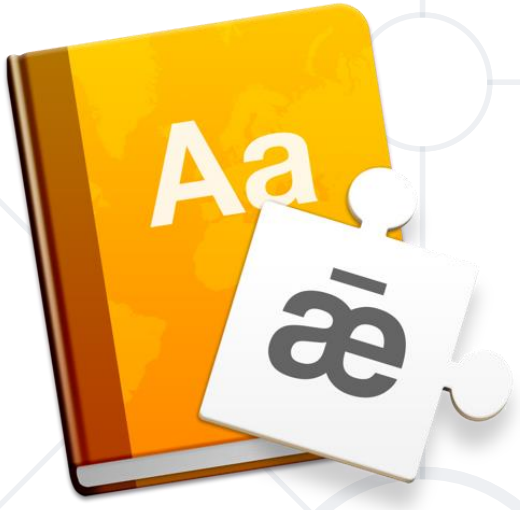
Dictionaries



SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#fund-python

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
```

Problem: Bakery

- You will receive a single line containing some **food (keys)** and **quantities (values)**
- They will be separated by a single space (the first element is the key, the second – the value and so on)
- Create a **dictionary** and **print** it on the console

```
bread 10 butter 4 sugar 9 jam 12
```

```
{'bread': 10, 'butter': 4, 'sugar': 9, 'jam': 12}
```

Solution: Bakery

```
elements = input().split(" ")
bakery = {}  # bakery = dict()
for i in range(0, len(elements), 2):
    key = elements[i]
    value = elements[i + 1]
    bakery[key] = int(value)
print(bakery)
```

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
```

Problem: Stock

- You will be given **key-value pairs** of products and quantities
- On the next line you will be given products to **search** for
- Check for each product, you have **2 possibilities**:
  - If you have it, print "`We have {quantity} of {product} left`"
  - Otherwise, print "`Sorry, we don't have {product}`"

Solution: Stock

```
elements = input().split(" ")
bakery = {}
# Fill in the products in the dictionary

searched_products = input().split(" ")
for product in searched_products:
    if product in bakery:
        print(f"We have {bakery[product]} of {product} left")
    else:
        print(f"Sorry, we don't have {product}")
```

Problem: Statistics

- You will be receiving **key-value** pairs on separate lines separated by ": " until you receive the command "**statistics**"
- Sometimes you may receive a product **more than once**. In that case, add up the **quantities**
- When you receive the "**statistics**" command, print the output as in the example

```
bread: 4
cheese: 2
ham: 1
bread: 1
statistics
```

```
Products in stock:
- bread: 5
- cheese: 2
- ham: 1
Total Products: 3
Total Quantity: 8
```

Solution: Statistics

```
products = {}
command = input()
while command != "statistics":
    # split the command and get the product and the quantity
    if product not in products:
        products[product] = 0
    products[product] += quantity
    command = input()
# TODO: print the result
```

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"}
```

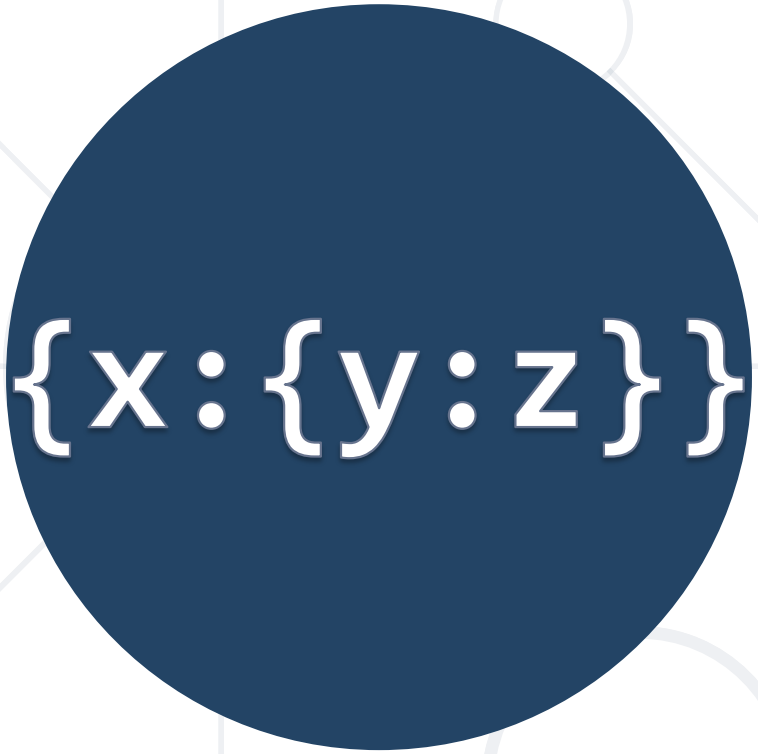
Dictionary Methods

- `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
```



Nested Dictionaries

What is Nested Dictionary?

