Program

It is a set a set of instructions that governs the processing. We give some inputs to a program, the program carries out some instructions on given inputs and produces an output.

PYTHON

It is high level cross platform interpreted based language used to create various types of application.

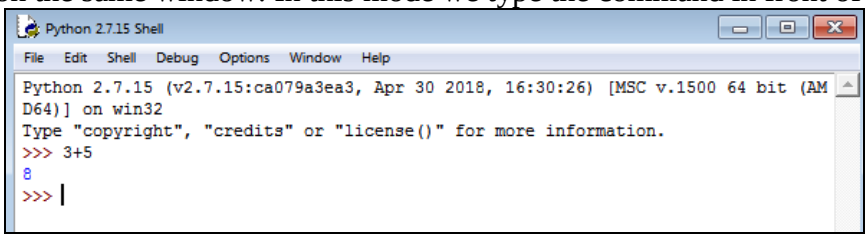
Features:

i. Highly case sensitive.
ii. Cross platform i.e. used in windows as well as linux based system.
iii. Interpreted based language.
iv. Create different types of application.
v. It is easy to write programs in python.
vi. No need to declare data types.

Python works in 2 modes

1. Interactive mode – it means you type command (one command at a time) and it executes it gives out the output on the same window. In this mode we type the command in front of python command prompt >>>.

for e.g.



To start this mode click on start menu →all programs →python 2.x OR python 3.x →IDLE (python GUI). It will open python shell where we will see the python prompt (three > signs i.e. >>>). In this mode we can't save our program. Outputs are sandwiched in between the commands.

2. Script mode – when we want to save all the commands in the form program file and want to see all output lines together rather than sandwiched between commands this mode is useful. To start script mode -
   i. click on start menu →all programs →python 2.x OR python 3.x →IDLE (python GUI)
   ii. Click file → new in IDLE python shell
   iii. Type your program and save it using .py extension by clicking on file→save
   iv. To run the progam click on the menu Run →Run module or use short cut key F5.

BASIC DATA TYPES

1. Numbers – used to save numeric values
   i. Integers – There is no fix size of integers in python. We can store up to whatever is supported by the memory of the machine. It does not contain fractional part.
      a. Plain integers e.g. - 5,0,-3
      b. long integers e.g. - 10L
      c. Boolean True and False
   ii. Floating point numbers – It contains fractional parts. E.g. 75.1425
   12 is an integer but 12.0 is a floating point number.
      a. Fractional form (normal decimal notation) – e.g. 5.12, -3.156
      b. Exponent notation e.g. 3.5245E03, 1.4457E-06

Advantage of floating point over intrgers

a. it can represent values between integers
b. it can represent much greater range of integer.

but the **disadvantage** is, it is slower than integer operations.

2. Strings – It holds string data. Any number of valid characters within quotation marks. String data may contain inside either single, double or triple quotes.
   a. Normal ASCII string – Contains only ASCII characters e.g. 'Computer', "Hello world",
   b. Unicode string – contains Unicode characters. Written same as ASCII but prefixed by u. e.g. - u'abc'

3. List – Python knows a number of compound data types, used to group together other values. The most versatile is the list, which as a list of comma-separated values (items) between square brackets. Lists might contain items of different types. List are mutable i.e. we can easily change the value of list if required.

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
```

```
>>> squares[0]  # indexing returns the item
1
>>> squares[-1]
25
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]
```

