Chapter 1: Python Basics

- **Topics**: Data types, variables, operators, conditionals (if, elif, else), and loops (for, while).
- **Project**: *System Health Checker* – A simple script to simulate CPU, memory, and disk health checks, printing a "healthy" or "alert" status based on threshold values.

1.1 Data Types

Data types are the kinds of values you can work with in Python.

- **Integer** (`int`): Represents whole numbers without a decimal. Examples include `10`, `-5`, and `0`.
- **Float** (`float`): Represents numbers with a decimal point, like `3.14` or `-0.001`.
- **String** (`str`): A sequence of characters, typically text, enclosed in quotes. Examples: `"Hello"`, `'SRE Script'`.
- **Boolean** (`bool`): Represents `True` or `False` values, used to control logic flow in code.

Example:

```
age = 30            # Integer
price = 19.99       # Float
name = "Alice"      # String
is_active = True    # Boolean
```

1.2 Variables

Variables act as containers for storing data values. They are created when you assign a value to a name using the `=` operator.

Rules:

- Variable names must start with a letter or an underscore, and can contain letters, numbers, and underscores.
- They are case-sensitive (`Name` and `name` are different variables).

Example:

```
username = "SRE_Engineer"
max_retries = 5
timeout = 30.5
```

In this example:

- `username` holds a string value.
- `max_retries` holds an integer.
- `timeout` holds a float.

1.3 Operators

Operators let you perform operations on variables and values.

Types of Operators:

1. **Arithmetic Operators**: Used for mathematical calculations.
   - `+` (addition), `-` (subtraction), `*` (multiplication), `/` (division), `%` (modulus), `**` (exponentiation), `//` (floor division).

   ```
   result = 10 + 5  # Result is 15
   quotient = 10 // 3  # Result is 3, discarding the remainder
   ```

2. **Comparison Operators**: Compare two values and return `True` or `False`.
   - `==` (equal), `!=` (not equal), `>`, `<`, `>=`, `<=`.

   ```
   is_equal = (10 == 10)  # True
   is_greater = (5 > 2)  # True
   ```

3. **Logical Operators**: Combine multiple conditions.
   - `and`: True if both conditions are true.
   - `or`: True if at least one condition is true.
   - `not`: Inverts the result.

   ```
   is_within_range = (5 > 2) and (5 < 10)  # True
   ```

1.4 Conditionals (If Statements)

Conditionals allow you to execute specific code blocks based on conditions.

Syntax:

```
if condition:
    # Code to execute if condition is true
elif another_condition:
    # Code to execute if the second condition is true
else:
    # Code to execute if no conditions above are true
```

Example:

```
threshold = 70
usage = 85
if usage > threshold:
    print("Alert: High usage!")
else:
    print("Usage is within limits.")
```

Here, if `usage` exceeds `threshold`, the message "Alert: High usage!" is printed. Otherwise, "Usage is within limits" is printed.

1.5 Loops

Loops allow you to repeat a set of instructions.

1. **For Loop**: Used to iterate over a sequence (like a list, tuple, or range).

   ```
   for i in range(3):  # Loops 3 times (i = 0, 1, 2)
       print("Iteration", i)
   ```

2. **While Loop**: Repeats as long as a condition is `True`.

   ```
   count = 0
   while count < 3:
       print("Count:", count)
       count += 1  # Increments count by 1 each time
   ```

Project Explanation: System Health Checker

Objective: Write a script to check system resource usage (CPU, memory, and disk) and print a warning if any resource exceeds a set threshold.

1. **Define thresholds** for each resource.
   - For this example, `cpu_threshold`, `memory_threshold`, and `disk_threshold` are set.
2. **Check each resource** against its threshold using an `if` statement.
   - If a resource usage (like `cpu_usage`) is above its threshold (`cpu_threshold`), a warning message is printed.
   - Otherwise, it prints that the usage is normal.

Code Walkthrough:

```python
# Define current usage values (simulated)
cpu_usage = 65
memory_usage = 70
disk_usage = 80

# Define threshold limits
cpu_threshold = 75
memory_threshold = 75
disk_threshold = 85

# Check CPU usage
if cpu_usage > cpu_threshold:
    print("Warning: High CPU usage!")
else:
    print("CPU usage is normal.")

# Check memory usage
if memory_usage > memory_threshold:
    print("Warning: High Memory usage!")
else:
    print("Memory usage is normal.")

# Check disk usage
if disk_usage > disk_threshold:
    print("Warning: High Disk usage!")
else:
    print("Disk usage is normal.")
```

- **Practical Application**: In real-world scenarios, we could replace the simulated values (`cpu_usage`, `memory_usage`, etc.) with actual system data from monitoring tools to automate system health checks.
