# Tutorial IV.II - Dictionaries

## Programming: Everyday Decision-Making Algorithms

## Introduction

Welcome to this tutorial on dictionaries in Python! In the realm of personal task scheduling, organizing tasks efficiently is crucial. Imagine managing your daily activities like checking emails, planning finances, and scheduling meetings. Python's dictionaries can be a helpful tool for handling such tasks by allowing you to store and retrieve information using key-value pairs. As always, follow the structured instructions, implement your code in the designated blocks, and affirm your comprehension with `assert` statements.

## Section 1 - Creating and Accessing Dictionaries

Think of it this way:

- A dictionary consists of a collection of key-value pairs
- Each key-value pair has a unique key (like for example the task name)
- And an associated value (like for example the task's priority level)
- Keys are unique, but values can be repeated
- Dictionaries can grow and shrink as needed

Let's see some examples:

```
# Creating a dictionary of tasks
tasks = {
    "Check Emails": 1,
    "Liquidity Planning": 2,
    "Team Meeting": 3
}
print(tasks)
```

```
{'Check Emails': 1, 'Liquidity Planning': 2, 'Team Meeting': 3}
```

To access the value associated with a key, you can use the following syntax:

```
print(tasks["Check Emails"])
```

```
1
```

New entries can be added to the dictionary or existing entries can be modified using the following syntax:

```
tasks["Team Meeting"] = 4
tasks["Answer Emails"] = 7
print(tasks)
```

```
{'Check Emails': 1, 'Liquidity Planning': 2, 'Team Meeting': 4, 'Answer
Emails': 7}
```

To delete entries, you can use the `del()` function:

```
del(tasks["Answer Emails"])
print(tasks)
```

```
{'Check Emails': 1, 'Liquidity Planning': 2, 'Team Meeting': 4}
```

> ⚠ Warning
>
> You will have to create the dictionary first before you can add new entries to it!

To check if a key exists in the dictionary, you can use the following syntax:

```
print("Check Emails" in tasks)
```

```
True
```

Furthermore, we can nest a dictionary in a dictionary:

```
# Creating a dictionary of projects
projects = {
    "Project Alpha": {"status": "In Progress", "deadline": "2023-12-31"},
    "Project Beta": {"status": "Completed", "deadline": "2023-06-30"}
}
print(f"The deadline of Project Alpha is {projects['Project Alpha']
['deadline']}.")
```

## Exercise 1.1 - Create and Modify a Dictionary

Add a new task called “Prepare Presentation” with the priority level `5` to the existing `tasks` dictionary. Note, that you will have to execute the code cell above that creates the dictionary first before you can add the new task!

```
# YOUR CODE BELOW
```

```
# Test your answer
assert "Prepare Presentation" in tasks, "The task 'Prepare Presentation'
was not added to the dictionary."
```

```
print("Great! You've successfully added a new task to the tasks
dictionary.")
```

### Exercise 1.2 - Check if a Key Exists

Check if the key "Team Meeting" exists in the `tasks` dictionary using the `in` operator and a conditional statement. If it exists, save the message: `'Team Meeting is in the dictionary'` to the variable `message`. If it does not exist, save the message: `'Team Meeting is not in the dictionary'` to the variable `message`. Finally, print the value of the `message` variable.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert message == "Team Meeting is in the dictionary", "The message is not
correct. It should be 'Team Meeting is in the dictionary'."
print("Great! You've successfully checked if a key exists in the
dictionary.")
```

### Exercise 1.3 - Change the Value of a Key

Change the value associated with the key "Check Emails" to `6`.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert tasks["Check Emails"] == 6, "The value associated with the key
'Check Emails' is not correct. It should be 6."
print("Great! You've successfully changed the value of a key in the
dictionary.")
```

## Section 2 - Advanced Dictionary Operations

Dictionaries can do more than just store simple information. Let's explore some features.

With the `keys()` method, you can get all the keys of a dictionary as a dictionary view object. You can convert it to a list using the `list()` function.

```
task_names = tasks.keys() # Get all keys
print("Tasks:", task_names) # Still a dictionary view object
print("Tasks as list:", list(task_names)) # Convert to a list
```

```
Tasks: dict_keys(['Check Emails', 'Liquidity Planning', 'Team Meeting'])
Tasks as list: ['Check Emails', 'Liquidity Planning', 'Team Meeting']
```

To get all the values from a dictionary, you can use the `values()` method to get a dictionary view object. Again, you can convert it to a list using the `list()` function.

```
all_deadlines = tasks.values()
print("All deadlines:", list(all_deadlines))
```

```
All deadlines: [1, 2, 4]
```

We can also loop through the dictionary using a `for` loop. To do so, we can use the `items()` method which returns a dictionary view object containing the individual key-value pairs of the dictionary.

```
for task, priority in tasks.items():
    print(f"Task: {task}, Priority: {priority}")
```

```
Task: Check Emails, Priority: 1
Task: Liquidity Planning, Priority: 2
Task: Team Meeting, Priority: 4
```

To check the length of a dictionary, you can use the `len()` function.

```
print(len(tasks))
```

```
3
```

## Exercise 2.1 - Compute the Average Priority

Compute the average priority of the tasks in the `tasks` dictionary. Save the result to the variable `average_priority`.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert len(tasks) == 4, "The number of tasks is not correct. It should be 4
based on the previous excercises."
assert average_priority == 4.25, "The average priority is not correct. It
should be 4.25."
print("Great! You've successfully computed the average priority of the
tasks.")
```

## Exercise 2.2 - Get the Task with the Highest Priority

Get the task with the highest priority from the `tasks` dictionary. Save the result to the variable `highest_priority_task`. Note, that a priority of 1 is the highest priority.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert highest_priority_task == "Liquidity Planning", f"The task
```

```
{highest_priority_task} as the task with the highest priority is not
correct. It should be 'Liquidity Planning'."
print("Great! You've successfully gotten the task with the highest
priority.")
```

### Exercise 2.3 - Remove a Task

First, check if the key `"Liquidity Planning"` exists in the `tasks` dictionary. If it exists, remove it using the appropriate method.

> Tip
>
> You can use the `del()` function to remove a key from a dictionary.

```
# YOUR CODE BELOW
```

```
# Test your answer
assert "Liquidity Planning" not in tasks, "The task 'Liquidity Planning'
was not removed from the dictionary."
print("Great! You've successfully removed a task from the dictionary.")
```

### Conclusion

Great! You've just navigated through the basics of dictionaries in Python. Dictionaries are powerful data structures that allow for efficient data organization and retrieval. Remember:

- Dictionaries can store information using key-value pairs
- Accessing them using keys is efficient and easier as accessing them using indices
- Loops can iterate over dictionaries to perform operations on each key-value pair

## Solutions

You will likely find solutions to most exercises online. However, we strongly encourage you to work on these exercises independently without searching explicitly for the exact answers to the exercises. Understanding someone else's solution is very different from developing your own. Use the lecture notes and try to solve the exercises on your own. This approach will significantly enhance your learning and problem-solving skills.

Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities. If you encounter difficulties, review the lecture materials, experiment with different approaches, and don't hesitate to ask for clarification during class discussions.

Later, you will find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them next week. To access the solutions, click on the Github button on the lower right and search for the folder with today's lecture and tutorial. Alternatively, you can ask ChatGPT or Claude to explain them to

you. But please remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.

---

In the next tutorial, we'll dive deeper into DataFrames and how to use them!