```
In [ ]:
```

```
In [5]: i=30
        i

Out[5]: 30
```

```
In [6]: type(i)

Out[6]: int
```

```
In [7]: print(type(i))

       <class 'int'>
```

```
In [8]: a,b,c=2,3,4
        print(a,b,c)

       2 3 4
```

```
In [ ]:
```

```
In [ ]:
```

```
In [9]: f=110.23
        f

Out[9]: 110.23
```

```
In [10]: type(f)

Out[10]: float
```

```
In [11]: f1,f2,f3=2.3,3.4,5.1
         f1

Out[11]: 2.3
```

```
In [12]: print(f)
         print(f1)
         print(f2)
         print(f3)

        110.23
        2.3
        3.4
        5.1
```

```
In [13]: f1=1e0
         f1

Out[13]: 1.0
```

```
In [14]: f2=3e2
         f2
```

Out[14]:
```
300.0
```

In [15]:
```
f4=3e3
f4
```

Out[15]:
```
3000.0
```

In [16]:
```
f5=2.4e2
f5
```

Out[16]:
```
240.0
```

onlay e is allow in flaot data type

in python bool is always is T and F only captial letter allow In python error are 3 type compile time error-use while write code,missimg ,missing. run time erroe-no user side error 0division error- e0-1,e1-10.0,e2-100.0,e3-1000.0

In [17]:
```
b=true
b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[17], line 1
----> 1 b=true
      2 b

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

In [ ]:
```
b=True
b
```

In [ ]:
```
b1=False
b1
```

In [ ]:
```
print(b)
print(b1)
```

In [ ]:
```
True+  False
```

In [ ]:
```
True-False
```

In [ ]:
```
False-True
```

In [ ]:
```
True+True+True+False-True
```

In [ ]:
```
False*True
```

In [ ]:
```
True*True
```

In [18]:
```
False/True
```

```
Out[18]:  0.0
```

```
In [19]:  True/False        #0division error
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[19], line 1
----> 1 True/False

ZeroDivisionError: division by zero
```

complex type

```
In [20]:  c=10+20j
          c
```

```
Out[20]:  (10+20j)
```

```
In [21]:  import keyword      #key word
          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]:  len(keyword.kwlist)
```

Out[22]: 35

In [23]:
```
p,q,r=20,20,20
p,q,r
```

Out[23]:
```
(20, 20, 20)
```

In [ ]:

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

Out[24]:
```
complex
```

In [25]:
```
(1+20j)
```

Out[25]:
```
(1+20j)
```

In [26]:
```
c.real
```

Out[26]:
```
10.0
```

In [27]:
```
c.imag
```

Out[27]:
```
20.0
```

In [28]:
```
c1=10+20j
c2=30+40j
c1+c2
```

Out[28]:
```
(40+60j)
```

In [29]:
```
print(c1-c2)
print(c1+c2)
```

```
(-20-20j)
(40+60j)
```

python data type ---------Every value has a data type and variable can holds the value

there are 5 types of data type in python 1----numeric in numeric -----intiger =a value with out decimal point -----complex=i+bj -----float=a value with decimal point

2----dictionary 3----boolean= hold only two value True and False 4-----set 5-----sequence type in seqence------string= single qute ,double cout and threeple qute ------list ------tupel

In [30]:
```
a=10
b="hi python"
c=10.5
print(type(a))
print(type(b))
print(type(c))
```

```
<class 'int'>
<class 'str'>
<class 'float'>
```

In [31]:

```
a=5
print("The type of a",type(a))
b=40.5
print("The type of b",type(b))
c=1+3j
print("The type of c",type(c))
print("c is a complex number",isinstance(1+3j,complex))
```

```
The type of a <class 'int'>
The type of b <class 'float'>
The type of c <class 'complex'>
c is a complex number True
```

python 3

In [32]:

```
s='nit' #' use for single line
s
```

Out[32]:

```
'nit'
```

In [33]:

```
type(s)
```

Out[33]:

```
str
```

In [34]:

```
s1="hello python"
s1
```

Out[34]:

```
'hello python'
```

In [35]:

```
s2='''nit
         hello python'''   #''' use for multiline comment
s2
```

Out[35]:

```
'nit\n         hello python'
```

python index begains with 0 2type ----1=forward index(0,1,2............forward) 2=backward index(-1,-2,-3.........backward)

In [36]:

```
s1[0]
```

Out[36]:

```
'h'
```

In [37]:

```
s1[-4]
```

Out[37]:

```
't'
```

In [38]:

