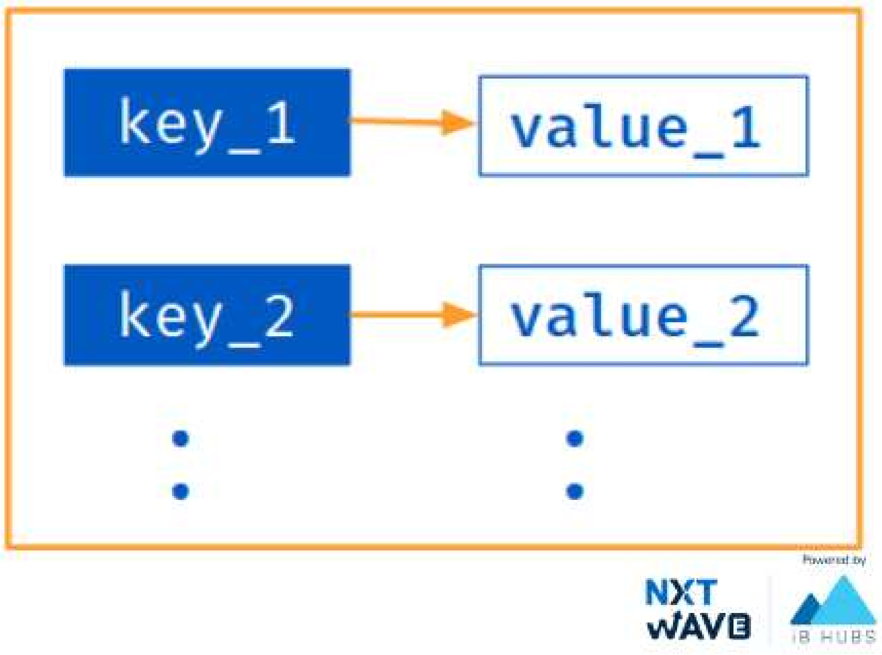
Cheat Sheet

Dictionaries

Unordered collection of items.

Every dictionary item is a **Key-value** pair.



Creating a Dictionary

Created by enclosing items within **{curly}** brackets

Each item in dictionary has a key - value pair separated by a **comma**.

Code

PYTHON

```python
dict_a = {
  "name": "Teja",
  "age": 15
}
```

Key - Value Pairs

Code

PYTHON

```python
dict_a = {
  "name": "Teja",
  "age": 15
}
```

In the above dictionary, the

- keys are `name` and `age`
- values are `Teja` and `15`

Collection of Key-Value Pairs

Code

PYTHON

```python
dict_a = { "name": "Teja",
           "age": 15 }
print(type(dict_a))
print(dict_a)
```

Output

```
<class 'dict'>
{'name': 'Teja','age': 15}
```

Immutable Keys

Keys must be of immutable type and must be unique.

Values can be of any data type and can repeat.

Code

PYTHON

```python
dict_a = {
  "name": "Teja",
  "age": 15
  "age": 15,
  "roll_no": 15
}
```

Creating Empty Dictionary

Code - 1

PYTHON

```python
dict_a = dict()
print(type(dict_a))
print(dict_a)
```

Output

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

Code - 2

PYTHON

```python
dict_a = {}
print(type(dict_a))
print(dict_a)
```

Output

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

Accessing Items

To access the items in dictionary, we use square bracket

`[ ]` along with the `key` to obtain its value.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
print(dict_a['name'])
```

Output

```
Teja
```

Accessing Items - Get

The

get() method returns None if the key is not found.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
print(dict_a.get('name'))
```

Output

```
Teja
```

Code

PYTHON

```python
dict_a = {
    'name': 'Teja',
    'age': 15
}
print(dict_a.get('city'))
```

Output

```
None
```

KeyError

When we use the square brackets

[] to access the key-value, **KeyError** is raised in case a key is not found in the dictionary.

