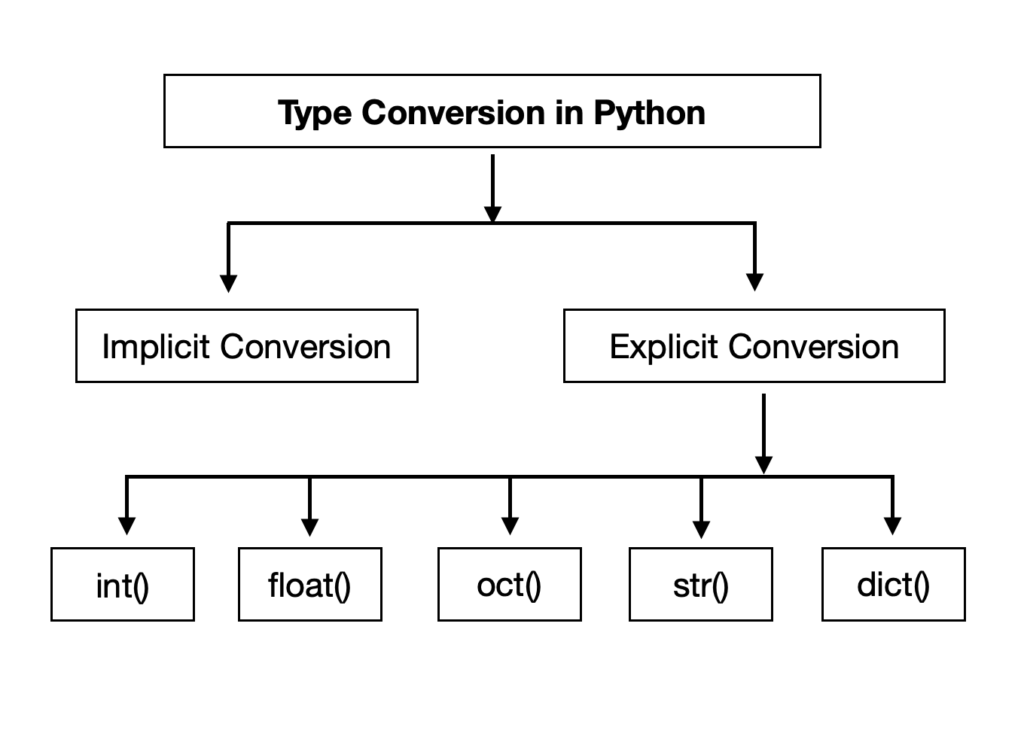
**Type Conversion**

In Python programming, type conversion, also known as typecasting, is converting data from one data type to another.

This feature is vital as it allows programmers to manipulate data effectively and perform various operations across different data types. Let's delve into the details of type conversion in Python:

**In Python, there are two types of data conversion**

****

**1. Implicit Type Conversion:**

Python automatically converts one data type to another when required. This process is known as implicit type conversion or coercion.

It occurs during arithmetic operations where operands are of different data types. Python promotes the lower data type (e.g., integer) to the higher data type (e.g., float) to perform the operation.

**Example:**

```python

num\_int = 10

num\_float = 5.5

# Implicit conversion of num\_int to float

result = num\_int + num\_float

```

**2. Explicit Type Conversion:**

Explicit type conversion, also called type casting, involves converting the data type explicitly using predefined functions like int(), float(), str(), etc.

This method gives programmers control over the conversion process and ensures accuracy when dealing with different data types.

**Example:**

```python

num\_str = "10"

# Explicit conversion of num\_str to integer

num\_int = int(num\_str)

```

**3. Functions for Type Conversion:**

**Python provides some built-in functions to do explicit type conversion:**

- int(x): Converts x to an integer.

- float(x): Converts x to a floating-point number.

- str(x): Converts x to a string.

- bool(x): Converts x to a Boolean value (True or False).

- list(x): Converts x to a list.

- tuple(x): Converts x to a tuple.

- dict(x): Converts x to a dictionary.

- set(x): Converts x to a set.

**Example:**

```python

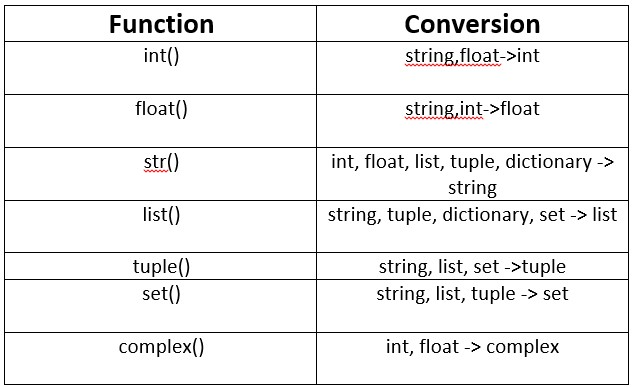
num\_float = 3.7

# Convert float to integer

num\_int = int(num\_float)

```

**Internally it follows the type-conversion Order as shown:**

****

**Handling Exceptions while** **performing explicit type conversion:**

it's crucial to handle exceptions that may occur if the conversion is not possible.

Python provides try-except blocks to handle such exceptions gracefully, ensuring smooth program execution.

**Example:**

```python

str\_num = "hello"

try:

# Try converting string to integer

num = int(str\_num)

except ValueError:

# Handle ValueError if conversion fails

print("Conversion failed: Not a valid integer.")

```

Key Points to Remember:

**1. Type Conversion:** It's like changing the form of something from one type to another, such as turning a number into a word or vice versa.

**2. Implicit Type Conversion:** This happens automatically in Python without us having to do anything. Python is smart and tries to avoid losing any information when converting.

3. **Explicit Type Conversion:** Also known as Type Casting, it's when we deliberately change the type of an object using special functions provided by Python. However, this can sometimes lead to loss of information.

**4. Data Loss Caution:** While Python tries to handle conversions carefully, be cautious as there might still be cases where you lose some information, especially with explicit conversions.

Type conversion in Python is like changing one thing into another. Sometimes Python does it by itself, and sometimes we need to tell it what to do. We use it to make sure our data fits together correctly, like changing a number to a word or a word to a number. Knowing how to convert data types helps us write better and more flexible code, making Python a powerful tool for solving different kinds of problems.