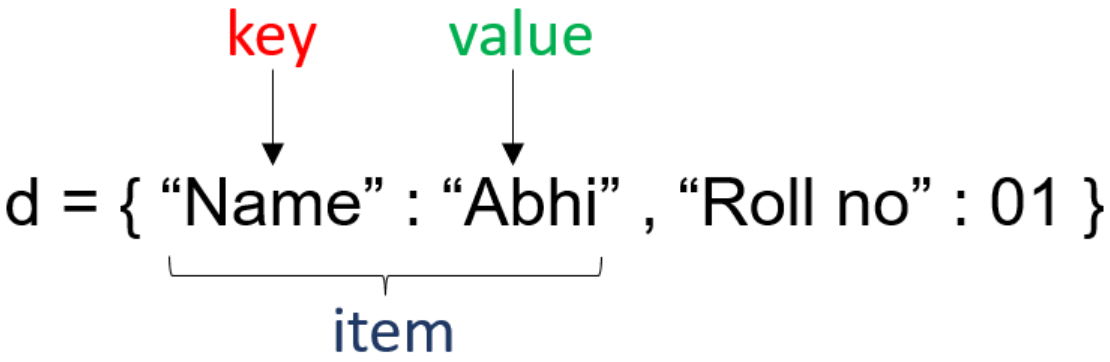
Dictionary

A Python dictionary is a collection of key:value pairs.

- It is the mutable data-structure.
- Keys must consist of just one element.
- Value can be any type such as list, tuple, integer, etc.
- keys are unique identifiers that are associated with each value.
- Like list, dictionary are also dynamic that is it can grow and shrink as needed.
- A dictionary can contain another dictionary. A dictionary can also contain a list, and vice versa.
- Dictionary elements are accessed via keys.

You can define a dictionary by enclosing a comma-separated list of key-value pairs in curly braces ({}). A colon (:) separates each key from its associated value.

syntax: d = {<"key1">:<"value">, <"key2">:<"value">, <"key3">:<"value">.....<"keyn">:<"value">}



Note:The key in dictionary must be unique whereas the value can be repeat. The combination of one key value pair is called one item.

In [2]:
```
# We can use curly bracket without any element to create the empty dictionary.
d = {}
print(d)         # printing the empty dictionary.
print(type(d))   # type() function tells the datatype of the object.
```

```
{}
<class 'dict'>
```

In [6]:
```
# Enter the elements as key value pair and separate by comma inside the curly brac
d={"key1":1, "key2":2, "key3":3}
print(d)
print(type(d))
```

```
{'key1': 1, 'key2': 2, 'key3': 3}
<class 'dict'>
```

In [11]:
```
# We can use tuple to create the dictionary using dict() function. But note the va
t=(
    ("Name","Broke"),
   ("Age",20),
   ("Course","DBDA"),
```

```
   ("City","Mumbai")
  )
d=dict(t)   # here, we are passing a pair of value inside the dict() function to cr
print(d)
print(type(d))
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
<class 'dict'>
```

In [12]:
```
# We can use list to create the dictionary using dict() function. But note the valu
t=[
    ["Name","Broke"],
   ["Age",20],
   ["Course","DBDA"],
   ["City","Mumbai"]
]
d=dict(t)   # here, we are passing a pair of value inside the dict() function to cr
print(d)
print(type(d))
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
<class 'dict'>
```

In [23]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d[0])    # Here we are trying to access the item using 0 index.
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[23], line 2
      1 d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
----> 2 print(d[0])

KeyError: 0
```

In [24]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d['roll no'])    # Here we are trying to access the item with that key that
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[24], line 2
      1 d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
----> 2 print(d['roll no'])

KeyError: 'roll no'
```

You may have noticed that the interpreter raises the same exception, KeyError, when a dictionary is accessed with either an undefined key or by a numeric index:

Note: Dictionary elements are not accessed by numerical index although they are ordered. A value is retrieved from a dictionary by specifying its corresponding key in square brackets [].

Accessing items from the dictionary.

