More About Input Functions

Now, you may have noticed that most of the Python code samples we've used include the line

#!/usr/bin/env python3

Now, this is important, because it sets the Python version to Python 3.

There are some subtle differences in how data streams are handled in Python 3 and older versions, such as Python 2. Let's just focus on input() and raw_input(), because they work differently in Python 2 and 3, and you would want to use one or the other depending on the Python version.

In Python 2

Taking an input from a user, raw_input should be used:

```
>>> my_number = raw_input('Please Enter a Number: \n')
Please Enter a Number:
1337
>>> print(my_number)
1337
>>>
```

Now, this is important, because, raw_input does *not* evaluate an otherwise valid Python expression. In simple terms, raw_input will just get a string from a user, where input will actually perform basic maths and the like. See below:

```
>>> my_raw_input = raw_input('Please Enter a Number: \n')
Please Enter a Number:
123 + 1  # This is treated like a raw string.
>>> my_input = input('Please Enter a Number: \n')
Please Enter a Number:
123 + 1 # This is treated like an expression.
>>> print(my_raw_input)
123 + 1
>>> print(my_input)
124 # See that the expression was evaluated!
```

In Python 2 input(x) is just eval(raw_input(x)). eval() will just evaluate a generic string as if it were a Python expression.

In Python 3

Taking an input from a user, input should be used. See the below sample:

```
>>> my_number = input('Please Enter a Number: \n')
Please Enter a Number:
123 + 1
>>> print(my_number)
123 + 1
>>> type(my_number)
<class 'str'>

```

Notice that the expression is treated just like a string. It is not evaluated. If we want to, we can call eval() and that will actually execute the string as an expression:

execute the string as an expression.

```
>>> my_number = input('Please Enter a Number: \n')
Please Enter a Number:
123 + 1
>>> print(my_number)
123 + 1
>>> eval(my_number)
124
```

Finally, it’s worth noting, raw_input doesn’t natively exist in Python 3, but there are some tricky ways to force the interpreter to evaluate raw_input in backwards compatible ways. This can be useful for modernizing legacy Python code without rewriting large portions of it. Research on this topic is better left to the reader, as there are lots of fun (and sometimes scary) ways of doing this.

Summary

Python 2 and Python 3 handle input and raw_input differently.

In Python 2

- input(x) is roughly the same as eval(raw_input(x))
- raw_input() is preferred, unless the author wants to support evaluating string expressions.
- eval() is used to evaluate string expressions.

Standard Library Docs:

- https://docs.python.org/2/library/functions.html#input
- https://docs.python.org/2/library/functions.html#raw_input
- https://docs.python.org/2/library/functions.html#eval

In Python 3

- Input handles string as a generic string. It does not evaluate the string as a string expression.
- raw_input doesn’t exist, but with some tricky techniques, it can be supported.
- eval() can be used the same as Python 2.

Standard Library Docs:

- https://docs.python.org/3/library/functions.html#input
- https://docs.python.org/3/library/functions.html#eval