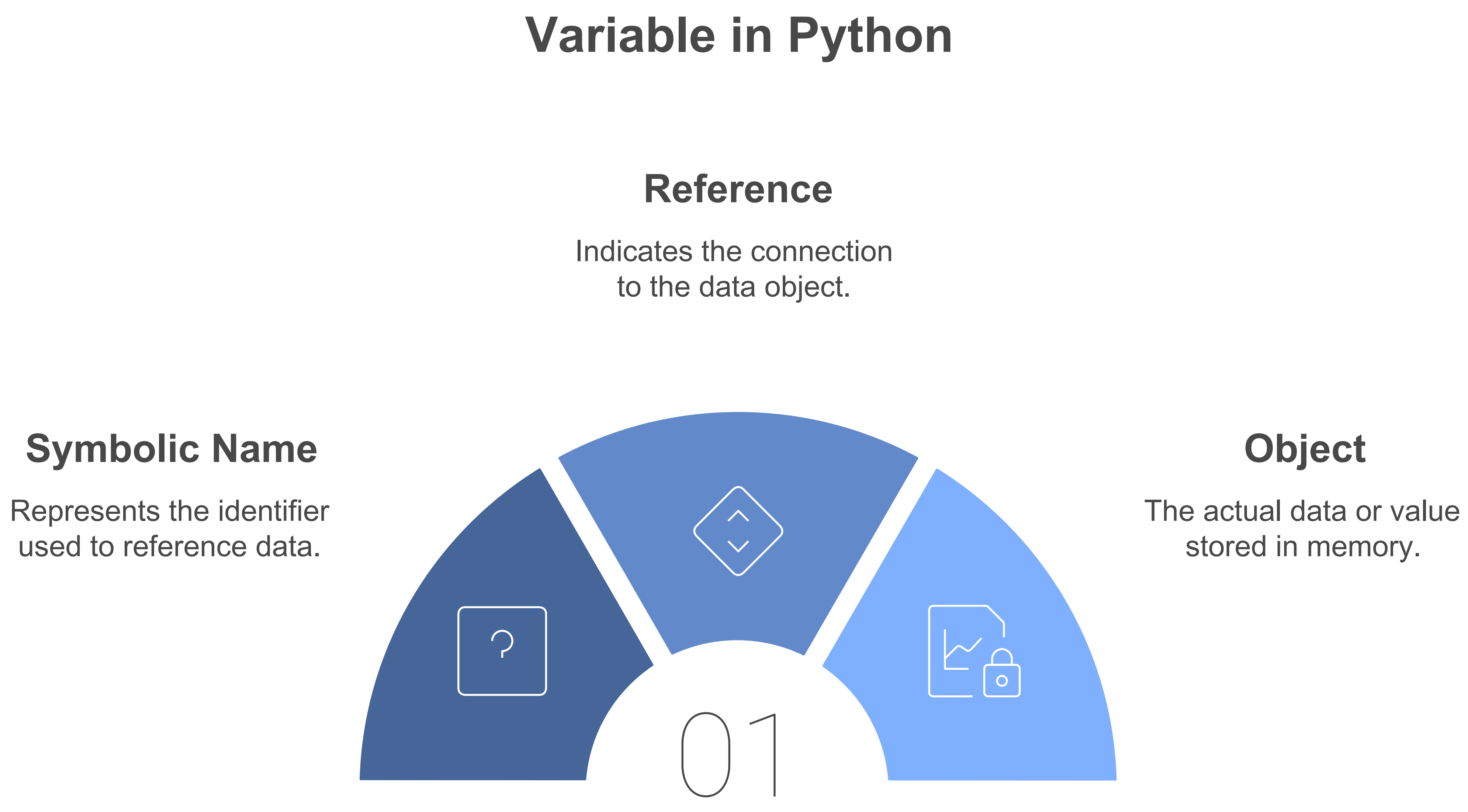
Python Variables: Types, Examples, and Rules

In this document, we will explore the concept of variables in Python, including the different types of variables, practical examples, and the rules that govern their usage. Understanding variables is fundamental to programming in Python, as they are used to store data values that can be manipulated throughout your code.

What is a Variable?

A variable in Python is a symbolic name that is a reference or pointer to an object. Variables are used to store information that can be referenced and manipulated in a program.


