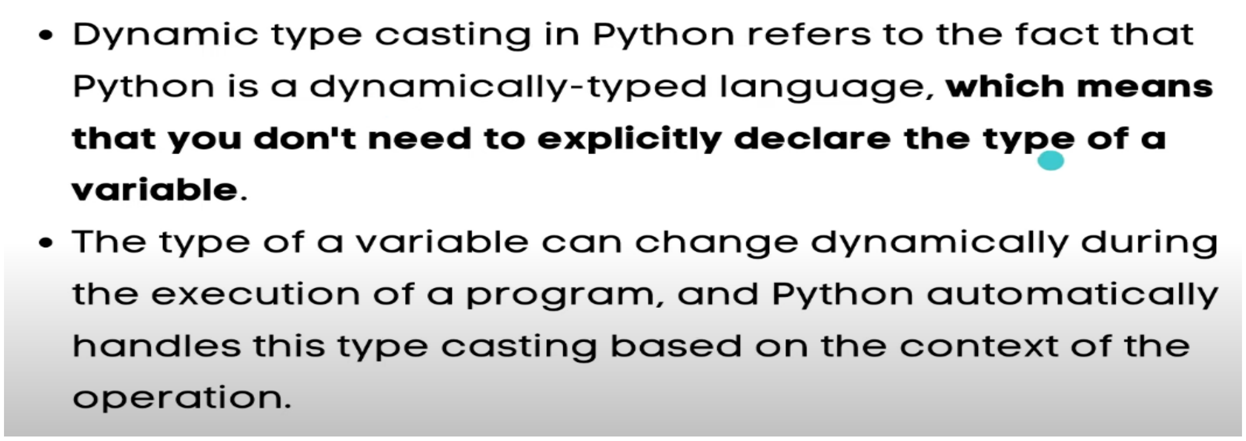
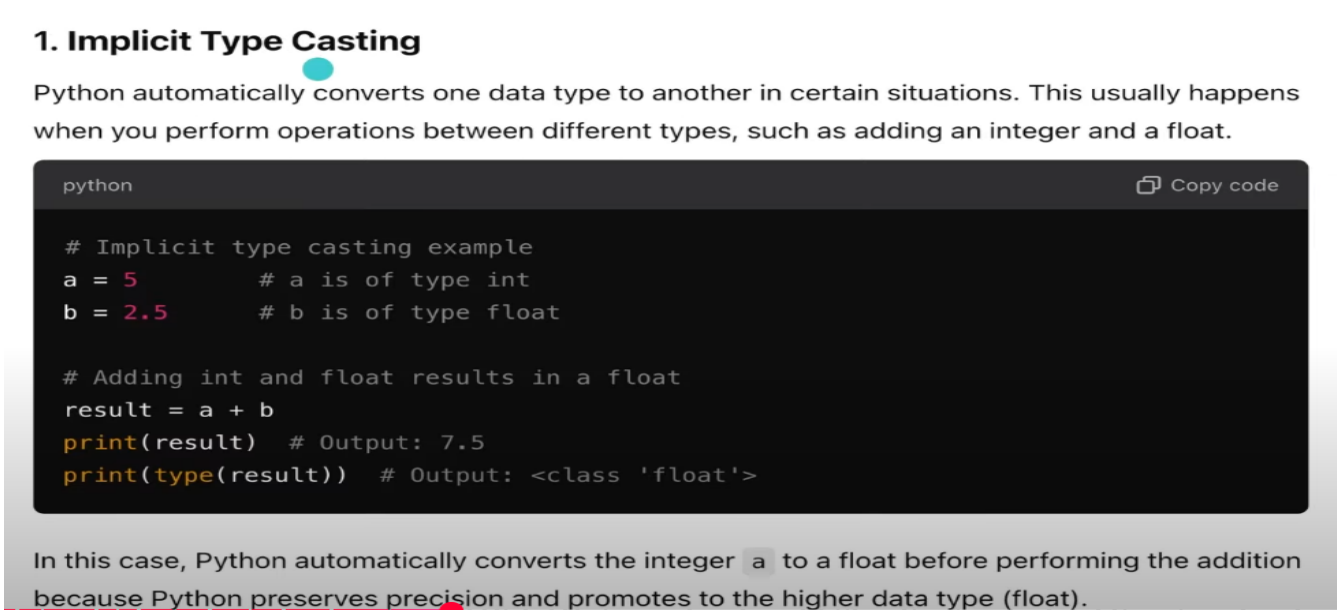


