Phase 1: The Absolute Basics (Foundation)

This phase is your entry point into the world of programming. It's all about learning the fundamental rules and tools to make your computer do simple tasks.

- **Introduction to Python:**
  - **What it is:** Python is a popular, high-level programming language known for its simplicity and readability. It's used in web development, data science, AI, automation, and more.
  - **Why learn it:** Easy to pick up, versatile, large community, and lots of available libraries.
  - **Getting started:** Involves installing Python and setting up a code editor (like VS Code) where you'll write and run your first programs.
- **Variables and Data Types:**
  - **Variables:** Think of these as named containers or labels that hold different pieces of information in your program. You give them a name, and they store a value.
  - **Data Types:** This defines *what kind* of information a variable holds.
    - **Integers (`int`):** Whole numbers (e.g., `5`, `-100`).
    - **Floats (`float`):** Numbers with decimal points (e.g., `3.14`, `0.5`).
    - **Strings (`str`):** Text, enclosed in single or double quotes (e.g., `"Hello"`, `'Python'`).
    - **Booleans (`bool`):** Represents truth values: `True` or `False`.
    - **NoneType (`None`):** Represents the absence of a value.
- **Operators:**
  - These are symbols that perform operations on values and variables.
  - **Arithmetic:** `+` (add), `-` (subtract), `*` (multiply), `/` (float division), `//` (integer division), `%` (remainder/modulo), `**` (exponent).
  - **Comparison:** `==` (equal to), `!=` (not equal to), `>` (greater than), `<` (less than), `>=` (greater than or equal to), `<=` (less than or equal to). These always result in `True` or `False`.
  - **Logical:** `and`, `or`, `not`. Used to combine or negate boolean conditions.
  - **Assignment:** `=` (assign), `+=` (add and assign), `-=` (subtract and assign), etc.
- **Input and Output:**
  - **`print()`:** The primary way to display information (text, variable values, results) to the user on the console. You can format output using f-strings for clarity.
  - **`input()`:** The primary way to get information *from* the user. It pauses your program and waits for the user to type something and press Enter. The input is always read as a string.
- **Conditional Statements (`if-elif-else`):**
  - These allow your program to make decisions and execute different blocks of code based on whether certain conditions are true or false.
  - **`if`:** Executes code if its condition is `True`.
  - **`elif` (else if):** Checks another condition if the preceding `if` (or `elif`) condition was `False`. You can have multiple `elif`s.

    - **`else` :** Executes code if none of the preceding `if` or `elif` conditions were `True` . It's the "catch-all."
- **Loops (`for` , `while`):**
    - Loops are used to repeat a block of code multiple times.
    - **`for` loop:** Ideal for iterating over a sequence (like a list of items, or a range of numbers) where you know how many times you want to repeat, or you want to process each item in a collection.
    - **`while` loop:** Repeats a block of code as long as a specified condition remains `True` . It's useful when you don't know in advance how many times the loop needs to run.
    - **`break` :** Immediately stops the entire loop and continues execution at the first line of code *after* the loop.
    - **`continue` :** Skips the rest of the current iteration of the loop and moves directly to the next iteration.
- **Data Structures (Part 1: Basic - Lists & Tuples):**
    - These are ways to store and organize collections of data.
    - **Lists (`[]`):** Ordered, mutable (changeable) collections. You can add, remove, or modify elements after the list is created. They are very flexible.
    - **Tuples (`()`):** Ordered, immutable (unchangeable) collections. Once a tuple is created, you cannot modify its elements. They are often used for fixed collections of related items (like coordinates).