Variable in Python
Reference
Indicates the connection to the data object.
Symbolic Name
Represents the identifier used to reference data.
Object
The actual data or value stored in memory.
01

Variable Types

Python supports various data types, and variables can hold values of any of these types. Here are some of the most common variable types in Python:

1. **Integer**: Whole numbers, both positive and negative.

```
age = 25
```

2. **Float**: Numbers with a decimal point.

```
height = 5.9
```

3. **String**: A sequence of characters enclosed in quotes.

```
name = "Alice"
```

4. **Boolean**: Represents one of two values: **True** or **False**.

```
is_student = True
```

5. **List**: An ordered collection of items, which can be of mixed types.

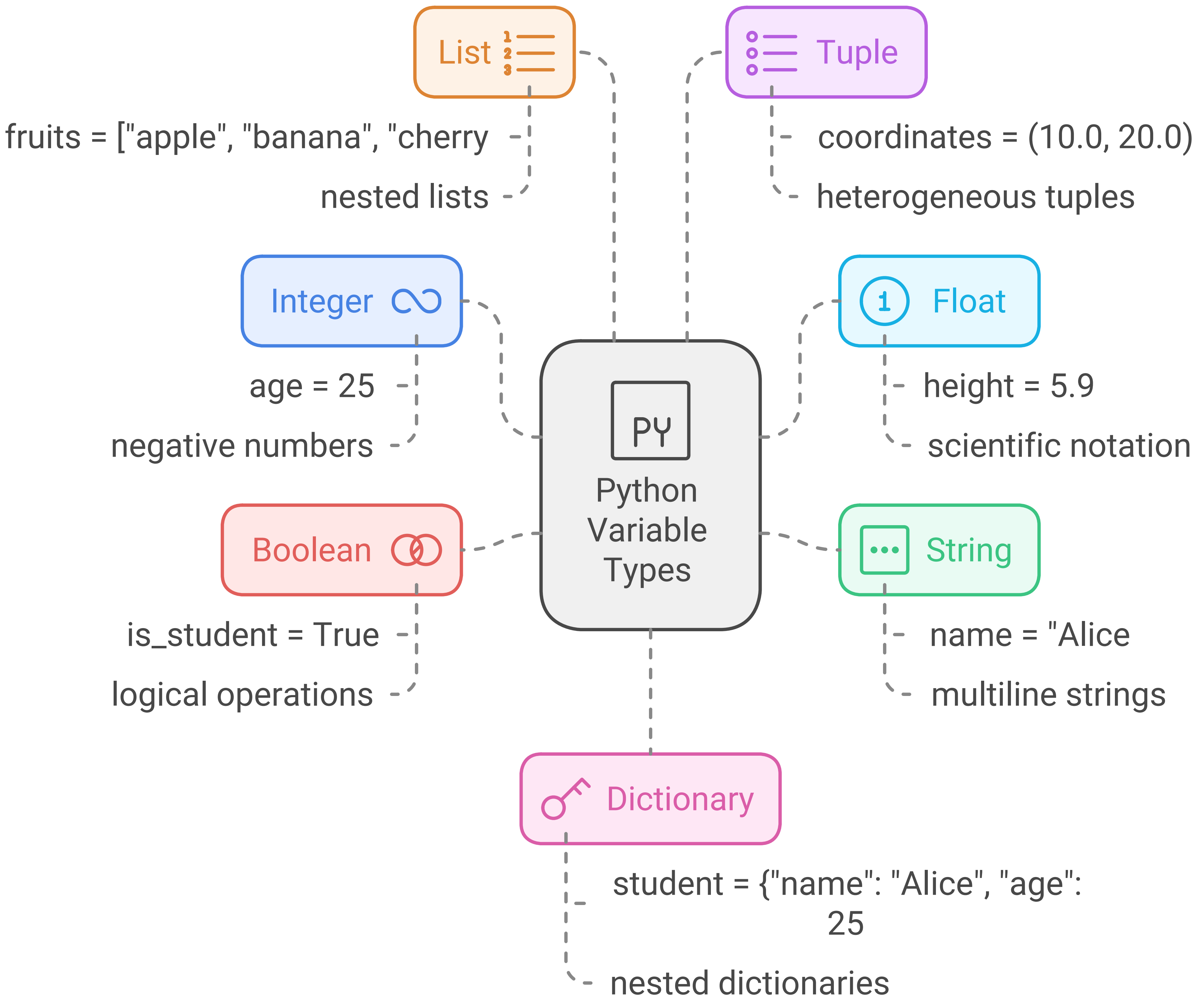
```
fruits = ["apple", "banana", "cherry"]
```

