In [1]:
```
s='abcdef'
print(s[0])
```
```
a
```

In [2]:
```
L=['data1','data2']
print(L[0])
```
```
data1
```

In [3]:
```
t=('D1','D2')
print(t[0])
```
```
D1
```

In [5]:
```
# dict - Collection of unordered data = key:value //pair - mutable (we can add/modi
#
# dictname = {}

product_info = {'pid':1234,'pname':'pA','pcost':1235.63,'pstatus':True}
print(type(product_info))
print(product_info)
print(len(product_info))
```
```
<class 'dict'>
{'pid': 1234, 'pname': 'pA', 'pcost': 1235.63, 'pstatus': True}
4
```

In [10]:
```
L = ['D1','D2',10,2.5,True]

# How to fetch nth item from list ?  => Listname[index] ->Value / IndexError

print(L[0])
# print(L[6]) IndexError: list index out of range

# How to modify an existing data from list ? => Listname[oldIndex] = updatedValue
print(L[1]) # 'D2'
L[1] = 'Data-2' # modification
print(L[1])
print(L)
```
```
D1
D2
Data-2
['D1', 'Data-2', 10, 2.5, True]
```

In [11]:
```
product_info = {'pid':1234,'pname':'pA','pcost':1235.63,'pstatus':True}
# How to fetch nth item from dict ?  => dictname[oldkey] ->Value / KeyError

print(product_info['pid'])
print(product_info['pname'])
# print(product_info['pNAME']) KeyError: 'pNAME'

# To modify an existing value from dict =>  dictname['oldKey'] = updatedValue
product_info['pname'] = 'pB'  # modification
print(product_info)
```

```
1234
pA
{'pid': 1234, 'pname': 'pB', 'pcost': 1235.63, 'pstatus': True}
```

In [12]:
```
L = ['D1','D2',10,2.5,True]

# How to add new data to an existing list ?
#  Listname.append(Value)  (or)  Listname.insert(<index>,value)

L.append('D3')
print(L)
```

```
['D1', 'D2', 10, 2.5, True, 'D3']
```

In [13]:
```
product_info = {'pid':1234,'pname':'pA','pcost':1235.63,'pstatus':True}

# How to add new data(key:value) to an existing dict?
# dictname['newKey'] = value   Vs   dictname['oldKey'] = updatedValue
#            ^^^^^                               ^^^^^^

print(product_info)
print('')
product_info['pvendor'] = 'oracle' # adding new data to an existing dict
product_info['pcost'] = 5909.32    # modification
print(product_info)
```

```
{'pid': 1234, 'pname': 'pA', 'pcost': 1235.63, 'pstatus': True}

{'pid': 1234, 'pname': 'pA', 'pcost': 5909.32, 'pstatus': True, 'pvendor': 'oracle'}
```

In [14]:
```
L=['D1','D2','D3',10,20,30,40,50,'Dx','Dy']
r = L.pop() # remove last index value - default
print(f'removed item:{r}')
print(L)
```

```
removed item:Dy
['D1', 'D2', 'D3', 10, 20, 30, 40, 50, 'Dx']
```

In [15]:
```
r = L.pop(5)
print(f'removed item:{r}')
print(L)
```

```
removed item:30
['D1', 'D2', 'D3', 10, 20, 40, 50, 'Dx']
```

In [16]:
```
d={'K1':'V1','K2':'V2','K3':123,'K4':45.23}
print(d)
# to delete nth item from dict =>  dictname.pop(<oldKey>) ->removed_value
r = d.pop('K2')
print(f'removed value:{r}')
print(d)
```

```
{'K1': 'V1', 'K2': 'V2', 'K3': 123, 'K4': 45.23}
removed value:V2
{'K1': 'V1', 'K3': 123, 'K4': 45.23}
```

In [ ]:
```
Task:
1. create an employee dictionary - empty dict => emp={}
```

```
2. add emp details(empID,empName,empDept,empDOB,empPay) to an existing dict
3. display emp details
4. modify emp - working department
5. delete empPay
6. Add empContact number
7. display updated emp records
```

