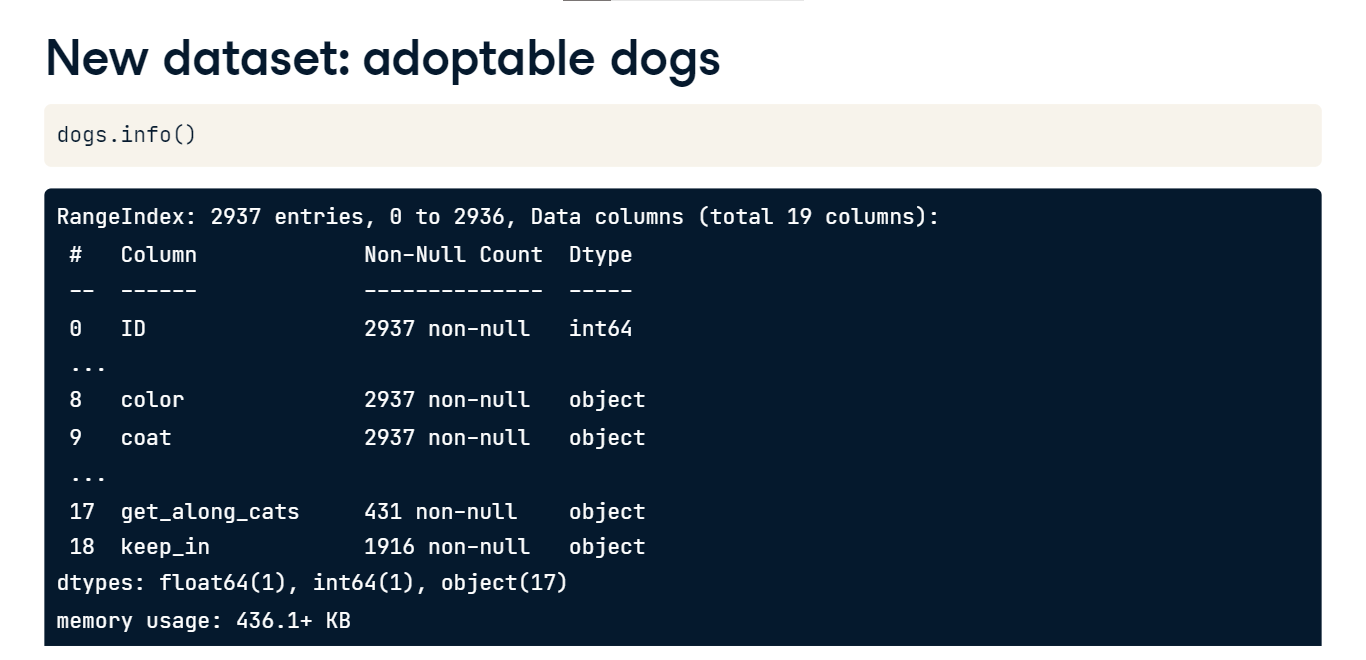
**Setting category variables**

To get the most out of using the pandas categorical dtype, we need to understand how to set, add, and remove categories.

