[Skip to main content](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/courseware/a71234edfd4f465e8d3e7e0082d04540/fa25de4817d044fea3b65cf24587a31b/3?activate_block_id=block-v1%3AIITBombayX%2BCS101.2x%2B2022_T1%2Btype%40vertical%2Bblock%404c98fb4b34b646f6a84c7cbc99afc968" \l "main)

**[IITBombayX Home Page](http://iitbombayx.in/)**

**IITBombayX: CS101.2xObject-Oriented Programming**

[Courses](http://iitbombayx.in/courses)

[Help](https://iitbombayx.in/contact_us)

[[https://courses.iitbombayx.in/static/images/profiles/default_50.3455a6581573.png](https://courses.iitbombayx.in/dashboard) Dashboard for:Paramhans\_Singh](https://courses.iitbombayx.in/dashboard)

1. [Course, current location](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/)

1. [Discussion](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/discussion/forum/)

1. [Progress](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/progress)

1. [Grading Policy](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/487375f5363a49a492641b63d5e814ec/)

1. [Course Slides](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/pdfbook/0/)

1. [Practice Programming](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/b5c2b3240ef9416ebb0ea5b2087cb4ee/)

1. [FAQs](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/f2e9c32b35ae4ef083ed716b4981ff0d/)

[Course](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/)  [Topic 2: Review of Pointers and Structures](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/#block-v1:IITBombayX+CS101.2x+2022_T1+type@chapter+block@a71234edfd4f465e8d3e7e0082d04540)  [Session 3: Use of Pointers in C++ Programs](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/#block-v1:IITBombayX+CS101.2x+2022_T1+type@sequential+block@fa25de4817d044fea3b65cf24587a31b)  Walkthrough

 PreviousNext

1. video Use of Pointers in C++ Programs
2. other Slides: Use of Pointers in C++ Programs
3. other Walkthrough
4. other Walkthrough

**Walkthrough**

 Bookmark this page

**Behaviour of Variables in Python**

**Objective:**

So far we have seen the behaviour of the variables only specific to C++ programming language. However, there are other chief programming languages used nowadays where the behaviour of variables is completely different. Hence, it's good to know the behaviour of variables in these programming languages (like in Python).

**Let's start:**

**Behaviour of Variables in C++ :**

When you create a variable in C++, on a higher abstraction level you can assume that the compiler creates a box for the variable and if we put a value in that variable the value is stored in that box. If we assign it some other value, then the old value is removed and a new value is added to the box. To make this more concrete, let's look at one example.

int a = 1;

A box is created and a value of 1 is put in that box.

Now if you do a = 2, then

The value 1 is removed and 2 is added to the box.

Now if you do, int b = a, then C++ creates a new box for **b** and copies the value of **a** to that box.

So overall, the point that we are trying to put is that both the variables **a** and **b** have different addresses and have no relation despite the fact that they carry the same value.

**Behaviour of Variables in Python:**

In Python, the things go in a bit different way. First of all, there is no concept of data-type at least when creating variables. Just doing a = 1 or a = 'hello' works. Now, writing a = 1, it assigns memory to value 1 and puts a tag of variable a to that value. If you change the value of the variable, it just changes the tag to the new value in the memory. Now, if you do b = a, then it does not create a new space, it just puts a tag of **b** on the same value.

So far we have discussed the higher abstraction of handling of variables done by python. Now, let's dive deeper and see what actually happens. For the latter part of the walkthrough, we will be using a function id() extensively. For our purpose, this function returns the memory address of the variable.

Now, let's see an example.

a = 10

b = 10

c = 10

print (id(a), id(b), id(c))

Firstly, to clear your doubts, we don't need to provide semicolons at the end of lines and print function is the same as cout function of C++.  So, assuming it to be a valid Python code what do you think will be the addresses of **a**, **b** and **c**. Surprisingly, all three addresses are the same. This might get a wrong intuition among C++ users that if these addresses are same, then changing one would change the others as well. But this is not true in python. a = 11, actually assigns memory to variable 11 and puts the tag of **a** there i.e. address of **a** is no more same as that of **b** and **c**.

This happens because unlike C++, variable is just a name in Python. Values are sprinkled in the heap, and variables just refer to this values. Multiple variables (names) can refer to the same value. Assigning a variable a value simply makes it to refer to that value. Hence other variables referring to the same value remain referring to the same value.

Now, one extremely weird behaviour that you will encounter is this:

a = 500

b = 500

print(id(a),id(b))

The id/addresses are no more the same. It's because of CPython optimisation techniques. What happens is that Cpython maintains an array of integer objects in the range -5 to 256 and if a variable is assigned value in this range then it simply back references to the address where these numbers are already stored (if any). This is what happens when we assign **a**, **b** and **c** to the value of 10. But when we assign values out of this range, python doesn't care about checking for the already created objects and just creates a new one. This type of behaviour is seen with other data types like strings as well.

**COnclusion:**

So, basically, the point we try to show here is that the variables in python behave a lot differently than other programming languages and it's necessary to know the way programming languages tend to optimise the variable memory handling with different heuristics.

Image Citation : http://foobarnbaz.com/2012/07/08/understanding-python-variables/

 PreviousNext

© All Rights Reserved

[IITBombayX Home Page](http://iitbombayx.in/)

1. [About](http://iitbombayx.in/about)
2. [Research](http://iitbombayx.in/research)
3. [FAQs](http://iitbombayx.in/faq)
4. [Contact Us](http://iitbombayx.in/contact_us)

* [MOOCs Copyright](http://iitbombayx.in/mooc-copyright-guidelines)
* [Terms of Service and Honour Code](http://iitbombayx.in/honor)
* [Authorization Document](http://iitbombayx.in/authorization)
* [Privacy Policy](http://iitbombayx.in/privacy)

[Powered by Drupal](https://www.drupal.org/)

[Creative Common License](https://creativecommons.org/)

[](http://open.edx.org/)