LOGICAL AND PHYSICAL LINE

A physical line is what you *see* when you write the program. A logical line is what *Python sees* as a single statement. Python implicitly assumes that each *physical line* corresponds to a *logical line*.

An example of a logical line is a statement like `print 'hello world'` - if this was on a line by itself (as you see it in an editor), then this also corresponds to a physical line.

Implicitly, Python encourages the use of a single statement per line which makes code more readable.

If you want to specify more than one logical line on a single physical line, then you have to explicitly specify this using a semicolon (;) which indicates the end of a logical line/statement. For example:

```
i = 5
print i
```

is effectively same as

```
i = 5;
```

```
print i;
```

which is also same as

```
i = 5; print i;
```

and same as

```
i = 5; print i
```

However, I **strongly recommend** that you stick to **writing a maximum of a single logical line on each single physical line**. The idea is that you should never use the semicolon. In fact, I have *never* used or even seen a semicolon in a Python program.

There is one kind of situation where this concept is really useful: if you have a long line of code, you can break it into multiple physical lines by using the backslash. This is referred to as *explicit line joining*:

```
s = 'This is a string. \
This continues the string.'
print s
```

Output:

```
This is a string. This continues the string.
```

Similarly,

```
print \
i
```

is the same as

```
print i
```

Sometimes, there is an implicit assumption where you don't need to use a backslash. This is the case where the logical line has a starting parenthesis, starting square brackets or a starting curly braces but not an ending one. This is called **implicit line joining**. You can see this in action when we write programs using lists in later chapters.

Source: http://www.swaroopch.com/notes/python/