Introduction to Programming in Python

Variables and Assignments

Dr. Bill Young
Department of Computer Science
University of Texas at Austin

Last updated: June 4, 2021 at 11:04

Assignment Statements

An assignment in Python has form:

Variable Name = Value

This means that variable is *assigned* **value**. I.e., after the assignment, **variable** "contains" **value**.

```
>>> x = 17.2
>>> y = -39
>>> z = x * y - 2
>>> print( z )
-672.8
```

Variables

A **variable** is a named memory location used to store values. We'll explain shortly how to name variables.

Unlike many programming languages, Python variables do not have associated types.

```
// C code
int x = 17;      // variable x has type int
x = 5.3;         // illegal
```

```
# Python code
x = 17           # x gets int value 17
x = 5.3          # x gets float value 5.3
```

A variable in Python actually holds a *pointer* (address) to an object, rather than the object itself.

Variables and Assignments

You can create a new variable in Python by assigning it a value. *You don't have to declare variables, as in many other programming languages.*

```
>>> x = 3                  # creates x, assigns int
>>> print(x)
3
>>> x = "abc"              # re-assigns x a string
>>> print(x)
abc
>>> x = 3.14               # re-assigns x a float
>>> print(x)
3.14
>>> y = 6                  # creates y, assigns int
>>> x * y                  # uses x and y
18.84
```

Meaning of a Variable

```
x = 17                  # Defines and initializes x
y = x + 3               # Defines y and initializes y
z = w                   # Runtime error if w undefined
```

This code defines three variables x, y and z. Notice that on the *left hand side* (lhs) of an assignment the variable is created (if it doesn't already exist), and given a value. On the lhs, it stands for a *location.*

On the *right hand side* (rhs) of an assignment, it stands for the current *value* of the variable. If there is none, it's an error.

Naming Variables

Below are (most of) the rules for naming variables:

- Variable names must begin with a letter or underscore ("_") character.
- After that, use any number of letters, underscores, or digits.
- Case matters: "score" is a different variable than "Score."
- You can't use *reserved words*; these have a special meaning to Python and cannot be variable names.

Naming Variables

Python Reserved Words:

> *and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, nonlocal, None, not, or, pass, raise, return, True, try, while, with, yield*

IDLE and many IDEs display reserved words in color to help you easily recognize them.

Not Reserved, but Don't Use

Function names like `print` are *not* reserved words. But using them as variable names is *a very bad idea* because it redefines them.

```
>>> x = 17
>>> print(x)
17
>>> print = 23
>>> print(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
```

Naming Variables

```
>>> ___ = 10                          # wierd but legal
>>> _123 = 11                         # also wierd
>>> ab_cd = 12                        # perfectly OK
>>> ab|c = 13                         # illegal character
  File "<stdin>", line 1
SyntaxError: can't assign to operator
>>> assert = 14                       # assert is reserved
  File "<stdin>", line 1
    assert = 14
           ^
SyntaxError: invalid syntax
>>> maxValue = 100                    # good one
>>> print = 8                         # legal but ill-advised
>>> print( "abc" )                    # we've redefined print
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
```

Naming Variables

In addition to the rules, there are also some conventions that good programmers follow:

- Variable names should begin with a lowercase letter.
- Choose meaningful names that describe how the variable is used. This helps with program readibility.

  Use `max` rather than `m`.
  Use `numberOfColumns` rather than `c`.
- One exception is that loop variables are often i, j, etc.

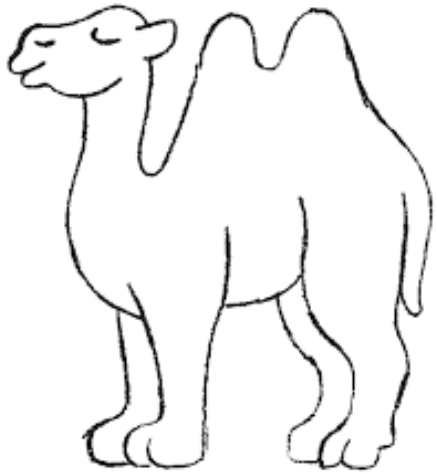
```
for x in lst: print( x )
```

rather than:

```
for listItem in lst: print( listItem )
```

Camel Casing

If you use a multi-word names (good practice), I prefer "camel casing": `avgHeight`, `countOfItems`, etc. Others prefer PEP-8: `avg_height`, `count_of_items`, etc.



These are just conventions; you'll see lots of counterexamples in real code. Adopt a style and use it!