Part 1: Variables and Memory Allocation

Exercise 1.1: Memory Exploration

Write a program that demonstrates how variables and memory allocation work in Python. Your program should:

1. Create variables of different types (int, float, string, list, tuple).
2. Print the ID (memory address) of each variable using the `id()` function.
3. Create a second variable that references the same object as one of your first variables.
4. Modify the original variable and observe what happens to the second one (for both mutable and immutable types).
5. Include comments explaining the behavior you observe.

Exercise 1.2: Variable Scope Investigation

Create a function that demonstrates variable scope in Python:

1. Define global variables outside the function.
2. Define local variables inside the function with the same names.
3. Try to modify a global variable both with and without the `global` keyword.
4. Print the IDs of all variables before and after modifications.
5. Explain what happens and why in your comments.

Part 2: Data Types and Type Conversion

Exercise 2.1: Type Exploration

Create a program that:

1. Creates at least one variable of each of these types: int, float, complex, bool, str, and None.
2. Uses the `type()` function to verify the type of each variable.
3. Uses `isinstance()` to check if variables are of specific types.
4. Demonstrates at least three examples where Python automatically converts types in expressions.
5. Includes comments documenting your observations about type behavior.

Exercise 2.2: Type Conversion Challenge

Write a function that:

1. Takes a string input containing a mixture of numbers and text (e.g., "I am 25 years old and my height is 5.9 feet").

2. Extracts all numbers from the string and converts them to their appropriate numeric types (int or float).
3. Returns a tuple containing two lists: one with all integers found and one with all floats found.
4. Handles potential conversion errors gracefully.

Part 3: Number Systems and Representation

Exercise 3.1: Base Converter

Create a set of functions that:

1. Converts decimal integers to binary, octal, and hexadecimal strings without using built-in functions like `bin()`, `oct()`, or `hex()`.
2. Converts binary, octal, and hexadecimal strings to decimal integers without using `int(x, base)`.
3. Demonstrates your functions with at least five different numbers.
4. Verifies your results by comparing with Python's built-in conversion functions.

Part 4: Floating-Point Precision

Exercise 4.1: Precision Problems

Write a program that demonstrates floating-point precision issues:

1. Create at least five examples where floating-point arithmetic gives unexpected results.
2. For each example, explain why the result occurs.
3. Implement a solution to each problem using at least two different approaches (rounding, epsilon comparison, Decimal, Fraction, etc.).
4. Compare the accuracy and performance of each solution.

Part 5: Deep Dive Modules

Exercise 5.1: Math Module Explorer

Write a program that explores the capabilities of the `math` module:

1. Use at least 10 different functions from the `math` module.
2. Create practical examples for each function.
3. Create a function that calculates the roots of a quadratic equation using `math` functions.
4. Create a function that converts between different angle units (degrees, radians) using `math` functions.
5. Include comments explaining each function and its application.