```
>>> cubes = [1, 8, 27, 65, 125]  # something's wrong here
>>> 4 ** 3  # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64  # replace the wrong value
>>> cubes
[1, 8, 27, 64, 125]
```

4. Tuple – Tuples are comma-separated values (items) between parenthesis ( ). Tuples might contain items of different types. Tuples are immutable i.e. we cannot change the value of list if required.
   Tup1 = (5, 6, 7)
   Tup2 = (‘a’,’1’, ‘c’)
5. Dictionary – Unlike all other data type that hold only a single value Dictionary keep all the records with a key value pair. It is written inside { }. Each key-value pair is separated by a colon: whereas each key is separated by a ‘comma’.
   Dict1={x:’one’,y:’two’,z:’three’,a:’four’,b:’five’}
   # To get value ‘three’
   Dict1[z]

TOKENS

In a passage of text, individual words and punctuation marks are called as tokens. The smallest individual units in a program are known as tokens. Python has following tokens – Keywords, Identifiers, Literals (Constants), Operators and punctuators.

1. **Keywords –** Keywords are the words which convey special meaning to the language. These are the reserved words for special purposes and must not be used as identifier names. E.g. – if, elif, else,and, or, not, int etc. float, while etc.

2. **Identifiers** – It is used for giving names to different parts of the program such as variables, objects, classes, function, list, tuples, dictionaries etc.
   Rules to form a identifiers:
   i. It a sequence of letters and digits.
   ii. The first character must be a letter or underscore (_).
   iii. Upper case and lower case are different.
   iv. The digits 0 to 9 can be part of identifier but it should not be the first character.
   v. Identifier must not be a keyword.
   vi. Identifier cannot contain any special character except underscore (_).

3. **Literals (Constants)** – Literals or constants are the items that have a fixed value. Python has following literals-
   i. String literals – It should be enclosed in quotes. E.g. ‘a’, “abc”.
   ii Numeric literals – e.g -1,5, 2.5, -17, 10+5j
   iii. Boolean literals – True or False
   iv. Special literals – None

4. **Operators** – It triggers some computation when applied to some variables and objects in an expressison. Variables and objects in which it is applied is called **operands.**

Unary operators – requires one operand

Unary + If a=5 then +a means 5
If a=0 then +a means 0
If a=-5 then +a means -5

Unary - If a=5 then -a means -5
If a=0 then +a means 0
If a=-5 then +a means 5

Binary operators – requires 2 operands. E.g. - a+b, 5*4

Arithematic operators

+ (Addition)	2+3 = 5
- (Subtraction)	5-4 = 1
* (Multiplicatiion)	3*7 = 21
/ (Division)	10/2 = 5
// (floor division)	17//4=4
% (remainder)	17%4=1
** (exponentiation)	2**4=16

Relational Operator- It returns True or False

<	(Less than)
>	(greater than)
<=	(less than or equal to)
>=	(greater than or equal to)
==	(equal to)
!=	(not equal to)

Assignment operators

=	(Assigment)
/=	(assign quotient)
+=	(assign sum)
-=	(assign difference)
*=	(assign product)
%=	(assign remainder)
**=	(assign exponent)
//=	(assign floor division)

Logical operator – AND, OR, NOT. It returns True or False. AND – True if both operands are True. OR – True if any one operand is True. NOT – returns True if operand is false.
Bitwise Operator - & (Bitwise AND), | (Bitwise OR), ^ (Bitwise XOR –set 1 only if only 1 bit is 1)
Shift Operator - << (left shift), >> (right shift)

5. **Punctuators** – These are the symbols which are used in a program to organize the structure in program. Examples of punctuators are : ‘ “ # \ ( ) {} [ ] , : ; ‘

COMMENTS IN PYTHON

This can be done in two ways:

(i) **Single line comment**: These are the readable information by the programmer but ignored by the python interpreter. In python comments begins with # and end with that particular line. E.g.

This is a comment.

A=10 # A is assign a value 10.
(ii) **Multiline comment** – This can be done by triple quoted multiline string:

```
''' Multiline comment
    Can be given
    Using triple quoted multiline string
'''
```

GETTING INPUT FROM USER

Pyhton 2.7 offers to functions **raw_input( ) and input( )** to obtain input from user and **print** statement to print the output.

raw_input( ) – This is the built in function to get input from user. It can be used in following manner-
variable_to_hold_the_value = raw_input(<prompt to be displayed>)
e.g.

```
>>>name=raw_input("Enter your name: ")
Enter your name: xyz
>>> age=raw_input("Enter age: ")
Enter age: 17
>>>print (name)
'xyz'
>>> print (age)
'17'
```

raw_input() always returns a sting type. So when you write
age=age+1
it will not give you 18, it will show you an error. So to remove this error you need to change the string type onto integer type as like this.

```
>>> age=int(raw_input("Enter age: "))
```

This method is called type conversion. This should be done in case of integer and floating point numbers.

input( ) – This evaluate the given value and returns accordingly.

```
>>> age=input("Enter age: ")
Enter age: 17
>>>age=age+1
>>>print (age)
18
```

With input function there is no need of type conversion.
In latest python release (3.x) the **raw_input()** has been renamed as **input()** and the old input() function of release (2.x) has been removed.

OUTOUT THROUGH PRINT STATEMENT

The print statement is away to print output in the console. The syntax is

```
>>>print "hello"
hello
```

Features of print statement

1. It converts the items into string. i.e if you are printing a numeric value firstly it will convert it into string and print it.
2. It inserts spaces automatically.
3. It appends new line unless a comma is given at the last position of the statement. For e.g.