In [14]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d['Name'])    # Here we are getting the value of the key 'Name'. We have to u
```

```
Broke
```

In [15]:
```
print(d['City'])
```

```
Mumbai
```

Adding the value to the dictionary.

In [17]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
d['Phone'] =8545657212  # Here we have to just assign the value to a particular key
print(d)
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai', 'Phone': 8545657212}
```

Updating/modifying the value in the dictionary.

In [19]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
d['Name'] ='Kevin'  # Here we have to just pass the vaue to the particular key whose
print(d)

# The value for the key "Name" change from "Broke" to "Kevin".
```

```
{'Name': 'Kevin', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
```

In [20]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
d['Course'] ='DAC'
print(d)
# The value for the key "Course" change from "DBDA" to "DAC".
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DAC', 'City': 'Mumbai'}
```

Note: Python is a case sensitive language.

Removing elements from Dictionary

We use the del statement to remove an element from the dictionary.

In [22]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(f'Before delete: {d}')
del d['City']      # Here we are deleting the item "City":"Mumbai"
print(f'Before delete: {d}')
```

```
Before delete: {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
Before delete: {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA'}
```

The del statement removes the element associated with the key

Nested dictionary

In [34]:
```
d={
    'Name': 'John',
    'Age': 20,
    'Siblings': ['Kevin', 'Gwen', 'Stacy'],
    'Parents':['Max', 'Kelly'],
    'City': 'Mumbai'
}
print(d)
print("--------------------")
print(d['Name'])
print("--------------------")
print(d['Siblings'])    # Here we are accesing the list of siblings
```

```
print("--------------------")
print(d['Siblings'][2])    # Here we are accesing the third elements in the sibling
print("--------------------")
print(d['Parents'])     # Here we are accesing the list of parents
```

```
{'Name': 'John', 'Age': 20, 'Siblings': ['Kevin', 'Gwen', 'Stacy'], 'Parents': ['M
ax', 'Kelly'], 'City': 'Mumbai'}
--------------------
John
--------------------
['Kevin', 'Gwen', 'Stacy']
--------------------
Stacy
--------------------
['Max', 'Kelly']
```

Dictionary Built-in methods

.clear() method

Syntax: Dictionary.clear()

This method clear the dictionary completely. It does not takes any argument.

In [37]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(f"Before using the method, {d}")
print("----------------------------------------")
d.clear()   # Here, we are using the clear() method to remove all items from the d
print(f"Before using the method, {d}")
```

```
Before using the method, {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'M
umbai'}
----------------------------------------
Before using the method, {}
```

.get() method

Syntax: Dictionary.get(key, value(optional))

- key: The keyname of the item you want to return the value from (Required).
- value: A value to return if the specified key does not exist (Optional). Default value None

This method returns the value of the mentioned key.

In [39]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.get('Name'))    # get() method return the indicated key. Here "Name" key is
```

```
Broke
```

In [40]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.get('Age'))
```

```
20
```

In [42]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.get('Country'))    # Here "Country" key is passed which is not in the dict
```

```
None
```

In [43]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.get('Country', 'India'))    # Here "Country" key is passed which is not in
```

```
India
```

.items() method

Syntax: Dictionary.items()

This method returns the key-value pairs of the dictionary, as tuples in a list. It does not take any arguments.

In [44]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.items())    # It returns all the key value pair.
```

```
dict_items([('Name', 'Broke'), ('Age', 20), ('Course', 'DBDA'), ('City', 'Mumba
i')])
```

In [45]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(list(d.items()))
# It returns all the key value pair inside a list. This means we can now use index
```

```
[('Name', 'Broke'), ('Age', 20), ('Course', 'DBDA'), ('City', 'Mumbai')]
```

In [46]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(list(d.items())[2])
```

```
('Course', 'DBDA')
```

.keys() method

Syntax: Dictionary.keys()

This method returns list of all keys. It does not take any arguments.

In [47]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.keys())
# It returns all the keys in the dictionary.
```

```
dict_keys(['Name', 'Age', 'Course', 'City'])
```