6. **Tuple**: An ordered collection of items, similar to a list but immutable.

```
coordinates = (10.0, 20.0)
```

7. **Dictionary**: A collection of key-value pairs.

```
student = {"name": "Alice", "age": 25}
```



Examples of Variable Usage

Here are some examples demonstrating how to use variables in Python:

Example 1: Basic Arithmetic Operations

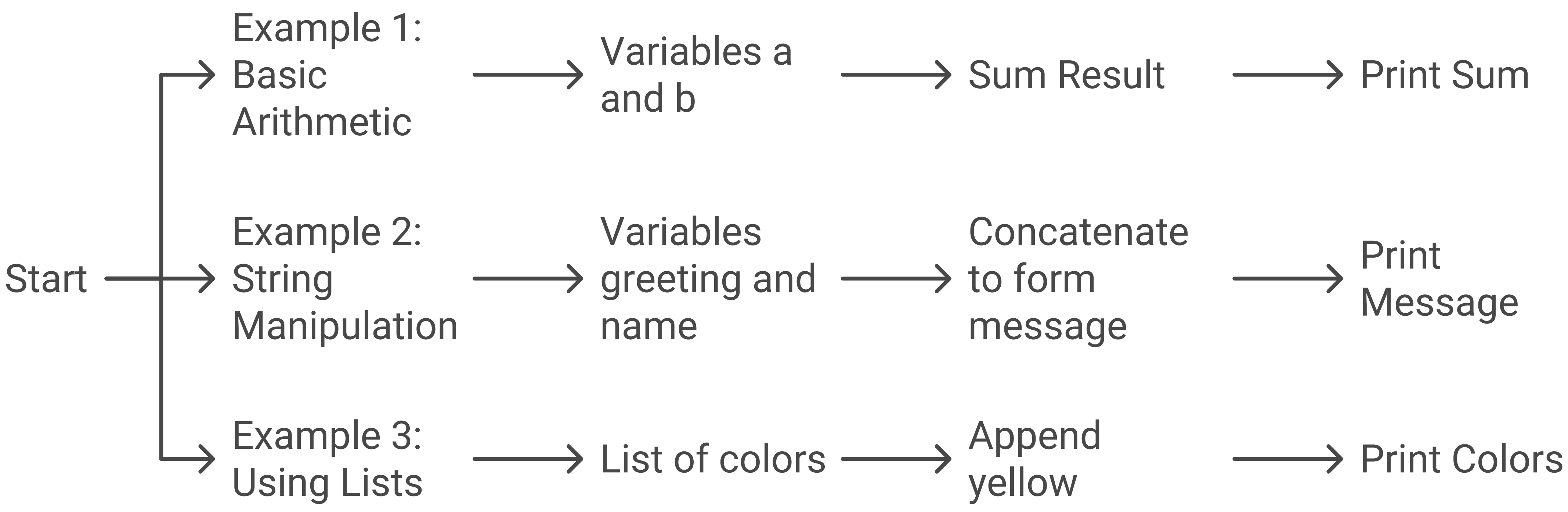
```
a = 10
b = 5
sum_result = a + b
print("Sum:", sum_result)  # Output: Sum: 15
```

Example 2: String Manipulation

```
greeting = "Hello"
name = "World"
message = greeting + ", " + name + "!"
print(message)  # Output: Hello, World!
```

Example 3: Using Lists

```
colors = ["red", "green", "blue"]
colors.append("yellow")
print(colors)  # Output: ['red', 'green', 'blue', 'yellow']
```

