In [1]:
```
emp={'eid':1234,'ename':"Theeba",'ecost':3313.221}
```

In [2]:
```
print(emp['eid'])
```

```
1234
```

In [3]:
```
print(emp['ename'])
```

```
Theeba
```

In [4]:
```
print(emp['ecost'])
```

```
3313.221
```

In [5]:
```
print(emp['EID'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-5-01556a872a38> in <module>
----> 1 print(emp['EID'])

KeyError: 'EID'
```

In [7]:
```
#How to modify a data from dict?
# dictname['oldkey']=updateValue

emp['ename']="Raghav"
print(emp['ename'])
print(emp)
```

```
Raghav
{'eid': 1234, 'ename': 'Raghav', 'ecost': 3313.221}
```

In [8]:
```
# How to add new data to dict?
#dictname['newkey']=Value

emp['ecity']='Bengaluru'
print(emp)
```

```
{'eid': 1234, 'ename': 'Raghav', 'ecost': 3313.221, 'ecity': 'Bengaluru'}
```

In [9]:
```
#How to delete nth data?

# del(dictname['key'])

del(emp['ename'])
print(emp)
```

```
{'eid': 1234, 'ecost': 3313.221, 'ecity': 'Bengaluru'}
```

In [10]:
```
'''
Task
====
1. create an empty dict
2. read hostname from user
3. read Ip address from user
4. use hostname and ipaddress as {k:v} pair and add it to dict

5. display the dict
```

```
'''

d={}
hostname=input("Enter the Hostname")
ip=input("Enter the ip address")

d[hostname]=ip

print(d)
```

```
Enter the Hostnamehost1
Enter the ip addressemp.eg.com
{'host1': 'emp.eg.com'}
```

In [11]:
```
len(d)
```

Out[11]:
```
1
```

In [12]:
```
L=['d1','d2','d3']

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

```
d1
d2
d3
```

In [13]:
```
for i in emp:   # iteration on keys not on value
    print(i)
```

```
eid
ecost
ecity
```

In [14]:
```
for i in emp:
    print(emp[i])
```

```
1234
3313.221
Bengaluru
```

In [15]:
```
for i in emp:
    print("{}\t{}".format(i,emp[i]))
```

```
eid     1234
ecost   3313.221
ecity   Bengaluru
```

In [16]:
```
"eid" in emp
```

Out[16]:
```
True
```

In [18]:
```
if "eid" in emp:
    print(emp["eid"])
else:
    print("Sorry key: k1 not exist")
```

```
1234
```

In [19]:
```
if "ename" not in emp:
    emp["ename"]="Hari"
```

```
else:
    print("Sorry key exist")
```

In [20]:
```
print(emp)
```

```
{'eid': 1234, 'ecost': 3313.221, 'ecity': 'Bengaluru', 'ename': 'Hari'}
```

In [21]:
```
help(dict)
```

```
Help on class dict in module builtins:

class dict(object)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a mapping object's
 |      (key, value) pairs
 |  dict(iterable) -> new dictionary initialized as if via:
 |      d = {}
 |      for k, v in iterable:
 |          d[k] = v
 |  dict(**kwargs) -> new dictionary initialized with the name=value pairs
 |      in the keyword argument list.  For example:  dict(one=1, two=2)
 |
 |  Built-in subclasses:
 |      StgDict
 |
 |  Methods defined here:
 |
 |  __contains__(self, key, /)
 |      True if the dictionary has the specified key, else False.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
 |
 |  __getattribute__(self, name, /)
 |      Return getattr(self, name).
 |
 |  __getitem__(...)
 |      x.__getitem__(y) <==> x[y]
 |
 |  __gt__(self, value, /)
 |      Return self>value.
 |
 |  __init__(self, /, *args, **kwargs)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  __iter__(self, /)
 |      Implement iter(self).
 |
 |  __le__(self, value, /)
 |      Return self<=value.
 |
 |  __len__(self, /)
 |      Return len(self).
 |
 |  __lt__(self, value, /)
 |      Return self<value.
 |
 |  __ne__(self, value, /)
```

