What are identifiers in python?

In Python, **identifiers** are the names used to identify variables, functions, classes, modules, or other objects in your code. An identifier must follow certain rules:

1. **Allowed Characters**: Identifiers can contain letters (A-Z or a-z), digits (0-9), and underscores (\_). However, they **cannot** start with a digit.
   * Valid: my\_variable, variable123, \_privateVar
   * Invalid: 123variable, my-variable!
2. **Case Sensitivity**: Identifiers are case-sensitive, meaning myVar and myvar would be considered different identifiers.
3. **Keywords Restriction**: You cannot use Python's reserved **keywords** (like if, else, class, etc.) as identifiers.
4. **Conventions**: While not mandatory, there are common naming conventions:
   * Variables and functions: snake\_case
   * Classes: PascalCase
   * Constants: UPPER\_CASE

Examples of identifiers:

my\_variable = 10

def my\_function():

pass

class MyClass:

pass

In this example, my\_variable, my\_function, and MyClass are all identifiers.

**You said:**

What are Variables and explain in details

**What Are Variables in Python?**

In Python, a **variable** is a symbolic name that references or holds a value. It's essentially a container for storing data that can be used and manipulated throughout a program. Variables make it possible to work with dynamic data rather than hardcoding values into your code.

**Characteristics of Variables**

1. **Dynamically Typed**:
   * Python is dynamically typed, meaning you don't need to explicitly declare the type of a variable when you create it. The type is inferred from the value assigned to the variable.
   * Example:

python

Copy code

x = 10 # 'x' is an integer

x = "Hello" # Now 'x' is a string

1. **Mutable and Immutable Data Types**:
   * Some data types in Python are **mutable**, meaning their values can be changed in place (e.g., lists, dictionaries).
   * Others are **immutable**, meaning their value cannot be changed after they're created (e.g., integers, strings, and tuples).

Example:

# Immutable

a = 5 # Integer (immutable)

a = a + 1 # Creates a new value

# Mutable

my\_list = [1, 2, 3] # List (mutable)

my\_list[0] = 4 # Modifies the first element of the list

**Naming Variables**

* Python variables follow the same rules as identifiers:
  1. Must start with a letter (A-Z, a-z) or an underscore (\_), but not a digit.
  2. Can contain letters, digits, and underscores.
  3. Are case-sensitive, meaning myVar and myvar are different variables.

**Assigning Values to Variables**

Assignment in Python is done using the = operator:

x = 5 # Assigns the integer value 5 to the variable 'x'

y = "Hello" # Assigns the string value 'Hello' to 'y'

z = 3.14 # Assigns a float value to 'z'

You can also assign multiple variables at once:

python

Copy code

a, b, c = 1, 2, 3 # Assigns 1 to 'a', 2 to 'b', and 3 to 'c'

Or assign the same value to multiple variables:

python

Copy code

x = y = z = 10 # All variables get the value 10

**Types of Variables**

1. **Local Variables**:
   * Variables defined within a function and only accessible within that function.

python

Copy code

def my\_function():

x = 10 # Local variable

print(x) # Accessible only inside the function

1. **Global Variables**:
   * Variables defined outside of any function and accessible throughout the entire program, including inside functions.

python

Copy code

x = 20 # Global variable

def my\_function():

print(x) # Can access the global variable

1. **Nonlocal Variables**:
   * These are variables used in nested functions, referring to a variable defined in the enclosing (but not global) scope.

python

Copy code

def outer\_function():

x = "outer"

def inner\_function():

nonlocal x # Refers to the 'x' in outer\_function

x = "inner"

inner\_function()

print(x) # Will print 'inner'

**Variable Scope**

The scope of a variable refers to the region of the program where the variable can be accessed. Variables can have different scopes:

1. **Local Scope**: Variables defined inside a function are only accessible within that function.
2. **Global Scope**: Variables defined outside of all functions can be accessed anywhere in the code.
3. **Nonlocal Scope**: Variables in a nested function, but not in the global scope, can be accessed using the nonlocal keyword.

**Variable Types and Conversion**

Python offers various types for variables:

* **Integers** (int): Whole numbers, positive or negative (x = 42).
* **Floats** (float): Numbers with a decimal point (y = 3.14).
* **Strings** (str): Text (name = "John").
* **Booleans** (bool): True or False values (is\_valid = True).
* **Lists**: Ordered, mutable sequences (fruits = ["apple", "banana"]).
* **Tuples**: Ordered, immutable sequences (colors = ("red", "blue")).
* **Dictionaries**: Unordered collections of key-value pairs (person = {"name": "Alice", "age": 30}).
* **Sets**: Unordered collections of unique elements (my\_set = {1, 2, 3}).

**Type Conversion**

You can convert between different data types using functions like int(), float(), str(), list(), etc.

python

Copy code

x = "100"

y = int(x) # Converts the string "100" to the integer 100

z = float(y) # Converts the integer 100 to the float 100.0

**Example of Using Variables**

python

Copy code

# Define variables

name = "Alice"

age = 25

height = 5.7

# Print values

print(f"Name: {name}, Age: {age}, Height: {height}")

# Perform calculations

years\_until\_30 = 30 - age

height\_in\_cm = height \* 30.48

# Print calculated values

print(f"Years until 30: {years\_until\_30}")

print(f"Height in cm: {height\_in\_cm:.2f}")

In this example:

* Variables name, age, and height hold different types of data (string, integer, float).
* Calculations and formatted printing demonstrate how variables can be manipulated.

**Summary**

* Variables store and reference data in Python.
* They are dynamically typed and follow specific naming rules.
* Variable scope defines where a variable can be accessed.
* Python supports different data types for variables and allows type conversion when necessary.

4o