Introduction to programming with Python (Lecture 2)

Imtiaz Ul Hassan

Python Variables

- Variables are containers for storing data
  - In Python, create variables to hold different data types.
- Naming Rules:
  - Must start with a letter or underscore.
  - Case-sensitive; follow specific naming conventions.
- Assignment:Use the = symbol for assignment.
  - Example: my_variable = 10

Python Variables

- Dynamic Typing:
  - Python dynamically determines variable types during runtime.
- Common Data Types:
  - Numeric Types: int, float
  - Text Type: str
  - Boolean Type: bool
- Variable Examples:

```
age = 25
height = 5.9
name = "John Doe"
is_student = True
```

Conditional Statements

- Introduction:
  - Conditional statements allow you to make decisions in your code based on certain conditions.
- Basic Syntax:
  - Use if, elif (optional), and else to create conditional statements.
- Example:

```
if condition:
    # Code block executed if condition is true
elif another_condition:
    # Code block executed if the first condition is false and
else:
    # Code block executed if none of the conditions are true
```

Conditional Statements

- Comparison Operators:
  - Used in conditions to compare values.
- Examples:
  - == (equal)
  - != (not equal)
  - <, >, <=, >=
- Logical Operators:
  - Combine multiple conditions.
- Examples:
  - and, or, not
- Nested Conditions:
  - Conditions can be nested within each other for more complex logic.

Arithmetic Operators

- Basic arithmetic operations in Python include:
  - Addition +
  - Subtraction -
  - Multiplication *
  - Division /
  - Exponentiation **
- Modulus Operator %:
  - Returns the remainder of the division of the left operand by the right operand.
  - remainder = 10 % 3 # Result: 1

Python Data Structures: Lists and Tuples

- Lists:
  - Ordered collection of items.
  - Mutable (can be modified after creation).
  - Created using square brackets: my_list = [1, 2, 3]
- Tuples:
  - Similar to lists but immutable.
  - Created using parentheses: my_tuple = (1, 2, 3)

Tuples and Lists

- Accessing Elements:
  - Elements are accessed by index (0-based).
  - first_element = my_list[0]
- Common Operations:
  - Adding elements: my_list.append(4).
  - Slicing: subset = my_list[1:3].
  - Concatenation: new_list = my_list + [5, 6].

Dictionaries and Sets

- Dictionaries:
  - Unordered collection of key-value pairs.
  - Keys are unique.
  - Created using curly braces: my_dict = {"name": "John", "age": 25}.
- Accessing Elements:
  - Values are accessed by keys.
  - person_name = my_dict["name"]
- Common Operations:
  - Adding/Updating elements: my_dict["gender"] = "Male".
  - Removing elements: del my_dict["age"].
  - Getting keys and values: keys = my_dict.keys(), values = my_dict.values().

Dictionaries and Sets

- Sets:
  - Unordered collection of unique elements.
  - Created using curly braces: my_set = {1, 2, 3}.
- Common Operations:
  - Adding elements: my_set.add(4).
  - Removing elements: my_set.remove(2).
- Set operations: Union, Intersection, Difference

In the Next Lecture We will cover

- Loops
  - While Loop
  - For Loop
  - Nested Loops
- Using loops with Lists
  - Finding greatest number
  - Finding smallest number
  - Sorting