- **created by**: Sudip Ghimire
- **URL**: https://www.sudipghimire.com.np
- **GitHub**: https://github.com/ghimiresdp

go to course contents

Chapter 2.6. Typecasting

Table of Contents

- Chapter 2.6. Typecasting
  - Implicit typecasting
  - Explicit typecasting

- Type casting or type conversion is the process of converting the data from one type to another.
- We perform type casting to unify the data types for better and efficient calculation
- Sometimes we perform typecasting to display the equivalent and readable value of the variable.

In case of python there are 2 types of type conversion

1. Implicit typecasting
2. Explicit typecasting

Implicit typecasting

- It is the type of typecasting when interpreter automatically converts data type while performing calculation
- It is done when we perform operations between lower and higher precision quantities.
- Implicit type casting always casts lower precision data type to higher precision one.
- example: `int` can be implicitly type casted to `float`

Example:

```
print(5 + 4.5)
```

In this case, the integer, `5` is automatically converted into `float` data type and is then added to `4.5` since 4.5 is higher precision number.

While Using F-Strings, data automatically gets typecasted to display the correct representation.

Example:

```
x = 45
sentence = f'The value of x is {x}'
```

In this case, the value of x will automatically be typecasted into `str` and replaced inside `{}`.

Explicit typecasting

If we want to manually convert one data type into another, then it is known as explicit typecasting.

- it is a manual process where we want to convert one type of data to another
- it is more useful when we want to convert numeric to non-numeric and vice-versa.

Example: *Converting string input to integer while parsing value from command line*

```
rate = 2
quantity = input('Enter the number of apples you want to buy: ')

# here, the value of quantity when entered by user will be taken as string

quantity = int(quantity)    # Explicit type conversion from str to int

amount = quantity * rate
print (f'The total payable amount is: {amount}.)
```

Type casting from higher precision data-types to lower-precision data types can also lead to data loss. so we have to make sure that we actually need to convert the higher precision data type into lower precision one.

example: *float to int type casting omits all numbers after the decimal point.*

Example 1: Explicit typecasting before adding `45.5` and `54.5` will result `99` instead of `100.0` in which the difference might be negligible.

```
x = 45.5
y = 54.5
print(x + y)                # 100.0
print(int(x) + int(y))      # 99
```

Example 2: Explicit typecasting before taking power of `10.9` by `5.7` will result `100000` instead of `777168.88` in which the difference is significant and will lead to wrong results.

```
x = 10.9
y = 5.7
print(x ** y)               # 777168.8877963118
print(int(x) ** int(y))     # 100000
```

So before typecasting, we need to be sure that we're doing it right and is not significantly changing the expected output.