Code

PYTHON

```python
dict_a = {'name': 'Teja','age': 15 }
print(dict_a['city'])
```

Output

```
KeyError: 'city'
```

Quick Tip

If we use the square brackets `[]`, `KeyError` is raised in case a key is not found in the dictionary. On the other hand, the `get()` method returns `None` if the key is not found.

Membership Check

Checks if the given key exists.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
result = 'name' in dict_a
print(result)
```

Output

```
True
```

Operations on Dictionaries

We can update a dictionary by

- Adding a key-value pair
- Modifying existing items
- Deleting existing items

Adding a Key-Value Pair

Code

PYTHON

```python
dict_a = {'name': 'Teja','age': 15 }
dict_a['city'] = 'Goa'
print(dict_a)
```

Output

```
{'name': 'Teja', 'age': 15, 'city': 'Goa'}
```

Modifying an Existing Item

As dictionaries are mutable, we can modify the values of the keys.

Code

PYTHON

```python
dict_a = {
    'name': 'Teja',
    'age': 15
}
dict_a['age'] = 24
print(dict_a)
```

Output

```
{'name': 'Teja', 'age': 24}
```

Deleting an Existing Item

We can also use the

`del` keyword to remove individual items or the entire dictionary itself.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
del dict_a['age']
print(dict_a)
```

Output

```
{'name': 'Teja'}
```

Dictionary Views

They provide a dynamic view on the dictionary’s entries, which means that when the dictionary changes, the view reflects these changes.

Dictionary Methods

- dict.keys()
  - returns dictionary Keys
- dict.values()
  - returns dictionary Values
- dict.items()
  - returns dictionary items(key-value) pairs

The objects returned by

keys() , values() & items() are **View Objects** .

Getting Keys

The

keys() method returns a view object of the type dict_keys that holds a list of all keys.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
print(dict_a.keys())
```

Output

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

Getting Values

The

`values()` method returns a view object that displays a list of all the values in the dictionary.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
print(dict_a.values())
```

Output

```
dict_values(['Teja', 15])
```

Getting Items

The

`items()` method returns a view object that displays a list of dictionary's (key, value) tuple pairs.

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
print(dict_a.items())
```

Output

```
dict_items([('name', 'Teja'), ('age', 15)])
```

Iterate over Dictionary Views

Example - 1

Code

PYTHON

```python
dict_a = {
    'name': 'Teja',
    'age': 15
}
for key in dict_a.keys():
   print(key)
```

Output

```
name
age
```

Example - 2

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
keys_list = list(dict_a.keys())
print(keys_list)
```

Output

```
['name', 'age']
```

Example - 3

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
for value in dict_a.values():
    print(value)
```

Output

```
Teja
15
```

Example - 4

Code

PYTHON

```python
dict_a = {
  'name': 'Teja',
  'age': 15
}
for key, value in dict_a.items():
    pair = "{} {}".format(key,value)
    print(pair)
```

Output

```
name Teja
age 15
```

Dictionary View Objects

keys() , values() & items() are called Dictionary Views as they provide a dynamic view on the dictionary's items.

Code

PYTHON

```python
dict_a = {
    'name': 'Teja',
    'age': 15
}
view = dict_a.keys()
print(view)
dict_a['roll_no'] = 10
print(view)
```

Output

```
dict_keys(['name', 'age'])
dict_keys(['name', 'age', 'roll_no'])
```

Converting to Dictionary

dict(sequence) takes any number of key-value pairs and converts to dictionary.

Code

PYTHON

```python
list_a = [
    ("name","Teja"),
    ["age",15],
    ("roll_no",15)
]
dict_a = dict(list_a)
print(dict_a)
```

Output

