

Overview

In a previous lesson, we looked at how some variable types don't play nice with others. This is especially important to know for when you want to use data retrieved from a user's input.

ALL USER INPUT IS A STRING EVEN IF THEY TYPED NUMBERS!

However, sometimes you want to do math with the input a user gave you. That's where type caseing comes in.

Important Information

When we start off with casting, we are going to assume that our users are nice and cooperative. The first time we use casting, we will assume that our user gave us the information we asked for.

However, users are not nice. Users break code. This lesson on casting is meant only as a demonstration for changing types, but is not robust enough to be considered "good code".

We will learn more about creating more robust code that users *can't* break in our next module.

Set Up

Create a new file called `typecasting.py`.

Copy

Consider a program where we ask for a user's age to determine what year they were born.

```
age = input("What is your age?")  
  
year_born = 2026 - age  
  
print("You were born in the year" + year_born)
```

Immediately when we run it, we get a `TypeError` because `age` is treated as a string, and `year_born` is trying to subtract a number and a string. If I typed `27` as the input, it is treated like `"27"` (notice the quotation marks) - a `string` value and not a number.

To get rid of this error, we can `Type Cast`. Type casting is when we change a values type so it is interpreted differently.

We can use the built in `int()` method to turn a `string` that looks like an `int` into an `int`.

```
# Have the user give us an age.  
age = input("What is your age?")  
# Turn that age into a number.  
age = int(age)
```

Our other really common casting methods are `float()` for turning a value into a `float` and `str()` for turning a value into a `string`.

Change

If you run the program (which should look like this):

```
# Have the user give us an age.  
age = input("What is your age?")  
# Turn that age into a number.  
age = int(age)  
  
year_born = 2026 - age  
  
print("You were born in the year" + year_born)
```

You are still getting a `TypeError`! Change the `year_born` variable to a `string` so it can be added properly to our message.

Once you have it running with no errors, see what happens if you are a not nice user. Instead of typing a number, type `"Hello!"` as your input. What happens?

You just discovered another kind of error! A `ValueError` occurs when you try to cast a value to a new type that can't work.

Python has no meaningful way to make `"Hello"` into an integer, which makes sense. `"Hello"` is a `string`. We won't look at dealing with type errors until our next module, but for now you should be aware of them.

Challenge

Ask the user for their favourite number, add 10 to it, and then tell them what their favourite number is plus 10.

Tips and Tricks

Any value can be placed between the brackets for type casting. You have to be careful about `ValueError`s, but you can shorten the process of getting an integer or float from a user by putting it all on one line.

```
age = int(input("What is your age? "))
```