# Tutorial II.V - Dictionaries

## Applied Optimization with Julia

## Introduction

Imagine you have a school directory where each student's name is associated with their unique student ID. This is similar to how dictionaries work in programming - they allow you to store and retrieve information using key-value pairs.

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 is like a lookup table
- Each entry has a unique key (like a student's name)
- And an associated value (like their ID number)

Let's see some examples:

```
# Creating a dictionary
student_ids = Dict(
    "Elio" => 1001,
    "Bob" => 1002,
    "Yola" => 1003
)
```

```
Dict{String, Int64} with 3 entries:
  "Elio" => 1001
  "Bob"  => 1002
  "Yola" => 1003
```

```
# Accessing values
println("Elio's ID: ", student_ids["Elio"])
```

```
Elio's ID: 1001
```

```
# Adding a new entry
student_ids["David"] = 1004
```

```
1004
```

```
# Checking if a key exists
if haskey(student_ids, "Eve")
    println("Eve's ID: ", student_ids["Eve"])
else
    println("Eve is not in the directory")
end
```

```
Eve is not in the directory
```

### Exercise 1.1 - Create and Modify a Dictionary

Add a new book called “Harry Potter and the Philosopher’s Stone” with the author “J.K. Rowling” to the created dictionary.

```
# Creates a dictionary of books and authors
books = Dict(
    "1984" => "George Orwell",
    "Nexus" => "Yuval Noah Harari"
)
# YOUR CODE BELOW
```

```
# Test your answer
@assert haskey(books, "Harry Potter and the Philosopher's Stone")
println("Great! You've successfully added a new book to the books
dictionary.")
```

### Exercise 1.2 - Modify a Dictionary

Change the author of “1984” to “Eric Blair” (George Orwell’s real name).

```
# YOUR CODE BELOW
```

```
# Test your answer
@assert books["1984"] == "Eric Blair"
println("Great! You've successfully modified the books dictionary.")
```

---

## Section 2 - Advanced Dictionary Operations

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

```
# A dictionary of student grades
grades = Dict(
    "Elio" => [85, 92, 78],
    "Bob" => [76, 88, 94],
    "Yola" => [90, 91, 89]
)
```

```
Dict{String, Vector{Int64}} with 3 entries:
  "Elio" => [85, 92, 78]
  "Bob"  => [76, 88, 94]
  "Yola" => [90, 91, 89]
```

```
# Get all the keys (student names)
student_names = keys(grades)
println("Students: ", student_names)
```

```
Students: ["Elio", "Bob", "Yola"]
```

```
# Get all the values (grade lists)
all_grades = values(grades)
println("All grades: ", all_grades)
```

```
All grades: [[85, 92, 78], [76, 88, 94], [90, 91, 89]]
```

```
# Calculate average grade for each student
for (student, grade_list) in grades
    average = sum(grade_list) / length(grade_list)
    println("$student's average grade: $average")
end
```

```
Elio's average grade: 85.0
Bob's average grade: 86.0
Yola's average grade: 90.0
```

> **i** Note
>
> The `(student, grade_list)` is a tuple that contains the key and value of each entry in the dictionary. We could also name the tuple as `(key, value)` or `(a, b)`.

---

## Conclusion

Great! You've just navigated through the basics of dictionaries in Julia. Dictionaries are powerful data structures that allow for efficient data organization and retrieval. Continue to the next file to learn more advanced Julia concepts.

# Solutions

You will likely find solutions to most exercises online. However, I 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 in next week's tutorial. 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.

## Bibliography