Hyperiondev

"Lecturer": Henri Branken
Lesson Topic: Python Dictionaries
Date: 2024-07-31

Review of Lesson Timeline Overview

Learning Objectives and Outcomes	**1** mins
Initial Assessment (Poll)	**5** mins
Introduction	**6** mins
Lesson Body	**35** mins
Check/s for Understanding - Poll/s	**5** mins
Lesson Conclusion and Recap	**6** mins
Homework or Follow-up Activities	**2** mins
Total	**60** mins

Materials and Resources Needed by Lecturer

- **Books/Readings:** None
- **Worksheets:** None
- **Technology:** **VSCode** on presentation device (Laptop) for [1]**Coding** & [2]**Demonstration** purposes.
- **Other Tools**: Pre-prepared Exercises. Quizizz Polls.

Learning Outcomes (1 mins)

By the end of this lesson, learners should be able to:

- **Explain** the concept of Python Dictionaries, including:
    - **Defining** what a Python Dictionary is.
    - **Elaborating** when this type of Data Structure needs to be used.
- **Create** their own instances of Python Dictionaries based on simple (and relatable) real-life scenarios.

Initial Assessment - Poll (5 mins)

- **Purpose:** In light of the fact that I want to build upon the students' existing knowledge, I am going to ask them about other Basic Python Structures they should already know at this point in time. This way, I prep them for another Data Structure I am about to introduce them to. My aim is to also facilitate evaluation between the different Python data structures.
  Contingency - What if my students show very poor understanding in this area?
  Work through a "backup" slideshow that recaps on important [1]properties, [2]definitions and [3]examples of Python Data structures my students should know at this point. That way, I can hopefully "re-scaffold" their knowledge to better absorb Python Dictionaries.
- **Tools/Methods Used:** Quizizz.
- **Questions:**
  1. Which of the following situations is best suited for using a Python list?
     [a] You need to store a sequence of items that should not change.
     [b] You need to store a collection of unique items.
     [c] You need to store a pair of values.
     [d] **You need to store a sequence of items that can be modified.**

  2. What will be the output of the following code?
     `my_list = [1, 2, 3]; my_list.append(4); print(my_list);`
     [a] `[1, 2, 3]`
     [b] **`[1, 2, 3, 4]`**
     [c] `[1, 2, 4]`
     [d] `[4, 1, 2, 3]`

  3. Which of the following benefits does a tuple provide over a list?
     [a] Tuples are mutable.
     [b] Tuples can contain duplicate elements.
     [c] **Tuples have a smaller memory footprint.**
     [d] Tuples allow adding and removing elements.

  4. Which of the following operations can be performed on a tuple?
     a. `my_tuple.append(4)`
     b. `my_tuple[1] = 10`
     [c] **`my_tuple + (4, 5)`**
     d. `my_tuple.remove(2)`

  5. Which operation is most efficient with a set compared to a list?
     [a] Retrieving an element by index.
     [b] **Checking if an element is present.**
     [c] Adding an element to the end.
     [d] Accessing the first element.

  6. Which of the following is a valid way to create a set in Python?
     [a] `my_set = [1, 2, 3]`
     [b] `my_set = (1, 2, 3)`

[c] **`my_set = {1, 2, 3}`**
[d] `my_set = <1, 2, 3>`

7. What is the result of the following code?
```
set1 = {1, 2, 3};
set2 = {2, 3, 4};
print(set1.intersection(set2));
```
[a] `{1, 2, 3, 4}`
[b] **`{2, 3}`**
[c] `{1}`
[d] `{4}`

8. Why would you choose a string over a list of characters?
[a] **You need the collection to be immutable.**
[b] You need to store non-character data.
[c] You need to modify individual characters.
[d] You need to perform arithmetic operations.

9. What is the output of the following code:
```
my_string = "43210";  print(my_string[-2]);
```
[a] **`1`**
[b] `0`
[c] `2`
[d] `4`

10. You are developing a function that needs to return a fixed collection of latitude and longitude coordinates. Which data structure should you use?
[a] List
[b] **Tuple**
[c] Set
[d] String

Introduction (2 mins)

Establish Relevance

- **Problem Statement:** Consider for example the following **"Configuration Settings"**:
  Theme ⇔ "Dark"
  Language ⇔ "English"
  Timeout ⇔ 30 seconds.

Also consider the following Geographical Data of **"Cities"**:
🔵 New York
  Population ⇔ 8 419 000
  Area ⇔ 468.9
  Coordinates ⇔ (40.7128, -74.0060)
🔵 Los Angeles

Population ⇔ 3 980 400
Area ⇔ 503
Coordinates ⇔ (34.0522, -118.2437)

What is the **most efficient way to elegantly** represent this information? Is there perhaps another Data Structure in Python for associating every unique key with a certain value. I.e.:
Key ⇔ Value.

- Context and Engagement:
    - Engage the Learners by asking them how an **English Dictionary** organises words and their meanings/definitions. They can do this in a Think-Pair-Share format.
    - Present the Learning Objectives, and emphasise the importance of leveraging Python Dictionaries to handle certain (and complex) Data Structures, and to code efficient and organised solutions.

Lesson Body - Approaches and Activities (40 mins)

Activation

- Introduce the new data structure, i.e. Python Dictionaries.
    - **Explanation**: Python Dictionaries are data structures that store key-value pairs, allowing for fast and efficient retrieval of values based on the unique keys.
    - **Use Case of Dictionaries**: Organise and manage data where each item (For Example: "Theme" ⇔ "Dark") can be accessed via a descriptive key (in this case: "Theme").

