TIP101 | Intro to Technical Interview Prep

Intro to Technical Interview Prep Fall 2025 (@ Section 3 | Tuesdays and Thursdays 5PM - 7PM PT)
Personal Member ID#: **134071**

Need help? Post on our **class slack channel** or email us at **support@codepath.org**

Session 1: Dictionaries in Python

Session Overview

The focus of this session is advanced data handling in Python, focusing on functions, lists, and dictionaries. Students will tackle practical programming challenges such as verifying subsequences, creating and manipulating dictionaries, and calculating values based on dynamic inputs.

The tasks are designed to enhance understanding of data structures, algorithmic thinking, and conditional logic, preparing students for more complex problem-solving scenarios involving data manipulation and retrieval.

> You can find all resources from today including the session deck, session recording, and more on the resources tab

📈 Part 1 : Instructor Lead Session

We'll spend the first portion of the synchronous class time in large groups, where the instructor will lead class instruction for 30-45 minutes.

👩‍💻 Part 2: Breakout Session

In breakout sessions, we will explore and collaboratively solve problem sets in small groups. Here, the **collaboration, conversation, and approach** are just as important as “solving the problem” - please engage warmly, clearly, and plentifully in the process!

In breakout rooms you will:

- Screen-share the problem/s, and verbally review them together
- Screen-share an interactive coding environment, and talk through the steps of a solution approach
    - ProTip: - An Integrated Development Environment (IDE) is a fancy name for a tool you could use for shared writing of code - like Replit.com, Collabed.it, CodePen.io, or other - your staff team will specify which tool to use for this class!
- Screen-share an implementation of your proposed solution

Your program leader/s will indicate which code sharing tool/s to use as a group, and will help break down or provide specific scaffolding with the main concepts above.

▶ **Note on Expectations**

🔎 Problem Solving Approach

To build a long-term organized approach to problem solving, we'll start with three main steps. We'll refer to them as **UPI: Understand, Plan, and Implement**.

We'll apply these three steps to most of the problems we'll see in the first half of the course.

We will learn to:

- **Understand** the problem,
- **Plan** a solution step-by-step, and
- **Implement** the solution

▶ **Comment on UPI**

▶ **UPI Example**

Breakout Problems Session 1

💡 Unit 2 Cheatsheet

To jump start your journey into Python dictionaries, we've put together a guide to common functions and syntax you will use throughout Unit 2 breakout problems. Use this cheatsheet as a quick reference guide as you work through the problems below.

▼ Problem Set Version 1

Problem 1: All In

Write a function `all_in()` that takes in a list of integers `a` and a list of integers `b` as parameters. Given these two lists, return `True` if *every* element in list `a` is in list `b`. Return `False` otherwise.

```
def all_in(a, b):
    pass
```

Example Usage:

```
print(all_in(lst_1, lst_2))
print(all_in(lst_2, lst_1))
```

Example Output:

```
True
False
```

▶ 💡 **Hint: Problem Type**

Problem 2: Create a Dictionary

Write a function `create_dictionary()` that takes in a list of `keys` and a list of `values` as parameters. The function returns a dictionary where each item in `keys` is paired with a corresponding item in `values`.

`keys[i]` should be paired with `values[i]` in the dictionary where `0` <= `i` <= `len(keys)`. You may assume `keys` and `values` are the same length.

```
def create_dictionary(keys, values):
    pass
```

Example Input:

```
keys = ['peanut', 'dragon', 'star', 'pop', 'space']
values = ['butter', 'fly', 'fish', 'corn', 'ship']
```

Example Output:

```
{'peanut': 'butter', 'dragon': 'fly', 'star': 'fish', 'pop': 'corn', 'space': 'ship'}
```

▶ ✨ **AI Hint: Dictionaries**

Problem 3: Print Pair

Write a function `print_pair()` that takes in a dictionary `dictionary` and a key `target` as parameters. The function looks for the `target` and when found, it prints the key and it's associated value as `"Key: <key>"` followed by `"Value: <value>"`. If `target` is not in `dictionary`, print `"That pair does not exist!"`.

```
def print_pair(dictionary, target):
    pass
```

```
dictionary = {"spongebob": "squarepants", "patrick": "star", "squidward": "tentacles"}
print_pair(dictionary, "patrick")
print_pair(dictionary, "plankton")
print_pair(dictionary, "spongebob")
```