- Dynamic type casting in Python refers to the fact that Python is a dynamically-typed language, **which means that you don't need to explicitly declare the type of a variable**.
- The type of a variable can change dynamically during the execution of a program, and Python automatically handles this type casting based on the context of the operation.



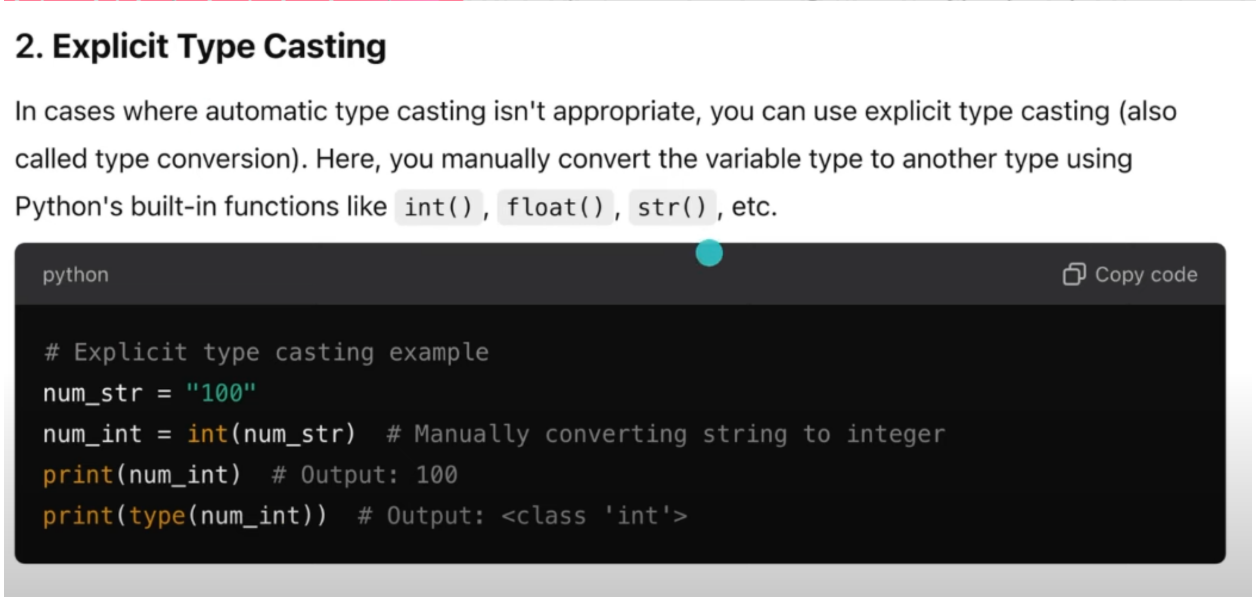
1. Implicit Type Casting

Python automatically converts one data type to another in certain situations. This usually happens when you perform operations between different types, such as adding an integer and a float.

```python
# Implicit type casting example
a = 5          # a is of type int
b = 2.5        # b is of type float

# Adding int and float results in a float
result = a + b
print(result)  # Output: 7.5
print(type(result))  # Output: <class 'float'>
```

