By Sai Kumar

DICTIONARY DATA STRUCTURE

- We can use List, Tuple and Set to represent a group of individual objects as a single entity.
- If we want to represent a group of objects as key-value pairs then we should go for Dictionary.

Eg:

rollno ---- name

phone number -- address

ipaddress --- domain name

- Duplicate keys are not allowed but values can be duplicated.
- Heterogeneous objects are allowed for both key and values.
- Insertion order is not preserved
- Dictionaries are mutable
- Dictionaries are dynamic
- indexing and slicing concepts are not applicable

Note: In C++ and Java Dictionaries are known as "Map" where as in Perl and Ruby it is known as "Hash"

How to Create Dictionary?

d = {} OR d = dict()

We are creating empty dictionary. We can add entries as follows

1) **d[100]="Sai"**

2) **d[200]="ravi"**

3) **d[300]="shiva"**

4) **print(d)** → {100: 'sai', 200: 'ravi', 300: 'shiva'}

If we know data in advance then we can create dictionary as follows

d = {100:'sai' ,200:'ravi', 300:'shiva'}

d = {key:value, key:value}

By Sai Kumar

How to Access Data from the Dictionary?

We can access data by using keys.

1) **d = {100:'sai',200:'ravi', 300:'shiva'}**

2) **print(d[100])** #sai

3) **print(d[300])** #shiva

If the specified key is not available then we will get **KeyError**

we have to use in operator.

if 400 in d:

print(d[400])

Example:

Write a Program to Enter Name and Percentage Marks in a Dictionary and Display Information on the Screen

```
rec={}
n=int(input("Enter number of students: "))
i=1
while i <=n:
name=input("Enter Student Name: ")
marks=input("Enter % of Marks of Student: ")
rec[name]=marks
i=i+1
print("Name of Student","\t","% of marks")
for x in rec:
print("\t",x,"\t\t",rec[x])
```

How to Update Dictionaries?

- 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.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d)**

3) **d[400]="pavan"**

4) **print(d)**

5) **d[100]="sunny"**

6) **print(d)**

How to Delete Elements from Dictionary?

1) **del d[key]**

It deletes entry associated with the specified key.

If the key is not available then we will get **KeyError.**

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d)**

3) **del d[100]**

4) **print(d)**

5) **del d[400]**

d.clear()

To remove all entries from the dictionary.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d)**

3) **d.clear()**

4) **print(d)**

3) **del d**

To delete total dictionary. Now we cannot access d.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d)**

3) **del d**

4) **print(d)**

By Sai Kumar

Important Functions of Dictionary

1) **dict():**

To create a dictionary

d = dict() → It creates empty dictionary

d = dict({100:"sai",200:"ravi"}) → It creates dictionary with specified elements

d = dict([(100,"sai"),(200,"shiva"),(300,"ravi")])

It creates dictionary with the given list of tuple elements

2) **len()**

Returns the number of items in the dictionary.

3) **clear():**

To remove all elements from the dictionary.

4) **get():**

To get the value associated with the 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.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d[100])** → sai

3) **print(d[400])** → KeyError:400

4) **print(d.get(100))** → sai

5) **print(d.get(400)) → None**

6) **print(d.get(100,"Guest"))** → sai

7) **print(d.get(400,"Guest"))** → Guest

5) **pop():**

d.pop(key)

It removes the entry associated with the specified key and returns the corresponding value.

If the specified key is not available then we will get KeyError.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d.pop(100))**

3) **print(d)**

4) **print(d.pop(400))**

6) **popitem():**

It removes an arbitrary item(key-value) from the dictionary and returns it.

1**) d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d)**

3) **print(d.popitem())**

4) **print(d)**

Output

{100: 'sai', 200: 'ravi', 300: 'shiva'}

(300, 'shiva')

{100: 'sai', 200: 'ravi'}

If the dictionary is empty then we will get **KeyError**

d={}

print(d.popitem()) ==>KeyError: 'popitem(): dictionary is empty'

7) **keys():**

It returns all keys associated with the dictionary.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d.keys())**

3) **for k in d.keys():**

    print(k)

8) **values():**

It returns all values associated with the dictionary.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d.values())**

3) **for v in d.values():**

    print(v)

9) **items():**

It returns list of tuples representing key-value pairs.

[(k,v),(k,v),(k,v)]

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **for k,v in d.items():**

    print(k,"--",v)

10) **copy():**

To create exactly duplicate dictionary (cloned copy)

d1 = d.copy();

11) **setdefault():**

d.setdefault(k,v)

If the key is already available then this function returns the corresponding value.

If the key is not available then the specified key-value will be added as new item to the dictionary.

1) **d={100:"sai",200:"ravi",300:"shiva"}**

2) **print(d.setdefault(400,"pavan"))**

3) **print(d)**

4) **print(d.setdefault(100,"sachin"))**

5) **print(d)**

12) **update():**

d.update(x)

All items present in the dictionary x will be added to dictionary d

*** **Happy Learning** ***