Example Output:

```
Key: patrick
Value: star
That pair does not exist!
Key: spongebob
Value: squarepants
```

▶ ✨ AI Hint: Accessing Values in a Dictionary

▶ 💡 **Hint: Dictionary Access options**

Problem 4: Keys Versus Values

Write a function `keys_v_values()` that takes in a dictionary `dictionary` whose keys and values are both integers. *Using at least one loop*, the function should find the sum of all keys in the dictionary and the sum of all values.
If the sum of all keys is greater than the sum of all values, the function should return the string `"keys"`.
If the sum of all values is greater than the sum of all keys, the function should return the string `"values"`.
If keys and values have equal sums, the function should return the string `"balanced"`.

```
def keys_v_values(dictionary):
    pass
```

Example Usage:

```
dictionary1 = {1:10, 2:20, 3:30, 4:40, 5:50, 6:60}
greater_sum = keys_v_values(dictionary1)
print(greater_sum)

dictionary2 = {100:10, 200:20, 300:30, 400:40, 500:50, 600:60}
greater_sum = keys_v_values(dictionary2)
print(greater_sum)
```

Example Output:

```
values
```

▶ 💡 **Hint: Accessing Keys, Values, and Key-Value Pairs**

Problem 5: Restock Inventory

Write a function `restock_inventory()` that updates an inventory dictionary based on a restock list. It accepts two parameters:

- `current_inventory`: a dictionary where each key-value pair represents an item and its current stock in the inventory
- `restock_list`: a dictionary where each key-value pair represents an item and the quantity to be added to the inventory

If an item in `restock_list` is not present in the `current_inventory`, it should be added. The function should return the updated dictionary `current_inventory`.

```
def restock_inventory(current_inventory, restock_list):
    pass
```

Example Input:

```
current_inventory = {
    "apples": 30,
    "bananas": 15,
    "oranges": 10
}

restock_list = {
    "oranges": 20,
    "apples": 10,
    "pears": 5
}
```

Example Output:

```
{
    "apples": 4 0,
    "bananas": 15,
    "oranges": 30,
    "pears": 5
}
```

▶ 💡 **Hint: Looping over Key-Value Pairs**

Write a function `calculate_gpa()` that calculates the grade point average (GPA) for a student based on their course grades and returns the `gpa` as a float. The function takes in a dictionary `report_card` as a parameter where each key-value pair represents a course and the grade received in that course respectively. The grades are represented as strings (`"A"`, `"B"`, `"C"`, `"D"`, `"F"`) and each grade corresponds to a certain number of grade points:

`"A"` = 4
`"B"` = 3
`"C"` = 2
`"D"` = 1
`"F"` = 0

A GPA is calculated by finding the average of all grade points.

```
def calculate_gpa(report_card):
    pass
```

Example Usage:

```
report_card = {"Math": "A", "Science": "C", "History": "A", "Art": "B", "English": "B", "Span
print(calculate_gpa(report_card))
```

Example Output: `3.3333333333333335`

Problem 7: Best Book

Imagine you are working on a book review software like Goodreads. Write a function named `highest_rated()` that returns the book with the highest rating.

The function should take in a **list of dictionaries** named `books` as a parameter. Each dictionary represents data associated with a book, including its title, author, and rating. The function should return the dictionary for the book with the highest rating.

```
def highest_rated(books):
    pass
```

Example Input:

```
books = [
    {"title": "Tomorrow, and Tomorrow, and Tomorrow",
     "author": "Gabrielle Zevin",
     "rating": 4.18
    },
    {"title": "A Fortune For Your Disaster",
     "author": "Hanif Abdurraqib",
     "rating": 4.47
    },
```

```
    "rating": 4.40
  }
]
```

Expected Output:

```
{"title": "A Fortune For Your Disaster",
 "author": "Hanif Abdurraqib",
 "rating": 4.47
}
```

Problem 8: Index-Value Map

Write a function `index_to_value_map()` that takes in a list `lst` and returns a dictionary that maps the index of each element in `lst` to its value.

```
def index_to_value_map(lst):
    pass
```

Example Input: `lst = ["apple", "banana", "cherry"]`

Example Output: `{0: "apple", 1: "banana", 2: "cherry"}`

Close Section

▸ **Problem Set Version 2**

▸ **Problem Set Version 3**