# 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**

Getting Started

Learning with AI ✨

IDE Setup

HackerRank Guide

Schedule

Course Progress

Unit 1

Unit 2

Unit 3

Unit 4

Unit 5

Unit 6

Unit 7

Unit 8

Unit 9

Unit 10

nission Guide

# Session 1: Python and Lists

## Session Overview

Students will be introduced to basic Python programming concepts necessary for technical interviews. The curriculum covers writing and calling functions, variable manipulation, and list operations through a series of problem sets.

These exercises include practical tasks such as organizing code snippets, performing mathematical operations, and implementing simple algorithms. By solving problems ranging from creating a lunch menu to playing a simplified version of Blackjack, students will gain familiarity with Python syntax, function definitions, and control structures.

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

## 📈 Part 1 : Instructor Led 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.

▶ **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 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: Hello World!

Given the following lines of code, work with your group members to place the lines in order and write and call your first Python function!

a. `print("Hello world!")`
b. `def hello_world():`
c. `hello_world()`

▶ 💡 **Hint: Python Functions**

## Problem 2: Today's Mood

The following function uses a variable, `mood` to print out `"Today's mood: 😎"`. Copy this code into your IDE and update the `mood` variable to print out your mood for today.

```
def todays_mood():
    mood = "😎"
    print("Today's mood: " + mood)

todays_mood()
```

Example Output: `Today's mood: 😎`

▶ 💡 **Hint: Variables**

## Problem 3: Lunch Menu

The following function accepts one parameter `menu`. Copy this code into your IDE and add a function call so that `"Lunch today is: 🍕"` is printed to the console.

```
def print_menu(menu):
    print("Lunch today is: " + menu)
```

Example Output: `Lunch today is: 🍕`

▶ 💡 **Hint: Parameters**

▶ ✨ **AI Hint: Parameters, Arguments, and Variables Oh My!**

## Problem 4: Sum of Two Integers

The following function returns the sum of two integers: `a` and `b`.

```
def sum(a, b):
    return a + b
```

Use the `sum()` function to calculate the sum of `13` and `27`. Then, use the `sum()` function again to **double** the calculated sum and print the result to the console.

*Note: Do not use any mathematical operators such as `+`, `-`, `*`, or `/` when solving this problem.*

## Problem 5: Product of Two Integers

Write a function `product()` that returns the product of two integers, `a` and `b`.

Example Input: `22` and `7`
Example Result: `154`

▶ ✨ AI Hint: To Print or to Return?

▶ 💡 **Hint: Print vs. Return Examples**

## Problem 6: Classify Age

Write a function `classify_age()` that takes an integer `age` as a parameter and returns `"child"` if the age is less than 18, and returns `"adult"` otherwise.

Example Usage:

```
output = classify_age(18)
print(output)
output = classify_age(7)
print(output)
output = classify_age(50)
print(output)
```

Output:

```
adult
child
adult
```

▶ ✨ AI Hint: Conditionals

## Problem 7: What time is it?

**Let's put what we learned in Problems 1-4 all together!** Write a function named `what_time_is_it()` that takes an integer `hour` as a parameter.
If `hour` is 2, the function should return `"taco time 🌮"`.

Example Usage:

```
time = what_time_is_it(2)
print(time)
time = what_time_is_it(7)
print(time)
time = what_time_is_it(12)
print(time)
```

Output:

```
taco time 🌮
nap time 😴
peanut butter jelly time 🥪
```

## Problem 8: Blackjack

In the game Blackjack, players try to draw a hand of cards that totals as close to 21 as possible. Players "bust" if their cards total more than 21. Players say "Hit me!" if they want the dealer to give them another card.

Write a function called `blackjack()` that takes an integer `score` as a parameter.
If `score` equals 21, print `"Blackjack!"`.
If `score` is greater than 21, print `"Bust!"`.
If `score` is greater than or equal to 17 and less than 21, print `"Nice hand!"`.
If `score` is less than 17, print `"Hit me!"`.

Example Usage:

```
blackjack(21)
blackjack(24)
blackjack(19)
blackjack(10)
```

Output:

```
Blackjack!
Bust!
Nice hand!
Hit me!
```

## Problem 9: First Item

Write a function `get_first()` that takes in a list as a parameter and returns the first item in the list. Return `None` if the list is empty.

Example Input: `[3,1,6,7,5]`
Example Output: `3`

*Note: `pass` is a keyword that is used as a placeholder for future code*

▶ ✨ **AI Hint: List indexing**

## Problem 10: Last Item

Write a function `get_last()` that takes in a list as a parameter and returns the last item in the list. Return `None` if the list is empty.

```
def get_last(lst):
    pass
```

Example Input: `[3,1,6,7,5]`
Example Output: `5`

▶ ✨ **AI Hint: Getting the length of a list**

## Problem 11: Counter

Write a function `counter()` that uses the `range` function to print numbers between 1 and a given `stop` value (inclusive).

```
def counter(stop):
    pass
```

Example Usage: `counter(7)`. Example Output:

```
1
2
3
4
5
6
7
```

▶ ✨ **AI Hint: For Loops**

## Problem 12: Sum of 1 to 10

Write a function `sum_ten()` that returns the sum of numbers from 1 to 10.

```
def sum_ten():
    pass
```

Example Usage: `output = sum_ten()`
Example Result: `output = 55`

▶ ✨ AI Hint: Accumulator Variable

## Problem 13: Total Sum

Write a function `sum_positive_range()` that returns the sum of numbers from 1 to a given `stop` value (inclusive).

```
def sum_positive_range(stop):
    pass
```

Example Usage: `sum = sum_positive_range(6)`
Example Result: `sum = 21`

## Problem 14: Total Sum in Range

Write a function `sum_range()` that returns the sum of numbers from a given `start` value to a given `stop` value (inclusive).

```
def sum_range(start, stop):
    pass
```

Example Usage: `sum = sum_range(3, 9)`
Example Result: `sum = 42`

## Problem 15: Negative Numbers

Write a function `print_negatives()` that takes a list of integers `lst` and prints all negative numbers in the list.

Example Usage: `print_negatives([3,-2,2,1,-5])`

Example Output:

```
-2
-5
```

Example Usage: `print_negatives([1,2,3,4,5])`

Example Output:

```
None
```

Close Section

## ▸ Problem Set Version 2