```
s1[4]
```

Out[38]:

```
'o'
```

```
In [39]: s1[5] #forward space
```

```
Out[39]: ' '
```

```
In [40]: s1[-7] #backword space
```

```
Out[40]: ' '
```

```
In [41]: print(s[0])
         print(s[1])
         print(s[2])
```

```
n
i
t
```

sring slicing :=print the string

```
In [42]: s1[:]
```

```
Out[42]: 'hello python'
```

```
In [43]: s1[2:7]
```

```
Out[43]: 'llo p'
```

```
In [44]: s3='data analyst'
         s3
```

```
Out[44]: 'data analyst'
```

```
In [45]: s3[0:10]
```

```
Out[45]: 'data analy'
```

```
In [46]: s3[0:11]
```

```
Out[46]: 'data analys'
```

```
In [47]: s3[0:12]
```

```
Out[47]: 'data analyst'
```

```
In [48]: s3[0:13]
```

```
Out[48]: 'data analyst'
```

```
In [49]: s3[9:12]
```

```
Out[49]: 'yst'
```

```
In [50]: s3[0:11:2]
```

```
Out[50]: 'dt nls'
```

```
In [51]: s3[0:11:3] #string slicing
```

```
Out[51]: 'dany'
```

```
In [52]: s3[2:-2]
```

```
Out[52]: 'ta analy'
```

```
In [53]: print(s)
         print(s1)
         print(s2)
         print(s3)
```

```
nit
hello python
nit
         hello python
data analyst
```

```
In [54]: for i s3:
          print(i)
```

```
  Cell In[54], line 1
    for i s3:
          ^
SyntaxError: invalid syntax
```

type conversion /python

type casting

```
In [55]: int(2.3) #float to int
```

```
Out[55]: 2
```

```
In [56]: int(True)#bool to int
```

```
Out[56]: 1
```

```
In [57]: int(1+2j) #complex to int not possible
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[57], line 1
----> 1 int(1+2j)

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

```
In [58]:  int('10') #strin to int
```

```
Out[58]: 10
```

```
In [59]: int('ten') #only digit convert word is not convert
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[59], line 1
----> 1 int('ten')

ValueError: invalid literal for int() with base 10: 'ten'
```

```
In [ ]:
```

```
In [60]: np.nan
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[60], line 1
----> 1 np.nan

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

```
In [61]: import numpy as np
         a=np.nan
```

```
In [62]: type(a)

Out[62]: float
```

25oct complex

```
In [63]: float(3)

Out[63]: 3.0
```

```
In [64]: float(True)

Out[64]: 1.0
```

```
In [65]: float(1+2J)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[65], line 1
----> 1 float(1+2J)

TypeError: float() argument must be a string or a real number, not 'complex'
```

```
In [66]: float(3,4) #in float we can not pass 2 argument
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[66], line 1
----> 1 float(3,4)

TypeError: float expected at most 1 argument, got 2
```

```
In [67]: float('10') #we convert all other data type to float except complex

Out[67]: 10.0
```

```
In [68]: float('ten')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[68], line 1
----> 1 float('ten')

ValueError: could not convert string to float: 'ten'
```

```
In [69]: complex(10)
```

```
Out[69]: (10+0j)
```

```
In [70]: complex(10,20)
```

```
Out[70]: (10+20j)
```

```
In [71]: complex(10,20,30)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[71], line 1
----> 1 complex(10,20,30)

TypeError: complex() takes at most 2 arguments (3 given)
```

```
In [72]: complex(2.3+0j)
```

```
Out[72]: (2.3+0j)
```

```
In [73]: complex(2.3,10)
```

```
Out[73]: (2.3+10j)
```

```
In [74]: complex(True)
```

```
Out[74]: (1+0j)
```

```
In [75]: complex(False)
```

```
Out[75]: 0j
```

```
In [76]: complex('10')
```

```
Out[76]: (10+0j)
```

```
In [77]: bool(1)
```

```
Out[77]: True
```

```
In [78]: bool(0)
```

```
Out[78]: False
```

```
In [79]: bool(2.3)
```

```
Out[79]: True
```

```
In [80]: bool()
```

```
Out[80]:  False
```

```
In [81]:  bool( )
```

```
Out[81]:  False
```

```
In [82]:  bool('nit')
```

```
Out[82]:  True
```

```
In [83]:  bool(10+2j)
```

```
Out[83]:  True
```

```
In [84]:  bool(0+0)
```

```
Out[84]:  False
```

```
In [85]:  bool(0+0J)
```

```
Out[85]:  False
```

```
In [86]:  print(str(2))
           print(str(2.3))
           print(str(True))
           print(str(1+2j))
```

