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: Lists Continued

Session Overview

Students will delve deeper into Python programming by tackling a diverse array of problems that enhance their understanding of list operations, control structures, and basic algorithmic thinking.

The problems range from manipulating lists by printing, doubling, or flipping signs of their elements to more complex tasks such as finding the maximum difference within a list, counting items below a certain threshold, and identifying even numbers.

> 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 2

💡 Unit 1 Cheatsheet

To jumpstart your journey into Python fundamentals and lists, we've put together a guide to common Python functions and syntax you will use throughout Unit 1 breakout problems. Use this cheatsheet as a quick reference guide as you work through the problems below.

▼ Problem Set Version 1

Problem 1: Print List

Write a function `print_list()` that takes in a list `lst` as a parameter and prints out each item in the list.

```
def print_list(lst):
    pass
```

Example Input: `lst = ["squirtle", "gengar", "charizard", "pikachu"]`

```
squirtle
gengar
charizard
pikachu
```

Problem 2: Print Doubled List Items

Write a function `doubled()` that takes in a list of integers `lst` as a parameter and prints each item in the list multiplied by two.

```
def doubled(lst):
    pass
```

Example Input: `lst = [1,2,3]`
Example Output:

```
2
4
6
```

Problem 3: Return Doubled List

Modify the function `doubled()` so that instead of printing the items, it returns a new list of the doubled numbers.

Example Usage:

```
lst = [1,2,3]
new_lst = doubled(lst)
print(new_lst)
```

Example Output:

```
[2, 4, 6]
```

Problem 4: Flip Signs

Write a function `flip_sign()` that takes in a list of integers `lst` as a parameter and returns a new list where each number in the original list has been multiplied by -1.

```
def flip_sign(lst):
    pass
```

Example Usage:

```
lst = [1,-2,-3,4]
flipped_lst = flip_sign(lst)
print(flipped_lst)
```

Example Output:

```
[-1, 2, 3, -4]
```

Problem 5: Max Difference

Write a function `max_difference()` that takes in a list of integers `lst` and returns the difference between the smallest and largest value in the list.

```
def max_difference(lst):
    pass
```

Example Usage:

```
lst = [5,22,8,10,2]
max_diff = max_difference(lst)
print(max_diff)
```

Example Output: `20`

Problem 6: Below Threshold

Write a function `count_less_than()` that takes in a list of integers `numbers` and an integer `threshold` as parameters and returns the number of items in `numbers` that are less than `threshold`.

```
def count_less_than(numbers, threshold):
    pass
```

Example Usage:

```
numbers = [12,8,2,4,4,10]
counter = count_less_than(numbers,5)
print(counter)
```

Example Output: `3`

Problem 7: Evens List

Write a function `get_evens()` that takes in a list of integers `lst` as a parameter and returns a list of all even numbers in the list.

```
def get_evens(lst):
    pass
```

Example Usage:

```
lst = [1,2,3,4]
evens_lst = get_evens(lst)
print(evens_lst)
```

Example Output:

```
[2, 4]
```

▶ 💡 **Remainders with Modulus Division**

Problem 8: Multiples of Five

Write a function `multiples_of_five()` that prints out multiples of 5 between 1 and 100 (inclusive).

```
def multiples_of_five():
    pass
```

Example Output:

```
5
10
15
20
25
....
90
95
100
```

▶ ✨ **AI Hint:** `range()`

Problem 9: Divisors

Write a function `find_divisors()` that takes in an integer `n` as a parameter that returns a list of all divisors of `n`.

```
def find_divisors(n):
    pass
```

```
lst = find_divisors(6)
print(lst)
```

Example Output:

```
[1, 2, 3, 6]
```

Problem 10: FizzBuzz

Write a function `fizzbuzz()` that takes in an integer `n` as a parameter and prints the numbers from 1 to `n`.
For multiples of 3, print `"Fizz"` instead of the number.
For multiples of 5, print `"Buzz"` instead of the number.

```
def fizzbuzz(n):
    pass
```

Example Usage: `fizzbuzz(13)`

Example Output:

```
1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
```

Problem 11: Print the Index

Write a function `print_indices()` that takes in an integer list `lst` as a parameter and prints out the index of each item in the given list.
Use the function `range()` to loop through the list indices.

```
def print_indices(lst):
    pass
```

Example Usage:

Example Output:

```
0
1
2
3
4
```

Problem 12: Linear Search

Write a function `linear_search()` that takes in a list `lst` and value `target` as parameters. The function returns the index of `target` in `lst` if found. If `target` is not found in `lst`, return `-1`.

```
def linear_search(lst, target):
    pass
```

Example Usage:

```
lst = [1,4,5,2,8]
position = linear_search(lst,5)
print(position)
```

Example Output: `2`

Example Usage:

```
lst = [1,4,5,2,8]
position = linear_search(lst,10)
print(position)
```

Example Output: `-1`

Close Section

▸ Problem Set Version 2