# Theory: Taking input

12 minutes 14 / 14 problems solved

Start practicing

25458 users solved this topic. Latest completion was about 2 hours ago.

Sometimes programs need to interact with users, either to receive some data or to deliver some sort of result. And that's when the `input()` function steals the show.

## §1. Reading input from a user

The input data we want to get is nothing but some value entered by the user. The `input()` function reads this value and returns it in a program as a string. For example, the following program reads the user name and prints a greeting.

```
user_name = input()
print('Hello, ' + user_name)
```

In the first line, the program will wait for the user to enter something as input, which we will assign to a variable so we can use it later. In the second line, the program appends the entered name to the end of `'Hello, '` string and prints the whole phrase as a result.

If a user enters `Sauron`, this program prints:

```
Hello, Sauron
```

So, your program prints a result that depends on the user's input (name).

## §2. Clear messages

It is highly recommended to state clearly what type of input we expect from our user. To do so, the `input()` function may take an optional argument, that is a message:

```
user_name = input('Please, enter your name: ')
print('Hello, ' + user_name)
```

The program starts, the user sees the message, enters their name and gets the result as follows:

```
Please, enter your name: Sauron
Hello, Sauron
```

Another way to do it is to print the message separately:

```
print('Enter your name: ')
user_name = input()
print('Hello, ' + user_name)
```

There's no big difference actually: in the previous example, the input will be printed in the same line as the message, while in this case it will be written in the next line. So, you may choose whatever you like.

> Although it is recommended to print messages for users, avoid them in our educational programming challenges, otherwise your code may not pass our tests.

## §3. Important details

Let's dig into some details. First of all, how long can the user's input be? The second question is: how does the program understand that the person entered everything they wanted?

Current topic:

✓ Taking input 17★ Stage 1

Topic depends on:

✓ Variables 17★ Stage 1

Topic is required for:

✓ String formatting 7★ Stage 1

✓ Program with numbers 17★ Stage 1

✓ Basic string methods 13★ Stage 1

Text corpora in NLTK

Table of contents:

Here's a thing about the `input()` function: as soon as the program has started executing this function, it stops and waits for the user to enter some value and **press Enter**. That also means that if there's no input from the user, the program will not execute any further.

What else should you remember? Well, this: any value you enter, the function sees as a **string**. It doesn't matter if you enter digits or letters, the input will be converted to a string.

If you want a number to be a **number**, you should write it explicitly:

```
print("What's your favorite number?")
value = int(input())  # now value keeps an integer number
```

However, be careful: in these circumstances, if a user enters a non-integer value, an `Error` will appear.

To read several inputs, you should call the function more than once:

```
day = int(input())  # 4
month = input()     # October
```

Brilliant! Why this date? It's simple:

```
print('Cinnamon roll day is celebrated on', month, day)
# Cinnamon roll day is celebrated on October 4
```

Congratulations, now you know how to work with `input()`, that is, a function that helps you interact with the user. Believe us, this is something you will definitely appreciate when programming.

Report a typo

**1900** users liked this theory. **28** didn't like it. **What about you?**

😍 🙂 😐 😕 😡

Start practicing

Comments (16) Hints (0) Useful links (1) Show discussion