In [1]:

```
import numpy as np
np.ones(3)
```

Out[1]:

```
array([1., 1., 1.])
```

In [2]:

```
2 + 3
```

Out[2]:

```
5
```

In [ ]:

```
a = 10
b= 20
print(a+b)
print(a-b)
print(a*b)
print(a/b)
print(a%b)
print(a**b)
print(a//b)
```

```
30
-10
200
0.5
10
100000000000000000000
0
```

In [2]:

```
print('hello','hai','how are you',sep='----->')
```

```
hello----->hai----->how are you
```

In [3]:

```
print(3,'.',sep='')
```

```
3.
```

In [4]:

```
print(1,2,end=' ')
print(3,'.',sep='')
```

```
1 2 3.
```

In [5]:

```
letter = 'p'
print(letter*10)
```

```
pppppppppp
```

In [6]:

```
#### Unpacking characters
language = 'Python'
a,b,c,d,e,f = language # unpacking sequence characters into variables
print(a) # P
print(b) # y
print(c) # t
print(d) # h
print(e) # o
print(f) # n
```

```
P
y
t
h
o
n
```

In [9]:

```
# Accessing characters in strings by index
language = 'Python'
first_letter = language[0]
print(first_letter) # P
```

```
P
```

In [10]:

```
second_letter = language[1]
print(second_letter) # y
```

```
y
```

In [11]:

```
last_index = len(language) - 1
last_letter = language[last_index]
print(last_letter) # n
```

```
n
```

In [12]:

```
# If we want to start from right end we can use negative indexing. -1 is the las
language = 'Python'
last_letter = language[-1]
print(last_letter) # n
second_last = language[-2]
print(second_last) # o
```

```
n
o
```

In [14]:

```
language = 'Python'
first_three = language[0:3]
last_three = language[3:6]
print(last_three) # hon
```

```
hon
```

In [15]:

```
language = 'Python'
pto = language[0:6:2] #
print(pto) # pto
```

```
Pto
```

In [16]:

```
l = []
l
```

Out[16]:

```
[]
```

In [17]:

```
len(l)
```

Out[17]:

```
0
```

In [18]:

```
l.append(10)
```

In [19]:

```
l
```

```
Out[19]:  [10]
```

```
In [20]:  len(l)
```

```
Out[20]:  1
```

```
In [21]:  import keyword
          keyword.kwlist
```

```
Out[21]:  ['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 [22]:  d = {}
          d
```

```
Out[22]:  {}
```

```
In [24]:  type(d)
```

```
Out[24]:  dict
```

```
In [25]:  d1 = {1: 'one', 2: 'two'}
          d1
```

```
Out[25]:  {1: 'one', 2: 'two'}
```

In [26]:

```
keys = {'ram','b', 'c','d'}
value = [10,20,30]
mydict = dict.fromkeys(keys,value)
mydict
```

Out[26]:

```
{'c': [10, 20, 30], 'd': [10, 20, 30], 'b': [10, 20, 30], 'ram': [10, 20, 30]}
```

In [27]:

```
mytuple = ('one','two','three')
```

In [28]:

```
mytuple[0:3]
```

Out[28]:

```
('one', 'two', 'three')
```

In [29]:

```
mytuple[0:3:2]
```

Out[29]:

```
('one', 'three')
```

In [30]:

```
mytuple[0:3:2]
```

Out[30]:

```
('one', 'three')
```

In [31]:

```
if 'three' in mytuple:
    print('Thre is present in mytuple')
else:
    print('elevenis  present in mytuple')
```

```
Thre is present in mytuple
```

In [32]:

```
mytuple.index('three')
```

Out[32]:

```
2
```

In [ ]:

```
```