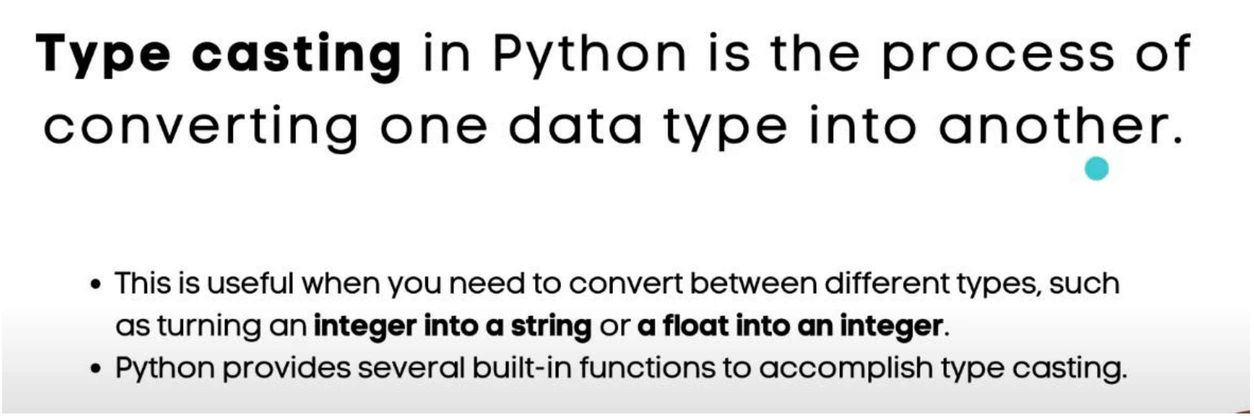
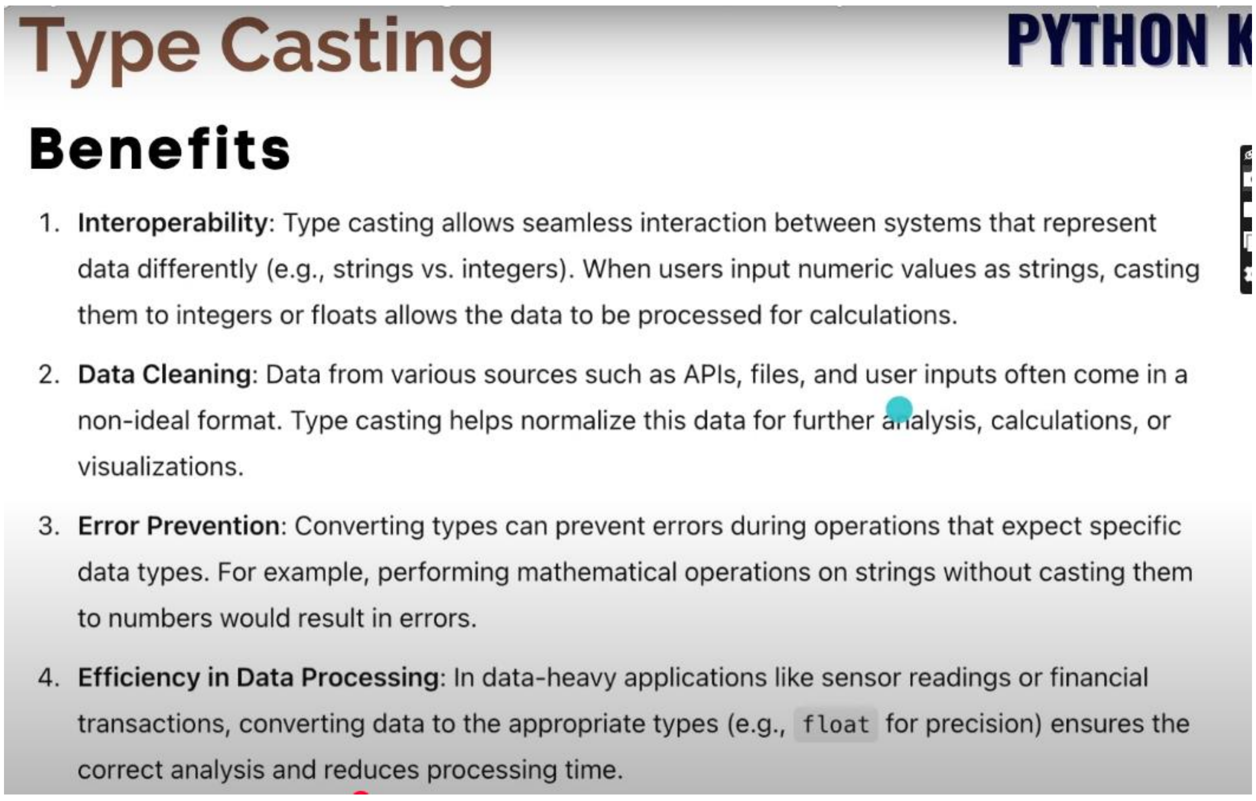


Type casting in Python is the process of converting one data type into another.

