In [1]:

```
#Dictionary datastructure
# If we want to represent group of objects as key-value pairs then we should go fo dictionary.
# Eg: 1.rollno -- name
#     2.phone no--address
#     3.Ip address--domainname
#Note: In c++ and java Dictionaries are also known as map where perl and ruby it is "hash"
```

In [2]:

```
# 1. Duplicate keys are not allowed but values can be dulicated.
# 2. Heterogenous objects are allowed for both keys and values
# 3. Insertion order is not preserved
# 4. Dictionaries are mutble.
# 5. Dictionaries are dynamic.
# 6. Indexing and slicing concepts are not applicable.
```

In [4]:

```
#creating a dictionary and printing the dictionary
student={'name':'aravind','age':21,'courses':['java','c','python']}
print(student)
```

```
{'name': 'aravind', 'age': 21, 'courses': ['java', 'c', 'python']}
```

In [7]:

```
#Access data from dictionary
# We can access data by using keys.
print(student['name'])
print(student['courses'])
```

```
aravind
['java', 'c', 'python']
```

In [6]:

```
#Access data from dictionary
# If the specified key is not available we will get Key Error
print(student['address'])
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-6-18d0f4c5c2f7> in <module>
      1 #Access data from dictionary
      2 # If the specified key is not available we will get Key Error
----> 3 print(student['address'])

KeyError: 'address'
```

In [12]:

```
#To avoid above error you can use get function
#get()
# to get the value associated with key
#d.get(key)
#  if the key is available then returns the corresponding value otherwise returns none.It won't raise any error
#d.get(key,defaultvalue)
# If the key is available then returns the corresponding value otherwise returns default value
student={'name':'aravind','age':21,'courses':['java','c','python']}
print(student.get('address'))
print(student.get('address','Not_found'))
```

```
None
Not_found
```

In [13]:

```
#updating the dictionary
# Syntax:d[key]=value
#If the key is not available then a new entry will be added to the dictionary with the specified key-value pair
# If the key is already available then old value will be replaced with new value.
student['address']='lasvegas'
print(student.get('address','Not_found'))
```

```
lasvegas
```

In [15]:

```
#updating the dictionary
student['name']='johnny deep'
print(student)
```

```
{'name': 'johnny deep', 'age': 21, 'courses': ['java', 'c', 'python'], 'address': 'lasvegas'}
```

In [16]:

```
#updating the multiple values at a time
student.update({'name':'jack','age':22,'address':'newyork'})
print(student)
```

```
{'name': 'jack', 'age': 22, 'courses': ['java', 'c', 'python'], 'address': 'newyork'}
```

In [17]:

```
#delete elements from dictionary
#Syntax : del d[key]
#It deletes an entry associated with specified key. If the key is not available then we will get key error
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
del student['address']
print(student)
```

```
{'name': 'jack', 'age': 22, 'courses': ['java', 'c', 'python']}
```

In [18]:

```
#pop() method to remove elements in dictionary
#Syntax: d.pop(key)
# It removes the entry associated with specific key and returns corresponding value
# If the specified key is not available we will get keyerror.
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
age=student.pop('age')
print(student)
print(age)
```

```
{'name': 'jack', 'courses': ['java', 'c', 'python'], 'address': 'newyork'}
22
```

In [19]:

```
#len() - returns no of items in a dictionary
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
print(len(student))
```

```
4
```

In [20]:

```
#clear - To remove all items in a dicitonary
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
student.clear()
print(student)
```

```
{}
```

In [23]:

```
#keys - It returns all keys associated with dicitonary
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
print(student.keys())
for k in student.keys():
    print(k)
```

```
dict_keys(['name', 'age', 'courses', 'address'])
name
age
courses
address
```

In [26]:

```
#values - It returns all values associated with dictionary
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
print(student.values())
for v in student.values():
    print(v)
```

```
dict_values(['jack', 22, ['java', 'c', 'python'], 'newyork'])
jack
22
['java', 'c', 'python']
newyork
```

In [28]:

```
#items() - It returns list of tuples representing key-value pairs
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
for k,v in student.items():
    print(k,'...',v)
```

```
name ... jack
age ... 22
courses ... ['java', 'c', 'python']
address ... newyork
```

In [29]:

```
#copy() - To create exactly duplicate dictionary(cloned copy)
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
student1=student.copy()
print(student1)
```

```
{'name': 'jack', 'age': 22, 'courses': ['java', 'c', 'python'], 'address': 'newyork'}
```

In [34]:

```
#setdefault()
# Syntax:d.setdefault(k,v)
# If the key is already available then function returns the corresponding value
# If the key is not available then specified key value will be added as new item to dictionary
student={'name':'jack','age':22,'courses':['java','c','python'],'address':'newyork'}
print(student.setdefault('phone',555555555))
print(student)
print(student.setdefault('phone',555555555))
print(student)
```

```
555555555
{'name': 'jack', 'age': 22, 'courses': ['java', 'c', 'python'], 'address': 'newyork', 'phone': 55555
5555}
555555555
{'name': 'jack', 'age': 22, 'courses': ['java', 'c', 'python'], 'address': 'newyork', 'phone': 55555
5555}
```