Fundamentals of the Python:

Token is nothing but smallest individual component in the program

Python Tokens Are :

1.Character SET

Set of characters which are supported by Python Language interpreter

* Python will support Unicode char SET
* Unicode = ASCII and Non ASCII
* Non Ascii = Other Language Characters
  [National | International]
* Range : 0 to 65535
* We can develop Language Friendly Application

2.Variables

* It is a space to store the data or
* It is named container which enable you to store the data temporally during the program execution

Java :

Syn:[modifiers]b<datatype>b<identifiers>[=value];

Actually here b is nothing a space

Python:

Syn:<identifier>=<value>[;]

```
eno=10
ename="Ramesh";
```

3.Datatypes

4.Operators

5.Identifiers

Are nothing but all the names which are declared by us for our programming requirements, Such as : Variable names, Function names, Class names.....

Rules :

1. It must starts with an alphabet or _
2. It may be in Lower | Upper | Mixed cases
3. No Limit in the length of identifiers
4. It May Have digits
   Eg: e00no; ena98me; [valid]
5. It may have a Special character [i.e. : _]
6. It should not be Python Keyword

6.Keywords

* These are nothing but reserved words
* Every keyword is having its importance in the program
* The meaning of the keyword can't be changed
* To know the keywords existed in the python then we have to use kwlist [predefined Variable of type <class 'list'>]
  Existed in keyword module

```
import keyword
keyword.kwlist   [shell and idle]
In script Mode --> print(keyword.kwlist)
```

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

```
>>> type(keyword.kwlist)
<class 'list'>
>>> len(keyword.kwlist)   #35
```

iskeyword() :
It returns True if the given String is a keyword else it will return False
Eg: >>> keyword.iskeyword('Roja') #False
>>>keyword.iskeyword('as') #True

Print( ) :

- It an output functions from builtins module
- It used to display the result on the console

```
x=10
print(x)  #10
print("welcome")  #welcome
```

Eg2:

```
x=10
y=3.14
z="welcome"

print(x) #10
print(y) #3.14
print(z) #welcome

print(value,value,value,.....)
print(x,y,z)   #10 3.14 welcome
```

Note : While printing multiple values using print function, then each value is separated by a space, if you want provide any literal then we have to use sep attribute in print( )

Eg3: Using sep attribute

```
x=10
y=3.14
z="welcome"

print(x,y,z)    #if u want any literal as sep
                  #sep attribute in print( )
                  #default value for sep attribute is space [ sep=' ' ]

print(x,y,z,sep=' ') # 10 3.14 welcome
print(x,y,z,sep='-') # 10-3.14-welcome
print(x,y,z,sep=',') # 10,3.14,welcome
print(x,y,z,sep='') #103.14welcome
```

Note: While printing the values using print (), it will print the result on the screen and throws the cursor to the newline. Just because the default value of “end” attribute is ‘\n’. if you want print in result in same line then we have to use “end” attribute with end=’’.

Eg 4: Using end attribute in print( )

```
x=10
y=3.14
z="welcome"

print(x)   # print( ) -> printf( ) + "\n" [In C]
print(y)   # end attribute in print( )
print(z)   # the default value for end attribute is \n

print(x,end='\n')
```

```
print(y,end='\n')
print(z,end='\n')
print(x,end='')
print(y,end='')
print(z,end='')  # print(x,y,z,sep='')
```

We can use the Esc Sequence Char in both end or sep attribute of print( ) if required

\n New Line
\a Bell sound
\t Horizontal .Tab(4)
\b Backspace

\\ It will print \ symbol
\' It will print ' symbol
\" It will print " Symbol ...

```
x=10
y=3.14
z="welcome"

print(x,y,z,sep=' ')
print(x,y,z,sep='\t')


print(x,end='\t')
print(y,end='\t')
print(z,end='\t')
```

Printing the value of variables using format specifier's

Format Specifies : These are used to specify what type values to be formatted during output.

int --> %d

float --> %f
string -> %s

Syn: print("formatspecifier" %variable)
print("formatspecifiers" %(list of variables))

```
x=10                  #int
y=3.14                #float
z="welcome"       #str

print("%d" %x)
print("%f" %y)

print("x val is : %d " %x)
print("y val is : %f " %y)

print("%d %f %s" %(x,y,z))
print("x:%d y:%f z:%s " %(x,y,z))
          #x:10 y:3.14000 z:welcome
```

print( ) using replacement fields { }

- while working with replacement field then we have to use format( ) from the class : <class 'str'>.

Syn: print("msg | values ".format(variables))

Replacement Fields Can Be
- Index based
- Non Index Based
- Name based

```
name="sudha"
age=27
```

#Non Index Based
print("name :{}".format(name)) # name: sudha
print("name is :{} and age is :{}".format(name,age))

#IndexBased 0 1 –index values
print("name is :{1} and age is:{0} ".format(age,name))

#NameBased
print("Name is :{} and age is :{}".format(name,age))
print("Name is :Mr|Mrs.{n} and age is :{a},{n} is from Hyd". format(n=name,a=age))

#Output:
#Name is :Mr|Mrs.sudha and age is :27,sudha is from Hyd