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 2: Strings and Lists

Session Overview

Students will continue to work with string manipulation and advanced list operations, enhancing their ability to handle data structures in Python. Students will learn to perform operations such as string reversals, substring management, and list deduplication.

The focus will be on developing skills in string interpolation, list manipulation, and the efficient use of loops and conditionals, providing a solid foundation for more complex programming challenges.

> 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
- Independently follow-along, or create an implementation, in your own IDE.

Your program leader/s will indicate which code sharing tool/s to use as a group, and will help break down

🔎 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 2

💡 **Unit 3 Cheatsheet**

To jumpstart your journey into Python strings, we've put together a guide to common functions and syntax you will use throughout Unit 3 breakout problems. Use this cheatsheet as a quick reference guide as you work through the problems below.

▼ **Problem Set Version 1**

Problem 1: Sum of Strings

Write a function `sum_of_number_strings()` that takes in a list of strings `nums`. Each string is a representations of integers. The function should return the sum of these strings as an integer.

```
def sum_of_number_strings(nums):
    pass
```

Example Usage:

```
nums = ["10", "20", "30"]
sum = sum_of_number_strings(nums)
print(sum)
```

Example Output: `60`

Problem 2: Remove Duplicates

Write a function `remove_duplicates()` that takes in a *sorted* list of integers `nums` as a parameter and removes all duplicates in the list. The function returns the modified list.

```
def remove_duplicates(nums):
    pass
```

Example Input: `nums = [1,1,1,2,3,4,4,5,6,6]`
Example Output: `no_dups = [1,2,3,4,5,6]`

▶ ✨ **AI Hint: While Loops**

▶ 💡 **Hint: While Loops (more detail)**

Problem 3: Reverse Letters

Write a function `reverse_only_letters()` that takes in a string `s` as a parameter. The function reverses the order of the letters in the string and returns the new string. Non-letter characters should remain in their original positions.

```
def reverse_only_letters(s):
    pass
```

Example Usage:

```
s = "a-bC-dEf-ghIj"
reversed_s = reverse_only_letters(s)
print(reversed_s)
```

Example Output: `j-Ih-gfE-dCba`

Problem 4: Longest Uniform Substring

Write a function `longest_uniform_substring()` that takes in a string `s` and returns the length of the longest uniform substring. A uniform substring consists of a single repeated character.

```
def longest_uniform_substring(s):
    pass
```

Example Usage:

```
s1 = "aabbbbCdAA"
```

```
s2 = "abcdef"
l2 = longest_uniform_substring(s2)
print(l2)
```

Example Output:

```
4
1
```

Problem 5: Teemo's Attack

In the game **League of Legends**, Teemo attacks his enemy Ashe with poison arrows. Write a function `find_poisoned_duration()` that takes in two parameters: `time_series` (*the time at which Teemo's attacks hits Ashe*) and `time_duration` (*the duration of the poisoning effect*). The function returns the total time that Ashe is in a poisoned condition.

`time_series` is a list of integers that represents the times at which Teemo attacks and makes Ashe poisoned for the exact `time_duration`.

If Teemo hits Ashe while she is still poisoned, the poison's duration starts over. For example, if Teemo attacks at times `1` and `4` for 3 seconds, the states at each time would be:

```
1: attacked
2: in poison state
3: in poison state
4: attacked, poison duration resets to 3
5: in poison state
6: in poison state
7: in poison state
8: in normal state
```

This means that the total time that Ashe is in a poisoned condition is 5.

```
def find_poisoned_duration(time_series, duration):
    pass
```

Example Usage:

```
time_series = [1,4,9]
damage = find_poisoned_duration(time_series, 3)
print(damage)
```

Example Output: `8`

Problem 6: Sum Unique Elements

Write a function `sum_of_unique_elements()` that takes in two lists of integers, `lst1` and `lst2`, as parameters and returns the sum of the elements that are unique in `lst1`

- it appears exactly once in `lst1`
- it does not appear in `lst2`

```
def sum_of_unique_elements(lst1, lst2):
        pass
```

Example Usage:

```
lstA = [1, 2 ,3, 4]
lstB = [3, 4, 5, 6]
lstC = [7, 7, 7, 7]

sum1 = sum_of_unique_elements(lstA, lstB)
print(sum1)

sum2 = sum_of_unique_elements(lstC, lstB)
print(sum2)
```

Example Output

```
3
0
```

Close Section

▸ Problem Set Version 2

▸ Problem Set Version 3