Print Statement:

Docstring: print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

In [2]:
```
print(False)
```

```
False
```

In [3]:
```
print()
```

In [4]:
```
print('A','B')
```

```
A B
```

In [6]:
```
print('A','B',sep='/')
```

```
A/B
```

Shilft + tab 'for documentaionn view'

In [7]:
```
print('hello')
print('hi')
```

```
hello
hi
```

In [9]:
```
print('hello',end=' ')
print('hi')
```

```
hello hi
```

In [10]:
```
print(False,'hello',3.14,5)
```

```
False hello 3.14 5
```

In [ ]:
```

```

Python support 3 categories of data types

- Basic data type : integer, float, boolean and string
- Conatiner type : list, tuple, set, dict
- User defined type : class

Basic data type : integer, float, boolean and string

In [17]:
```
print(1e308)

# max int limit 1e309
print(1e309)
```

```
1e+308
inf
```

In [16]:
```
print(1.7e308)

# max float limit 1.7e309
print(1.7e309)
```

```
1.7e+308
inf
```

In [18]:
```
# boolenn

print(True)
print(False)
```

```
True
False
```

In [19]:
```
# complex
print(1+2j)
```

```
(1+2j)
```

In [21]:
```
# string
print('hello')
print("hi")
print("""Listen to BBC news""")
```

```
hello
hi
Listen to BBC news
```

Conatiner type : list, tuple, set, dict

In [23]:
```
print([1,2,3,4]) #List
```

```
[1, 2, 3, 4]
```

In [24]:
```
print((1,2,3,4)) #Tuple
```

```
(1, 2, 3, 4)
```

In [25]:
```
print({1,2,3,4}) #Set
```

```
{1, 2, 3, 4}
```

In [26]:
```
print({'name':'harsh','age':12,'gender':'male'})  #dict
```

```
{'name': 'harsh', 'age': 12, 'gender': 'male'}
```

In [ ]:
```

```

Comment :

- A piece of code which is not exectuable by the compiler or interpreter.
- Used to improve code readibility

In [27]:
```
# comment
```

Python does not support multi line comment

- Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:

In [30]:
```
"""
This is a comment
written in
more than just one line
"""
print('hi')
```

```
hi
```

In [ ]:
```

```

Variables :

- Variables are containers for storing data values.
- no need to mention data type : python support dynamic typing
- one variable can store multiple data tyoes : dynamic binding
- Python has no variable declaration

Dynamic typing

In [31]:
```
x = 5
y = "John"
z = 3.14
k = True
print(x,type(x))
print(y,type(y))
print(z,type(z))
print(k,type(k))
```

```
5 <class 'int'>
John <class 'str'>
3.14 <class 'float'>
True <class 'bool'>
```

Dynamic binding

In [33]:
```
x = 5
print(x,type(x))

x = "John"
print(x,type(x))

x = 3.14
print(x,type(x))
```

```
5 <class 'int'>
John <class 'str'>
3.14 <class 'float'>
```

Special declaration syntex

In [35]:
```
a=1;b=2;c=3
print(a,b,c)
```

```
1 2 3
```

In [36]:
```
d,e,f=3,3,3
print(d,e,f)
```

```
3 3 3
```

In [37]:
```
g=h=i=3
print(g,h,i)
```

```
3 3 3
```

In [ ]:
```

```

Identifiers and Keywords

- Python is a case sensitive programming launguage
- In Python, keywords are case sensitive. There are 33 keywords in Python 3.7.

Keywords :

- Keywords in Python are reserved words that can not be used as a variable name, function name, or any other identifier.
- Every programming laung. has a set of keywords that cannot be used as a variable name
- Because compiler will be confused, if we use the keywords as variable name

In [38]:
```
# python has 33 keywords

import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'bre
ak', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'fina
lly', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'n
onlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'wit
h', 'yield']
```

Identifiers :

- Python Identifiers is a name which is used to identify a variable, function, class, module or other object

or

- Identifiers are user-defined names. It helps to distinguish one identity from another.

Rules for identifiers ::

Rule 1: Can only start with alphabets or _

We can use either a lowercase (A to Z) or an uppercase (A to Z) sequence of letters. However, you can also add digits (0 to 9) or an underscore (_) while writing identity in python.

For Ex: Names like myClass, my_1, and upload_image_to_db are all valid identifiers.

Rule 2: You cannot write digit with the start of an identifier. This will assume invalid. But you can write digit with the end of the identifier.

For Ex:- If you write identifier like 1variable, it is invalid, but variable1 is perfectly fine.

Rule 3: Reserved Keywords cannot be used as identifiers.

Like del, global, not, with, as, if, etc.

In [26]:
```
_h1 = 'hello'
```

In [27]:
```
_h1
```

Out[27]:
```
'hello'
```

In [28]:
```
0n = 'char'
```

```
  File "<ipython-input-28-493593717a51>", line 1
    0n = 'char'
     ^
SyntaxError: invalid syntax
```

In [29]:
```
first-name = 'harsh'
```

```
  File "<ipython-input-29-0e24842eb422>", line 1
    first-name = 'harsh'
                       ^
SyntaxError: can't assign to operator
```

In [31]:
```
first_name = 'harsh'
first_name
```

Out[31]:
```
'harsh'
```

In [34]:
```
last__name = 'shah'
last__name
```

Out[34]:
```
'shah'
```

In [ ]:
```

```