- This is useful when you need to convert between different types, such as turning an **integer into a string** or **a float into an integer**.
- Python provides several built-in functions to accomplish type casting.

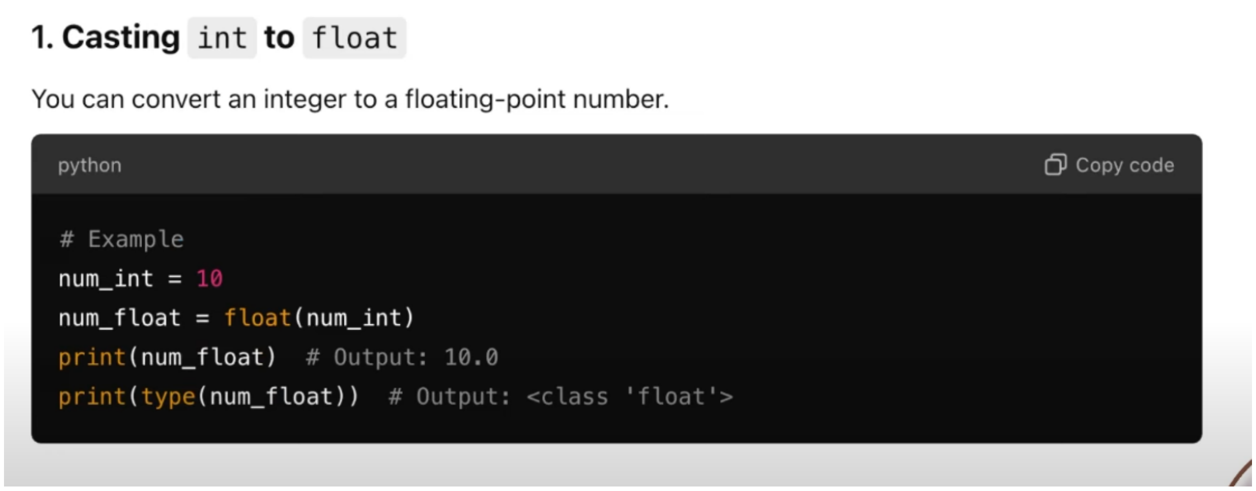


Type Casting

PYTHON K

Benefits

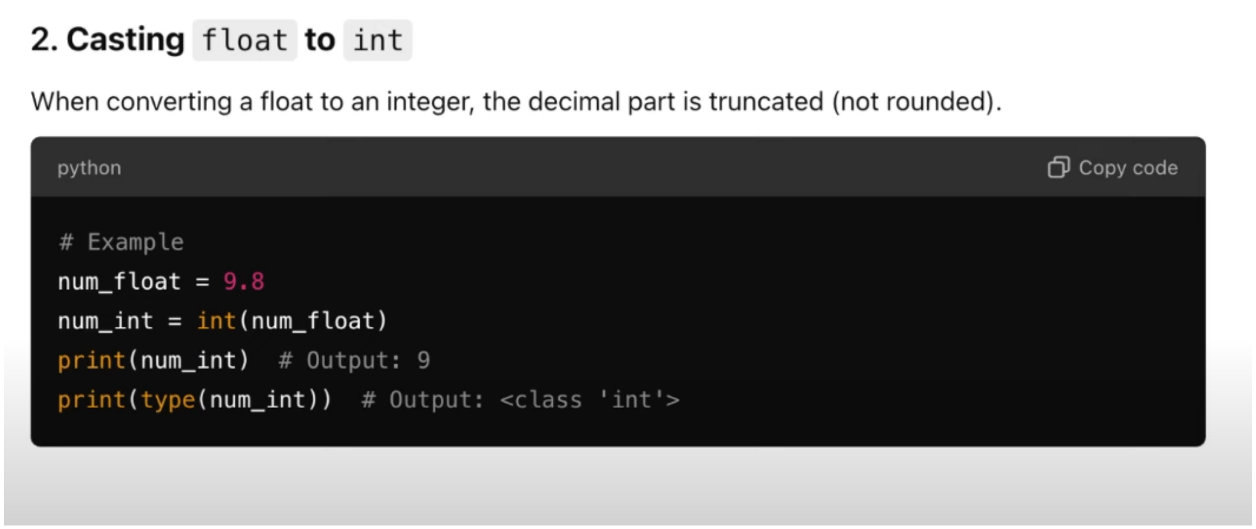
1. **Interoperability**: Type casting allows seamless interaction between systems that represent data differently (e.g., strings vs. integers). When users input numeric values as strings, casting them to integers or floats allows the data to be processed for calculations.
2. **Data Cleaning**: Data from various sources such as APIs, files, and user inputs often come in a non-ideal format. Type casting helps normalize this data for further analysis, calculations, or visualizations.
3. **Error Prevention**: Converting types can prevent errors during operations that expect specific data types. For example, performing mathematical operations on strings without casting them to numbers would result in errors.
4. **Efficiency in Data Processing**: In data-heavy applications like sensor readings or financial transactions, converting data to the appropriate types (e.g., `float` for precision) ensures the correct analysis and reduces processing time.



1. Casting `int` to `float`

You can convert an integer to a floating-point number.

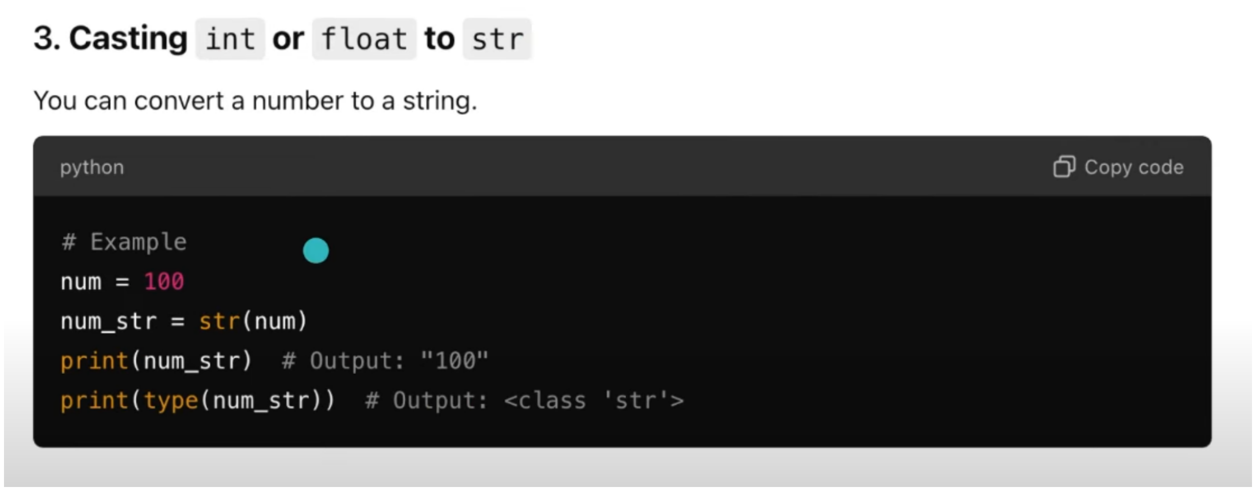
```python
# Example
num_int = 10
num_float = float(num_int)
print(num_float)  # Output: 10.0
print(type(num_float))  # Output: <class 'float'>
```



2. Casting `float` to `int`

When converting a float to an integer, the decimal part is truncated (not rounded).

```python
# Example
num_float = 9.8
num_int = int(num_float)
print(num_int)  # Output: 9
print(type(num_int))  # Output: <class 'int'>
```



3. Casting `int` or `float` to `str`

You can convert a number to a string.

```python
# Example
num = 100
num_str = str(num)
print(num_str)  # Output: "100"
print(type(num_str))  # Output: <class 'str'>
```

4. Casting `str` to `int` or `float`

If a string contains numeric characters, you can convert it to an integer or float.

python Copy code

```python
# Example
num_str = "50"
num_int = int(num_str)
print(num_int)  # Output: 50
print(type(num_int))  # Output: <class 'int'>

num_str_float = "123.45"
num_float = float(num_str_float)
print(num_float)  # Output: 123.45
print(type(num_float))  # Output: <class 'float'>
```