Demonstration

- **Demonstrate**:
    - The **Syntax** for creating and initialising dictionaries in Python.
    - **Accessing** the value of a dictionary
    - **Adding** a new key-value pair
    - **Update** an existing value
    - **Removing** a key-value pair
- Demonstrate some Dictionary **Methods**. E.g.: `.get()` `.keys()` etc.
- Demonstrate **Iterating** over a dictionary.

Application

- **Code Along**:
  I supply students with Contact Information:
  "john_doe":
  "phone": "123-456-7890"
  "email": "**john@example.com**"
  "jane_smith":
  "phone": "987-654-3210"
  "email": "**jane@example.com**"
  I ask for their suggestions on how I can create a dictionary that stores this information for "John" and "Jane".

- **Individual Attempt**:
  Ask students to code the Geographic Data (given in the “Problem Statement” bullet) into a Python Dictionary.
  Provide guidance and support as they work through this exercise.

Check/s for Understanding - Poll/s (5 mins)

Integration by means of a Quizizz Poll.

1. What is a Python Dictionary?
   [a] An ordered collection of elements indexed by integers.
   [b] **An unordered collection of key-value pairs.**
   [c] A mutable sequence of characters.
   [d] An immutable collection of unique elements.

2. Which of the following correctly defines a Python dictionary?
   [a] `my_dict = [1, 2, 3, 4]`
   [b] `my_dict = (1, 2, 3, 4)`
   [c] `my_dict = {1, 2, 3, 4}`
   [d] **`my_dict = {“name”: “Alice”, “age”: 25}`**

3. In which scenario is it most appropriate to use a Python Dictionary?
   [a] You need to store a list of student names in a specific order.
   [b] You need to store unique elements with no duplicates.
   [c] **You need to map employee IDs to employee details.**
   [d] You need to perform a series of arithmetic operations on a list of numbers.

4. Which of the following is a real-world use case for a Python dictionary?
   [a] **Managing a set of key-value pairs such as configuration settings.**
   [b] Keeping track of unique values in a dataset.
   [c] Storing a collection of items in a specific order.
   [d] Representing a sequence of characters in a string.

5. How do you add a new key-value pair to an existing dictionary my_dict?
   [a] **`my_dict.append("email", "alice@example.com")`**
   [b] **`my_dict["email"] = "alice@example.com"`**
   [c] **`my_dict.add("email", "alice@example.com")`**
   [d] **`my_dict.update("email", "alice@example.com")`**

Lesson Conclusion and Recap (5 mins)

Recap the key concepts and techniques covered during the lesson.

- **Python Dictionaries** are:

    - A Collection of **Key-Value Pairs**.
      They are defined by using curly braces `{}` with key-value pairs separated by commas.
    - **Unordered** and **Mutable**.
- Dictionary **values** are accessed by using their corresponding **keys** inside **square brackets**.
    - **Keys** must be unique inside a dictionary.
    - The **Values** can be of any type: **numbers**, **strings**, **lists**, **tuples**, other **dictionaries**.

Emphasise the Importance of dictionaries in any application that needs a flexible and efficient way to manage key-value pairs. The Python Dictionary is a vital data-structure because of its efficient storage of complex datasets.

Examples of Real-Life Scenarios:

- **Web Development**
- **Data Processing (JSON | API)**
- **Machine Learning**
- **Database Access**
- **Etc...**

Homework or Follow-up Activities (2 min)

Integration

Given that you understand:
- How Python List comprehension works
- The basics of Python Dictionaries (their syntax and creation)

Give an attempt to solve the following problem relying on **Dictionary Comprehension**.

Challenge 1:
Use dictionary comprehension to generate a dictionary where the keys are the integers from the list and the values are their squares. You're welcome to Google resources to clue yourself in on how dictionary comprehension works.

```
# Step 1: Create a list of integers
numbers = [1, 2, 3, 4, 5]

# Step 2: Use dictionary comprehension to create the dictionary
# Attempt your solution here
squares_dict =

# Step 3: Print the resulting dictionary
print(squares_dict)  # Expected Output → {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

Challenge 2 (Optional):
Modify your program to only include even integers from the list in the dictionary.

```
# Step 1: Create a list of integers
numbers = [1, 2, 3, 4, 5]

# Step 2: Use dictionary comprehension to create the dictionary with even numbers only
# Attempt your solution here
even_squares_dict =

# Step 3: Print the resulting dictionary
print(even_squares_dict)  # Expected Output → {2: 4, 4: 16}
```

Challenge 3 (Optional):
You are given a dictionary that contains student names as keys and their respective grades as values. Complete the following Python script in which we want to print the Student's name and grade in a nicely formatted way. (*Google* resources to figure out iterating through a dictionary):

```
# 1. The student names and their grades
grades = {
    "Alice": 85,
    "Bob": 78,
    "Charlie": 92,
    "David": 88,
    "Eve": 95
}

# 2. Print the names and grades
# Attempt your solution here:
for . . . . . .:
    print(. . . .)
```

Evaluating their performance on this Homework:
I will set up these homework questions in **TestGorilla**, and ensure that the results I get back are anonymised. I can also build basic "Unit Testing" into the Python scripts they need to complete at the `*# Attempt your solution here*` sections. That way:

- Students can feel more confident in knowing that they will not be "picked upon" if their solution turns out to be wrong.
- Students can critically think about their own solutions by running the unit-test cases and see whether these fail or pass.

Motivation:
I need to gauge how much my students learned and determine if they're managing the application of the new concept (Python Dictionaries) they learned.

Educator Resources

- **Examples to Include:**
  - Website Configuration Settings
  - City Geographical Data
  - Contact Information Data (John Doe & Jane Smith)
- **Interactive Tools for the Lesson:**

    - **Quizizz**
    - VS Code **Live Share**
- Homework: **Challenges 1 - 3**. **TestGorilla**.