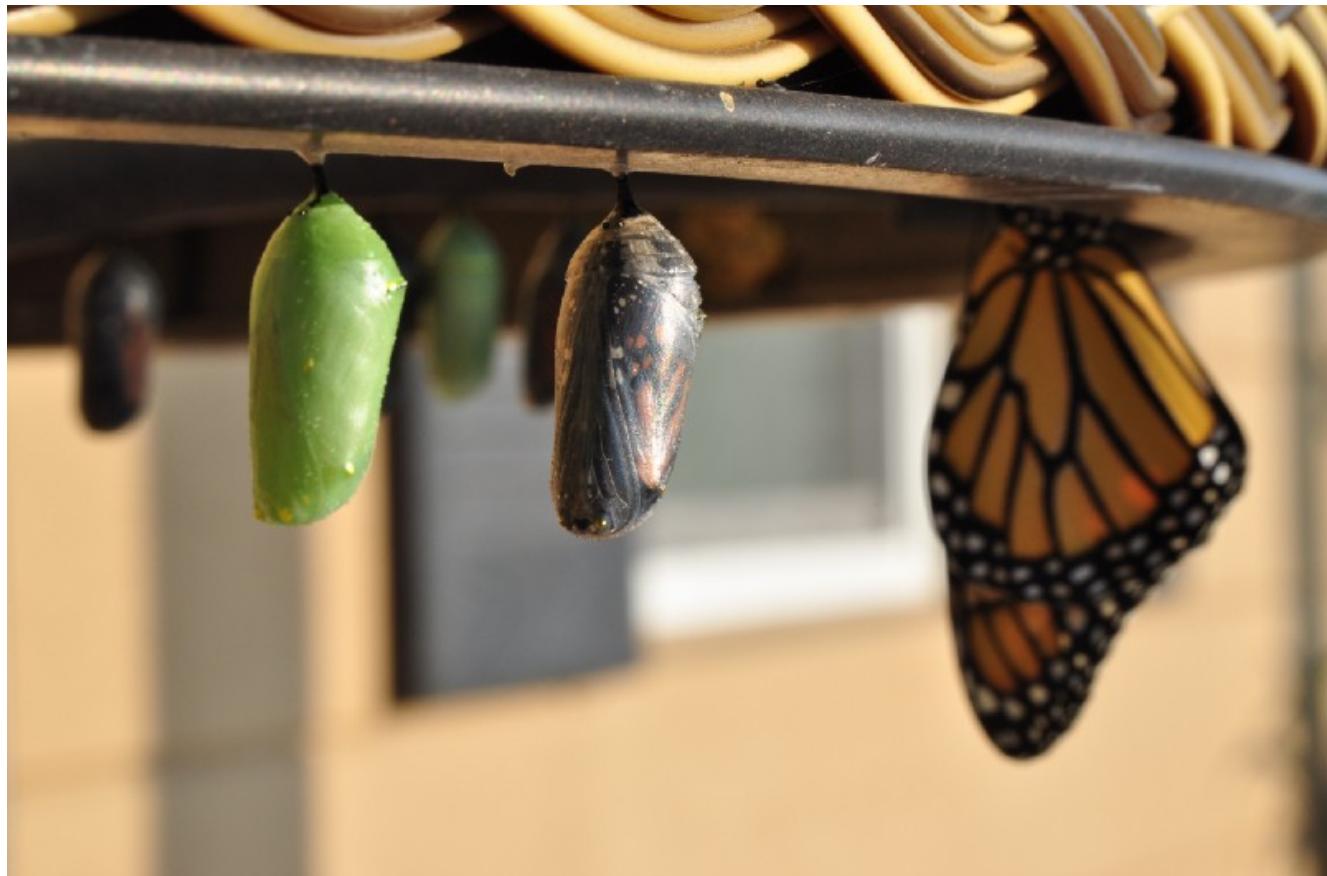


[Donate](#)

Stay safe, friends. Learn to code from home. Use our free 2,000 hour curriculum.

5 MARCH 2019 / #PROGRAMMING

Learn typecasting in Python in five minutes



by PALAKOLLU SRI MANIKANTA

A crash course on Typecasting and Type conversion in Python in a very non-verbose manner

[Donate](#)~~The process of converting one data type to another data type is~~

called Typecasting or Type Coercion or Type Conversion.

The topics that I'll be focusing on in this article are:

1. Implicit Type Conversion
2. Explicit Type Conversion
3. Advantages
4. Disadvantages

Implicit Type Conversion

When the type conversion is performed automatically by the interpreter without the programmer's intervention, that type of conversion is referred to as **implicit type conversion**.

Example Program:

```
myInt = 143      # Integer value.myFloat = 1.43  # Float value.

myResult = myInt + myFloat  # Sum result

print("datatype of myInt:", type(myInt))print("datatype of myFlo
```

[Donate](#)

Output:

The output for the above program will be:

```
datatype of myInt: <class 'int'>datatype of myFloat: <class 'float'>
```

In the above program,

- We add two variables myInt and myFloat, storing the value in myResult.
- We will look at the data type of all three objects respectively.
- In the output, we can see the datatype of myInt is an integer , the datatype of myFloat is a float .
- Also, we can see the myFloat has float data type because Python converts smaller data type to larger data type to avoid the loss of data.

This type of conversion is called **Implicit Type conversion (or) UpCasting**.

