

I/O: Input-Output

I/O is short for input-output.

Input usually means the data entered by the **end-user** of the program.

When you use Microsoft Word, you are playing the role of the end-user (sometimes shortened to just plain 'user'). The player of a video game is its 'end-user'.

When programming, we often switch between playing the role of programmer and end-user.

We're the end-user when we are testing whether the program works. We usually call the program itself the system, as in 'when the end-user enters a number, the system responds by...'

Input can also come from a file, e.g.,

when you type 'python someprog.py' the program called 'python' is getting its input from a file called someprog.py.

Output is what the program produces.

For command-line programs, its just whatever is printed on the screen.

Of course for programs with graphical user interfaces, its much more complex.

User Input in Python

Python has two key functions to deal with end-user input, one called `raw_input()` and one called `input()`. When you call these functions, the system will wait for the end-user to enter something.

The system returns the user's entry as a string result from the function.

If you execute the statement:

```
>> result = raw_input()
```

Python will wait for you (the end-user) to enter something. Enter some text and hit return. Then type

```
>> print result
```

This should print whatever the end-user typed.

Generally, we want to prompt the user for input. You can do this by adding a parameter to the 'raw_input()' call. Open a text editor and add create a file with the following lines:

```
x = raw_input('please enter a string:')  
print x
```

will look like:

```
please enter a string: hello  
'hello'
```

User input and types

The function raw-input() always returns a string. So if the user enters 7, Python interprets it as the string '7'.

Sometimes this is fine, but often we want to perform computations on numbers, and as far as the computer is concerned, '7' is a symbol and not a number.

If you want to 'interpret' input as an integer, you need to convert it. Python provides some type conversion functions to do so, such as:

```
x=int('7') # puts the integer 7 into x.  
f=float('7.7') #puts the floating point number 7.7 into f.
```

Using a type converter, one could get an integer from the user, and perform a computation on it, with the following:

```
s=raw_input('enter a number:')
x = int(s)
y = x+3
print y
```

This will print out 3 more than whatever the user enters.

Because this converting is a bit laborious, Python provides the 'input()' function, which converts things automatically. 'input()' basically looks at what the user enters, and automatically determines the correct type.

```
x = input('enter a number:')
y = x+3
```

The only problem with 'input' is you have to assume the user will enter what you are expecting. If they don't, you are in trouble. For instance, if a user entered 'xyz' for the input statement above, the program would bomb.

For now, just use the input() function when you are expecting an integer from the end-user, and raw_input when you are expecting a string. Later, will talk about validating user input in detail.

In-Class Assignment:

1. Mastermind beginnings: Write a python program that prompts the user for four colors, gets the input, and builds a list named 'guess' which has four elements of type string. Here's a sample of how your program should behave (with user input in bold)

```
Please enter a color: red
Please enter a color: blue
Please enter a color: green
Please enter a color: red
Your guess is: [red, blue, green, red]
```