Name:**Katricia Lopez**
Sec:**CS3C**

Variable declaration
● What is variable?
-A variable is a symbolic name for a variable value in programming. It functions as a memory address-based storage space that you may access to alter data in your code by using the variable name that has been allocated to it.

● Fundamentals of Python Programming

1.Syntax: Understand the language's structure, indentation, and rules.

2.Variables and Data Types: Learn how to declare variables and work with data types like int, float, str, etc.

3.Control Flow: Master concepts like if statements, loops (for, while), and conditional expressions.

4.Functions: Create and use functions for code modularity and reusability.

5.Data Structures: Explore lists, tuples, dictionaries, and sets to organize and manipulate data.

6.File Handling: Read from and write to files using file I/O operations.

7.Exception Handling: Handle errors and exceptions gracefully in your code.

8.Modules and Libraries: Utilize built-in modules and external libraries to enhance functionality.

9.Object-Oriented Programming (OOP): Understand classes, objects, inheritance, and encapsulation.

10.Basic Algorithms: Grasp basic algorithms and problem-solving approaches.

● Rules in Declaring a Variable in Python

1.Naming Convention: Use descriptive names that reflect the variable's purpose. Follow the snake_case naming convention (lowercase with underscores).
2.Start with a Letter or Underscore: Variable names must begin with a letter (a-z, A-Z) or an underscore (_).

3.No Spaces or Special Characters: Avoid spaces and special characters in variable names, except for underscores.

4.Case-Sensitive: Python is case-sensitive, so 'myVariable' and 'myvariable' are different.

5.Avoid Keywords: Don't use Python reserved words (keywords) as variable names.

6.Follow PEP 8 Guidelines: Adhere to the PEP 8 style guide for Python code, which includes conventions for variable naming.

● Keywords in Python

-Keywords in Python are reserved words that have special meanings and cannot be used as identifiers (variable names, function names, etc.). Here are the keywords in Python:

1. **and*
2. **as*
3. **assert*
4. **break*
5. **class*
6. **continue*
7. **def*
8. **del*
9. **elif*
10. **else*
11. **except*
12. **False*
13. **finally*
14. **for*
15. **from*
16. **global*
17. **if*
18. **import*
19. **in*
20. **is*
21. **lambda*
22. **None*
23. **nonlocal*
24. **not*
25. **or*

26. **pass*
27. **raise*
28. **return*
29. **True*
30. **try*
31. **while*
32. **with*
33. **yield*

● Rules for local and global variables in Python

-In Python, local and global variables follow specific rules:

***** Variables:****
1. **Scope:** Local variables are defined within a function or a block of code. They are only accessible within that specific scope.
2. **Lifetime:** Their lifetime is limited to the duration of the function or block in which they are declared.
3. **Shadowing:** A local variable can have the same name as a global variable, but it will overshadow the global one within its scope.

Example:
```***python***
***def example_function():***
    ***local_var = 10***
    ***print(local_var)***

***example_function()***
***# Uncommenting the line below would result in an error since local_var is not defined outside the function.***
***# print(local_var)***
```

****Global Variables:****
1. **Scope:** Global variables are declared outside any function or block, making them accessible throughout the entire program.
2. **Lifetime:** They persist as long as the program is running.
3. **Access within Functions:** To modify a global variable within a function, use the `global` keyword.

Example:
```***python***
***global_var = 20***
```

def another_function():
global global_var
global_var += 5
print(global_var)

another_function()
print(global_var) # Output will be 25
```

● Operators

***-In Python, operators are symbols that perform operations on variables and values. Here are some fundamental types of operators:***

***1. **Arithmetic Operators:****
***- `+` (Addition)***
***- `-` (Subtraction)***
***- `*` (Multiplication)***
***- `/` (Division)***
***- `%` (Modulus)***
***- `**` (Exponentiation)***
***- `//` (Floor Division)***

***2. **Comparison Operators:****
***- `==` (Equal to)***
***- `!=` (Not equal to)***
***- `<` (Less than)***
***- `>` (Greater than)***
***- `<=` (Less than or equal to)***
***- `>=` (Greater than or equal to)***

***3. **Logical Operators:****
***- `and` (Logical AND)***
***- `or` (Logical OR)***
***- `not` (Logical NOT)***

***4. **Assignment Operators:****
***- `=` (Assignment)***
***- `+=` (Addition assignment)***
***- `-=` (Subtraction assignment)***
***- `*=` (Multiplication assignment)***
***- `/=` (Division assignment)***
***- `%=` (Modulus assignment)***
***- `**=` (Exponentiation assignment)***
```

- `//=` (Floor division assignment)

*5. **Identity Operators:**
- `is` (True if the operands are identical objects)
- `is not` (True if the operands are not identical objects)*

*6. **Membership Operators:**
- `in` (True if a value is found in the sequence)
- `not in` (True if a value is not found in the sequence)*

*7. **Bitwise Operators:**
- `&` (Bitwise AND)
- `|` (Bitwise OR)
- `^` (Bitwise XOR)
- `~` (Bitwise NOT)
- `<<` (Left shift)
- `>>` (Right shift)*

Understanding and using these operators is essential for manipulating data and controlling flow in Python programs.