Python Institute: Comprehensive Python Exercise Sheet

Disclaimer

This exercise sheet is provided by the **Python Institute** for educational purposes. It is designed to help you practice and master the fundamental concepts of Python programming.

Instructions

1. This exercise is to be completed **individually**. Collaboration is not permitted.
2. Save your solutions in a Python file (.py). Name the file as: YourName_PythonComprehensiveExercises.py
3. Submit the completed file to your instructor by the given deadline.
4. **Use comments** in your code to explain the purpose and logic of your solutions.
5. Your code should be clean, indented properly, and free of syntax errors.
6. Avoid using any external help or automated tools to solve the exercises. They are designed to challenge and build your skills.

Part 1: Basics

1. Python Syntax

- Write a Python script that:
    1. Prints the message: "Welcome to Python Programming!".
    2. Creates two variables: course with the value "Python Fundamentals" and hours with the value 4.
    3. Prints: "You are enrolled in <course> for <hours> hours."
    4. Includes a comment explaining each line of your code.

---

2. Python Comments

Scenario: You are working on a Python project with a team. To make your code readable, you need to use comments.

- Write a Python script that includes:
    - A **single-line comment** describing the purpose of the script.
    - A **multi-line comment** listing three benefits of commenting in code.

- Your script should also calculate and print the product of two numbers.

---

3. Python Variables

Scenario: You are creating a simple student database.

- Write a Python script to:
    1. Create variables to store a student's:
        - Name (string)
        - Age (integer)
        - GPA (float)
        - Is enrolled (boolean)
    2. Print each variable with a descriptive message.
    3. Use comments to document why you chose specific data types for each variable.

Part 2: Data Types and Operations

4. Python Data Types

Scenario: You are building a system to process survey data.

- Write a script to:
    1. Create variables for:
        - Total participants (integer)
        - Average age (float)
        - Survey feedback (string: "Great session!")
        - Participant IDs (list: [101, 102, 103])
    2. Print the data type of each variable using the type() function.
    3. Use comments to explain why each data type is appropriate.

---

5. Python Numbers and Casting

- Create a Python script that:
    1. Asks the user to enter two numbers.

2. Calculates and prints:
    - Sum
    - Difference
    - Product
    - Quotient
3. Cast one number to a float and one to an integer and print their types.

---

6. Python Strings

Scenario: You are building a program for a library to process book titles.

- Write a Python script that:
    1. Stores the book title "Harry Potter and the Philosopher's Stone" in a variable.
    2. Prints the title:
        - In uppercase
        - In lowercase
        - In title case
    3. Extracts and prints:
        - The first 5 characters
        - The last 5 characters
    4. Replaces the word "Philosopher's" with "Sorcerer's" and prints the updated title.

Part 3: Logical Structures

7. Python Booleans and Operators

Scenario: You are developing an attendance system.

- Write a Python script that:
    1. Assigns boolean variables is_present = True and has_completed_homework = False.
    2. Uses logical operators to:

        - Check if the student is eligible for a bonus (both conditions must be true).
        - Check if they attended class OR completed the homework.
    3. Prints appropriate messages based on the results.

---

8. Python If...Else

Scenario: You are writing software to assess exam grades.

- Write a Python program that:
    1. Asks the user to input their exam score (out of 100).
    2. Checks if the score is:
        - Greater than or equal to 90: Print "Excellent"
        - Between 70 and 89: Print "Good"
        - Below 70: Print "Needs Improvement"

Part 4: Data Structures

9. Python Lists

Scenario: You are designing a food delivery app.

- Write a Python script to:
    1. Create a list of 5 dishes offered in your app.
    2. Add a new dish to the list.
    3. Remove the second dish from the list.
    4. Print the updated menu and the number of dishes.

---

10. Python Tuples

Scenario: You are maintaining a database of top-performing employees.

- Write a script to:
    1. Create a tuple containing the names of 3 employees.
    2. Try adding another name to the tuple. (Explain what happens in a comment.)

3. Print the second employee’s name.

---

11. Python Sets

Scenario: You are analyzing customer preferences.

- Write a Python script to:
    1. Create two sets:
        - set1 = {'pizza', 'burger', 'pasta'}
        - set2 = {'burger', 'sushi', 'pasta'}
    2. Print:
        - All unique dishes (union of the sets).
        - Common dishes (intersection).
        - Dishes in set1 but not in set2.

---

12. Python Dictionaries

Scenario: You are creating a contact management system.

- Write a script to:
    1. Create a dictionary with keys: "Name", "Phone", and "Email".
    2. Add a new key "Address" with a sample value.
    3. Update the phone number.
    4. Print all keys and their corresponding values.

Part 5: Loops and Functions

13. Python While Loops

Scenario: You are automating a countdown timer.

- Write a Python script that:
    1. Prints numbers from 10 to 1 in descending order using a while loop.
    2. Prints "Blast off!" when the loop ends.

---

14. Python For Loops

Scenario: You are building a system to analyze customer purchases.

- Write a script that:
    1. Iterates over the list [200, 300, 150, 400, 100] representing purchase amounts.
    2. Prints:
        - Each purchase amount.
        - The total amount spent (use a loop to calculate this).

---

15. Python Functions

Scenario: You are building a simple calculator.

- Write a function called calculate() that:
    1. Accepts three parameters: two numbers and an operator (+, -, *, /).
    2. Performs the operation and returns the result.
    3. Call the function with different inputs to test it.

Bonus Challenge

Scenario: You are designing a simple text-based game.

1. Create a random number between 1 and 100 using random.randint().
2. Ask the user to guess the number.
3. Provide feedback for each guess:
    - "Too high!" if the guess is greater than the number.
    - "Too low!" if the guess is less than the number.
4. Continue until the user guesses the number correctly and print "You win!".