Dictionaries

Dictionaries:

Словари используются для хранения значений данных в парах ключ:значение.
Словарь представляет собой упорядоченный* набор, изменяемый и не допускающий дублирования.

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

Элементы словаря представлены парами ключ:значение, и на них можно ссылаться, используя имя ключа.

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

Ford
```

В словарях не может быть двух элементов с одним и тем же ключом:

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

{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

The values in dictionary items can be of any data type:

```
thisdict = {
  "brand": "Ford",
  "electric": False,
  "year": 1964,
  "colors": ["red", "white", "blue"]
}
```

Accessing Items:

You can access the items of a dictionary by referring to its key name, inside square brackets:

```
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")
```

keys() method.

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

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

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

values() method.

Метод `values()` вернет список всех значений в словаре.

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

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

items() method.

Метод `items()` вернет каждый элемент словаря в виде кортежей в списке.

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
```

```
  "year": 1964
}
x = thisdict.items()
print(x)

dict_items([('brand', 'Ford'), ('model', 'Mustang'), ('year', 1964)])
```

Чтобы определить, присутствует ли указанный ключ в словаре, используйте `in` ключевое слово:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Change and Add Values:

You can change the value of a specific item by referring to its key name:

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

update() method.

The `update()` method will update the dictionary with the items from the given argument.

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

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

Removing Items:

There are several methods to remove items from a dictionary:

1. The `pop()` and `del` method removes the item with the specified key name:

```
thisdict = {
  "brand": "Ford",
```

```
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)

{'brand': 'Ford', 'year': 1964}

# The del keyword can also delete the dictionary completely!!!!!!!!!!!!!!!!!!!!!

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

2. The `popitem()` method removes the last inserted item

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

{'brand': 'Ford', 'model': 'Mustang'}
```

3. The `clear()` method empties the dictionary:

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

Loop Through a Dictionary:

При переборе словаря возвращаемым значением являются *ключи* словаря, но есть и методы для возврата *значений* .

1. Print all key names in the dictionary, one by one:

```
for x in thisdict:
  print(x)

# You can use the keys() method to return the keys of a dictionary:

for x in thisdict.keys():
```

```
  print(x)
```

2. Print all *values* in the dictionary, one by one:

```
for x in thisdict:
  print(thisdict[x])

# You can also use the values() method to return values of a dictionary:

for x in thisdict.values():
  print(x)
```

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

```
thisdict =      {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x, y in thisdict.items():
  print(x, y)

brand Ford
model Mustang
year 1964
```

Copy Dictionaries:

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`.
There are ways to make a copy, one way is to use the built-in Dictionary method `copy()`.

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

Another way to make a copy is to use the built-in function `dict()`.

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

```
print(mydict)
```

Nested Dictionaries:

Словарь может содержать словари, это называется вложенными словарями.

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

Or, if you want to add three dictionaries into a new dictionary:

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

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

Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary