## 30.

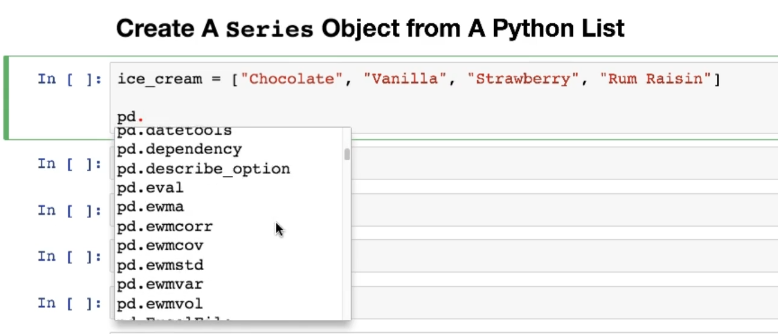
## 31. Create A Series Object from a Python List

Pandas series Objects. A series is a **one dimensional labeled array.** So the easiest way that I can describe it is sort of **like a more powerful version of a python list**.

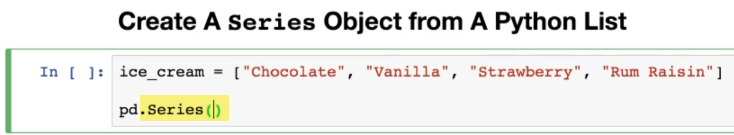
A Pandas series stores data in a **sequenced order**, and it's just one dimensional data. **It's like one column of information**.

Now, just like whenever we're working with any type of data, **consistency i**s the most important thing here. So our pandas series can consist of any data type. It can have string values or integer values or floating point number of values or boolean values. The data type doesn't matter, but **consistency should always be the goal.**

So in order to create a pandas series, object will have to call a method on our pandas library. **It's actually called the series method** and it f**alls into a category called Constructor Methods.** That name comes from the fact that these methods construct a brand new object for us to work with. And in our case, our new object will be a new series.

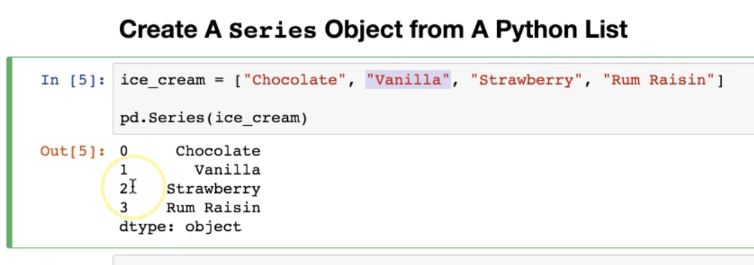


Whenever we have a library or an object, we always have to follow the reference with a dot or a period. This is called **dot notation**, and you can think of the dot as saying, go one level deep into this thing, go one layer down or step into the lobby of this library. At this point, we're kind of entering pandas and now we can issue a command.





You'll see this little **dtype** tidbit and that's **short for data type**. And that's going to tell us a little bit about the data type that **makes up the values** within my pandas series.

Now, if you're working with something like integers or floats, it's just going to say into or floats here. It says **object and object is actually the internal pandas lingo for string.**

Now, on the left here, we have an additional component and that's called the index. So the index in this case resembles basically an index for a python list.

Now, the one of the **key advantages of a pandas is over a python list is that the index labels do not have to be numeric.** They can be any data type.

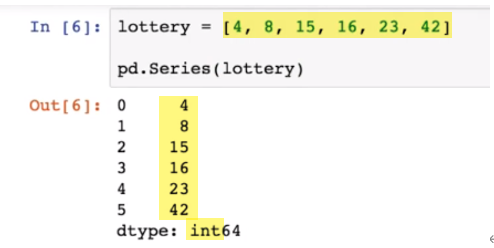
For now, the reason that pandas is giving us the default numeric one is because that's the standard process. That's the default start count at zero.

Now, just like with most things in programming like arrays or lists, that count is going to start at 0.

So because it doesn't start at one, but rather zero, the very last indexed position will always be one less than the total number of actual values within the series.

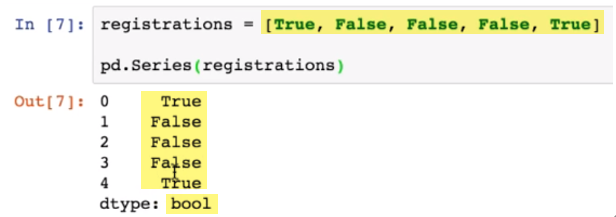
### More example: integer values within a series

Now, as I mentioned, **the values within this series do not have to be strings that can be anything**. Again, **this dtype is referring to the values, not the index**.



### More example: Boolean within a series

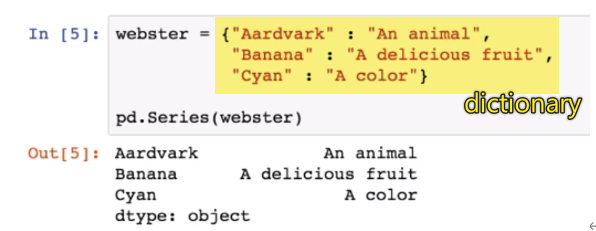
Once again, the values from the Python list are now the values in the series.



## 32. Create A Series Object from a Python Dictionary

The index does not have to be numeric. It can be any data type.

I'm going to create **one series object from a python dictionary**.



And to the left, you'll see that pandas has taken **the keys** from our dictionary and **turned them into the index labels within our series.**

On the bottom, we have our data type, which is listed as **object**. Again, this is **referring to the values of the series**. (wlc note: not to the index of the series)

So although **we have both a string index and a string values, this is referring to the values**.

So as we can see here, our **series object this time does not have a numeric index**.

### Advantage 1: index in python list vs. a series

So that's one of the advantages that **a series has over a python list that allows you to have identifiers or index labels that are not numeric.**

### Advantage 2: a series have over a dictionary

One of the advantages that **a series has over a dictionary is that the index labels in a series actually do not have to be unique.**

In this example they are, but as we'll see in a few lessons, you can **add multiple identical index labels throughout the series**.

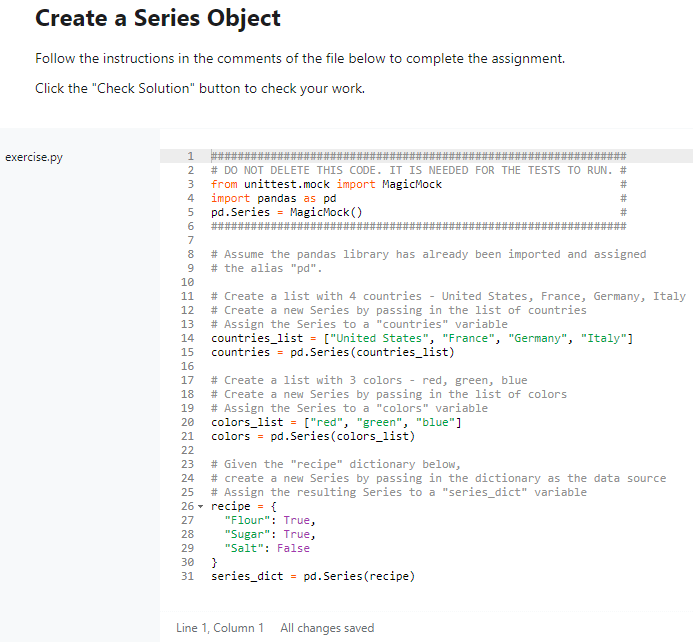
### The key takeaway here is:

A series is kind of like a **mishmash of a python list and a python dictionary that combines the best features of both of them and as a whole slew of available functionalities on top**.

In the next lesson, we'll start diving into some of the **attributes** that are available on a **serious object.**

## Coding Exercise 1: Create a

**Series Object**



## 33. Coding Exercise SOLUTION: Create a Series Object



## 34. Intro to Attributes on a Series Object

So objects in python **have attributes and they have methods.**

And a pandas series is just one type of object. It's a thing that exists within this language.

### Attributes within pandas object

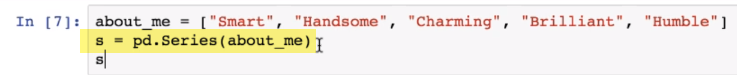
Now that we've imported pandas and **attributes return information about the objec**t, they deliver some kind of detail about it. They deliver perhaps a summary of it or a way to look at it. From a certain perspective, **attributes do not modify an object so they don't actually manipulate it in any way or destroy it or edit it**. All they do is kind of view it, observe it and give us some information about it.

That's **in comparison to methods which do operate upon an object. They do perform some kind of manipulation or operation or calculation.**



So what's happening here is that pandas is executing the very last line within the cell. So it's giving us that series, but it's not actually saving the series anywhere in memory. So once it's previewed to us on the screen, since it's not being stored in a variable, it's discarded from memory.

In order to use and reference the series later, we'll have to store it in a variable or rather point a variable to it.



### What happens from both side of the equal sign

And again, the easiest way to always remember this is **that whatever happens on the right side of the equals sign always takes place first**.

So the series is going to be generated first. It's going to use the values from the About Me list.

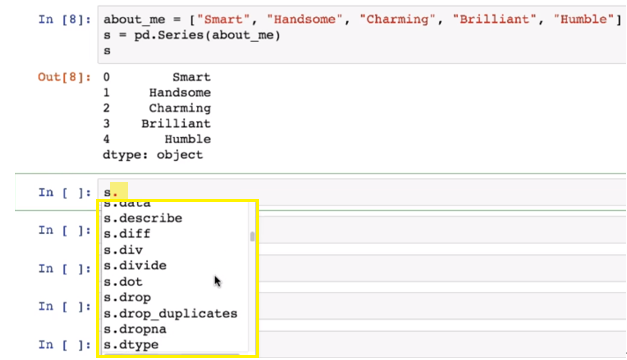
It's going to generate a numeric index because we didn't specify what index to use.



Once that series object is entirely constructed, it's going to be the variable s is going to be used to point to it. And then we're going to have our series stored in the variable is now simply executing.

**It's not exactly a method, so it's not going to require parentheses**, but otherwise the syntax is identical. We're going to begin with our object, which in this case is s we're then going to place a dot that's called **dot notation.**

### .value attribue

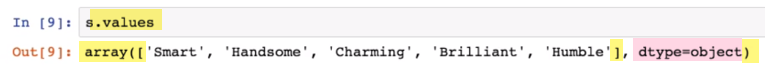


And there it is, so estate values, **it's an attribute. So it does not require parentheses at the end.**

**That's one syntactical difference between attributes and methods.**

And let's execute the sale and see what happens. So values is going to return an array of all of these values within the series.

So you can think of it as saying, take this little chunk, take this section of our series and just give it back to me as an array.



**Now, again, it's a attribute. So it's not manipulating, it's not changing those values. It's not shuffling them around. It's just giving them to me as an array.**

It's kind of like extracting that piece of the puzzle and outputting it to me.

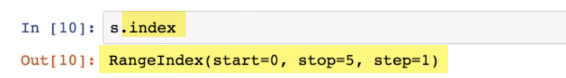
### .index attribute

Similarly, there's a complementary attribute called index. And **what that does is focus on the index component of our series. This is going to return an object. It's called the arranged index.**

So it's a little bit different from the values result above.

But basically think of all of these things as different puzzle pieces or different machine parts that come together to make the final series. And right now, we're kind of focusing on one of them at a time. So we're taking a look at the values and now we're taking a look at the object that makes up the index that we see on the left, right here.

And **the index is made up of its own separate object called range index, which comes from pandas.**



And here we can see a little bit of a description about how it operates. We can see that it starts at zero, which is true. We have our zero up here. It stops at five, which is true. It goes up to four and stops there. And this last part is called Step, and that's telling us that it's incrementing by one. So it's starting at zero and then going to one and then going to two and so on.

So this is the object that makes up the index in an actual data frame or rather our series.

### .dtype attribute

The dtype attribute, and that's just the chunk that we see down here. If we execute the self, it's going to give us a dtype of **O, which is short for object, which again is internal pandas lingo for a string.**



So these attributes we've used them to kind of take apart that big series and look at it chunk by chunk, piece by piece.

We took a look at the values of it by looking at the values attribute. We looked at the index of it by looking at the index attribute and we asked pandas to tell us about these type by using the dtype attribute. And all of these have been accessed on our series, which is stored in the s variable.

So again, **attributes are just a way** that we can use to find out information about the components that make up a series. Sometimes we'll dive into more attributes later, sometimes attributes. **Give us a little bit of a summary or description about a series, but they don't actually modify it or do anything with it. They're very passive.**

**That's in comparison to methods which do operate or perform some kind of operation or calculation.**

## 35. Intro to Methods on a Series Object