In [17]:
```
emp = {} # empty dict
# adding emp details to an existing dict
emp['eid'] = 123
emp['ename'] = 'Mr.Leo'
emp['edept'] = 'sales'
emp['edob'] = '1st Jan'
emp['epay'] = 12562.32
print(emp) # display emp details
```

```
{'eid': 123, 'ename': 'Mr.Leo', 'edept': 'sales', 'edob': '1st Jan', 'epay': 12562.3
2}
```

In [18]:
```
emp['edept'] = 'production' # modification
print(emp)
```

```
{'eid': 123, 'ename': 'Mr.Leo', 'edept': 'production', 'edob': '1st Jan', 'epay': 12
562.32}
```

In [19]:
```
r = emp.pop('epay') # delete emp cost
print(emp)
```

```
{'eid': 123, 'ename': 'Mr.Leo', 'edept': 'production', 'edob': '1st Jan'}
```

In [20]:
```
emp['contact'] = '080-6651423' # adding new data
emp
```

Out[20]:
```
{'eid': 123,
 'ename': 'Mr.Leo',
 'edept': 'production',
 'edob': '1st Jan',
 'contact': '080-6651423'}
```

In [ ]:
```
d={}  # OK
d['K1']='V1'  # OK
print(d['K1']) # OK
print(d['K2']) # KeyError

d={1:True,():False,[]:'OK'} #
    ----   ------  ====== Error
d.pop() # Error
```

In [24]:
```
s='abababababab'
print(len(s),s)

L=['Data1','Data1','Data1','Data1']
print(len(L),L)

T=('Data1','Data1','Data1','Data1')
print(len(T),T)
```

```
d={'K1':'Data1','K2':'Data1','K3':'Data1'}
print(len(d),d)

print('') # Empty line

s={'data1','data1','data1','data1',10,20,10,20,10,20}
print(type(s))
print(len(s))
print(s)
```

```
12 abababababab
4 ['Data1', 'Data1', 'Data1', 'Data1']
4 ('Data1', 'Data1', 'Data1', 'Data1')
3 {'K1': 'Data1', 'K2': 'Data1', 'K3': 'Data1'}

<class 'set'>
3
{10, 20, 'data1'}
```

In [25]:
```
L=['Data1','Data2']
L.append('Data1')
L.append('Data2')
L.append('Data1')
L.append('Data2')
L.append('Data1')
L.append('Data2')
L
```

Out[25]:
```
['Data1', 'Data2', 'Data1', 'Data2', 'Data1', 'Data2', 'Data1', 'Data2']
```

In [26]:
```
set(L) #typecast to set
```

Out[26]:
```
{'Data1', 'Data2'}
```

In [27]:
```
sL = set(L) #typecast to set
list(sL) # typecast to list
```

Out[27]:
```
['Data2', 'Data1']
```

In [ ]:
```
python operators
-----------------
1. Arithmetic operators =>  + - * / // % **  (inputTypes: int,float -> int,float)
```

In [28]:
```
2 ** 3
```

Out[28]:
```
8
```

In [30]:
```
10 / 5
```

Out[30]:
```
2.0
```

In [31]:
```
10 // 5
```

Out[31]:
```
2
```

In [32]:
```
10 % 3
```
Out[32]:
```
1
```

In [ ]:
```
python operators
-----------------
1. Arithmetic operators =>  + - * / // % **  (inputTypes: int,float -> int,float)
2. string operators =>  + * (inputTypes: str,int ->str)
```

In [33]:
```
print(10+20) # 30
print('A'+'B')
```
```
30
AB
```

In [34]:
```
print('40'+str(50)) # 4050
```
```
4050
```

In [35]:
```
print('A'+10)
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[35], line 1
----> 1 print('A'+10)

TypeError: can only concatenate str (not "int") to str
```

In [36]:
```
print('A'+str(10))
```
```
A10
```

In [38]:
```
print('Hello' * 5)
```
```
HelloHelloHelloHelloHello
```

In [39]:
```
print('AB'*3)
```
```
ABABAB
```

In [40]:
```
s='''This is sample test message
about vector db corpus
data\n'''

print(s*5)
```

```
This is sample test message
about vector db corpus
data
This is sample test message
about vector db corpus
data
This is sample test message
about vector db corpus
data
This is sample test message
about vector db corpus
data
This is sample test message
about vector db corpus
data
```

In [41]:
```
print('-'*50)
```

```
--------------------------------------------------
```