```
|      Return self!=value.
|
|  __repr__(self, /)
|      Return repr(self).
|
|  __reversed__(self, /)
|      Return a reverse iterator over the dict keys.
|
|  __setitem__(self, key, value, /)
|      Set self[key] to value.
|
|  __sizeof__(...)
|      D.__sizeof__() -> size of D in memory, in bytes
|
|  clear(...)
|      D.clear() -> None.  Remove all items from D.
|
|  copy(...)
|      D.copy() -> a shallow copy of D
|
|  get(self, key, default=None, /)
|      Return the value for key if key is in the dictionary, else default.
|
|  items(...)
|      D.items() -> a set-like object providing a view on D's items
|
|  keys(...)
|      D.keys() -> a set-like object providing a view on D's keys
|
|  pop(...)
|      D.pop(k[,d]) -> v, remove specified key and return the corresponding value.
|      If key is not found, d is returned if given, otherwise KeyError is raised
|
|  popitem(self, /)
|      Remove and return a (key, value) pair as a 2-tuple.
|
|      Pairs are returned in LIFO (last-in, first-out) order.
|      Raises KeyError if the dict is empty.
|
|  setdefault(self, key, default=None, /)
|      Insert key with a value of default if key is not in the dictionary.
|
|      Return the value for key if key is in the dictionary, else default.
|
|  update(...)
|      D.update([E, ]**F) -> None.  Update D from dict/iterable E and F.
|      If E is present and has a .keys() method, then does:  for k in E: D[k] = E[k]
|      If E is present and lacks a .keys() method, then does:  for k, v in E: D[k] = v
|      In either case, this is followed by: for k in F:  D[k] = F[k]
|
|  values(...)
|      D.values() -> an object providing a view on D's values
|
|  ----------------------------------------------------------------------
|  Class methods defined here:
|
|  fromkeys(iterable, value=None, /) from builtins.type
|      Create a new dictionary with keys from iterable and values set to value.
|
|  ----------------------------------------------------------------------
|  Static methods defined here:
|
|  __new__(*args, **kwargs) from builtins.type
|      Create and return a new object.  See help(type) for accurate signature.
|
```

```
|  ----------------------------------------------------------------------
|  Data and other attributes defined here:
|
|  __hash__ = None
```

In [ ]:
```
for i in emp:                          (or)      for i in emp.keys()       # iterates ov
    print(emp[i])
```

In [24]:
```
for i in emp.items():
    print(i)
```

```
('eid', 1234)
('ecost', 3313.221)
('ecity', 'Bengaluru')
('ename', 'Hari')
```

In [ ]:
```
#How to add new data to dict (Method)

dictname["newkey"]=Value      (or)    dictname.setdefault("newKey",Value)
```

In [26]:
```
d={}
d.setdefault("k1","v1")
print(d)
d['k2']='v2'
print(d)
```

```
{'k1': 'v1'}
{'k1': 'v1', 'k2': 'v2'}
```

In [ ]:
```
#how to fetch single data from dict
dictname['oldkey']--> Value/keyError     (or)   dictname.get("oldkey")----> Value/None
```

In [27]:
```
print(d['k3'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-27-6182ff4954f6> in <module>
----> 1 print(d['k3'])

KeyError: 'k3'
```

In [29]:
```
print(d.get("k3"))
```

```
None
```

In [30]:
```
print(d['k2'])
```

```
v2
```

In [31]:
```
print(d.get("k2"))
```

```
v2
```

In [ ]:
```
#How to delete nth data from dict?

#del(d["key"])                  or        dictname.pop("existingkey")-----> removed val
```

In [32]:
```
d
```

Out[32]:
```
{'k1': 'v1', 'k2': 'v2'}
```

In [33]:
```
d.pop("k2")
```

Out[33]:
```
'v2'
```

In [34]:
```
d
```

Out[34]:
```
{'k1': 'v1'}
```

In [35]:
```
d.pop("k2")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-35-5b3e047fc0b3> in <module>
----> 1 d.pop("k2")

KeyError: 'k2'
```

In [36]:
```
for i in d:
    print(i)
```

```
k1
```

In [37]:
```
d['k2']="v2"
d.setdefault("k3","v3")
```

Out[37]:
```
'v3'
```

In [42]:
```
for i in d.keys():
    print(i)
```

```
k1
k2
k3
```

In [39]:
```
for i in d.items():
    print(i)
```

```
('k1', 'v1')
('k2', 'v2')
('k3', 'v3')
```

In [41]:
```
for i in d.values():
    print(i)
```

```
v1
v2
v3
```

In [ ]:
```
List                                                vs                        dict
------                                                                       ------

collection of oredered  items                                          collection of uno
```

```
mutable          dat-single                                          data-key:value pai

index based access                                                              keybased

[]                                                                   { }

|d1|d2|d3|                                                              |key|value|
----------                                                            ------------
 0 1   2                                                               |k1| v1|
                                                                       |k2|v2|
listname[index]---> value/IndexError                              dictcname["key"]---> v

l.append(value)
l.insert(index,value)                                              dictnname["newkey"]=v

l.pop() (or) l.pop(index)                                        dictname.pop("key") (or)
```

In [ ]:
```
L= [(), (),(), {}]
T= ([],(),{})
d={"k1":"v1"}

d={"k1":{}, "k2":[],"k3":()}
        |         |      |
       dit     list      tuple
```

In [ ]:
```
variable['key']
|
dict


variable[index]
|
list/tuple
```

In [ ]:
```
variable=()----> tuple
varibele()----> method/function
```

In [ ]:
```

```