```
2
2.3
True
(1+2j)
```

complele type casting

```
In [87]:  index='HELLOPYTHON'
           index
```

```
Out[87]:  'HELLOPYTHON'
```

```
In [88]:  index[:]         #string slicing
```

```
Out[88]:  'HELLOPYTHON'
```

```
In [89]:  index[::-1] #reverse string
```

```
Out[89]:  'NOHTYPOLLEH'
```

```
In [90]:  index[::-2]
```

```
Out[90]:  'NHYOLH'
```

```
In [91]:  index
```

```
Out[91]:  'HELLOPYTHON'
```

```
In [92]: index[::-4]
```

```
Out[92]: 'NYL'
```

```
In [93]: index
```

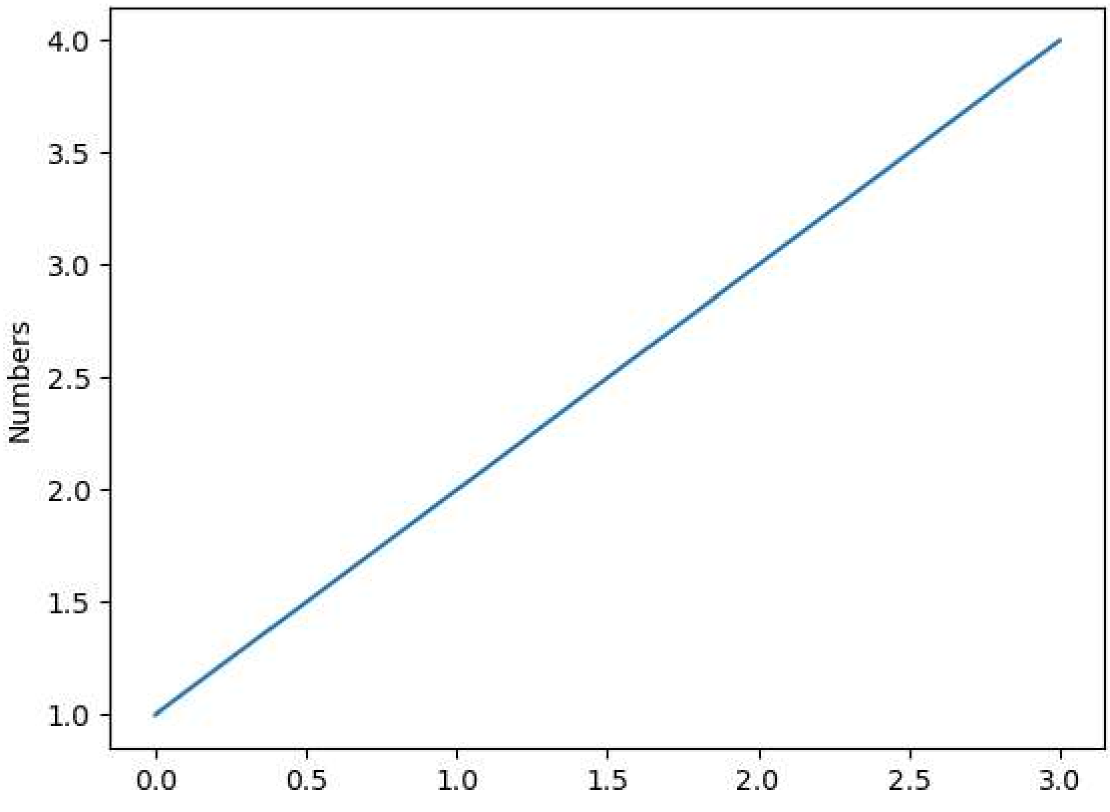
```
Out[93]: 'HELLOPYTHON'
```

index[:]----all element index[2:4]----print 2nd index to 4-1=3rd index[:4]---print the element till 3rd index index[4:]---- 4th index

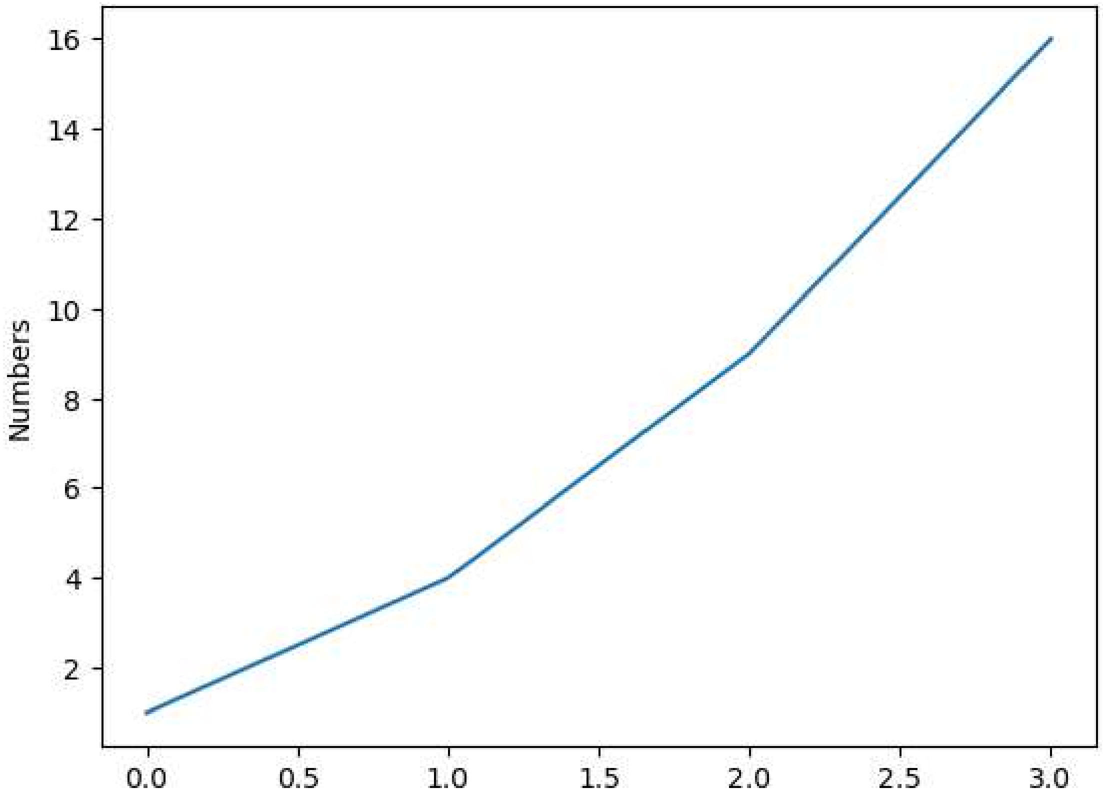
```
In [94]: index[1:10:3]
```

