In [3]:
```
a=10
b=20
print(a+b+10)
```
```
40
```

In [6]:
```
print("This is New Block")
```
```
This is New Block
```

In [9]:
```
# user Input
# python 2 vs python 3 (input)
# Python 2 -> 1) raw_input("Enter Name")-> String  2) input("Enter Number") -> base
d on data
# Python 3 -> input("Enter String")

first_name=input("Enter First Name:")
last_name=input("Enter Second Name:")
# String Conacate
full_name=first_name+" "+last_name
print(full_name)
```
```
Enter First Name:s
Enter Second Name:s
s s
```

In [16]:
```
# eval()
data = eval(input("Enter data:"))
print(type(data))
```
```
Enter data:1.2
<class 'float'>
```

In [18]:
```
#in Python Every Data type is internally object
# int data type
num1=eval(input("Enter Number 1"))
num2=eval(input("Enter Number 2"))
# String Addtion
num3=num1+num2
print(num3)
```
```
Enter Number 11
Enter Number 22
3
```

In [21]:
```
# hint for assignment
a=1
if a==1:
    print("add")
elif a==2:
    print("sub")
else:
    print("div")
```
```
add
```

In [32]:
```
# ways to specify int
# Number System -> 1) Decimal Number System(10) 2)Binary Number System(2) 3)ocatal
Number System(8) 4)Hexadecimal Number System (0-9 A-F)

# Binary Number System
binary=0B1111
print(binary)

# ocatal Number System
octal=0O754
print(octal)

# Hexadecimal Number System
hex=0X09ABF
print(hex)

# Built Functions
# hex(int)
# bin(int)
# bin(int)
```

```
15
492
39615
```

In [36]:
```
# Float Data Type
num=1.2
print(type(num))
print(num)
num=1.2E10
# only decimal
# num=0b111.0101
print(num)
```

```
  File "<ipython-input-36-b19babc5dfdd>", line 7
    num=0b111.0101
                 ^
SyntaxError: invalid syntax
```

In [40]:
```
# complex data type
complex=2+5j
# 2-> real Part
# 5-> imaginary part
print(complex)
# incuilt attribuites
print(complex.real)
print(complex.imag)

# int in real not in imag
complex=0b0101+5j
print(complex)
# complex=1+0b111j
# print(complex)
```

```
(2+5j)
2.0
5.0
(5+5j)
```

In [43]:

```
# check keyword
import keyword
print(keyword.kwlist)
```

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

In [48]:

```
# bool
bool=True
print(bool)
bool2=False
print(bool2)
bool3=bool-bool2
print(bool3)
```

```
True
False
1
```

In [52]:

```
# String -> Collections of characters
name="This is String"
name='This is String'
print(name)
mutil='''This
is
mutltiline '''
print(mutil)
```

```
This is String
This
is
mutltiline
```