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: Strings and Lists

Session Overview

Students will explore complex string operations like determining if sentences are pangrams, compressing strings, and converting Roman numerals to integers. By engaging with these problems, students will deepen their understanding of string interpolation, slicing, and looping, essential skills for effective string manipulation and broader programming tasks.

> 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 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 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: Calling Mississippi

Copy and paste the following function:

```
def count_mississippi(limit):
    for num in range(1, limit):
        print( f"{num} mississippi")
```

Call the function so that it prints out the following to the console (*without calling the function more than once*):

```
1 mississippi
2 mississippi
3 mississippi
4 mississippi
```

▶ ✨ AI Hint: String Interpolation

Problem 2: Swap Ends

Write a function `swap_ends()` that accepts a string `my_str` as a parameter and returns a new string where the first and last characters from `my_str` are swapped.

```
def swap_ends(my_str):
    pass
```

Example Usage:

```
my_str = "boat"
swapped = swap_ends(my_str)
print(swapped)
```

Example Output: `toab`

▶ ✨ AI Hint: String Indexing

▶ 💡 **What's the difference between strings and lists?**

Problem 3: Is Pangram

Write a function `is_pangram()` that takes in a string `my_str` as a parameter and returns `True` if the string is a **pangram** and `False` if not. A **pangram** is a sentence containing every letter in the English alphabet.

```
def is_pangram(my_str):
    pass
```

Example Usage:

```
my_str = "The quick brown fox jumps over the lazy dog"
print(is_pangram(my_str))

str2 = "The dog jumped"
print(is_pangram(str2))
```

Example Output:

```
True
```

▶ 💡 **Hint: Capitalization**

▶ ✨ **AI Hint: String Looping**

Problem 4: Reverse String

Write a function `reverse_string()` that takes a string `my_str` as a parameter and returns the string reversed.

```
def reverse_string(my_str):
    pass
```

Example Usage:

```
my_str = "live"
print(reverse_string(my_str))
```

Example Output: `evil`

Problem 5: First Unique

Write a function `first_unique_char()` that given a string `my_str` as a parameter, it finds the first non-repeating character in it and returns its index. If it does not exist, then return `-1`.

```
def first_unique_char(my_str):
    pass
```

Example Usage:

```
my_str = "leetcode"
print(first_unique_char(my_str))

str2 = "loveleetcode"
print(first_unique_char(str2))

str3 = "aabb"
print(first_unique_char(str3))
```

Example Output

```
0
2
-1
```

Problem 6: Minimum Distance

Write a function `min_distance()` that takes in a list of strings `words` and two strings `word1` and `word2'` as parameters. The function should return the minimum distance between `word1` and `word2` in the list of words. The distance between one word and an adjacent word in the list is 1.

```
def min_distance(words, word1, word2):
    pass
```

Example Usage:

```
words = ["the", "quick", "brown", "fox", "jumped", "the"]
dist1 = min_distance(words, "quick", "jumped")
dist2 = min_distance(words, "the", "jumped")
print(dist1)
print(dist2)

words2 = ["code", "path", "code", "contribute",  "practice"]
dist3 = min_distance(words2, "code", "practice")
print(dist3)
```

Example Output:

```
3
1
2
```

Close Section

▸ Problem Set Version 2

▸ Problem Set Version 3