**1. Primitive Data Types in Python**

Python supports the following primitive data types:

* **int**: Represents integer numbers. Example: x = 10
* **float**: Represents floating-point numbers. Example: y = 3.14
* **bool**: Represents Boolean values (True or False). Example: is\_true = True
* **str**: Represents strings of characters. Example: name = "Alice"
* **complex**: Represents complex numbers. Example: z = 2 + 3j

**2. Mutable vs. Immutable Variables**

* **Mutable:** Values can be changed after assignment. Examples: Lists, dictionaries.
* **Immutable:** Values cannot be changed after assignment. Examples: Numbers, strings, tuples.

**3. Local and Global Variables**

* **Local:** Defined within a function and only accessible inside that function.
* **Global:** Defined outside a function and accessible from anywhere in the program.

To modify a global variable inside a function, use the global keyword:

Python

Global variable

x = 10    
  
def modify\_global():  
    global x  
    x = 20  
  
modify\_global()  
print(x)

Output: 20

**4. Lists and Tuples**

* **List:** An ordered collection of items, mutable. Example: my\_list = [1, 2, 3]
* **Tuple:** An ordered collection of items, immutable. Example: my\_tuple = (1, 2, 3)

Use a list when you need to modify the elements, and use a tuple when you want to ensure the data remains unchanged.

**5. Dictionaries (dict)**

A dictionary is an unordered collection of key-value pairs. Example: my\_dict = {"name": "Alice", "age": 30}

**6. NoneType Variables**

A NoneType variable represents the absence of a value. It's often used as a default value or to indicate that a variable has not been assigned a value yet.

**7. Sets and Lists**

* **Set:** An unordered collection of unique elements. Example: my\_set = {1, 2, 3}
* **List:** An ordered collection of elements, can contain duplicates. Example: my\_list = [1, 2, 2, 3]

**8. Creating and Changing Variable Types**

Use type casting to convert a variable to a specific type:

Python

x = 10    
y = float(x)    
z = str(x) 

**9. Assigning Multiple Variables**

Python

a, b, c = 1, 2, 3

**10. Shallow Copy vs. Deep Copy**

* **Shallow Copy:** Creates a new object but references the same underlying data.
* **Deep Copy:** Creates a new object with a completely independent copy of the data.

Python

import copy  
  
Shallow copy  
list1 = [1, 2, [3, 4]]  
list2 = list1.copy()  
list2[0] = 10    
  
 Deep copy  
list3 = copy.deepcopy(list1)  
list3[0] = 20 