In [48]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(list(d.keys()))
```

```
['Name', 'Age', 'Course', 'City']
```

.values() method

Syntax: Dictionary.values()

This method returns list of all values in the list. It does not take any arguments.

In [51]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(d.values())
# It returns all the values in the dictionary.
```

```
dict_values(['Broke', 20, 'DBDA', 'Mumbai'])
```

In [52]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(list(d.values()))
# It returns all the values in the dictionary.
```

```
['Broke', 20, 'DBDA', 'Mumbai']
```

.pop() method

Syntax: Dictionary.pop()

The pop() method removes the specified item from the dictionary. The value of the removed item is the return value of the pop() method.

In [55]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(f"Before using the method, {d}")
print("----------------------------------------")
d.pop('Age')   # Here, we are using the pop() method to remove 'Age' key from the d
print(f"Before using the method, {d}")
```

```
Before using the method, {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'M
umbai'}
----------------------------------------
Before using the method, {'Name': 'Broke', 'Course': 'DBDA', 'City': 'Mumbai'}
```

.popitem() method

Syntax: Dictionary.popitems()

The popitem() method removes the item that was last inserted into the dictionary. The value of the removed item is the return value of the pop() method.

In [57]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
print(f"Before using the method, {d}")
print("----------------------------------------")
d.popitem()   # Here, we are using the pop() method to remove 'Age' key from the di
print(f"Before using the method, {d}")
```

```
Before using the method, {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'M
umbai'}
----------------------------------------
Before using the method, {'Name': 'Broke', 'Age': 20, 'Course': 'DBDA'}
```

Note: Here the last key-value pair removed from the dictionary. pop(key) remove a particuar key value pair from the dictionary whereas popitem() remove the last inserted key-vaue pair.

.fromkeys() method

Syntax: Dictionary.fromkeys(key, value)

The fromkeys() method returns a dictionary with the specified keys and the specified value.

In [58]:
```
collegename=['vit','srm','iitb','iit','nit']
d={}
d=d.fromkeys(collegename)
print(d)
# Here we passed the collegename list as a key into the .fromkeys() method whereas
```

```
{'vit': None, 'srm': None, 'iitb': None, 'iit': None, 'nit': None}
```

In [60]:
```
collegename=['vit','srm','iitb','iit','nit']
d={}
d=d.fromkeys(collegename,100)
```

```
print(d)
# Here we passed the collegename list as a key into the .fromkeys() method whereas
```

```
{'vit': 100, 'srm': 100, 'iitb': 100, 'iit': 100, 'nit': 100}
```

.update() method

Syntax: Dictionary.update(iterable)

The update() method inserts the specified items to the dictionary. The specified items can be a dictionary, or an iterable object with key value pairs.

In [61]:
```
d1={"c1":100, "c2":200, "c3":300}
d2={"c2":600, "c3":700, "c6":800}
d1.update(d2)
print(d1)
# Here the value of c2 and c3 changes as per d2 dictionary. Also, c6 key value pai
```

```
{'c1': 100, 'c2': 600, 'c3': 700, 'c6': 800}
```

In [62]:
```
d1 = {'a': 10, 'b': 20, 'c': 30}
d1.update(b=200, d=400)
print(d1)
```

```
{'a': 10, 'b': 200, 'c': 30, 'd': 400}
```

.copy() method

Syntax: Dictionary.copy()

The copy() method returns a copy of the specified dictionary.

In [63]:
```
d={'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
x=d.copy()
print(x)
# .copy() method makes a shallow copy which means if we make change in the original
# not affect.
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
```

In [64]:
```
d['Country']='India'
print(d)
print(x)
# Here you can see we added one more key value pair to dictionary d but there will
```

```
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai', 'Country': 'Indi
a'}
{'Name': 'Broke', 'Age': 20, 'Course': 'DBDA', 'City': 'Mumbai'}
```

Click this link to learn more: 👉 https://github.com/Abhishekk-Git