Ans. 1. Keywords in Python are reserved words that have special meanings and purposes in the language. These keywords cannot be used as variable names, function names, or any other identifiers.

Ans. 2. In Python, variables are containers that store values. When creating a variable in Python, you need to follow certain rules. Here are the rules for creating variables in Python:

A Python variable name must start with a letter or an underscore (\_).

A Python variable name can only contain alphanumeric characters (letters and numbers) and underscores (\_). Special characters such as @, $, and % are not allowed in variable names.

Variable names are case-sensitive in Python. This means that variables with different capitalization are considered different variables. For example, "name", "Name", and "NAME" are three different variables.

Python keywords (reserved words) cannot be used as variable names. These keywords have special meanings in the Python language and are used for defining the syntax and structure of the language.

Ans. 3. Use descriptive and meaningful names

Use lowercase letters and underscores for variable names

Avoid using single-letter variable names

Be consistent with naming conventions

Avoid using reserved words or built-in names

Ans. 4. If a keyword is used as a variable name in Python, it will result in a compilation error. Python reserves certain keywords for its own use, and using them as variable names goes against the language syntax. When a programmer attempts to use a keyword as a variable name, the Python interpreter will trigger a compilation error.

Ans. 5. The def keyword in Python is used to define functions or methods. It is an essential keyword for creating user-defined functions and methods within classes. The basic syntax for defining a function using def.

Ans. 6. The backslash character \ is used for various purposes in different contexts.

Escape character in strings - In Python and many other programming languages, the backslash is used as an escape character. It is used to indicate that the character following the backslash should be treated in a special way. For example, \n represents a newline character, \t represents a tab character, and \" represents a double quotation mark within a string.

Ans. 7.

Homogeneous list:

A homogeneous list contains elements of the same data type. In the following example, all the elements in the list are integers:

my\_list = [1, 2, 3, 4, 5]

Heterogeneous set: A heterogeneous set contains elements of different data types. In this example, the set includes integers, strings, and a boolean value:

my\_set = {1, "Hello", True, 3.14}

Homogeneous tuple: A homogeneous tuple contains elements of the same data type. In the following example, all the elements in the tuple are strings:

my\_tuple = ("apple", "banana", "cherry", "durian")

Ans. 8.

Immutable Data Types: Immutable data types are those whose values cannot be modified after they are created. If you need to change the value of an immutable object, you create a new object with the desired value.

x = 10

y = x

y += 5

print(x) # Output: 10

print(y) # Output: 15

In this example, x is an immutable integer. When y is assigned the value of x, it creates a new variable y with the same value. However, when y is incremented by 5, it doesn't affect the original value of x.

Mutable Data Types: Mutable data types are those whose values can be modified after they are created. Changes made to a mutable object affect its current state directly.

my\_list = [1, 2, 3]

my\_list.append(4)

print(my\_list) # Output: [1, 2, 3, 4]

In this case, my\_list is a mutable list. The append() method modifies the list in-place by adding an element to the end.

It's important to note that when an immutable object appears to be modified, what actually happens is the creation of a new object with a different value. On the other hand, mutable objects allow modification of their internal state without creating a new object.