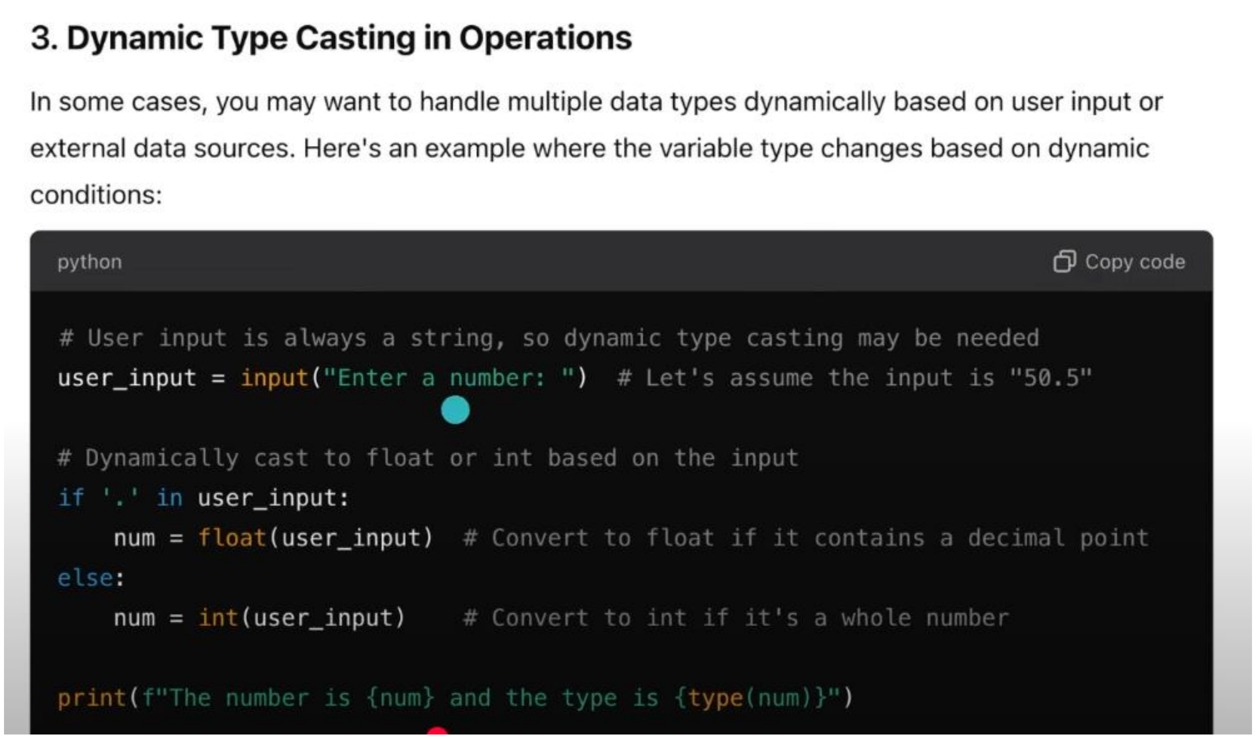
In this case, Python automatically converts the integer `a` to a float before performing the addition because Python preserves precision and promotes to the higher data type (float).



2. Explicit Type Casting

In cases where automatic type casting isn't appropriate, you can use explicit type casting (also called type conversion). Here, you manually convert the variable type to another type using Python's built-in functions like `int()`, `float()`, `str()`, etc.

```python
# Explicit type casting example
num_str = "100"
num_int = int(num_str)  # Manually converting string to integer
print(num_int)  # Output: 100
print(type(num_int))  # Output: <class 'int'>
```



