In [13]:
```
#Eval Functions always takes  a string it will first evaluate it and after evaluation
#convert that string into an object
x=eval("10+20")
print(x)
print(type(x))
```

```
(10+20))
<class 'complex'>
```

In [ ]:
```
[10,20,30,40,50]
{10:20,30:40,50:60}
(10,20,30,40)
(10,20,30,40)
[10,20,[20,30,40,50],10]
```

In [14]:
```
x = eval(input("Enter Your data"))
print(type(x))
print(x)
#"[10,20,30,40,50,60]"
```

```
Enter Your data[10,20,30,40,50]
<class 'list'>
[10, 20, 30, 40, 50]
```

In [17]:
```
x = eval(input("Enter Your data"))
print(type(x))
print(x)
```

```
Enter Your data{10:20,30:40,50:60}
<class 'dict'>
{10: 20, 30: 40, 50: 60}
```

In [18]:
```
x = eval(input("Enter Your data"))
print(type(x))
print(x)
```

```
Enter Your data{10,20,30,40,50,60}
<class 'set'>
{50, 20, 40, 10, 60, 30}
```

In [20]:
```
x = eval(input("Enter Your data"))
print(type(x))
print(x)
```

```
Enter Your data(10,20,30,40)
<class 'tuple'>
(10, 20, 30, 40)
```

In [24]:
```
#Aliasing and Cloning of List
#Aliasing means gives a shorter name
x=[10,20,30,40,50]
y=x
y[2]=200
print(id(x))
print(id(y))
print(x)
print(y)
#The major disadvantage of aliasing is if we are going to change any data through first reference
#then due to that change the second reference will also gets effected
```

```
2804790378432
2804790378432
[10, 20, 200, 40, 50]
[10, 20, 200, 40, 50]
```

In [ ]:
```
#Cloning  --> Copying the list object
1.First way is slicing
```

In [26]:
```
x=[10,20,30,40,50,60]
y=x[:]
y[1]=200
x[2]=5000
print(id(x))
print(id(y))
print(x)
print(y)
```

```
2804790340608
2804761440512
[10, 20, 5000, 40, 50, 60]
[10, 200, 30, 40, 50, 60]
```

2.second way is copy function

In [28]:
```
x=[10,20,30,40,50,60]
y=x.copy()
y[1]=200
x[2]=5000
print(id(x))
print(id(y))
print(x)
print(y)
```

```
2804790385488
2804790387584
[10, 20, 5000, 40, 50, 60]
[10, 200, 30, 40, 50, 60]
```

In [ ]:
```
#Pass Statement -->,pass is a keyword in python
in our programming i want to only declare any function , loop or condition. it is used to create
a empty block
```

In [30]:
```
def add():
    pass
```

In [35]:
```
def comp():
    pass
```

```
  Input In [35]
    def comp():
                ^
IndentationError: expected an indented block
```

In [34]:
```
if True:
    pass
```

In [32]:
```
for i in range(1,2):
    pass
```

```
  Input In [32]
    for i in range(1,2):
                        ^
IndentationError: expected an indented block
```

In [37]:
```
#Dictionary Comphersion --> for creation of dictionary into a concise manner
#Comphersion of dictionary is also possible
square = {x:x*x for x in range(1,6)}
square
```

Out[37]:
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [40]:
```
square = {x:2*x for x in range(1,6)}
type(square)
```

Out[40]:
```
dict
```

In [39]:
```
#Tuple Comphersion --> tuple comphersion is not possible because it will give you a generatir object
t=(x**2 for x in range(1,6))
print(t)
```

```
<generator object <genexpr> at 0x000002BD0ABDF270>
```

In [41]:
```
#Set Comphersion
t={x**2 for x in range(1,6)}
print(t)
```

```
{1, 4, 9, 16, 25}
```

In [ ]:
```
Q - What can be the code to loop over the keys of a dictionary?

My_dictionary = {0: 'a', 1: 'b', 2: 'c'}

A - for i in My_Dictionary:

print(keys)

B - for i in My_Dictionary:

print(My_Dictionary)

C - Not possible.

D - for i in My_Dictionary:

print(i)
```

In [ ]:
```
Q - What is the output of print str * 2 if str = 'Hello World!'?

A - Hello World!Hello World!

B - Hello World! * 2

C - Hello World!

D - None of the above.
```

In [43]:
```
#* --> repition operator  , + --> concatenation
str="Hello World"
str*2
```

Out[43]:
```
'Hello WorldHello World'
```

In [ ]:
```
Q - There are different basic operators in python and work according to the order of their precedence.

Arrange the order of precedence of the following operator –

i)- Division

ii)- Multiplication

iii)- Parentheses

iv)- Exponential

v)- Addition

vi)- Subtraction

A - i, ii, iii, iv, v, vi.

B - iv, iii, ii, i, vi, v.

C - iii, iv, i, ii, v, vi.

D - iv, iii, i, ii, v, vi.
```

In [ ]:
```
Q - What is the output for –

'you are doing well' [2:999]

A - 'you are doing well'

B - ' '

C - Index error.

D - 'u are doing well'
```

In [ ]:
```
insert()--> first argument will always be the index and second is the data
```

In [ ]:
```
Q - What is output of following code –

l = [1,2,6,5,7,8]
l.insert(6)
A - l=[9,1,2,6,5,7,8]

B - l=[1,2,6,5,9,7,8] (insert randomly at any position)

C - l=[1,2,6,5,7,8,9]

D - Type Error
```

In [ ]:
```
Q - Which of the following function checks in a string that all characters are alphanumeric?

A - shuffle(list)

B - capitalize()

C - isalnum()

D - isdigit()
```

In [ ]:
```
Q - What is the output for –

'Tutorials Point' [100:200]?

A - Index error.

B - ' '

C - 'Tutorials Point'

D - Syntax error
```

In [ ]:
```
Q - What is the output of the following code?

print(0.1+0.2==0.3)
A - True

B - False

C - No

D - Yes
```

In [ ]:
```
Q - Which of the following operator in python performs the division on operands where the result
is the quotient in which the digits after the decimal point are removed?

A - /

B - //

C - %

D - None
```

In [ ]:
```
 Which of the following statement is used when a statement is required syntactically
    but you do not want any command or code to execute?

A - break

B - continue

C - pass

D - None of the above.
```

In [ ]:
```
Q - What is the output of print tinylist * 2 if tinylist = [123, 'john']?

A - [123, 'john', 123, 'john']

B - [123, 'john'] * 2

C - Error

D - None of the above.
```

In [ ]:
```
Q - What is the output of print tinytuple * 2 if tinytuple = (123, 'john')?

A - (123, 'john', 123, 'john')

B - (123, 'john') * 2

C - Error

D - None of the above.
```

In [ ]:
```
Q - Which of the following operator in python performs exponential (power) calculation on operands?

A - **

B - //

C - is

D - not in
```

In [ ]:
```
Q - How can we swap two numbers a = 10, b = 20 in python without using third variable?

A - a = b

b = a

B - a,b = b,a

C - both a & b

D - b = a

a = b
```

In [ ]:
```
Which of the following functions convert string into any object?
a.eval()
b.input()
c.id()
d.enumerate()
```