Variable Assignment

Assignment Operator

The equal sign (=) is the main Python assignment operator. In Python, objects are referenced, so on assignment, a reference (not a value) to an object is what is being assigned, whether the object was just created or was a pre-existing object.

```
anInt = -12
aString = 'cart'
aFloat = -3.1415 * (5.0 ** 2)
anotherString = 'shop' + 'ping'
aList = [3.14e10, '2nd elmt of a list', 8.82-4.371j]
```

Augmented Assignment

Augmented assignment refers to the use of operators, which imply both an arithmetic operation as well
as an assignment.

```
+= -= *= /= %= **=
<<= >>= &= ^= |=
```

Multiple Assignment

```
>>> x = y = z = 1
>>> x
1
>>> y
1
>>> z
1
```

An integer object (with the value 1) is created, and `x`, `y`, and `z` are all assigned the same reference to that object. This is the process of assigning a single object to multiple variables. It is also possible in Python to assign multiple objects to multiple variables.

"Multuple" Assignment

multiple variables is using what we shall call the "multuple" assignment. This is not an official Python term, but we use "multuple" here because when assigning variables this way, the objects on both sides of the equal sign are tuples.

```
>>> x, y, z = 1, 2, 'a string'
>>> x
1
>>> y
2
>>> z
'a string'
```

In the above example, two integer objects (with values 1 and 2) and one string object are assigned to `x`, `y`, and `z` respectively. Parentheses are

normally used to denote tuples, and although they are optional, we recommend them anywhere they make the code easier to read:

Identifier

Identifiers are the set of valid strings that are allowed as names in a computer language. From this all encompassing list, we segregate out those that are *keywords*, names that form a construct of the language. Such identifiers are reserved words that may not be used for any other purpose, or else a syntax error (`SyntaxError` exception) will occur.

Valid Python Identifiers

The rules for Python identifier strings are like most other high-level programming languages that come from the C world:

- First character must be a letter or underscore (_)
- Any additional characters can be alphanumeric or underscore
- Case-sensitive

No identifiers can begin with a number, and no symbols other than the underscore are ever allowed. The easiest way to deal with underscores is to consider them as alphabetic characters. *Case-sensitivity* means that identifier `foo` is different from `Foo`, and both of those are different from `FOO`.

List and tuples

Lists and tuples can be thought of as generic "arrays" with which to hold an arbitrary number of arbitrary Python objects. The items are ordered and accessed via index offsets, similar to arrays, except that lists and tuples can store different types of objects.

There are a few main differences between lists and tuples. Lists are enclosed in brackets (`[ ]`) and their elements and size can be changed. Tuples are enclosed in parentheses (`( )`) and cannot be updated (although their contents may be). Tuples can be thought of for now as "read-only" lists.
Subsets can be taken with the slice operator (`[]` and `[ : ]`) in the same manner as strings.

```
>>> aList = [1, 2, 3, 4]
>>> aList
[1, 2, 3, 4]
>>> aList[0]
1
>>> aList[2:]
[3, 4]
>>> aList[:3]
[1, 2, 3]
>>> aList[1] = 5
>>> aList
[1, 5, 3, 4]
```