Explicit Type Conversion

In Explicit Type Conversion, users convert the data type of an object to the required data type. We use predefined in-built functions like:

1. int()
2. float()

[Donate](#)

4. `bool()`

5. `str()`

The syntax for explicit type conversion is:

`(required_datatype)(expression)`

This type of conversion is called **Explicit Type conversion (or) DownCasting.**

Int Conversion

We can use this function to convert values from other types to int.

For example:

```
>>> int(123.654)123
```

```
>>>int(False)0
```

```
>>> int("10")10
```

```
>>> int("10.5")ValueError: invalid literal for int() with base 10
```

[Donate](#)

```
>>> int("ten")ValueError: invalid literal for int() with base 10
```

```
>>> int("0B1111")ValueError: invalid literal for int() with base
```

```
>>> int(10+3j)TypeError: can't convert complex to int
```

Note:

1. You can't convert complex datatype to int
2. If you want to convert string type to int type, the string literal must contain the value in Base-10

Float Conversion

This function is used to convert any data type to a floating point number.

For example:

```
>>> float(10) 10.0
```

[Donate](#)

```
>>> float(False)0.0

>>> float("10")10.0

>>> float("10.5")10.5

>>> float("ten")ValueError: could not convert string to float: ``

>>> float(10+5j)TypeError: can't convert complex to float

>>> float("0B1111")ValueError: could not convert string to float
```

Note:

1. You can convert complex type to float type value.
2. If you want to convert string type to float type, the string literal must contain the value in base-10.

[Donate](#)

Complex Conversion

This function is used to convert real numbers to a complex (real, imaginary) number.

Form 1: complex (x)

You can use this function to convert a single value to a complex number with real part x and imaginary part 0.

For example:

```
>>> complex(10)10+0j
```

```
>>> complex(10.5)10.5+0j
```

```
>>> complex(True)1+0j
```

```
>>> complex(False)0+0j
```

```
>>> complex("10")10+0j
```

```
>>> complex("10.5")10.5+0j
```

[Donate](#)

```
>>> complex("ten")ValueError: complex() arg is a malformed string
```

Form 2: **complex (x, y)**

If you want to convert X and Y into complex number such that X will be real part and Y will be imaginary part.

For example:

```
>>> complex(10, -2)10-2j
```

```
>>> complex(True, False)1+0j
```

Boolean Conversion

This function is used to convert any data type to boolean data type easily. It is the most flexible data type in Python.

For example:

```
>>> bool(0)False
```

```
>>> bool(1)True
```

[Donate](#)

```
>>> bool(10)True
```

```
>>> bool(0.13332)True
```

```
>>> bool(0.0)False
```

```
>>> bool(10+6j)True
```

```
>>> bool(0+15j)True
```

```
>>> bool(0+0j)False
```

```
>>> bool("Apple")True
```

```
>>> bool("")False
```

[Donate](#)

type of datatype into boolean and the output will be - For all values it will produce True except 0, 0+0j and for an Empty String.

String Conversion

This function is used to convert any type into a string type.

For example:

```
>>> str(10)'10'
```

```
>>> str(10.5)'10.5'
```

```
>>> str(True)'True'
```

```
>>> str(False)'False'
```

```
>>> str(10+5j)'10+5j'
```

```
>>> str(False)'False'
```

[Donate](#)

Example Program:

```
integer_number = 123 # Intstring_number = "456" # String

print("Data type of integer_number:", type(integer_number))print(
    string_number = int(string_number))print("Data type of string_num

number_sum = integer_number + string_number

print("Sum of integer_number and num_str:", number_sum)print("Data type of num
    str:", type(number_sum))
```

Output:

When we run the above program the output will be:

```
Data type of integer_number: <class 'int'>Data type of num_str b
```

In the above program,

[Donate](#)

- We add string_number and integer_number variable.
- We converted string_number from string(higher) to integer(lower) type using `int()` function to perform addition.
- After converting string_number to an integer value Python adds these two variables.
- We got the number_sum value and data type to be an integer.

Advantages Of Typecasting

1. More convenient to use

Disadvantages Of Typecasting

1. More complex type system
2. Source of bugs due to unexpected casts

I covered pretty much everything that is required to perform any type of typecasting operation in Python3.

Hope this helped you learn about Python Typecasting in a quick and easy way.

If you liked this article please click on the clap and leave me your valuable feedback.

If this article was helpful, [tweet it.](#)

[Donate](#)

helped more than 40,000 people get jobs as developers.

[Get started](#)

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) nonprofit organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

You can [make a tax-deductible donation here](#).

Trending Guides

[JavaScript Closure](#)[JavaScript Promise](#)[CSS Box Shadow](#)[What is GitHub?](#)[Python List Append](#)[Python Sort List](#)[JavaScript Array Sort](#)[Comments in JSON](#)[Symlink in Linux](#)[What is Kanban?](#)[Linux Grep Command](#)[Python Write to File](#)[What is DNS?](#)[CSS Media Queries](#)[Primary Key SQL](#)[HTML Entities](#)[SQL Update Statement](#)[Excel VBA](#)[Screenshot on PC](#)[LOOKUP in Excel](#)[What is a Proxy Server?](#)[Arrow Function JavaScript](#)[Cat Command in Linux](#)[Remove Duplicates in Excel](#)[CSS Background Image](#)[dllhost.exe COM Surrogate](#)

[Donate](#)

Our Nonprofit

[About](#) [Alumni Network](#) [Open Source](#) [Shop](#) [Support](#) [Sponsors](#) [Academic Honesty](#)
[Code of Conduct](#) [Privacy Policy](#) [Terms of Service](#) [Copyright Policy](#)