Dictionary

What is Dictionary?

- Dictionaries are used to store data values in `key:value` pairs.
- Dictionary has no indexing
- Dictionary Keys → Immutable, Values → they can be mutable
- Key Should be Unique
- A dictionary is a collection which is ordered*, changeable and do not allow duplicates.

```
Empty_dict = {}

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict) # {'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["brand"]) # Ford
```

```
Duplicate values will overwrite existing values:

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 1920
}
print(thisdict) #{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Accessing

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]

#There is also a method called get() that will give you the same result:
x = thisdict.get("model")
```

`get` method will not work with 2D dictionary

```
# if dictionary is like this
# {'name':'adi', 'marks':{'maths':54, 'eng': 63, 'sci': 74}}
# i want to get the sci marks
# I can't use get method here get only work with single key
# instead

x = thisdict['marks']['sci']
print(x) # 74
```

Get Keys

The `keys()` method will return a list of all the keys in the dictionary.

```
x = thisdict.keys()

dict_keys(['brand', 'model', 'year'])
```

The list of the keys is a *view* of the dictionary, meaning that any changes done to the dictionary will be reflected in the keys list.

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change

# dict_keys(['brand', 'model', 'year'])
# dict_keys(['brand', 'model', 'year', 'color'])
```

Get Values

The `values()` method will return a list of all the values in the dictionary.

```
x = thisdict.values()

# dict_values(['Ford', 'Mustang', 1964])
```

The list of the values is a *view* of the dictionary, meaning that any changes done to the dictionary will be reflected in the values list.

```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.values()

print(x) #before the change

car["year"] = 2020

print(x) #after the change

car["color"] = "red"

print(x) #after the change

# dict_values(['Ford', 'Mustang', 1964])
# dict_values(['Ford', 'Mustang', 2020])
# dict_values(['Ford', 'Mustang', 2020, 'red'])
```

Check if Key Exists

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes")

# Yes
```

Change Dictionary Items

Change Values

to change value we use `[]` like `thisdict[‘year’] = 2018`

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018

print(thisdict) # {'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Update Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
```

The argument must be a dictionary, or an iterable object with key:value pairs.

Add Items - the square bracket

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
# {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Update Dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
print(thisdict)

# # {'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Remove Items

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")

print(thisdict) # {'brand': 'Ford', 'year': 1964}
```

`DEL`

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict) # {'brand': 'Ford', 'year': 1964}
```

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict
print(thisdict) #this will cause an error because "thisdict" no longer exists.
```

`Clear` - empties the dictionary

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict) # {}
```

Looping through Dictionary

```
# Print all key names in the dictionary, one by one:

for x in thisdict: # for x in thisdict.keys():
  print(x)

brand
model
year
```

```
# Print all values in the dictionary, one by one:

for x in thisdict: # for x in thisdict.values():
  print(thisdict[x])

Ford
Mustang
1964
```

```
# Loop through both keys and values, by using the items() method:

for x, y in thisdict.items():
  print(x, y)

brand Ford
model Mustang
year 1964
```

Copy a Dictionary

You cannot copy a dictionary simply by typing `dict2 = dict1`, because: `dict2` will only be a *reference* to `dict1`, and changes made in `dict1` will automatically also be made in `dict2`.

`copy`

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
```

Nested Dictionaries

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}
```

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
```

Access item in Nested Dict

```
print(myfamily["child2"]["name"]) # Tobias
```

Loop Through Nested Dictionaries

```
for x, obj in myfamily.items():
  print(x)

  for y in obj:
    print(y + ':', obj[y])
```

```
child1
name: Emil
year: 2004
child2
name: Tobias
year: 2007
child3
name: Linus
year: 2011
```