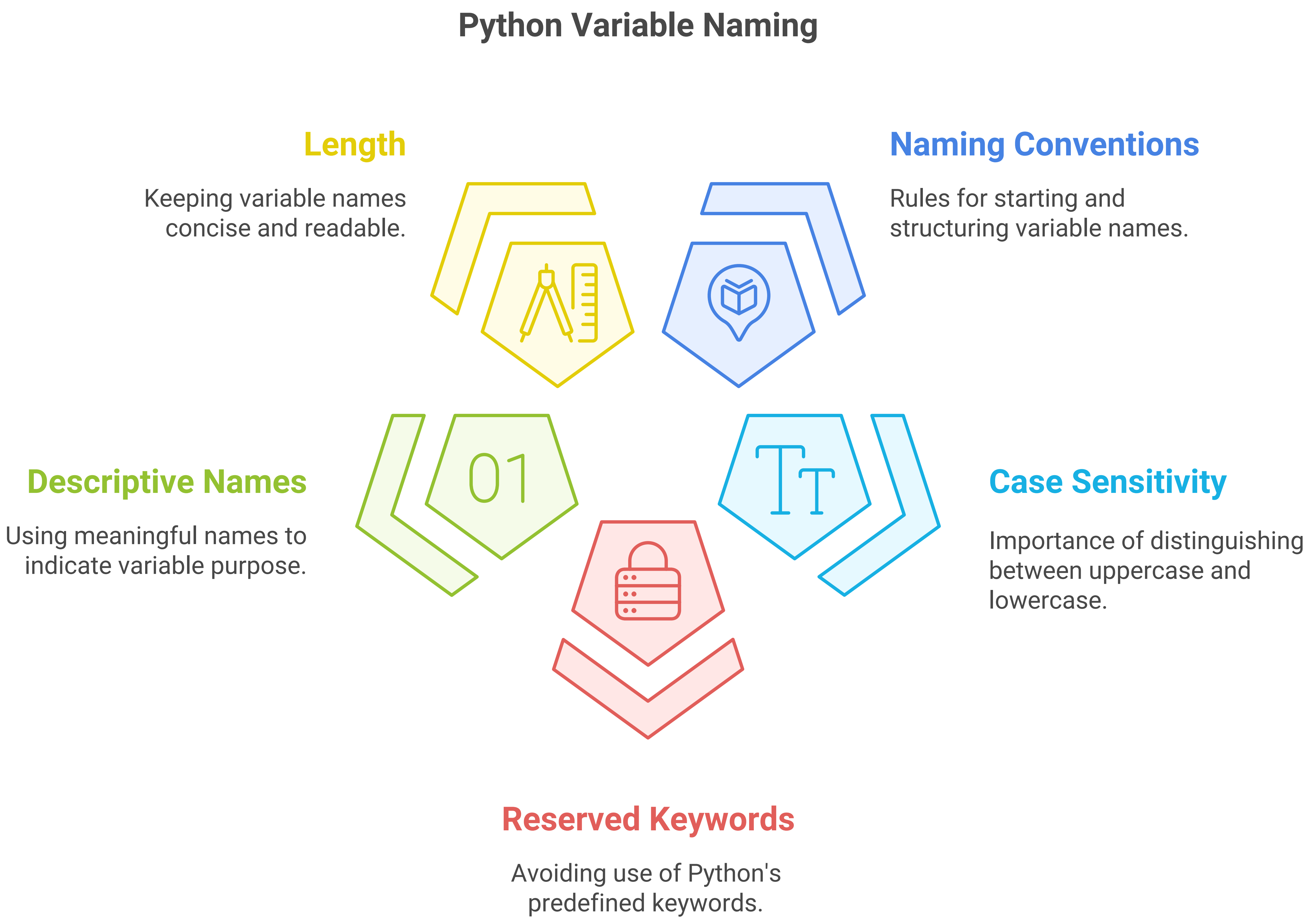


Rules for Naming Variables

When creating variables in Python, there are certain rules and conventions to follow:

1. **Naming Conventions**:
   - Variable names can include letters (a-z, A-Z), digits (0-9), and underscores (_).
   - Variable names must start with a letter or an underscore, not a digit.
2. **Case Sensitivity**:
   - Variable names are case-sensitive. For example, **age**, **Age**, and **AGE** are considered different variables.

