Computer Programming

Course Code: CS 501, Spring 2025

SECTION : (Friday) 2:00PM – 5:00 PM, Class Room : 614

Presented by
Dr. Rubaiyat Islam
Associate Professor,
Department of Software Engineering
Daffodil international University
Omdena Bangladesh Chapter Lead
Crypto-economist Consultant
Sifchain Finance, USA.

KEYWORDS IN PYTHON

- Keywords are the reserved words in Python and can't be used as an identifier.
- Examples:

- False, None, True, and, as, assert

- break, class, continue, def, del, elif, else, except

- Finally, for, from, global, if, import, in, is, lambda

- Python contains 35 keywords in total.

IDENTIFIERS IN PYTHON

- An identifier is a name given to entities like classes, functions, variables, etc.

- Rules:
- - Cannot start with a digit
- - Cannot use special symbols
- - Cannot use keywords
- - Can contain letters (a-z, A-Z), digits (0-9), or underscores (_).

- Examples:
- - Correct: val2, val_
- - Incorrect: 1var, val2@, import

STATEMENTS IN PYTHON

- Statements are instructions that the Python interpreter can execute.

- Types:
- - Expression Statements: Compute and write a value.
- - Assignment Statements: Bind a value to a variable.
- - Control Flow Statements: Direct the order of execution (e.g., if, for, while).

DOCSTRINGS IN PYTHON

- String literals that appear right after the definition of a function, method, class, or module.

- Purpose: Document the code.
- Access: Via the `__doc__` attribute.
- Example:
- def add(a, b):
- """Add two numbers and return the result."""
- return a + b

VARIABLES IN PYTHON

- Variables are containers for storing data values.

- Dynamic Typing: No need to declare a variable with a specific type.
- Example:
- x = 5
- y = 'Hello'

MULTIPLE ASSIGNMENTS IN PYTHON

- Assign multiple variables in a single statement.

- Example:
- a, b, c = 1, 2, 3
- Swapping Variables:
- a, b = b, a

DATA TYPES IN PYTHON

- Python supports various data types:

- - Numeric: int, float, complex
- - Sequence: str, list, tuple
- - Mapping: dict
- - Set: set, frozenset
- - Boolean: bool
- - None: NoneType

STRINGS IN PYTHON

- A sequence of characters.

- Creation: Using single, double, or triple quotes.
- Example:
- my_string = 'Hello, World!'

STRING INDEXING IN PYTHON

- Access individual characters in a string.

- Syntax: string[index]
- Example:
- my_string = 'Hello'
- first_char = my_string[0] # 'H'
- last_char = my_string[-1] # 'o'

STRING SLICING IN PYTHON

- Access a substring using slicing.

- Syntax: string[start:stop:step]
- Example:
- my_string = 'Hello, World!'
- sub_string = my_string[0:5] # 'Hello'

STRING CONCATENATION IN PYTHON

- Combine strings using:

- - `+` operator to add strings.
- - `*` operator to repeat strings.

- Example:
- str1 = 'Hello'
- str2 = 'World'
- result = str1 + ' ' + str2 # 'Hello World'
- repeated = str1 * 3 # 'HelloHelloHello'

ITERATION THROUGH A STRING IN PYTHON

- Loop through each character in a string.

- Example:
- my_string = 'Hello'
- for char in my_string:
- print(char)

PARTITIONING STRINGS IN PYTHON

- Split a string into parts using `partition(separator)`.

- Example:
- my_string = 'Hello, World!'
- parts = my_string.partition(', ') # ('Hello', ', ', 'World!')

STRING FUNCTIONS IN PYTHON

- Common Methods:

- - `upper()`: Converts to uppercase.
- - `lower()`: Converts to lowercase.
- - `strip()`: Removes leading/trailing whitespace.
- - `replace(old, new)`: Replaces substrings.
- - `split(separator)`: Splits the string.

- Example:
- my_string = ' Hello, World! '
- print(my_string.upper()) # ' HELLO, WORLD! '
- print(my_string.strip()) # 'Hello, World!'

LISTS IN PYTHON

- Ordered, mutable collections of items.

- Creation: Using square brackets.
- Example:
- my_list = [1, 2, 3, 'apple', 'banana']

Thank You