

In other words, this equation is saying, “add 5 to 30, then multiply the result by 20, and divide that result by 10.” Here’s what happens:

- Adding 5 to 30 gives 35.
- Multiplying 35 by 20 gives 700.
- Dividing 700 by 10 gives the final answer of 70.

If we had not used parentheses, the result would be slightly different:

---

```
>>> 5 + 30 * 20 / 10
65.0
```

---

In this case, 30 is first multiplied by 20 (giving 600), and then 600 is divided by 10 (giving 60). Finally, 5 is added to get the result of 65.

**WARNING** *Remember that multiplication and division always go before addition and subtraction, unless parentheses are used to control the order of operations.*

## VARIABLES ARE LIKE LABELS

The word *variable* in programming describes a place to store information such as numbers, text, lists of numbers and text, and so on. Another way of looking at a variable is that it’s like a label for something.

For example, to create a variable named `fred`, we use an equal sign (=) and then tell Python what information the variable should be the label for. Here, we create the variable `fred` and tell Python that it labels the number 100 (note that this doesn’t mean that another variable can’t have the same value):

---

```
>>> fred = 100
```

---

To find out what value a variable labels, enter `print` in the shell, followed by the variable name in parentheses, like this:

---

```
>>> print(fred)
100
```

---

We can also tell Python to change the variable `fred` so that it labels something else. For example, here's how to change `fred` to the number 200:

---

```
>>> fred = 200
>>> print(fred)
200
```

---

On the first line, we say that `fred` labels the number 200. In the second line, we ask what `fred` is labeling, just to confirm the change. Python prints the result on the last line.

We can also use more than one label (more than one variable) for the same item:

---

```
>>> fred = 200
>>> john = fred
>>> print(john)
200
```

---

In this example, we're telling Python that we want the name (or variable) `john` to label the same thing as `fred` by using the equal sign between `john` and `fred`.

Of course, `fred` probably isn't a very useful name for a variable because it most likely doesn't tell us anything about what the variable is used for. Let's call our variable `number_of_coins` instead of `fred`, like this:

---

```
>>> number_of_coins = 200
>>> print(number_of_coins)
200
```

---

This makes it clear that we're talking about 200 coins.

Variable names can be made up of letters, numbers, and the underscore character (`_`), but they can't start with a number. You can use anything from single letters (such as `a`) to long sentences for variable names. (A variable can't contain a space, so use an underscore to separate words.) Sometimes, if you're doing something quick, a short variable name is best. The name you choose should depend on how meaningful you need the variable name to be.

Now that you know how to create variables, let's look at how to use them.

## USING VARIABLES

Remember our equation for figuring out how many coins you would have at the end of the year if you could magically create new coins with your grandfather's crazy invention in the basement? We have this equation:

---

```
>>> 20 + 10 * 365
3670
>>> 3 * 52
156
>>> 3670 - 156
3514
```

---

We can turn this into a single line of code:

---

```
>>> 20 + 10 * 365 - 3 * 52
3514
```

---

Now, what if we turn the numbers into variables? Try entering the following:

---

```
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
```

---

These entries create the variables `found_coins`, `magic_coins`, and `stolen_coins`.

Now, we can reenter the equation like this:

---

```
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3514
```

---

You can see that this gives us the same answer. So who cares, right? Ah, but here's the magic of variables. What if you stick a scarecrow in your window, and the raven steals only two coins

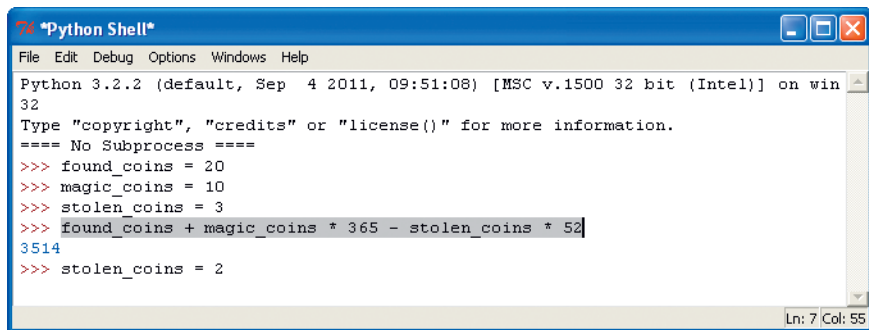


instead of three? When we use a variable, we can simply change the variable to hold that new number, and it will change everywhere it is used in the equation. We can change the `stolen_coins` variable to 2 by entering this:

```
>>> stolen_coins = 2
```

We can then copy and paste the equation to calculate the answer again, like so:

1. Select the text to copy by clicking with the mouse and dragging from the beginning to the end of the line, as shown here:

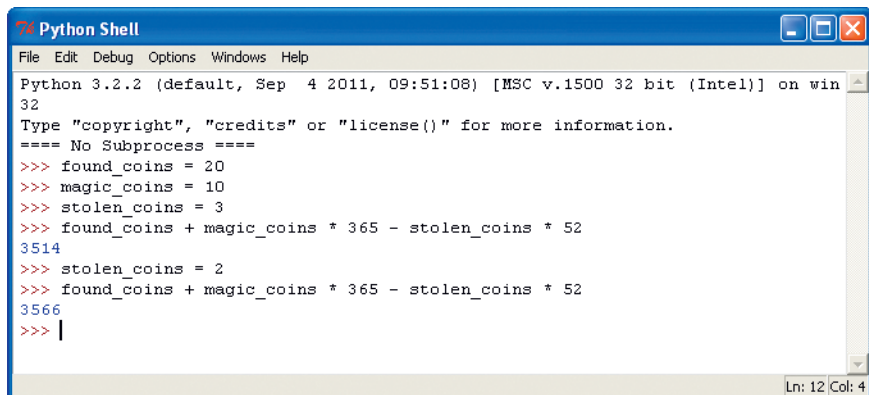


A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Debug", "Options", "Windows", and "Help". The main text area shows the following text:

```
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3514
>>> stolen_coins = 2
```

The line `>>> found_coins + magic_coins * 365 - stolen_coins * 52` is selected with the mouse. The status bar at the bottom right shows "Ln: 7 | Col: 55".

2. Hold down the CTRL key (or, if you're using a Mac, the ⌘ key) and press C to copy the selected text. (You'll see this as CTRL-C from now on.)
3. Click the last prompt line (after `stolen_coins = 2`).
4. Hold down the CTRL key and press V to paste the selected text. (You'll see this as CTRL-V from now on.)
5. Press ENTER to see the new result:



A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Debug", "Options", "Windows", and "Help". The main text area shows the following text:

```
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3514
>>> stolen_coins = 2
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3566
>>> |
```

The status bar at the bottom right shows "Ln: 12 | Col: 4".

Isn't that a lot easier than retyping the whole equation? It sure is.

You can try changing the other variables, and then copy (CTRL-C) and paste (CTRL-V) the calculation to see the effect of your changes. For example, if you bang the sides of your grandfather's invention at the right moment, and it spits out an extra 3 coins each time, you'll find that you end up with 4661 coins at the end of the year:

---

```
>>> magic_coins = 13
>>> found_coins + magic_coins * 365 - stolen_coins * 52
4661
```

---

Of course, using variables for a simple equation like this one is still only *slightly* useful. We haven't gotten to *really* useful yet. For now, just remember that variables are a way of labeling things so that you can use them later.

## WHAT YOU LEARNED

In this chapter you learned how to do simple equations using Python operators and how to use parentheses to control the order of operations (the order in which Python evaluates the parts of the equations). We then created variables to label values and used those variables in our calculations.