```
a,b=10,20
print "a=",a
print "b=",b
```

it will give you output:
a=10
b=20

and if you write like
a,b=10,20
print “a=”,a,
print “b=”,b

it will give you output:
a=10 b=20

CONDITIONAL AND ITERATIVE STATEMENTS

Generally a program executes its statement from beginning to the end but not all the programs. Depending upon the need sometimes it executes one of the available alternatives or even repeat some set of instructions. Python, of course, provides such tools to attain this.

CONDITIONAL STATEMENT

It helps us to select one statement out of 2 or more available alternatives.

if statement

It is the simplest form, if statement check the condition and if it is true, it carries out some instructions and does nothing if the condition is false. E.g.

```
if (age>18) :
        print “you are eligible to vote.”
```

if may contain single statement or multiple statements inside if condition. if statement must be ended with colon (:). Statement/s inside if must be properly indented at the same level. E.g.

```
if (age>18) :
        print “Hello”,
        print “you are eligible to vote.”
```

if – else statement

In this type if the condition is true, it carries out some instructions and if the condition is false, it carries out some other set of instructions. E.g.

```
if (age>18) :
        print “you are eligible to vote.”
else :
        print “you are not eligible to vote.”
```

if – elif statement

In this type there are multiple blocks of checking condition and if any block condition is true it executes its statement and if all the condition are false then else part is executed. E.g.

```
if (run > 100) :
        print (“batsman scored century”)
elif (run > 50) :
        print (“batsman scored half century”)
else :
        print (“batsman neither scored century or half century”)
```

ITERATIVE STATEMENT

When we need to execute a part of statement/s more than one time based on some checking condition it is said to be iterative statement or simply looping.

Before we go through we must know **range( )** function in python which is used with for statement. To declare a range of items the syntax is : range(lower limit, upper limit)
both the limit should be integer. Also note that lower limit is included but upper limit is excluded.

```
>>>a=range(1,6)
>>>print(a)
```

This will produce list as [1,2,3,4,5]
The default step value will be +1

To change the step value we will write as range(lower limit, upper limit, step value)

```
>>>b=range(0,10,2)
>>>print (b)
```

Output: [0,2,4,6,8]

```
>>>c=range(5,0,-1)
>>>print (c)
```

Output: [5,4,3,2,1]

Another form of range is : range(number) : it will create from 0 to number-1.
range(5) : Output: [0,1,2,3,4]

for statement

for loop is a counting loop i.e. the loop repeats a certain number of times. It is designed to process the items of any sequence such as list or string. Examples.

e.g.1

```
for a in [1,3,5] :
    print (a)
    print(a*a)
```

Output
1
1
3
9
5
25

e.g.2

```
for ch in'computer' :
    print (ch)
```

Output:
c
o
m
p
u
t
e
r

e.g.3

```
num=5
for a in range(0,6) :
    print (num,' x ',a,' = ',num*a)
```

Output:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25

program to print sum of natural nos. from 1 to 10 using for loop.

```
sum=0
for n in range(1,11) :
    sum = sum + n
print ("Sum of nos. = ", sum)
```

while loop

while loop is a conditional loop i.e. it is a loop that repeats till some condition is true. For e.g.

e.g.1

```
a=5
while a > 0 :
    print ("hello ", a)
    a = a – 3
print ("Loop over!")
```

Output :
hello 5
hello 2
Loop over!

e.g.2

```
n=1
while n<5 :
    print ("Square of ", n, " is ", n*n)
    n=n+1
print ("Loop over")
```

Output:
square of 1 is 1
square of 2 is 4
square of 3 is 9
square of 4 is 16
Loop over

While loop contains four parts:

1. Initialization : before entering a while loop its loop variable must be initialized.
2. Test expression : Expression whose truth value (True or False) decides whether loop will execute or not.
3. Body of loop : The statements that are executed as long as condition is True.
4. Update expression : It changes value of the loop variable. It is given as a statement inside body of the loop.

While lop may also contains else part as like for loop.
e.g.

```
n=10                    # Line 1 - Initialization
while n > 0 :           # Line 2 - Test expression
        print (n)       # Line 3 & 4 – Body of the loop
        n=n-1           # Line 4 – Update expression
```

The loop variable must be updated inside body of while loop in a way that condition becomes false after a number of times otherwise loop becomes endless. For e.g.

```
a=5
while a > 0 :
        print (a)
print (“Thankyou”)     # In this case program continuous prints 5. It will never end.
```

Program to print factorial of a number

```
num = int(input("Enter a number : "))
fact = 1
a=1
while a <= num :
   fact=fact*a
   a=a+1
print ("fact of ",num," is ",fact)
```

Output :

```
Enter a number : 5
fact of 5 is 120
```

Program to print sum up to n natural nos using while loop.

```
num = int (input("Enter a number:"))
sum=0
while (num>0) :
   sum = sum + num
   num=num-1
print ("sum is ",sum)
```

Output:

```
Enter a number:6
sum is 21
```