# Typecasting in Python

## What is Typecasting?

- **Typecasting** is the process of converting a variable's data type into another data type.
- This is useful in situations where you need to manipulate data in a format different from how it was originally stored or collected.

## Types of Typecasting

### 1. Implicit Typecasting

- Python automatically converts one data type to another to prevent data loss or errors during operations.
- This happens only when data types are **compatible**.

#### Example:

```
# Implicit conversion from int to float
num_int = 5
num_float = num_int + 2.5
print(num_float)  # Output: 7.5
print(type(num_float))  # Output: <class 'float'>
```

### 2. Explicit Typecasting

- Developers manually convert a variable to another data type using Python’s built-in functions.
- This is useful when Python does not automatically handle the conversion (e.g., string to int).

#### Common Typecasting Functions:

1. `str()` : Converts data to a string.

   ```
   value = 123
   string_value = str(value)
   print(string_value)  # Output: "123"
   print(type(string_value))  # Output: <class 'str'>
   ```

2. `int()` : Converts data to an integer (if possible).

```
value = "42"
int_value = int(value)
print(int_value)  # Output: 42
print(type(int_value))  # Output: <class 'int'>
```

3. **`float()`** : Converts data to a float (if possible).

```
value = "3.14"
float_value = float(value)
print(float_value)  # Output: 3.14
print(type(float_value))  # Output: <class 'float'>
```

## Other Typecasting Functions

- **`ord()`** : Converts a character to its Unicode integer value.

```
print(ord('A'))  # Output: 65
```

- **`hex()`** : Converts an integer to a hexadecimal string.

```
print(hex(255))  # Output: '0xff'
```

- **`oct()`** : Converts an integer to an octal string.

```
print(oct(8))  # Output: '0o10'
```

- **`list()`** : Converts a sequence to a list.

```
value = "abc"
print(list(value))  # Output: ['a', 'b', 'c']
```

- **`tuple()`** : Converts a sequence to a tuple.

```
value = [1, 2, 3]
print(tuple(value))  # Output: (1, 2, 3)
```

- **`set()`** : Converts a sequence to a set.

```
value = [1, 2, 2, 3]
```

```
print(set(value))  # Output: {1, 2, 3}
```

- **`dict()`** : Converts key-value pairs into a dictionary.

```
pairs = [('a', 1), ('b', 2)]
print(dict(pairs))  # Output: {'a': 1, 'b': 2}
```

## Important Notes

1. **Compatibility Matters**: Python will raise a `TypeError` if the conversion isn't possible.

```
value = "abc"
int_value = int(value)  # Raises ValueError
```

2. **Precision in Conversions**: Converting float to int truncates the decimal.

```
float_value = 4.99
int_value = int(float_value)
print(int_value)  # Output: 4
```

3. **String Conversion**: Almost all data types can be converted to strings.

## Practice Examples

1. Convert a floating-point number to an integer.
2. Convert a string representation of a number into a float.
3. Use `ord()` to find the Unicode value of a character.
4. Convert a list of numbers into a set.