Formatted Strings and Printing

You can control how the *print( )* function in Python works. The function can print several fields, e.g.

```
print(x, y, z)
```

This statement will print x, y, and z, whatever they are, with a space between x and y and another space between y and z, and it then terminates the output line, so the next print statement prints on a new line.

The parameters sep and end control what print( ) places between the fields and what it places at the end of the line of output. By default, sep=" " (a single space) and end="\n" (the newline character, which ends the current line and starts a new one). We can change those to anything we want.

```
        print( "Billy", "Bob")
```

prints

```
                Billy Bob
```

but `print( "Billy", "Bob", sep="")`

prints

```
                BillyBob
```

Similarly,

```
print( "John", "George")
print( "Paul", "Ringo" )
```

will print

```
John George
Paul Ringo
```

while

```
print( "John", "George", sep="***", end="#")
print( "Paul", "Ringo", sep=!!!")
```

will print

```
John***George#Paul!!!Ringo
```

Formatted Strings have the form

pattern %(*values*)

The pattern is allowed to have placeholders:

- %d is a placeholder for an integer
- %s is a placeholder for a string
- %f is a placeholder for a float

The placeholders get their values from the list of values. For example if variable **who** is "Mom" and variable **howMany** is 5

```
"Send %s %d flowers"%(who,howMany)
```

is

```
"Send Mom 5 flowers"
```

The print statement in fancy.py in Lab 1 could have been written

```
print( 'Welcome back, %s "%s" %s!' %(first, nick, last))
```

Placeholders can even assign fieldwidths to their values. Placeholder %5d says to use 5 spaces for whatever value goes in for this placeholder, and pad with blanks if it needs less than 5.   If you just say print(x, y, z) twice and the first time the values are 1, 2, 3 and the second time 100, 200, 300, the output looks like

```
1 2 3
100 200 300
```

If your print statement is

```
        print( "%5d  %5d  %5d"%(x, y, z))
```

your output will be

```
    1      2      3
  100    200    300
```

Your output is coming out in columns!

The float placeholder %f can even specify how many decimal places to use:

```
%6.3f
```

says to use at least 6 spaces for the float, with 3 after the decimal point.

If we say print("pi is %6.3f" % 3.1415926535)
it will actually print

```
pi is  3.142
```