Introduction to Python Basics

Python is a high-level, interpreted, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant indentation. It is widely used in web development (server-side), software development, mathematics, and system scripting.-----**1. Getting Started: Installation and Setup**

To begin using Python, you need to:

- **Download and Install:** Download the latest version of Python from the official website. The installer often includes `pip`, the package installer for Python.
- **Verify Installation:** Open your terminal or command prompt and type `python --version` (or `python3 --version`).
- **The Interpreter:** Python code can be executed line-by-line in an interactive shell (REPL - Read-Eval-Print Loop) or by running a `.py` file.

-----2. Basic Syntax and Structure

Indentation
Python uses whitespace (spaces or tabs) to define code blocks (e.g., within functions, loops, or conditional statements) instead of using curly braces `{}`. This is mandatory and contributes to its readability.

Comments
Comments are used to explain code and are ignored by the interpreter.
Single-line comments: Start with a `#`.
# This is a comment

- print("Hello")

Multi-line comments (docstrings): While not strictly comments, triple quotes (`"""` or `'''`) are often used to create multi-line strings that can serve as documentation (docstrings) for modules, classes, and functions.
"""
This is a multi-line string
used as a docstring.

- """

-----3. Data Types

Python has several built-in data types:

- **Numbers:**
  - int: Integers (e.g., 10, -5).
  - float: Floating-point numbers (e.g., 3.14, -0.01).
  - complex: Complex numbers (e.g., 1j).
- **Boolean:** Represents truth values (True or False).

Strings (str): Sequences of characters, enclosed in single quotes ('...') or double quotes ("...").
name = "Alice"

- message = 'Python is fun'
- **Lists:** Ordered, changeable (mutable) sequences of items. Elements are enclosed in square brackets [] and can be of different data types.
  my_list = [1, "apple", 3.14, True]
- **Tuples:** Ordered, unchangeable (immutable) sequences of items. Elements are enclosed in parentheses ().
  my_tuple = (1, "banana", 3.14)
- **Dictionaries (dict):** Unordered, changeable collections of key-value pairs. Keys must be unique and immutable. Enclosed in curly braces {}.
  person = {"name": "Bob", "age": 30}
- **Sets:** Unordered, unchangeable (but you can add/remove items), and unindexed collections of unique items. Enclosed in curly braces {}.
  my_set = {"a", "b", "c"}

-----4. Variables

Variables are containers for storing data values.

- **Declaration:** Python has no explicit command for declaring a variable. A variable is created the moment you first assign a value to it.
- **Naming Rules:**
  - Must start with a letter (a-z, A-Z) or an underscore (_).
  - Can only contain alphanumeric characters and underscores (A-z, 0-9, and _).
  - Case-sensitive (age, Age, and AGE are three different variables).

Type Inference: Python automatically infers the data type based on the assigned value.
x = 5   # x is an int

- greeting = "Hello" # greeting is a str

-----5. Operators

Operators are used to perform operations on variables and values.

Arithmetic Operators:

Operator	Name	Example
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
//	Floor Division	a // b (rounds down to the nearest whole number)
%	Modulus	a % b (remainder)
**	Exponentiation	a ** b

Comparison Operators:

Operator	Name	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

Logical Operators:

Operator	Description
and	Returns True if both statements are true
or	Returns True if one of the statements is true
not	Reverses the result, returns False if the result is true

-----6. Control Flow

Control flow statements determine the order in which the program's code is executed.

Conditional Statements (if, elif, else)

Used to execute code blocks only if certain conditions are met.

```
age = 18
if age >= 18:
    print("You are an adult.")
elif age > 12:
    print("You are a teenager.")
else:
    print("You are a child.")
```

Loops (for and while)

for loop: Used for iterating over a sequence (list, tuple, dictionary, set, or string).

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
```

- print(x)

while loop: Executes a set of statements as long as a condition is true.

```
i = 1
while i < 6:
    print(i)
```

- i += 1

-----7. Functions

A function is a block of code which only runs when it is called.

- **Definition:** A function is defined using the def keyword, followed by the function name, parentheses (), and a colon :.
- **Calling:** You call a function by using its name followed by parentheses.
- **Arguments/Parameters:** Information can be passed into functions as arguments.
- **Return Value:** The return statement is used to exit a function and pass a result back to the caller.

```
def greet(name):
    """This function greets the person passed in as a parameter."""
    return f"Hello, {name}!"

message = greet("Abdullah") # Calls the function
print(message)             # Output: Hello, Abdullah!
```

-----8. Input and Output

Output (print())

The print() function is used to output data to the standard output device (usually the screen).

```
print("This is output.")
print("The answer is", 42)
```

Input (input())

The input() function allows the program to take input from the user via the console. The input is always returned as a string.

```
user_input = input("Enter your name: ")
print(f"Your name is {user_input}.")
```