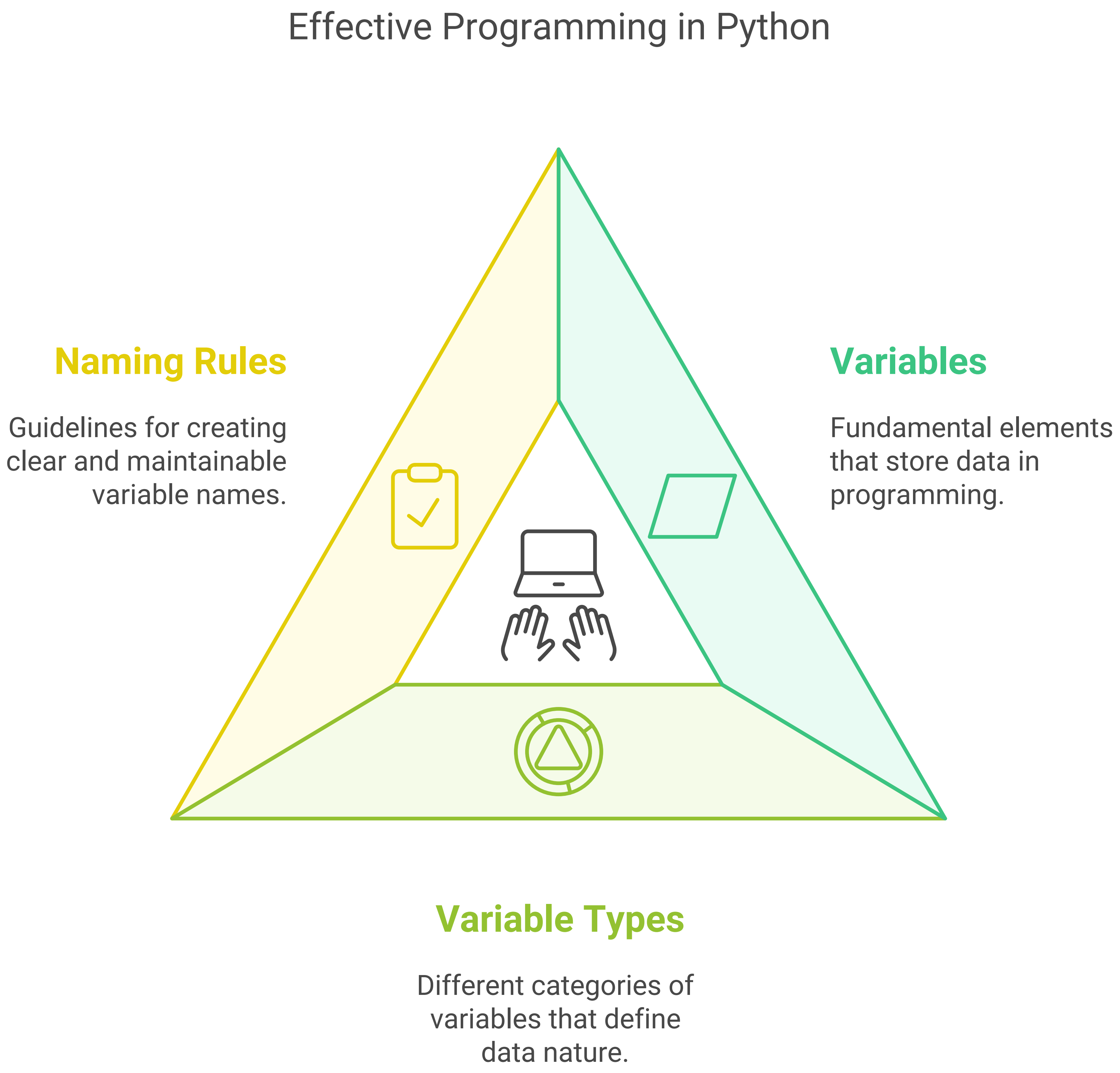
3. **Reserved Keywords**:
    - Avoid using Python's reserved keywords (like **if**, **else**, **while**, **for**, etc.) as variable names.
4. **Descriptive Names**:
    - Use descriptive names that convey the purpose of the variable. For example, use **total_price** instead of **tp**.
5. **Length**:
    - While there is no strict limit on the length of variable names, keeping them concise and readable is recommended.



Python Variable Naming
Length
Keeping variable names concise and readable.
Naming Conventions
Rules for starting and structuring variable names.
01
Tt
Descriptive Names
Using meaningful names to indicate variable purpose.
Case Sensitivity
Importance of distinguishing between uppercase and lowercase.
Reserved Keywords
Avoiding use of Python's predefined keywords.

Conclusion

Understanding variables, their types, and the rules for naming them is essential for effective programming in Python. By following the guidelines outlined in this document, you can create clear and maintainable code that leverages the power of variables to store and manipulate data efficiently.



Effective Programming in Python
Naming Rules
Guidelines for creating clear and maintainable variable names.
Variables
Fundamental elements that store data in programming.
Variable Types
Different categories of variables that define data nature.