```
Out[94]: 'EOT'
```

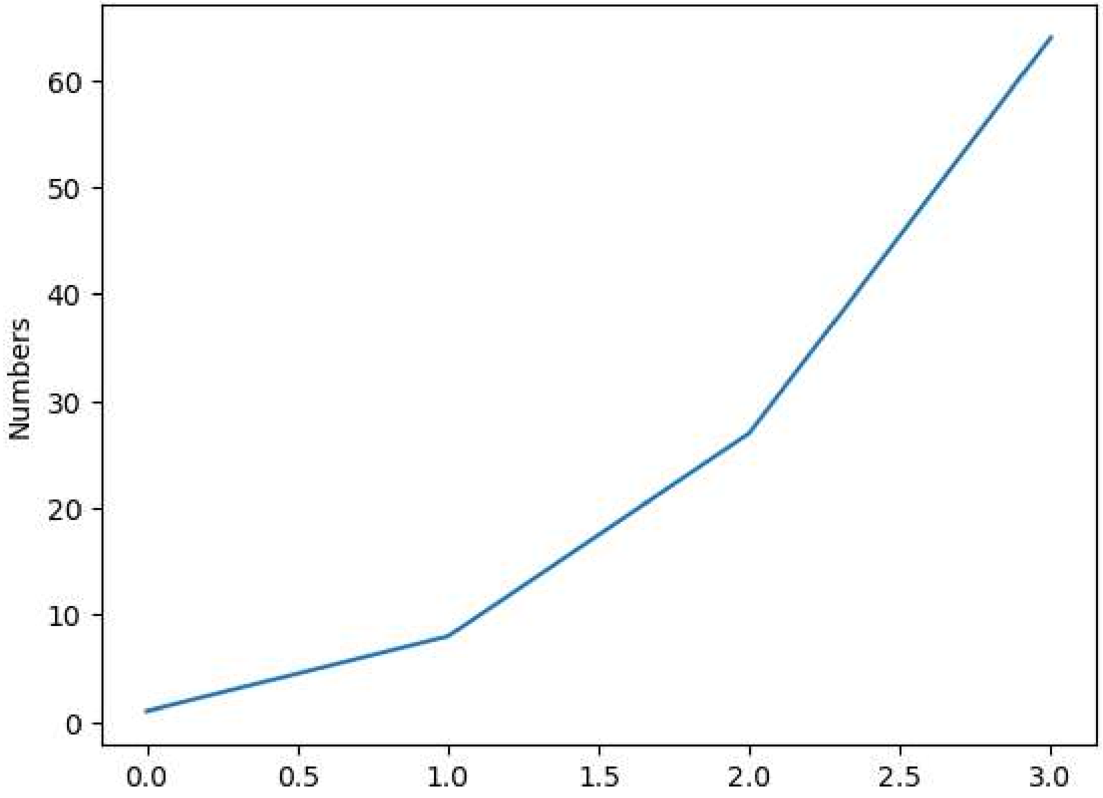
```
In [95]: import matplotlib.pyplot as plt
         plt.plot([1,2,3,4])
         plt.ylabel('Numbers')
         plt.show()
```



