NUMBERS AND STRINGS

NUMBERS

Python supports five basic numerical types, three of which are integer types.

- int (signed integers)
    - ❍ long (long integers)
    - ❍ bool (Boolean values)
- float (floating point real numbers)
- complex (complex numbers)
    - ❖ Numeric types of interest are the Python long and complex types. Python long integers should not be confused with C longs. Python longs have a capacity that surpasses any C long. You are limited only by the amount of (virtual) memory in your system as far as range is concerned. If you are familiar with Java, a Python long is similar to numbers of the Big Integer class type.
    - ❖ Moving forward, ints and longs are in the process of becoming unified into a single integer type. Beginning in version 2.3, overflow errors are no longer reported the result is automagically converted to a long. In a future version of Python, the distinction will be seamless because the trailing "L" will no longer be used or required.
    - ❖ Boolean values are a special case of integer. Although represented by the constants true and False, if put in a numeric context such as addition with other numbers, true is treated as the integer with value 1, and False has a value of 0.

- Complex numbers (numbers that involve the square root of -1, so-called "imaginary" numbers) are not supported in many languages and perhaps are implemented only as classes in others. There is also a sixth numeric type, decimal, for decimal floating numbers, but it is not a built-in type. You must import the decimal module to use these types of numbers. They were added to Python (version 2.4) because of a need for more accuracy. For example, the number 1.1 cannot be accurately representing with binary floating point numbers (floats) because it has a repeating fraction in binary. Because of this, numbers like 1.1 look like this as a float

```
>>> 1.1
1.1000000000000001
>>> print decimal.Decimal('1.1')
1
```

STRINGS

Strings in Python are identified as a contiguous set of characters in between quotation marks. Python allows for either pairs of single or double quotes. Triple quotes (three consecutive single or double quotes) can be used to escape special characters. Subsets of strings can be taken using the index ([ ]) and slice ([:]) operators, which work with indexes starting at 0 in the beginning of the string and working their way from -1 at the end. The plus (+) sign is the string concatenation operator, and the asterisk (*) is the repetition operator. Here are some examples of strings and string usage:

```
>>> pystr = 'Python'
>>> iscool = 'is cool!'
>>> pystr[0]
```

```
'P'
>>> pystr[2:5]
 'tho'
>>> iscool[:2]
 'is'
>>> iscool[3:]
'cool!'
>>> iscool[-1]
 '!'
>>> pystr + iscool
 'Pythonis cool!'
>>> pystr + ' ' + iscool
'Python is cool!'
>>> pystr * 2
'PythonPython' >
>> '-' * 20
 '--------------------'
>>> pystr = '''python
 ... is cool'''
 >>> pystr
'python\nis cool'
>>> print pystr
Python
 is cool >>>
```