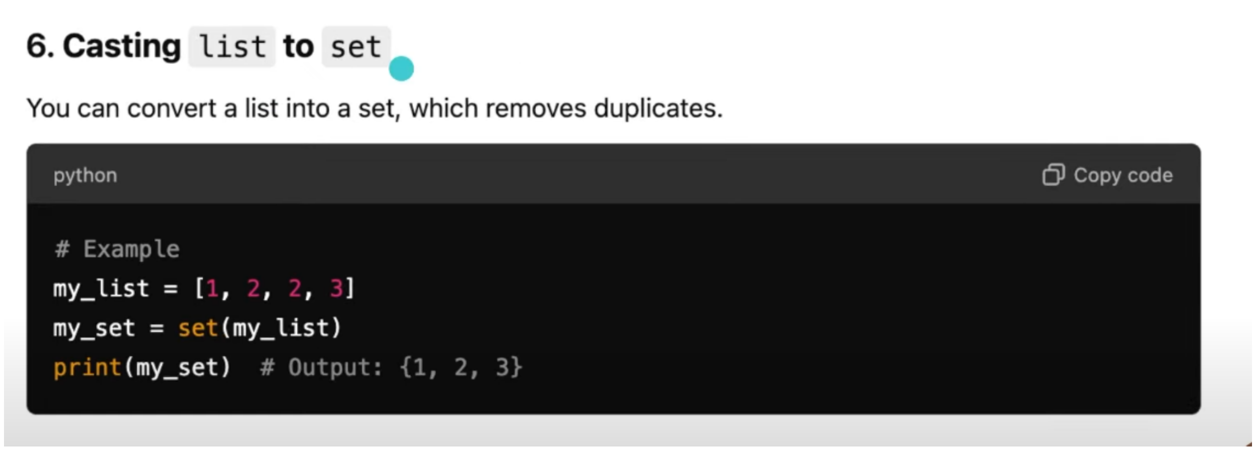
5. Casting `list` to `tuple` and vice versa

Lists can be converted into tuples and vice versa.

python Copy code

```python
# Example: List to tuple
my_list = [1, 2, 3]
my_tuple = tuple(my_list)
print(my_tuple)  # Output: (1, 2, 3)

# Example: Tuple to list
my_new_list = list(my_tuple)
print(my_new_list)  # Output: [1, 2, 3]
```

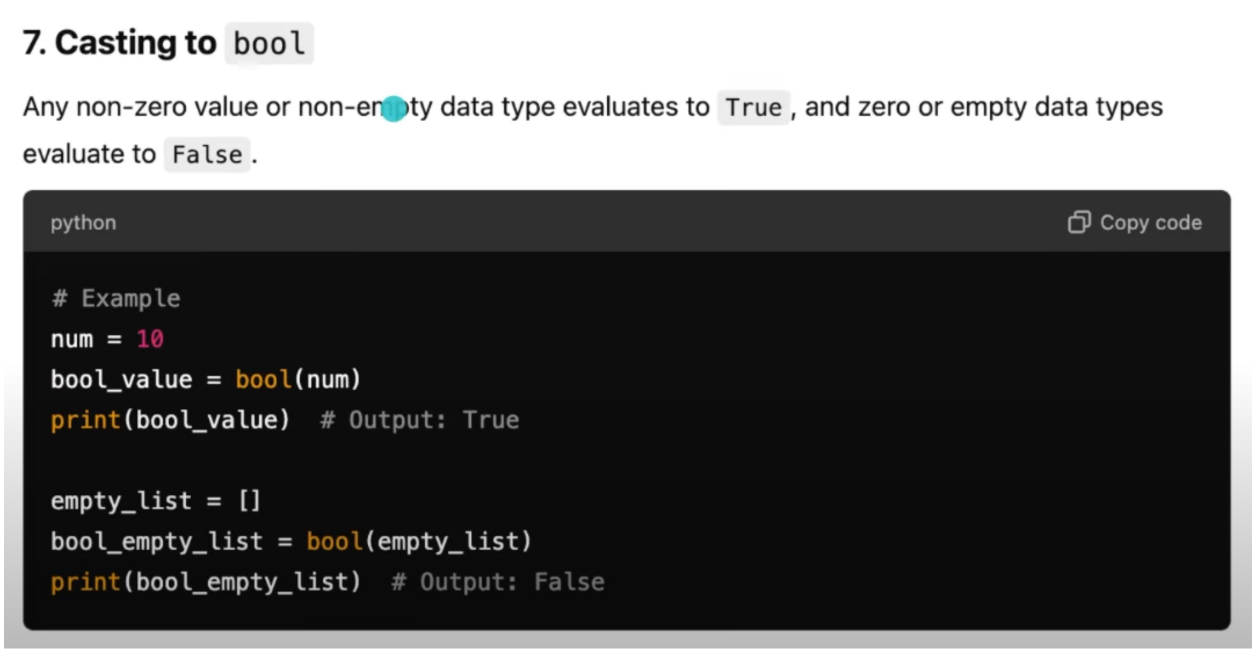


6. Casting `list` to `set`

You can convert a list into a set, which removes duplicates.

python Copy code

```python
# Example
my_list = [1, 2, 2, 3]
my_set = set(my_list)
print(my_set)  # Output: {1, 2, 3}
```



7. Casting to `bool`

Any non-zero value or non-empty data type evaluates to `True`, and zero or empty data types evaluate to `False`.

python Copy code

```python
# Example
num = 10
bool_value = bool(num)
print(bool_value)  # Output: True

empty_list = []
bool_empty_list = bool(empty_list)
print(bool_empty_list)  # Output: False
```