3. Dynamic Type Casting in Operations

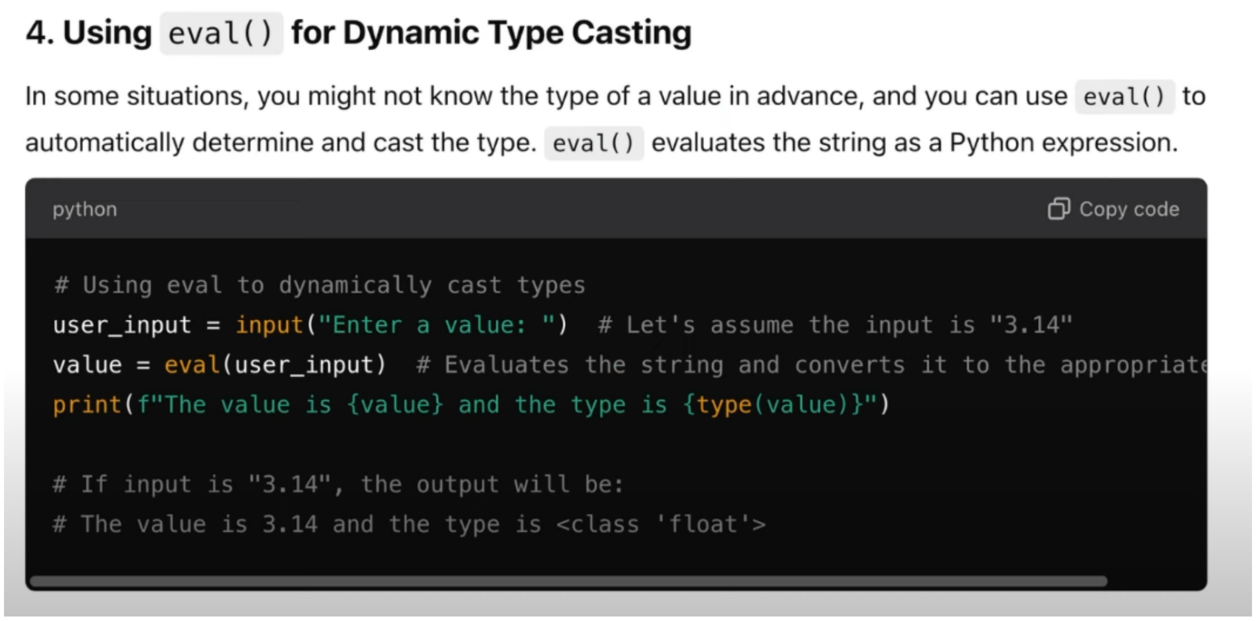
In some cases, you may want to handle multiple data types dynamically based on user input or external data sources. Here's an example where the variable type changes based on dynamic conditions:

python Copy code

```python
# User input is always a string, so dynamic type casting may be needed
user_input = input("Enter a number: ")  # Let's assume the input is "50.5"

# Dynamically cast to float or int based on the input
if '.' in user_input:
    num = float(user_input)  # Convert to float if it contains a decimal point
else:
    num = int(user_input)    # Convert to int if it's a whole number

print(f"The number is {num} and the type is {type(num)}")
```



4. Using `eval()` for Dynamic Type Casting

In some situations, you might not know the type of a value in advance, and you can use `eval()` to automatically determine and cast the type. `eval()` evaluates the string as a Python expression.

python Copy code

```python
# Using eval to dynamically cast types
user_input = input("Enter a value: ")  # Let's assume the input is "3.14"
value = eval(user_input)  # Evaluates the string and converts it to the appropriat
print(f"The value is {value} and the type is {type(value)}")

# If input is "3.14", the output will be:
# The value is 3.14 and the type is <class 'float'>
```

5. Changing Variable Types Dynamically

You can also dynamically reassign a variable to another type during runtime:

```python
# Dynamically changing variable types
x = 100    # Initially an integer
print(type(x))  # Output: <class 'int'>

x = "Python"  # Now x is a string
print(type(x))  # Output: <class 'str'>

x = 3.1415  # Now x is a float
print(type(x))  # Output: <class 'float'>
```

In this example, `x` is initially an integer, then reassigned to a string, and later reassigned to a float, illustrating how Python dynamically handles type casting based on variable reassignment.

6. Casting during Mathematical Operations

Python dynamically promotes types during mathematical operations. When different types are involved in a single expression, Python converts them to a compatible type.

```python
# Example of dynamic casting during operations
a = 5  # int
b = 2.7  # float

result = a * b  # Python automatically casts the result to a float
print(result)  # Output: 13.5
print(type(result))  # Output: <class 'float'>
```