Variables and Data Types in Python

Premanand S

Assistant Professor
School of Electronics Engineering
Vellore Institute of Technology
Chennai Campus

premanand.s@vit.ac.in

November 7, 2024

Introduction to Variables

- A variable is a named location in memory to store data.
- Python is **dynamically typed** - types are inferred during assignment.

Example (Python Code)

```
x = 10              # Integer
name = "Alice"      # String
price = 99.99       # Float
is_active = True    # Boolean
```

Variable Naming Rules and Conventions

Rules

- Must start with a letter or underscore, not a number.
- Can contain letters, numbers, and underscores.
- Case-sensitive (e.g., `Name` and `name` are different).

Conventions

- Use lowercase and underscores (e.g., `user_name`).
- Constants are written in all caps (e.g., `PI = 3.14159`).

Primitive Data Types - Integer (‘int‘)

- Used for whole numbers, positive or negative.

Example (Python Code)

```
age = 25
balance = -500
```

Primitive Data Types - Float ('float')

- Used for decimal numbers.

Example (Python Code)

```
temperature = 36.6
height = 5.3
```

Primitive Data Types - String (‘str‘)

- Text data enclosed in quotes.
- Can use single, double, or triple quotes.

Example (Python Code)

```
name = "Python"
greeting = "Hello, " + name
paragraph = """This is a
multi-line string."""
```

Primitive Data Types - Boolean ('bool')

- Represents truth values: `True` or `False`.

Example (Python Code)

```
is_logged_in = True
has_permission = False
```

Compound Data Types - List (‘list‘)

- Ordered collection that can hold mixed data types.
- Lists are mutable (modifiable).

Example (Python Code)

```
fruits = ["apple", "banana", "cherry"]
mixed_list = ["Premanand", 38, True]
```

Compound Data Types - Tuple ('tuple')

- Ordered and immutable (cannot be modified after creation).

Example (Python Code)

```
coordinates = (10, 20)
dimensions = (1920, 1080)
```

Compound Data Types - Dictionary (‘dict‘)

- Collection of key-value pairs.
- Keys are unique and typically strings.

Example (Python Code)

```
user = {"name": "Premanand", "age": 38, "is_admin": True}
```

Compound Data Types - Set ('set')

- Unordered collection of unique items (no duplicates).

Example (Python Code)

```
unique_numbers = {1, 2, 3, 4, 5}
unique_letters = {"a", "b", "c"}
```

Advanced Data Types: `frozenset` and `complex`

- `frozenset`: Immutable set, useful for fixed collections.
- `complex`: For complex numbers, represented as `a + bj`.

Example (Python Code)

```
fs = frozenset([1, 2, 3])
c = 3 + 4j
print(c.real)  # Outputs: 3.0
print(c.imag)  # Outputs: 4.0
```

Special Data Types and Concepts - NoneType ('None')

- Represents the absence of a value, often used to initialize variables.

Example (Python Code)

```
result = None
```

Type Conversion and Casting

- **Explicit Conversion**: Use functions like `int()`, `float()`, `str()`.
- **Implicit Conversion**: Python automatically converts types where appropriate.

Example (Python Code)

```
x = int("10")    # Convert string to integer
y = str(3.14)    # Convert float to string
```

Type Checking

- Use `type()` to check a variable's type.
- Use `isinstance()` to verify if a variable is of a certain type.

Example (Python Code)

```
x = 10
print(type(x))          # <class 'int'>
print(isinstance(x, int))  # True
```

Mutable vs Immutable Data Types

- **Mutable**: Lists, dictionaries, and sets can be modified.
- **Immutable**: Strings, tuples cannot be modified after creation.

Example (Python Code)

```
# Mutable
my_list = [1, 2, 3]
my_list[0] = 10

# Immutable
my_tuple = (1, 2, 3)
# my_tuple[0] = 10  # Error
```

Constants in Python

- Python lacks true constants, but we use uppercase variable names to indicate constant values.

Example (Python Code)

```
PI = 3.14159
MAX_CONNECTIONS = 5
```

Type Hinting in Python

- Type hints provide a way to indicate variable and function types.
- Enhances code readability and helps detect type errors during development.

Example (Python Code)

```
def add_numbers(a: int, b: int) -> int:
 return a + b

age: int = 38
name: str = "Premanand"
```

Using __annotations__ for Type Hints

- `__annotations__` stores type hints for functions and global variables.

Example (Python Code)

```
def greet(name: str) -> str:
 return "Hello, " + name

print(greet.__annotations__)
# Output: {'name': <class 'str'>, 'return': <class 'str'>}
```

Memory Allocation and Reference Counting

- Python uses **reference counting** to manage memory.
- Each variable points to an object stored in memory.
- Use `id()` to check the memory address of a variable.

Example (Python Code)

```
x = 42
print(id(x))  # Outputs the memory address of x
```

Variable Scope and Lifetime

- **Local Scope**: Variables declared inside a function.
- **Global Scope**: Variables declared outside any function.
- **Nonlocal Scope**: Used with nested functions, allows access to variables in the enclosing scope.

Example (Python Code)

```
count = 10
def outer():
 count = 20
 def inner():
     nonlocal count
     count += 1
 inner()
 print(count)  # Output: 21
outer()
```

Shallow vs. Deep Copy

- **Shallow Copy**: Creates a new object but inserts references into it to the original object's elements.
- **Deep Copy**: Creates a new object and recursively copies all objects found, no references to original data.

Example (Python Code)

```
import copy
original = [1, [2, 3]]
shallow_copy = copy.copy(original)
deep_copy = copy.deepcopy(original)
```

Summary

- Variables store data and come in various data types.
- Primitive data types: integers, floats, strings, booleans.
- Compound data types: lists, tuples, dictionaries, sets.
- Important concepts include type conversion, type checking, mutability, and constants.