In [ ]:
```
python operators
-----------------
1. Arithmetic operators =>  + - * / // % **  (inputTypes: int,float -> int,float)

2. string operators =>  + * (inputTypes: str,int ->str)

3. relational operators =>  == != < <= > >= (inputTypes: int,float,str ->bool)
|
4. logical operators => and or not  (inputTypes: int,float,str ->bool)
   ------------------
    Single Conditional statement, test more than one condition

5. membership operators =>  in  not in (inputTypes: str,list,tuple,dict,set  -> boo
```

In [42]:
```
159.31 >100
```

Out[42]:
```
True
```

In [43]:
```
159.31 < 0.23
```

Out[43]:
```
False
```

In [44]:
```
name = 'admin'
name == 'root'
```

Out[44]:
```
False
```

In [45]:
```
name != 'root'
```

Out[45]:
```
True
```

In [46]:
```
name == 'Admin'
```

```
Out[46]: False
```

```
In [47]: # test app port number range is 501-599

         port=450
         port >500 # test1
```

```
Out[47]: False
```

```
In [48]: port <600 # test2
```

```
Out[48]: True
```

```
In [49]: port >500 and port <600
```

```
Out[49]: False
```

```
In [50]: port=650
         port >500 and port <600
```

```
Out[50]: False
```

```
In [51]: port=560
         port >500 and port<600
```

```
Out[51]: True
```

```
In [52]: # test app name is OLA (or) Uber any one app is matched - OK
         app = 'OLA'

         app == 'OLA' or app == 'Uber'
```

```
Out[52]: True
```

```
In [ ]: 5. membership operators =>  in  not in (inputTypes: str,list,tuple,dict,set  -> boo

        'searchpattern_string' in  inputCollection
```

```
In [53]: s='101,raj,sales,bglore'

         'sales' in s
```

```
Out[53]: True
```

```
In [54]: 'prod' in s
```

```
Out[54]: False
```

```
In [55]: files =['p1.log','p2.log','p3.csv','emp.csv','data.html','index.html']

         'p1.pdf' in files
```

```
Out[55]: False
```

```
In [56]: 'emp.csv' in files
```

```
Out[56]: True
```

```
In [57]: # test input key is existing or not

         d={'K1':'Value1','K2':'Value2'}

         'K1' in d
```

```
Out[57]: True
```

```
In [58]: 'Kx' in d
```

```
Out[58]: False
```

```
In [59]: 'Value1' in d
```

```
Out[59]: False
```

```
In [60]: s={'D1','D2'}
         'D1' in s
```

```
Out[60]: True
```

```
In [61]: 'D1' not in s
```

```
Out[61]: False
```

```
In [ ]: 1. Read a port number from Keyboard
        2. typecast to int
        3. test - input port number is above 500 and below 600
                    -> initialize app name is testApp (  app="TestApp")
                    otherwise app name is demoApp     (  app="demoApp")
          |
        4. display app name and port number
```

```
In [62]: port = input('Enter a port number:')
         port = int(port)
         if(port >500 and port <600):
             app = 'TestApp'
         else:
             app = 'demotApp'

         print(f'App name is:{app} running port number is:{port}')
```

```
App name is:demotApp running port number is:450
```

```
In [63]: port = input('Enter a port number:')
         port = int(port)
         if(port >500 and port <600):
             app = 'TestApp'
```

```
else:
    app = 'demotApp'

print(f'App name is:{app} running port number is:{port}')
```

```
App name is:demotApp running port number is:670
```

In [65]:
```
port = input('Enter a port number:')
port = int(port)
if(port >500 and port <600):
    app = 'TestApp'
else:
    app = 'demoApp'

print(f'App name is:{app} running port number is:{port}')
```

```
App name is:TestApp running port number is:567
```

In [66]:
```
for var in 'abcd':
    print(var)
```

```
a
b
c
d
```

In [67]:
```
for var in ['D1','D2',10,20,30,40]:
    print(f'var value is:{var}')
```

```
var value is:D1
var value is:D2
var value is:10
var value is:20
var value is:30
var value is:40
```

In [68]:
```
for var in ('D1','D2',10,20,30,40):
    print(f'var value is:{var}')
```

```
var value is:D1
var value is:D2
var value is:10
var value is:20
var value is:30
var value is:40
```

