Printing

Python offers several ways to print information. This handout presents two, using the `print` statement.

Unformatted Printing

Unformatted printing prints numbers and strings in a natural format: integers as ordinary numbers, floating point numbers with two decimal digits, and strings as they are typed. Thus:

```
num = 1;
flnum = 2.6;
print("The integer is", num, "and the float is", flnum)
```

prints

```
The integer is 1 and the float is 2.6
```

A comma between two arguments prints a space. The keyword argument `end=' '` puts a blank (or whatever string follows the `=` sign) at the end of the line and suppresses the skip to the next line:

```
print("Here is one print statement", end=' ')
print("And here is another")
```

prints

```
Here is one print statement And here is another
```

One problem with unformatted printing is that you cannot avoid adding spaces before numbers. So, if you try to print "$1.39" with this:

```
print("$", 1.39)
```

you get

```
$ 1.39
```

(notice the unwanted space). To print this, you need a formatted print statement.

Formatted Print

A formatted print statement allows you to control how the number or string is printed. You can do lots of things with it that you cannot do with an unformatted print statement.

Let's start with the last unformatted print problem. We want to print "$1.39". Here's how we do it:

```
print("$%f" % (1.39))
```

This prints

```
$1.390000
```

The string immediately following the printf is the *format string*. It is printed as typed, except when a "%" is seen. The sequence of characters following it control how the next arguments are to be printed. "%f" means to print a floating point number. "%d" would mean to print an integer as a decimal number ("d" stands for "decimal"). The "%" after the string says that a tuple containing the arguments for the printing follow.[1]

One problem with the above—too many decimal places. Let's restrict it to 2 decimal places by saying instead

```
print("$%.2f" % (1.39))
```

which prints

```
$1.39
```

[1] Actually, if there is only one argument to be printed, you can omit the parentheses. As it's a 1-tuple, the parentheses are unnecessary in this case. But if your tuple has more than one element, you *must* put the parentheses in.

The "%f" says to print the number as a floating point number. The ".2" between the "%" and "f" means to print 2 digits after the decimal point. As another example, if you want to print the value of π to 7 decimal places, say:

```
import math
print("The value of pi is %.7f" % (math.pi))
```

You get

```
The value of pi is 3.1415927
```

Now let's say we want to print several arguments. You do it this way:

```
print("%d + %d = %d" % (2, 2, 2 + 2))
```

gives

```
2 + 2 = 4
```

You can get very fancy:

```
print('%(language)s has %(#)03d quote types.' % {"#":2, 'language':"Python"})
```

gives

```
Python has 002 quote types.
```

What Is Actually Going On

Actually, the arguments to print are simply a way to write a string. Ignore the word "print". Then the string in quotes is called the *format string* and the parenthesized list following the "%" are the values. The values are substituted into the format string, resulting in a new string. To see this, type the following to the IDLE interpreter:

```
x = "2 * %f = %f" % (math.pi, 2*math.pi)
print(x)
```

and you will see

```
2 * 3.141593 = 6.283185
```