has different address ")
if (b is c):
    print (" b and c has same address ")
else:
    print (" b and c has different address ")
```

O/P : a and b has same address

b and c has different address.

Precedence and Associativity of operators

If an expression has operators with different precedence then the operator with higher precedence should be solved first.

If an expression has more number of operators of same precedence then it should be solved from right to left or from left to right. Python operators have left to Right Associativity.

eg: $x = 10 + 2 ** 3 * 7 + 7$

$= 10 + 8 * 7 + 7$

$= 10 + 56 + 7$

$= 66 + 7$

$x = 73$

0.0000010

1111101

1010

0101

5

0.0001011

Functions

Function is a block of statements that perform specific task. There are two types of functions.

Userdefined functions

Predefined functions (Built in Functions)

Userdefined functions - These are created by user based upon the need of user.

eg: def add ( ):

...

Here add is a userdefined function to add two numbers.

Predefined functions - These are the functions that are available in python.

eg: len (s) - It is a built in function to find length of strings.

Need for functions

* Divide the program into well defined smaller parts
* Understanding, testing the functions are easier when compared to large program
* Code Reusability and Code Size Reduction
* Speeds up program execution and program development.

User Defined Function

We want to define the function before we use it.

The function definition should start with a keyword def.

the keyword should be followed by function name and parantheses ().

The function name is the unique name to identify function.

The values to functions are passed as agruments or paramters.

After () colon should be placed.

The code block is properly indented to form a function block.

The function have the return statement which is optional. It can return some value or only it can return the control.

Function should be called by means of function call statement.

Function Definition :

It has two parts function header & function body.

Syntax:

```
def funname ( Variables ): // header
    function block
    return [expression]      } // body
```

Function call

Function call invokes the function. When a function is invoked the control of the program jumps to called function to execute the statements and when a return statement is encountered the control pass back to function call.

Syntax:

```
funname ( Variables )
```

Example : To add Two Numbers.

```
def add ( a,b ): // fun defenition
    X = a+b
    return x
Print ("enter two Values")
a = int (input())
b = int (input())
add_Val = add (a,b) // fun call.
print (" Result = ", add_Val)
```

O/P enter two values 10 20
Result = 30

Parameters & Arguments

The arguments that are used in function call is called actual arguments and the arguments that are used in function definition to receive the values is called formal arguments.

Actual arguments and formal arguments may be same or different.

eg: To find power of a number

```
def power_fun(num, p):          # formal arguments
    result = num ** p
    return result
print("enter the number")
n = int(input())
print("enter the power")
po = int(input())
x = power_fun(n, pow)           # actual arguments
print("Result = ", x)
```

O/P: enter the number 2
enter the power 3
Result = 8

Modules

Modules allows us to reuse one or more functions in our program that is defined in another program.

Module is a file with .py extension that has definitions of functions that you would like to reuse in another program.

We should import the particular module by import module-name at the first line.

Module-name.Variable is used to access the values

eg: import sys

If we want to include only particular variable or function from a module then you can use the from import statement.

eg: from math import pi

Illustrative Programs

① Exchange the values of two variables.

```
def exchange(a,b):
    temp = a
```

```
    a = b
    b = temp
    print ("After Swap", a, b)
print ("Enter the Values of a & b")
a = int (input())
b = int (input())
exchange (a, b)
```

```
O/P: Enter the Values
        of a & b
     10
     20
  After Swap 20 10
```

(2) Circulate the Values of N Variables.

```
def circulate (a, no, n):
    no = no % n
    return a[no:] + a[:no]
a = []
print ("enter size of list")
n = int (input())
print ("enter elements of list")
for i in range (0, n):
    a.append (int (input()))
print ("enter number of rotation")
no = int (input())
print (circulate (a, no, n))
```

```
O/P: enter size of list
        3
     enter elements
        10
```

20
30
enter number of rotation
2
[30, 10, 20]

③ Distance between two points

```
import math
def dis(x1, y1, x2, y2):
    return math.sqrt((x2-x1)**2
        +(y2-y1)**2)

print(" Enter Values of 1st Point")
x1= int(input())
y1= int(input())
print(" Enter Values of 2nd Point")
x2= int(input())
y2= int(input())
print(dis(x1, y1, x2, y2))
```

O/P

Enter Values of 1st Point
12
10
Enter Values of 2nd Point
13
15
5.099019513592