In [1]:
```
"python"
```

Out[1]:
```
'python'
```

In [2]:
```
list1=67
```

In [2]:
```
list1=10
```

In [4]:
```
list_1=10
```

In [5]:
```
list=10
```

In [6]:
```
import keyword
keyword.kwlist
```

Out[6]:
```
['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']
```

In [10]:
```
a=10
type(a)
```

Out[10]: int

In [9]:
```
d=9.8
type(d)
```

Out[9]: float

In [11]:
```
string="python"
type(string)
```

Out[11]: str

In [12]:
```
print(string)
```

```
python
```

In [ ]:
```
number,numerical=int, float, complex
text=str
mapping=dict
sequence= list,tuple,set,frozenset,range()
set=set, frozenset
boolean=True and False
```

In [13]:
```
c=10+1j
```

In [15]:
```
a=10
print(a)
```

```
10
```

In [16]:
```
f=2.4
print(f)
type(f)
```

```
2.4
```

Out[16]: float

In [17]:
```
string1="python"
string2='python'
string3="""python"""
string4='''python'''
```

In [ ]:
```
list1=[1,2,3,4,5,6]
tuple=(1,2,3,4,5,6)
```

In [18]:
```
list1=[1,2,3,4,5,6]
type(list1)
```

Out[18]:
```
list
```

In [19]:
```
tuple1=(1,2,3,4,5,6)

type(tuple1)
```

Out[19]:
```
tuple
```

In [20]:
```
set1={1,2,3,4,5}
type(set1)
```

Out[20]:
```
set
```

In [21]:
```
a=10
b=20
c=30
a,b,c=10,20,30
```

In [22]:
```
a,b,c=10,20
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[22], line 1
----> 1 a,b,c=10,20

ValueError: not enough values to unpack (expected 3, got 2)
```

In [23]:
```
a,c=10,20,30
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 1
----> 1 a,c=10,20,30

ValueError: too many values to unpack (expected 2)
```

In [24]:
```
a=5 ; b=20 ; c=30
```

In [25]:
```
print(a)
```

```
5
```

In [26]:
```
a=10
print(A)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call las
t)
Cell In[26], line 2
      1 a=10
----> 2 print(A)

NameError: name 'A' is not defined
```

In [ ]:
```
# python
```

In [63]:
```
def addition():
    '''Here i am adding a and b
    hfjakjehh
    dhyehf
    fkur'''
    a=20
    b=30
    add=a+b
    print(add)
    '''ahdh'''
print(addition.__doc__)
```

```
Here i am adding a and b
    hfjakjehh
    dhyehf
    fkur
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

int to float

In [28]:
```
a=10
b=float(a)
type(b)
```

Out[28]: float

In [29]:
```
a=4.8
b=int(a)
```

In [30]:
```
print(b)
```

```
4
```

In [31]:
```
a=10
b=complex(a)
b
```

Out[31]:
```
(10+0j)
```

In [32]:
```
b=2.8
c=complex(b)
```

In [33]:
```
c
```

Out[33]:
```
(2.8+0j)
```

In [34]:
```
s=8+8j
a=int(s)
print(a)
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[34], line 2
      1 s=8+8j
----> 2 a=int(s)
      3 print(a)

TypeError: int() argument must be a string, a bytes-like object or a real
number, not 'complex'
```

In [35]:
```
a=10
v=str(a)
```

In [37]:
```
print(v)
type(v)
```

```
10
```

Out[37]:
```
str
```

In [38]:
```
d=2.7
f=str(d)
type(f)
```

Out[38]:
```
str
```

In [ ]:
```
b=20        .... global variable
def a():
    a=10    .... local variable
```

In [39]:
```
a=20
b=30
add=a+b
print(add)
```

```
50
```

In [40]:
```
print("addition of a and b ",add)
```

```
addition of a and b  50
```

In [41]:
```
print(f" addition of {a} and {b} is {add}")
```

```
 addition of 20 and 30 is 50
```

In [45]:
```
c=40
d=20
add1 = c + d
print(f" additon of {c} and {d} is {add1}")
```

```
 additon of 40 and 20 is 60
```

In [47]:
```
c=40
d=20
add1 = c + d
print(" addition of {} and {} is {} ".format(c,d,add1))
```

```
 addition of 40 and 20 is 60
```

In [48]:
```
c=40
d=20
add1 = c + d
print("addition of %d and %d is %d" %(c,d,add1))
```

```
addition of 40 and 20 is 60
```

In [49]:
```
c=40
d=2.5
add1 = c + d
print("addition of %d and %d is %d" %(c,d,add1))
```

```
addition of 40 and 2 is 42
```

In [51]:
```
c=40
d=2.5
add1 = c + d
print("addition of %d and %f is %f" %(c,d,add1))
```

```
addition of 40 and 2.500000 is 42.500000
```

In [53]:
```
c=40
d=2.5
add1 = c + d
print("addition of %d and %.2f is %.2f" %(c,d,add1))
```

```
addition of 40 and 2.50 is 42.50
```

In [54]:
```
a1=10
b1=50
add=a1+b1
print(add)
```

```
60
```

In [55]:
```
print("addition of a and b is ",add)
```

```
addition of a and b is  60
```

In [56]:
```
print(f" addition of {a1} and {b1} is {add}")
```

```
 addition of 10 and 50 is 60
```

In [57]:
```
print("addition of {} and {} is {}".format(a1,b1,add))
```

```
addition of 10 and 50 is 60
```

In [58]:
```
print("addition of %d and %d is %d " %(a1,b1,add))
```

```
addition of 10 and 50 is 60
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```
+
-
*
/
%
//
**
```

In [ ]:
```
a=4
b=5
v=a+b
v=a-b
v=a*b
v=a/b
v=a%b
v=a//b
v=a**b
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```
==
!=
<=
>=
<
>
```

In [ ]:
```
a=5
a+=5    a=a+5
a-=5    a=a-5
a*=5    a=a*5
a/=5    a=a/5
a%=5    a=a*5
a//=5   a=a//5
```

In [ ]:
```
in not in
```

In [ ]:
```
is is not
```

In [ ]: