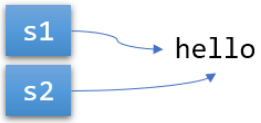
Mutability & Value Assignment

In C++ `string` objects are mutable; you may change the individual characters inside a **`string`** variable. Compare this with Java or Python, where string objects are immutable.

```
string str = "hello";
str[0] = 'j';
cout << str << endl;    // prints jello
```

In Java and in Python, assignment of object types means that the variables are copied, but that the objects are not. Here' a piece of Java code which creates a **`String s1`** and then creates a second, **`s2`** initialized with **`s1`**. The illustration shows what this looks like in memory.




```
String s1 = "hello";
String s2 = s1;
```

C++ works differently. In Java and Python, variables refer to objects; in C++ variables contain objects. In C++, assigning one **`string`** to another, **copies the underlying characters** into an entirely new **`string`**, in the same way that assigning one **`int`** variable to another creates a new, independent variable and value.

Languages (like C++) that work like this have **value semantics**. In C++, the statement

```
str1 = str2
```

overwrites any previous contents of **`str1`** with a **copy** of the characters contained in **`str2`**. The variables **`str1`** and **`str2`** therefore remain **independent**, which means that changing the characters in **`str1`** does not affect **`str2`**.