```
{'name': 'Teja', 'age': 15, 'roll_no': 15}
```

Code

PYTHON

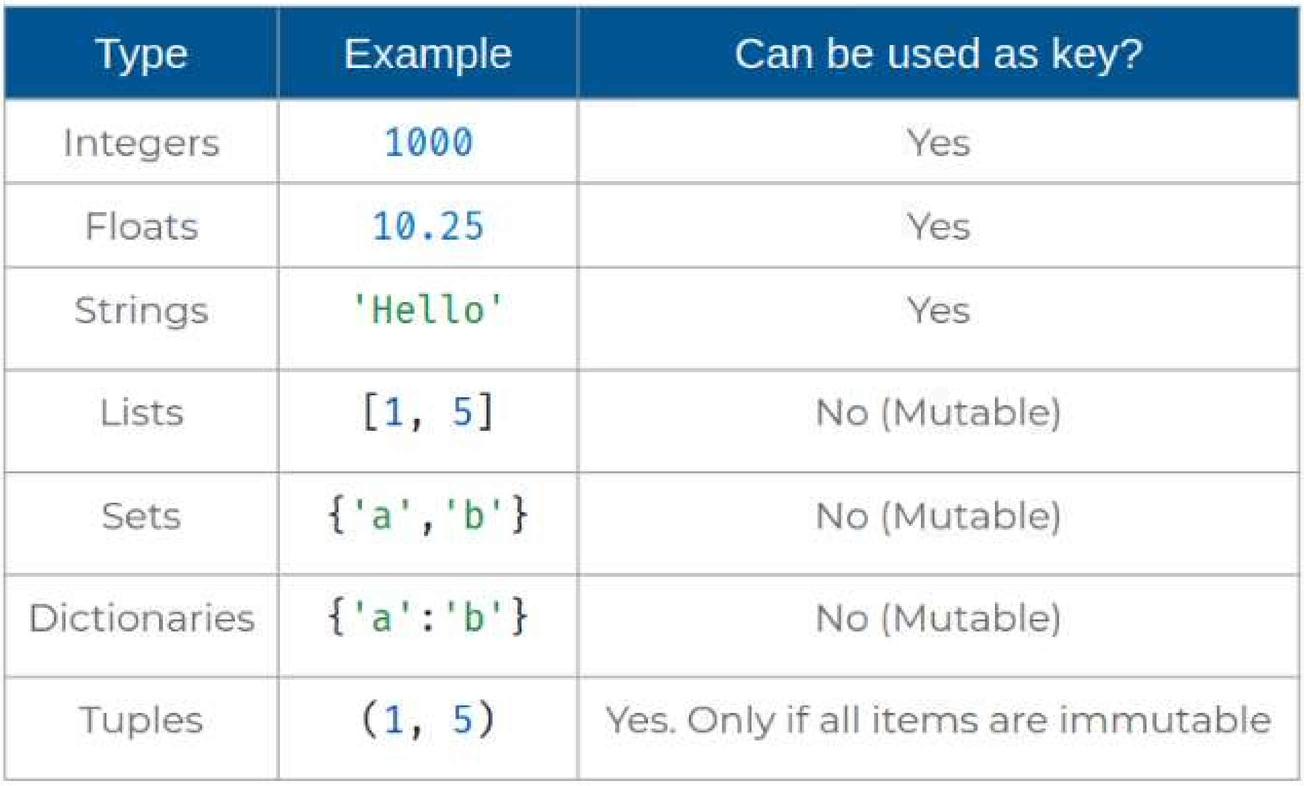
```python
list_a = ["name", "Teja", 15]
dict_a = dict(list_a)
print(dict_a)
```

Output

```
ValueError: dictionary update sequence element #0 has length 4; 2 is
```

Type of Keys

A dictionary key must be of a type that is immutable.



Type	Example	Can be used as key?
Integers	`1000`	Yes
Floats	`10.25`	Yes
Strings	`'Hello'`	Yes
Lists	`[1, 5]`	No (Mutable)
Sets	`{'a','b'}`	No (Mutable)
Dictionaries	`{'a':'b'}`	No (Mutable)
Tuples	`(1, 5)`	Yes. Only if all items are immutable

Submit Feedback