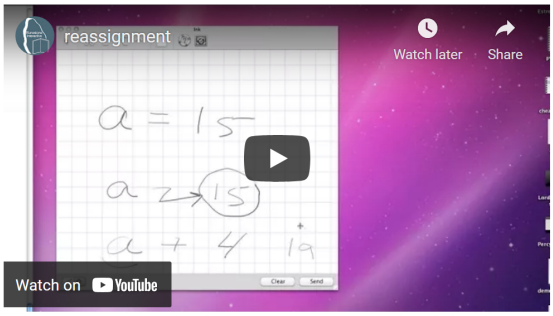
2.12. Reassignment



Activity: 1 -- Video: (reassignmentvid)

As we have mentioned previously, it is legal to make more than one assignment to the same variable. A new assignment makes an existing variable refer to a new value (and stop referring to the old value).

Save & Run 4/17/2021, 12:37:35 PM - 2 of 2 Show in CodeLens

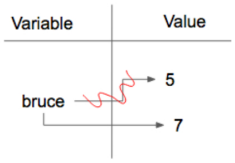
```
bruce = 5
print(bruce)
bruce = 7
print(bruce)

```

```
5
7
```

Activity: 2 -- ActiveCode (ac2_13_1)

The first time `bruce` is printed, its value is 5, and the second time, its value is 7. The assignment statement changes the value (the object) that `bruce` refers to.

Here is what **reassignment** looks like in a reference diagram:



It is important to note that in mathematics, a statement of equality is always true. If `a is equal to b` now, then `a will always equal to b`. In Python, an assignment statement can make two variables refer to the same object and therefore have the same value. They appear to be equal. However, because of the possibility of reassignment, they don't have to stay that way:

Save & Run 4/17/2021, 12:37:44 PM - 2 of 2 Show in CodeLens

```
a = 5
b = a    # after executing this line, a and b are now equal
print(a,b)
a = 3    # after executing this line, a and b are no longer equal
print(a,b)

```

```
5 5
3 5
```

Activity: 3 -- ActiveCode (ac2_13_2)

Line 4 changes the value of `a` but does not change the value of `b`, so they are no longer equal. We will have much more to say about equality in a later chapter.

2.12.1. Developing your mental model of How Python Evaluates

Its important to start to develop a good mental model of the steps the Python interpreter takes when evaluating an assignment statement. In an assignment statement, the interpreter first evaluates the code on the right hand side of the assignment operator. It then gives a name to whatever that is. The (very short) visualization below shows what is happening.

Next Step Reset

```
a = 5
b = a
```

```
a = 5
b = a
b = 5
```

Activity: 4 -- ShowEval (se_ac2_13_1a)

In the first statement `a = 5` the literal number 5 evaluates to 5, and is given the name `a`. In the second statement, the variable `a` evaluates to 5 and so 5 now ends up with a second name `b`.

You can step through the code and see how the variable assignments change below.

Python 3.3

```
1 a = 5
2 b = a    # after executing this line, a and b are now equal
3 print(a,b)
4 a = 3    # after executing this line, a and b are no longer equal
5 print(a,b)
```

<< First < Back Program terminated Forward > Last >>

line that has just executed
next line to execute

Visualized using Online Python Tutor by Philip Guo

Frames Objects

Global frame

a	3
b	5

Program output:

```
5 5
3 5
```

Activity: 5 -- CodeLens: (clens2_13_1)

Note

In some programming languages, a different symbol is used for assignment, such as `<-` or `:=`. The intent is that this will help to avoid confusion. Python chose to use the tokens `=` for assignment, and `==` for equality. This is a popular choice also found in languages like C, C++, Java, and C#.

Check your understanding

data-13-1: After the following statements, what are the values of x and y?

```
x = 15
y = x
x = 22
```

- A. x is 15 and y is 15
- B. x is 22 and y is 22
- C. x is 15 and y is 22
- D. x is 22 and y is 15

Check me Compare me

✔ Yes, x has the value 22 and y the value 15.

Activity: 6 -- Multiple Choice (question2_13_1)

You have attempted 6 of 6 activities on this page

2.11. Order of Operations">
2.11. Order of Operations">
2.13. Updating Variables">
✔ Completed. Well Done!
2.13. Updating Variables">Next Section - 2.13. Updating Variables

| Back to top