



# 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:

```
1  >>> my_number = raw_input('Please Enter a Number: \n')
2  Please Enter a Number:
3  1337
4  >>> print(my_number)
5  1337
6  >>>
```

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:

```
1  >>> my_raw_input = raw_input('Please Enter a Number: \n')
2  Please Enter a Number:
3  123 + 1 # This is treated like a raw string.
4  >>> my_input = input('Please Enter a Number: \n')
5  Please Enter a Number:
6  123 + 1 # This is treated like an expression.
7  >>> print(my_raw_input)
8  123 + 1
9  >>> print(my_input)
10 124 # 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:

```
1  >>> my_number = input('Please Enter a Number: \n')
2  Please Enter a Number:
3  123 + 1
4  >>> print(my_number)
5  123 + 1
6  >>> type(my_number)
7  <class 'str'>
8
```

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:

```
1  >>> my_number = input('Please Enter a Number: \n')
2  Please Enter a Number:
3  123 + 1
4  >>> print(my_number)
5  123 + 1
6  >>> eval(my_number)
7  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