**New dataset: adoptable dogs**

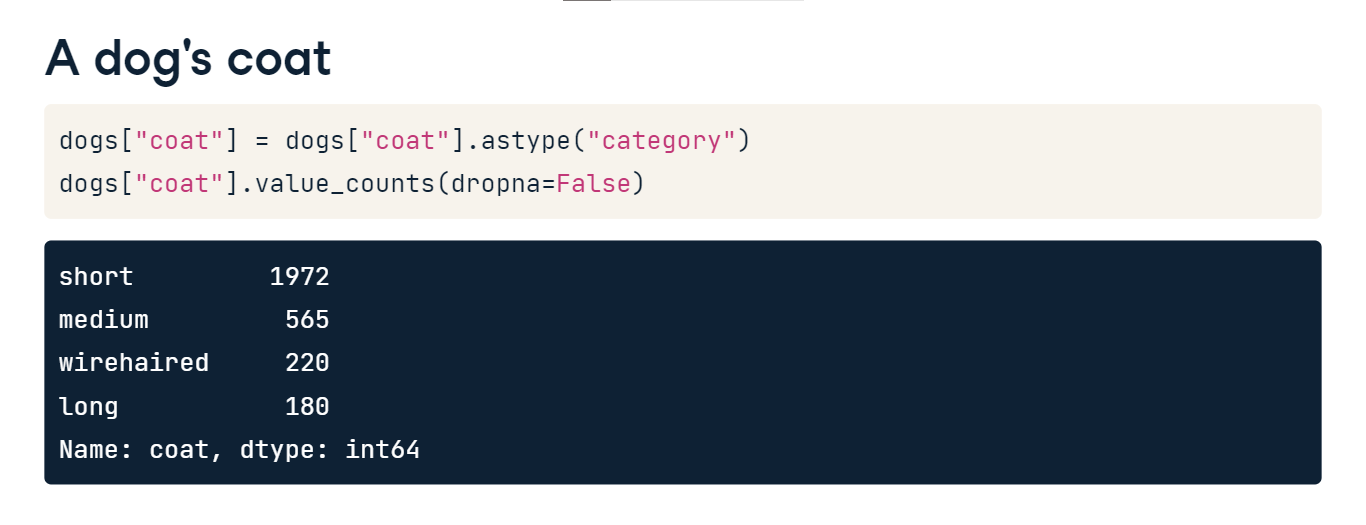
Before we begin, let's checkout another interesting dataset. The adoptable dogs dataset contains information on 2937 adoptable dogs and contains a lot of great categorical columns for us to explore.

1. 1 https://www.kaggle.com/jmolitoris/adoptable-dogs

**as**

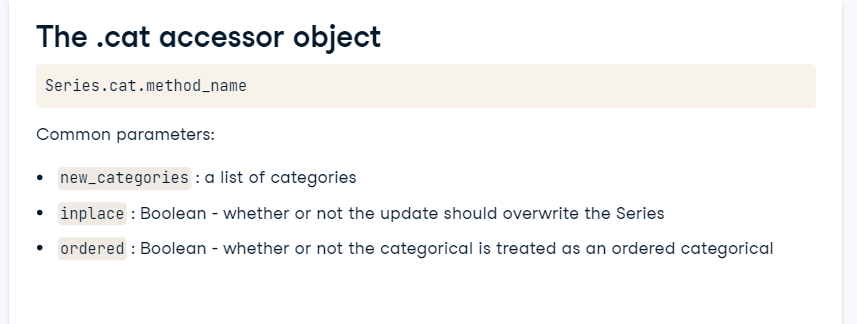
**A dog's coat**

Let's start by converting the coat variable to a category using the astype method, and then check the frequency distribution using the value counts method. We are setting the dropna parameter to false to check for any missing entries. We see that a short coat is the most common, while a long coat is the least common.