In [69]:
```
for var in {'K1':'V1','K2':'V2','K3':10,'K4':3.1}: # will get list of keys only
    print(var)
```

```
K1
K2
K3
K4
```

In [70]:
```
d={'K1':'V1','K2':'V2','K3':10,'K4':3.1}
d['K1']
```

Out[70]:
```
'V1'
```

In [71]:
```
for var in d:
    print(d[var])
```

```
V1
V2
10
3.1
```

In [72]:
```
# To get Key - value
for var in d:
    print(f'{var} - {d[var]}')
```

```
K1 - V1
K2 - V2
K3 - 10
K4 - 3.1
```

In [73]:
```
s='abcd'
for var in s:
    print(var)
```

```
a
b
c
d
```

In [ ]:
```
s[0]
s[1]
s[2]
s[3]
```

In [74]:
```
i=0
while(i < len(s)):
    print(s[i])
    i=i+1
```

```
a
b
c
d
```

In [77]:
```
# 1. Read a username from keyboard
# 2. test input user name is root (or) not
# 3.                         |         |

i=0
while(i < 3):
    name = input('Enter a username:')
    if(name == 'root'):
        print('Success')
    else:
        print('Sorry your not root user')
    i=i+1
```

```
Success
Success
Success
```

In [77]:
```
# 1. Read a username from keyboard
# 2. test input user name is root (or) not
# 3.                          |         |

i=0
while(i < 3):
    name = input('Enter a username:')
    if(name == 'root'):
        print('Success')
    else:
        print('Sorry your not root user')
    i=i+1
```

```
Success
Success
Success
```

In [79]:
```
# 1. Read a username from keyboard
# 2. test input user name is root (or) not
# 3.                          |         |

i=0
while(i < 3):
    name = input('Enter a username:')
    if(name == 'root'):
        print('Success')
        break # exit from loop
    else:
        print('Sorry your not root user')
    i=i+1
```

```
Sorry your not root user
Success
```

In [80]:
```
for var in ['app1','app2','app3','app4','app5']:
    if(var == 'app3'):
        break
    else:
        print(var)
```

```
app1
app2
```

In [81]:
```
for var in ['app1','app2','app3','app4','app5']:
    if(var == 'app3'):
        continue
    else:
        print(var)
```

```
app1
app2
app4
app5
```

In [83]:
```
pin = 1234
count = 0
```

```
while(count < 3):
    count = count + 1
    p = input('Enter a pinNumber:')
    if(int(p) == pin):
        print(f'Success pin is matched - {count}')
        break

if(int(p) != pin):
    print(f'Sorry your pin is blocked')
```

```
Success pin is matched - 1
```

In [84]:
```
pin = 1234
count = 0

while(count < 3):
    count = count + 1
    p = input('Enter a pinNumber:')
    if(int(p) == pin):
        print(f'Success pin is matched - {count}')
        break

if(int(p) != pin):
    print(f'Sorry your pin is blocked')
```

```
Success pin is matched - 2
```

In [ ]:

In [85]:
```
pin = 1234
count = 0

while(count < 3):
    count = count + 1
    p = input('Enter a pinNumber:')
    if(int(p) == pin):
        print(f'Success pin is matched - {count}')
        break

if(int(p) != pin):
    print(f'Sorry your pin is blocked')
```

```
Success pin is matched - 3
```

In [86]:
```
pin = 1234
count = 0

while(count < 3):
    count = count + 1
    p = input('Enter a pinNumber:')
    if(int(p) == pin):
        print(f'Success pin is matched - {count}')
        break

if(int(p) != pin):
    print(f'Sorry your pin is blocked')
```

```
Sorry your pin is blocked
```

In [87]:
```
'''create an empty list
   use len() - display number of items
   use while loop  5 times
        To read a hostname from <STDIN>
        To add a input hostname to existing list
    using for loop, display list of elements
    display number of items'''

hosts = []
print(f'No.of items:{len(hosts)}')

c=0
while(c < 5):
    h = input('Enter a hostname:')
    hosts.append(h)
    c=c+1

print('-'*15)
print('List of input hosts:')
print('-'*15)
for var in hosts:
    print(var)

print('')
print(f'No.of items:{len(hosts)}')
```

```
No.of items:0
---------------
List of input hosts:
---------------
host01
host02
host03
host04
host05

No.of items:5
```

In [ ]: