INPUT FROM KEYBOARD

THE INPUT FUNCTION

There are hardly any programs without any input. Input can come in various ways, for example from a database, another computer, mouse clicks and movements or from the internet. Yet, in most cases the input stems from the keyboard. For this purpose, Python provides the function input(). input has an optional parameter, which is the prompt string.

If the input function is called, the program flow will be stopped until the user has given an input and has ended the input with the return key. The text of the optional parameter, i.e. the prompt, will be printed on the screen.

The input of the user will be interpreted. If the user e.g. puts in an integer value, the input function returns this integer value. If the user on the other hand inputs a list, the function will return a list.

Let's have a look at the following example:

```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("So, you are already " + str(age) + " years old, " + name +
"!")
```

We save the program as "input_test.py" and run it:

```
$ python input_test.py
What's your name? "Frank"
Nice to meet you Frank!
Your age? 30
So, you are already 30 years old, Frank!
```

It is quite possible that you may have forgotten to include your name into quotes. If so, there is also a good chance that you may have been confused with the error message:

```
Traceback (most recent call last):
  File "input_test.py", line 1, in <module>
    name = input("What's your name? ")
  File "<string>", line 1, in <module>
NameError: name 'Frank' is not defined
```

We mentioned it at the beginning of this chapter on the input function: input interprets the input. That's the reason, why we had to cast the variable "age" into a string. If you don't wrap your name into quotes, Python takes your name as a variable. So, the error message makes sense!

Let's have a look at further examples. We will use the function type to check the type of the variables:

```
>>> name = input("What's your name? ")
What's your name? "John"
>>>
>>> age = input("Your age? ")
Your age? 38
>>> print(age, type(age))
(38, <type 'int'>)
>>> colours = input("Your favourite colours? ")
Your favourite colours? ["red","green","blue"]
>>> print(colours)
['red', 'green', 'blue']
>>> print(colours, type(colours))
(['red', 'green', 'blue'], <type 'list'>)
>>>
```

INPUT WITH RAW_INPUT()

raw_input does not interpret the input. It always returns the input of the user without changes, i.e. raw. This raw input can be changed into the data type needed for the algorithm. To accomplish this we can use either a casting function or the eval function. Once more, we use the interactive shell to demonstrate this behaviour:

```
>>> age = raw_input("Your age? ")
Your age? 38
>>> print(age, type(age))
('38', <type 'str'>)
>>>
>>> age = int(raw_input("Your age? "))
Your age? 42
>>> print(age, type(age))
(42, <type 'int'>)
>>>
```

```
>>> programming_language = raw_input("Your favourite programming
languages? ")
Your favourite programming languages? ["Python", "Lisp","C++"]
>>> print(programming_language, type(programming_language))
('["Python", "Lisp","C++"]', <type 'str'>)
>>>
>>> programming_language = eval(raw_input("Your favourite
programming languages? "))
Your favourite programming languages?  ["Python", "Lisp","C++"]
>>> print(programming_language, type(programming_language))
(['Python', 'Lisp', 'C++'], <type 'list'>)
>>>
```

Using the casting function list in the last example, doesn't return, what some readers might expect:

```
>>> programming_language = list(raw_input("Your favourite
programming languages? "))
Your favourite programming languages?  ["Python", "Lisp","C++"]
>>> print(programming_language, type(programming_language))
([' ', '[', '"', 'P', 'y', 't', 'h', 'o', 'n', '"', ',', ' ', '"',
'L', 'i', 's', 'p', '"', ',', '"', 'C', '+', '+', '"', ']'], <type
'list'>)
>>>
```

© 2011 - 2017, Bernd Klein, Bodenseo; Design by Denise Mitchinson adapted for python-course.eu by Bernd Klein