```
In [96]: import matplotlib.pyplot as plt
         plt.plot([1,4,9,16])
         plt.ylabel('Numbers')
         plt.show()
```



In [97]:
```
import matplotlib.pyplot as plt
plt.plot([1,8,27,64])
plt.ylabel('Numbers')
plt.show()
```



DATA TYPE VS DATA STR VS MATRIX

data type--store 1 variable data structure---store more than one variable matrix-----store data structure types of data structure---- *list

- tuple
- dict
- set

```
                                        date  ---    -- 26th
   *list*
        1=[]
```

In [98]:
```
l=[]
l
```

Out[98]:
```
[]
```

In [99]:
```
type(l)
```

Out[99]:
```
list
```

In [100…
```
l.append(10)
l
```

Out[100…
```
[10]
```

In [101…
```
l.append(20)
l
```

Out[101…
```
[10, 20]
```

In [102…
```
len(l)  #lenth function
```

Out[102…
```
2
```

In [103…
```
l[:]
```

Out[103…
```
[10, 20]
```

In [104…
```
l2=[]
l2
```

Out[104…
```
[]
```

In [105…
```
l2.append(1)
l2.append(2.3)
l2.append(True)
l2.append(1+2j)
l2.append('nit')
l2
```

Out[105…
```
[1, 2.3, True, (1+2j), 'nit']
```

In [106…
```
l3=[]
l3
```

```
Out[106…  []

In [107…  l3.append(1)
          l3.append(2.3)
          l3.append(True)
          l3.append(1+2j)
          l3.append('nit')
          l3

Out[107…  [1, 2.3, True, (1+2j), 'nit']

In [108…  l3.clear()

In [109…  len(l3)
          len

Out[109…  <function len(obj, /)>

In [110…  l3

Out[110…  []

In [111…  l3=[]
          l3.append(20)
          l3

Out[111…  [20]

In [112…  l2

Out[112…  [1, 2.3, True, (1+2j), 'nit']

In [113…  l3.extend(l2)

          l3

Out[113…  [20, 1, 2.3, True, (1+2j), 'nit']

In [114…  l3.index(1)

Out[114…  1

In [115…  l3.index(1+2j)

Out[115…  4

In [116…  l3.insert(5,'tech')
          l3

Out[116…  [20, 1, 2.3, True, (1+2j), 'tech', 'nit']

In [117…  l3.insert(3,False)
          l3

Out[117…  [20, 1, 2.3, False, True, (1+2j), 'tech', 'nit']
```

In [118…

```
l3.pop()
            #
```

Out[118…

```
'nit'
```

In [119…

```
l4=[10,100,3,45,76,24]
l4
```

Out[119…

```
[10, 100, 3, 45, 76, 24]
```

In [120…

```
l4.sort()
l4
```

Out[120…

```
[3, 10, 24, 45, 76, 100]
```

In [121…

```
l4.sort(reverse=True)
l4
```

Out[121…

```
[100, 76, 45, 24, 10, 3]
```

In [122…

```
l5=['z','m','c','w']    #always one one data type
l5
```

Out[122…

```
['z', 'm', 'c', 'w']
```

In [123…

```
l5.sort()
l5
```

Out[123…

```
['c', 'm', 'w', 'z']
```

In [124…

```
l3
```

Out[124…

```
[20, 1, 2.3, False, True, (1+2j), 'tech']
```

29th oct

muitable cocept or hasseble(changeable)

