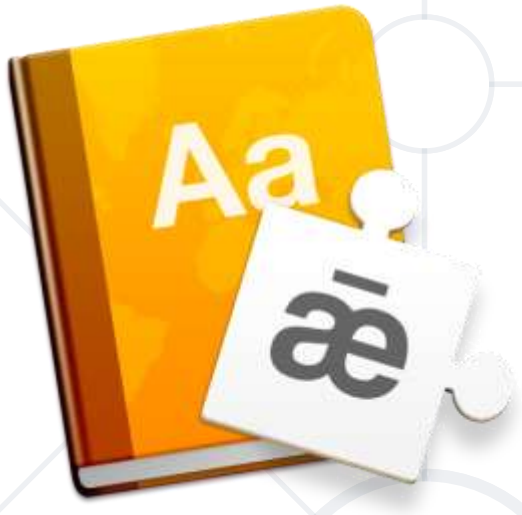
Dictionaries



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#PIDS

Table of Contents

Dictionary

Storing Key-Value Pairs

Definition

- In Python a dictionary is an **ordered collection** of items (Python 3.7+)
- While other data types have only value as an element, a dictionary has **key-value pairs**
- **Values** can be of any data type and **can repeat**
- **Keys** must be **immutable** and must be **unique**

Examples

- Creating a dictionary using curly braces {}

```
# empty dictionary
my_dict = {}
# dictionary with string keys
my_dict = {'fruit': 'apple', 'vegetable': 'cucumber'}
```

Key

Value

- Creating a dictionary using `dict()` function

```
dict_arguments = dict(name="George", age=22)
# {"name": "George", "age": 22}
```

key=value argument

Examples

- You can write a dictionary on **multiple lines**

```
my_dict = {
    'fruit': 'apple',
    'vegetable': 'cucumber',
    'diary': 'milk',
}
```

Indentation

Comma after every pair

Keys and Values

What is a Key?

- While indexing is used with other container types to access values, the dictionary uses **keys**
- Key can be used either inside **square brackets** or with the **`get()`** method

```
my_dict = {'name':'Jack', 'age': 26}
print(my_dict['name'])     # Jack
print(my_dict.get('age')) # 26
my_dict['address']         # KeyError
my_dict.get('address')     # None
```

Change Values

- Dictionary is a **mutable collection**
- We can add **new items** or **change** the value of existing items using an assignment operator
- If the key is already present, the value gets **updated**, else a new pair is **added** to the dictionary

```
my_dict = {'name':'Jack', 'age': 26}
my_dict['age'] = 27    # update
print(my_dict['age']) # 27
```

Iterating Through Dictionaries

Iterating through keys()

- Using the `keys()` method to get all the keys from a dictionary

```
squares = {1: 1, 2: 4, 3: 9}
for key in squares.keys():
    print(key, end=" ") # 1 2 3
```

- Changing the values by iterating through the keys

```
squares = {1: 1, 2: 4, 3: 9}
for key in squares.keys():
   squares[key] *= 2
# {1: 2, 2: 8, 3: 18}
```

Iterating through values()

- Using the `values()` method to get all the values

```
squares = {1: 1, 2: 4, 3: 9}
for value in squares.values():
    print(value, end=" ") # 1 4 9
```

- We can also use the keys to get the values

```
squares = {1: 1, 2: 4, 3: 9}
for key in squares.keys():
    print(squares[key], end=" ") # 1 4 9
```

Iterating Using items()

- Use the `items()` method to iterate through key-value pairs
- It returns `tuple` (key, value) pairs (tuples will be covered in the advanced course)

```
squares = {
    1: 1,
    2: 4,
    3: 9,
}
for (key, value) in squares.items():
    print(f"Key: {key}, Value: {value}")
```

Existence in Dictionary

Checking for Existence

- Check for **key** existence by using the **keys()** method

```
my_dict = {'name': 'Peter', 'age': 22}
if 'name' in my_dict.keys(): # You can skip keys()
    print(my_dict['name']) # Peter
```

- Check for **value** existence by using the **values()** method

```
my_dict = {'name': 'Peter', 'age': 22}
if 22 in my_dict.values():
    print("22 is a value in the dictionary")
# 22 is a value in the dictionary
```

Dictionary Methods

Dictionary Methods

- `clear()` - removes all the elements from a dictionary

```
my_dict = {1: 'apple', 2: 'banana'}
my_dict.clear()
print(my_dict) # {}
```

- `copy()` - returns a copy of a dictionary

```
my_dict = {1: 'apple', 2: 'banana'}
copied_dict = my_dict.copy()
print(my_dict == copied_dict) # True
```

Dictionary Methods

- `pop()` - removes and returns an item from a dictionary having the given key

```
my_dict = {"fruit": "apple", "vegetable": "cucumber"}
apple = my_dict.pop("fruit") # 'apple'
print(my_dict) # {'vegetable': 'cucumber'}
```

- `popitem()` - removes an item that was last inserted and returns it as a tuple - (key, value)

```
my_dict = {"fruit": "apple", "vegetable": "cucumber"}
print(my_dict.popitem()) # ("vegetable", "cucumber")
print(my_dict) # {"fruit": "apple"}
```

Dictionary Methods

- `del` keyword - removes an item with a specified key name

```
students = {"name": "George", "course": "Fundamentals"}
del students["course"]
print(students) # {"name": "George"}
```

- `del` keyword can also delete the dictionary completely

```
students = {"name": "George", "course": "Fundamentals"}
del students
print(students) # NameError
```

Summary

- We learned:
  - What dictionaries are
  - How to create dictionaries
  - How to iterate through dictionaries
  - Additional dictionary methods
  - Nested dictionaries
  - Dictionary comprehensions

Questions?

SoftUni

Software University | SoftUni Creative | SoftUni Digital | SoftUni Foundation | SoftUni Kids | СофтУни БУДИТЕЛ | Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

top EMPLOYER България Bulgaria 2024 FOR A BETTER WORLD OF WORK

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos, and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg