Dictionary: { } → Mutable

data

Format → key : value // any type for both key & value.

a_dict = { key1 : value1 , key2 : value2 .... }

Copy method.

sample_dict2 = sample_dict1.copy()

creates a new dict from existing but are different.

Another method of creating dict -

sample_dict3 = dict ([(pair1), (pair2), (pair3) ....])

list with pairs

eg: c = dict([("the", 1), (1, 2)])

⇒ c = {'the': 1, 1: 2}

~~pairs~~

pairs → (2 items) ⇒ created as key & value. // tuples

Methods for creating dict-

dict.fromkeys([key1, key2...])

default → values = None

dict.fromkeys([key1, key2...], value)

all keys will have same value

[No indexing in dictionary.]

Accessing elements -

① sample - dict[key-name] → prints corresponding value.

② sample - dict.get(key) → returns corresponding value

(2.1) sample_dict.get(non-existing-key) → returns None.

(2.1.1) sample_dict.get(non-existing-key, returnvalue) →
instead of none prints the return value.

get all keys -
sample_dict.keys()

get all values -
sample_dict.values()

get all the key-value pairs in pair form -
sample_dict.items()

} returns in a list format

looping -

for i in sample_dict:
1) print(i) ⟹ prints all keys
2) print(i, sample_dict[i]) ⟹ key & value separated by space.

```
for i in sample_dict.values():
    print(i) => prints all values.
```

Checking membership -

key in sample_dict → returns True / False

Adding data -

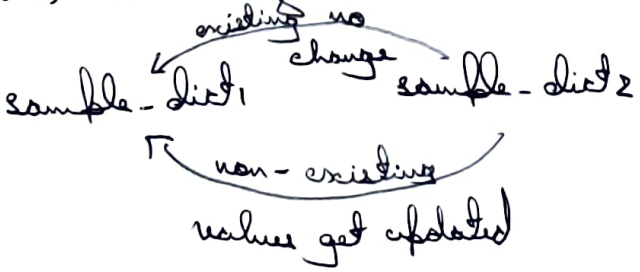
```
dict_name[key_name] = corresponding_value
```

Updating data -

```
dict_name[existing_key_name] = new_value
```

→ update() -

Given, 2 dicts. If both have some common elements.




```
sample_dict1.update(sample_dict2)
```

Removing data-

1. sample_dict.pop(key) → returns deleted value
2. del sample_dict[key]
3. sample_dict.clear() → deletes entire elements of dict
4. del sample_dict → deletes the dictionary itself.