Chapter 5: STRUCTURED TYPES AND MUTABILITY

Python challenges

CS 101 - Fall 2025

Putting It All Together: The Grand Finale!

Summary of Our Python Journey

- **Tuples**: Immutable, ordered data
- **Ranges**: Efficient number sequences
- **Lists**: Flexible, mutable collections
- **List Cloning**: Creating independent copies
- **List Comprehensions**: Elegant one-line creation
- **Nested Lists**: Organizing hierarchical data
- **2D Lists**: Grid-based data structures
- **Higher-Order Operations**: Functional programming tools
- **Sequence Types**: Common operations across types
- **Sets**: Unique collections with math operations

Now you're equipped with Python's most powerful data structures!

Practice Time!

Your Turn to Shine!

Let's practice with focused, manageable exercises! Each challenge takes 5-10 minutes and focuses on one key concept.
Choose your challenge level: - **Basic**: Single concept practice - **Intermediate**: Combine 2-3 concepts - **Advanced**: Combine multiple concepts
Remember: Start with basic challenges and work your way up!

Challenge 1: Tuple Practice

Basic Tuple Operations (5 minutes)

Practice creating and using tuples for storing related data.

Task

```
# Given data
student1 = ("Alice", 85, "Computer Science")
student2 = ("Bob", 92, "Mathematics")
student3 = ("Carol", 78, "Physics")

# Your tasks (complete each line):
# 1. Create a tuple with all three students
all_students = # Your code here

# 2. Get Alice's grade (second element of first tuple)
alice_grade = # Your code here

# 3. Get all student names in a list
names = # Your code here

# 4. Count how many students have grades above 80
high_grades = # Your code here
```

Challenge 2: List Comprehension Magic

List Comprehensions (7 minutes)

Create lists efficiently using comprehension syntax.

Task

```
# Given data
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
words = ["apple", "banana", "cherry", "date", "elderberry"]

# Your tasks:
# 1. Create a list of squares for even numbers only
even_squares = # Your code here

# 2. Create a list of words with more than 5 letters, in uppercase
long_words_upper = # Your code here

# 3. Create a list of numbers from 1-20 that are divisible by 3
divisible_by_3 = # Your code here

# 4. Create a list with first letter of each word
first_letters = # Your code here
```

Challenge 3: Set Operations

Working with Sets (5 minutes)

Practice set operations for finding unique elements and relationships.

Task

```
# Given data
class_a = {"Alice", "Bob", "Charlie", "David", "Eve"}
class_b = {"Charlie", "David", "Frank", "Grace", "Alice"}
grades = [85, 92, 78, 85, 91, 78, 88, 85, 92, 78]

# Your tasks:
# 1. Find students in both classes
both_classes = # Your code here

# 2. Find students only in class A
only_class_a = # Your code here

# 3. Find all unique grades
unique_grades = # Your code here

# 4. Find total number of unique students
total_students = # Your code here
```

Challenge 4: 2D List Basics

Working with 2D Lists (8 minutes)

Practice creating and manipulating 2D lists for grid-based data.

Task: Complete the following code

```
# Create a 3x3 tic-tac-toe board
board = [
    [' ', ' ', ' '],
    [' ', ' ', ' '],
    [' ', ' ', ' ']
]

# Your tasks:
# 1. Place 'X' in the center (row 1, column 1)
# Your code here
```

```
# 2. Place 'O' in top-left corner (row 0, column 0)
# Your code here

# 3. Create a list of all positions that are empty (' ')
empty_positions = # Your code here

# 4. Check if the center row has any 'X' in it
center_has_x = # Your code here
```

Challenge 5: List Cloning Practice

Safe List Operations (6 minutes)

Practice cloning lists to avoid unwanted side effects.

Task

```
# Original shopping list
original_list = ["apples", "bananas", "oranges"]

# Your tasks:
# 1. Create a proper copy of the original list
shopping_copy = # Your code here

# 2. Add "grapes" to the copy (original should remain unchanged)
# Your code here

# 3. Create another copy and remove "bananas" from it
fruit_copy = # Your code here
# Your code here

# 4. Verify original list is unchanged
print(f"Original: {original_list}")  # Should be ["apples", "bananas", "oranges"]
```

Challenge 6: Range and Map Practice

Higher-Order Functions (8 minutes)

Practice using ranges with map() and filter() functions.

Task

```
# Given data
prices = [19.99, 25.50, 12.75, 8.99, 45.00, 15.25]

# Your tasks:
# 1. Use range to create a list of numbers 0-9
numbers = # Your code here

# 2. Use map() to add tax (8%) to all prices
prices_with_tax = # Your code here

# 3. Use filter() to find prices under $20
affordable_prices = # Your code here

# 4. Use map() to convert all prices to integers (rounded down)
rounded_prices = # Your code here
```

Challenge 7: Combining Concepts

Integration Challenge (10 minutes)

Combine multiple concepts in a mini student database.

Task: Complete the following code

```
# Student data: (name, age, grades_list)
students = [
    ("Alice", 20, [85, 92, 78]),
    ("Bob", 19, [90, 88, 95]),
    ("Carol", 21, [76, 82, 85])
]
```

```
# Your tasks:
# 1. Create a set of all unique ages
unique_ages = # Your code here

# 2. Use list comprehension to get all student names
names = # Your code here

# 3. Create a list of average grades for each student
averages = # Your code here

# 4. Find students with average grade above 85
high_performers = # Your code here

# 5. Clone the students list and add a new student
students_copy = # Your code here
# Add ("David", 22, [88, 91, 87]) to the copy
```

Quick Check Solutions

Test Your Understanding

Before moving on, make sure you can explain:

- **Why** we use tuples vs lists
- **How** list comprehensions make code cleaner
- **When** to clone lists vs reference them
- **What** makes sets useful for data analysis
- **How** 2D lists represent grid data

Hint: If you can teach it to someone else, you've mastered it!

Great job practicing! These building blocks will serve you well!