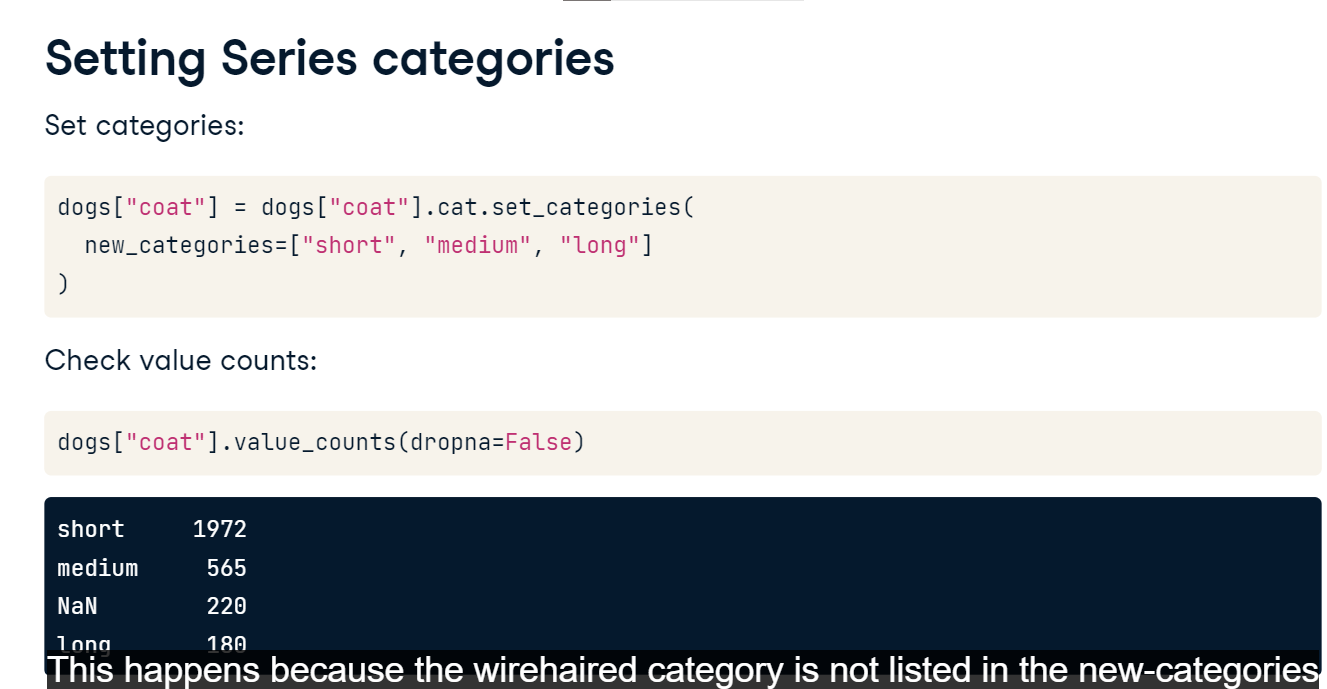
**The .cat accessor object**

We are going to use the dot-cat accessor object a lot in this chapter. This object let's us access and manipulate the categories of a categorical Series. Most of the methods we will introduce use the following parameters: new-categories - which is a list of new categories for the Series, inplace - which is a Boolean value for whether or not the method should overwrite the current Series, and ordered - which is a Boolean for whether or not the new Series should be treated as an ordered categorical or not. Our first example of using this object and these parameters will be setting new categories.

****

**Setting Series categories**

cat-dot-set categories is used to set specific categories for a Series. Any values not listed in the new-categories list will be dropped. Checking the value counts of this Series again, we see that the wirehaired responses have been set to NaN. This happens because the wirehaired category is not listed in the new-categories parameter and is no longer recognized.



**Setting order**

We can set the order of the categories using the ordered parameter. Checking the head of the pandas Series shows us that the Series now knows the categories have a specific order.