- It is a **collection of dictionaries** into one single dictionary
- The nested dictionary's key has **another** dictionary as value
- Nested dictionaries are useful if you want to store a **large** amount of data in a **structured** way

Nested Dictionary

Software University

- **Creating** a nested dictionary

```
students = {1: {'name': 'Peter', 'age': 22},
            2: {'name': 'Alex', 'age': 21}}
```

- **Accessing** an element

```
first_student_name = students[1]['name']
print(first_student_name) # Peter
```

- **Adding** an element

```
students[3] = {} # {3: {}}
students[3]['name'] = 'Amy' # {3: {'name': 'Amy'}}
students[3]['age'] = 25 # {3: {'name': 'Amy', 'age': 25}}
```

Iterate through Nested Dictionary

- We use nested for-loop

Using items() method

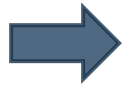
```
shopping_list = {
    "foods": {"nuts": "almonds"},
    "drinks": {"soft": "lemonade", "wine": "merlot"}
}

for key, value in shopping_list.items():
    for nested_key, nested_value in value.items():
        print(f'{nested_value} bought')
        shopping_list[key][nested_key] = 'bought'
```

Problem: Students

- You will be receiving names of students, their ID, and a course of programming they have taken in format "`{name}:{ID}:{course}`"
- On the last line you will receive the name of a course in **snake case** style
- You should print only the information of the students taken the corresponding course in the format: "`{name} - {ID}`" on separate lines

```
Peter:123:programming basics
John:5622:fundamentals
Maya:89:fundamentals
Lilly:633:fundamentals
fundamentals
```



```
John - 5622
Maya - 89
Lilly - 633
```

Solution: Students

```
students_dict = {}
command = input()
while ":" in command:
    info = command.split(":")
    name, id, course = info[0], info[1], info[2]
    if course not in students_dict:
        students_dict[course] = {}
    students_dict[course][id] = name
    command = input()

course = " ".join(command.split("_"))
for key, value in students_dict.items():
    if key == course:
        for id, name in value.items():
            print(f'{name} - {id}')
```

Dictionary Comprehensions

Dictionary Comprehensions

Software University

- Creating a dictionary using dictionary comprehension

```
data = [("Peter", 22), ("Amy", 18), ("George", 35)]
dictionary = {key:value for (key, value) in data}
# {'Peter': 22, 'Amy': 18, 'George': 35}
```

- Form a dictionary with cube values of numbers

```
nums = [1, 2, 3]
cubes = {num:num ** 3 for num in nums}
# {1: 1, 2: 8, 3: 27}
```

Problem: ASCII Values

- Write a program that receives a **list of characters** and creates a dictionary with each **character** as a **key** and its **ASCII** value as a **value**. Try solving that problem using **comprehensions**

```
a, b, c, a
```

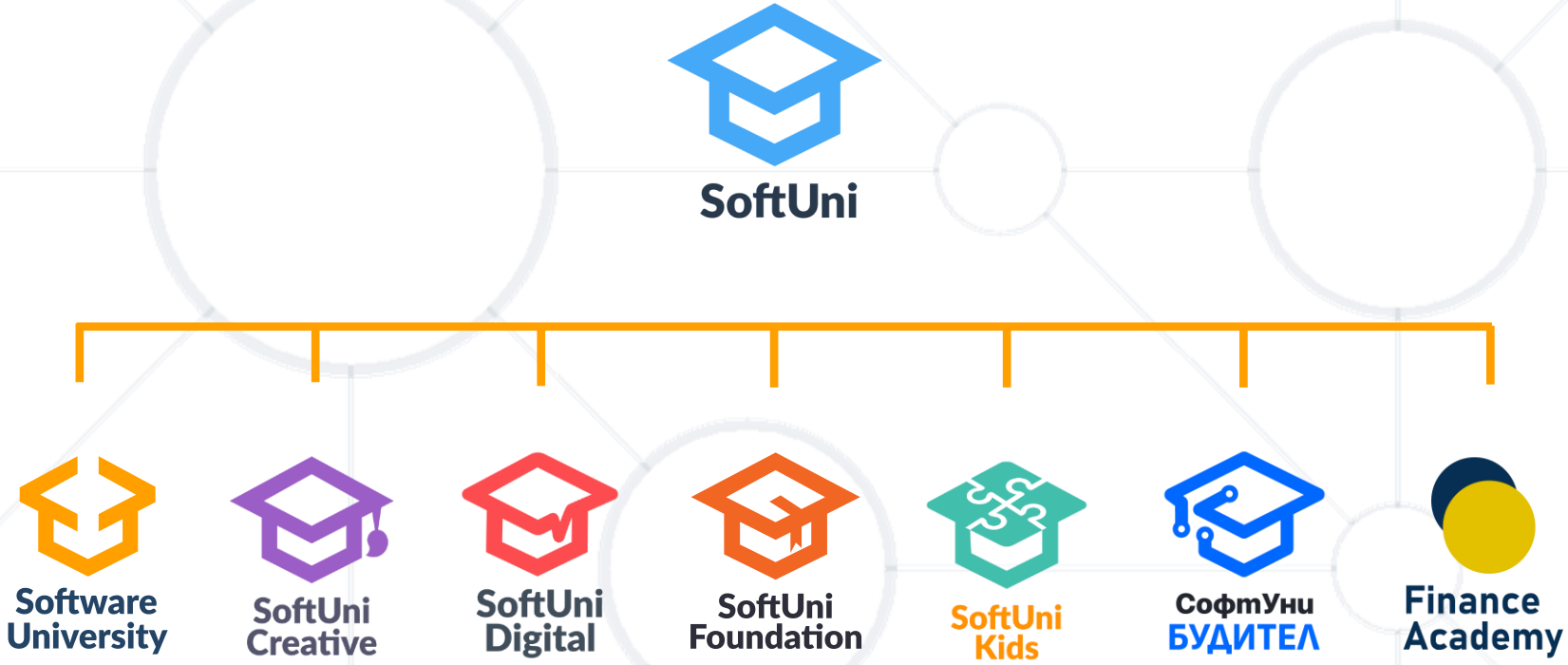
```
{'a': 97, 'b': 98, 'c': 99}
```

```
words = input().split(", ")
occurrences = {word: ord(word) for word in words}
print(occurrences)
```

Summary

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

Questions?



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