NAME : Sophia D Antoinette

SRN : PES2UG22EC137

PYTHON FOR DATA SCIENCE

Assignment -1

1) In Python, which operator is used to check if two variables refer to the same object in memory?
   a) ==
   b) =
   c) is
   d) in
   Answer: c) is

2) Explain the rules and conventions for naming variables in Python. Give suitable examples.
   Answer:

- Variable names are assigned using the assignment operator =.
- The first character must be an alphabet (lowercase or uppercase).
- Variables can contain alphanumeric characters and underscore _.
- Names should not clash with inbuilt functions.
- Avoid single-character variable names (except in loops).
- Common naming conventions include CamelCase, snake_case, and PascalCase.

3) Discuss the different types of **operators** in Python with examples.
   Answer:
   Python supports the following operators:
   Arithmetic Operators – Perform mathematical operations (+, -, *, /, %, **).
   Assignment Operators – Assign values (=, +=, -=, *=, /=).
   Relational Operators – Compare values (<, <=, >, >=, ==, !=).
   Logical Operators – Combine conditions (and, or, not).
   Bitwise Operators – Perform operations at binary level (&, |).
   Operator precedence decides the order of execution, e.g., parentheses > exponent > multiplication/division > addition/subtraction.

4) Which operator is used to check if an element exists in a sequence in Python?
   a) ==
   b) is
   c) in
   d) not
   Answer: c) in

5) Explain the different sequence operations available in Python with examples.

Answer:

Python provides several operations on sequences such as strings, lists, and tuples:

Concatenation (+) – Combines two sequences.

Repetition (*) – Repeats sequence elements.

Membership (in, not in) – Checks if an element exists.

Indexing and Slicing – Access specific elements or subparts.

These operations make sequences flexible and powerful for data manipulation.

6) Describe at least four important **sequence methods** in Python with examples.

Answer:

Some commonly used sequence methods are:

1. **count(x):** Returns number of occurrences of x.

   ```
   l = [1,2,2,3]
   print(l.count(2))  # 2
   ```

2. **index(x):** Returns first index where element x is found.

   ```
   l = [10,20,30]
   print(l.index(20))  # 1
   ```

3. **append(x):** Adds an element to the end of the list.

   ```
   l = [1,2]
   l.append(3)
   print(l)  # [1,2,3]
   ```

4. **extend(iterable):** Adds multiple elements from another iterable.

   ```
   l = [1,2]
   l.extend([3,4])
   print(l)  # [1,2,3,4]
   ```