```
after: by dufult four space
ex for i in l3
       print(i)
```

In [125…

```
l6=['a','b','c']
l7=['d','e','f']
family_bank=l6+l7     #concardination
family_bank
```

Out[125…

```
['a', 'b', 'c', 'd', 'e', 'f']
```

In [126…

```
l3=[2,'a',2.3]
l3
```

Out[126…

```
[2, 'a', 2.3]
```

In [127…

```
for i in enumerate(l3):
    print(i)
```

```
(0, 2)
(1, 'a')
(2, 2.3)
```

In [128…

```
for i in (l3):
    print(i)
i
```

```
2
a
2.3
```

Out[128…

```
2.3
```

set

```
        opperation ----union,intersection,symtric difference
```

In [129…

```
a5={1,2,3,4,5,6,7,8,9}
b5={3,4,5,6,7,8}
c5={10,20,30,40}
a5.issuperset(b5)
```

Out[129…

```
True
```

In [130…

```
a5={1,2,3,4,5,6,7,8,9}
b5={3,4,5,6,7,8}
c5={10,20,30,40}
a5.issubset(b5)
```

Out[130…

```
False
```

dict

In [131…

```
d={}
type(d)
```

Out[131…

```
dict
```

In [132…

```
d={1:'one',2:'two',3:'three',4:'four',5:'five'}
d
```

Out[132…

```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

In [133…

```
d1={'six':6,'seven':7,'eight':8,'nine':9,'ten':10}
d1
```

Out[133…

```
{'six': 6, 'seven': 7, 'eight': 8, 'nine': 9, 'ten': 10}
```

```
In [134…  print(len(d))
          print(len(d1))

         5
         5

In [135…  d

Out[135…  {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}

In [136…  d[1]

Out[136…  'one'

In [137…  d1

Out[137…  {'six': 6, 'seven': 7, 'eight': 8, 'nine': 9, 'ten': 10}

In [138…  d

Out[138…  {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}

In [139…  d.keys()

Out[139…  dict_keys([1, 2, 3, 4, 5])

In [140…  d.values()

Out[140…  dict_values(['one', 'two', 'three', 'four', 'five'])

In [141…  d1.keys()

Out[141…  dict_keys(['six', 'seven', 'eight', 'nine', 'ten'])

In [142…  d1.values()

Out[142…  dict_values([6, 7, 8, 9, 10])

In [143…  d2={1:2,2.3:4.8,'nit':'nit',True:False,1+2j:4+5j}
          d2

Out[143…  {1: False, 2.3: 4.8, 'nit': 'nit', (1+2j): (4+5j)}

In [144…  d.items()

Out[144…  dict_items([(1, 'one'), (2, 'two'), (3, 'three'), (4, 'four'), (5, 'five')])

In [145…  len(d.items())

Out[145…  5

In [146…  id(d)

Out[146…  2139714976128

In [147…  d.pop(1)
```

Out[147… 'one'

In [148…
```
d
```

Out[148… {2: 'two', 3: 'three', 4: 'four', 5: 'five'}

In [149…
```
d[1]='one'
d
```

Out[149… {2: 'two', 3: 'three', 4: 'four', 5: 'five', 1: 'one'}

In [150…
```
d.popitem()
```

Out[150… (1, 'one')

In [151…
```
d1
```

Out[151… {'six': 6, 'seven': 7, 'eight': 8, 'nine': 9, 'ten': 10}

In [152…
```
d2
```

Out[152… {1: False, 2.3: 4.8, 'nit': 'nit', (1+2j): (4+5j)}

In [153…
```
for i in d:
    print(i)
```

```
2
3
4
5
```

In [154…
```
for i in d:
    print(i,':',d[i])
```

```
2 : two
3 : three
4 : four
5 : five
```

In [155…
```
for i in d:
    print(d)
```

```
{2: 'two', 3: 'three', 4: 'four', 5: 'five'}
{2: 'two', 3: 'three', 4: 'four', 5: 'five'}
{2: 'two', 3: 'three', 4: 'four', 5: 'five'}
{2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

In [156…
```
range(5)
```

Out[156… range(0, 5)

In [157…
```
r1=range(0,1,2)
r1
```

Out[157… range(0, 1, 2)

In [158…
```
for i in r1:
    print(i)
```

```
          0

In [159…  x=3
          x

Out[159…  3

In [160…  y=3
          y

Out[160…  3

In [161…  _+x  # _means store previous out put

Out[161…  6

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:
```