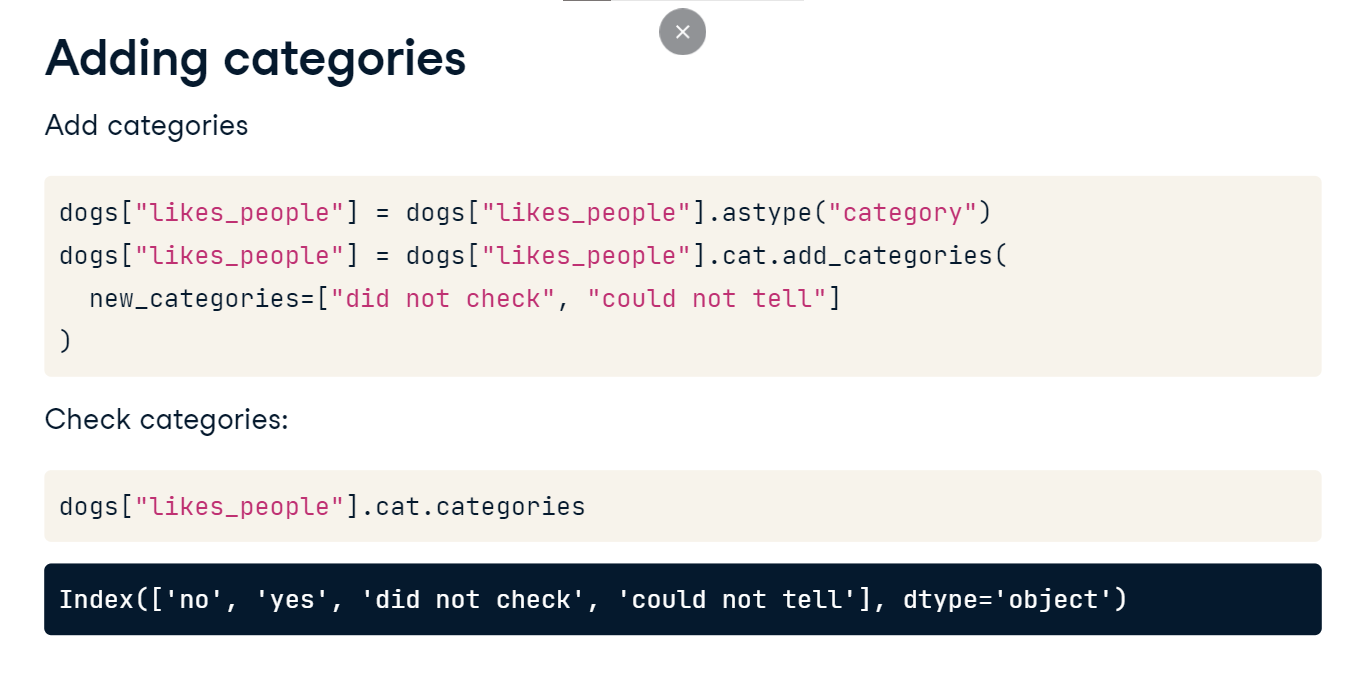
**Missing categories**

In the likes-people column, there are 938 rows without a response. Maybe the dog shelter did not check, or maybe they checked and could not tell. Let's add a couple of categories to clean this up.



**Adding categories**

We can add two categories using the cat-dot-add-categories method. Here we have added two categories, to help clarify what a missing value actually means. Notice that categories not listed in the new-categories parameter are not replaced with NaN values this time and are simply left alone. We can check the final categories using cat-dot-categories on our pandas Series. Awesome - both categories were added and can now be used in this Series.



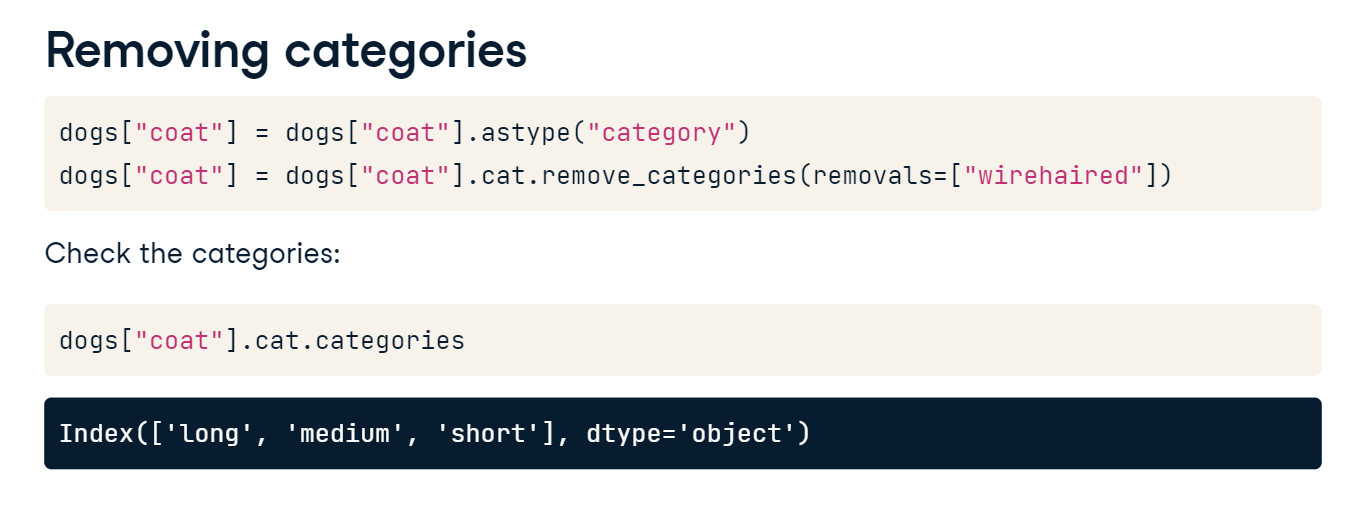
**New categories**

Although we added categories, this doesn't mean any rows of our data were set to these categories. Checking the value counts one more time verifies this. We will learn how to update values in a different lesson.



**Removing categories**

We can also remove categories using the cat-dot-remove categories method. This method takes a list of categories to remove using the removals parameter. In this example, we remove the wirehaired category altogether. This also means that all wirehaired values will be set to NaN values.



**Methods recap**

Let’s recap the methods covered in this lesson. We first learned how to set categories using the set-categories method, which drops values that are not specified. Add-categories can be used to add new categories, and categories not specified are left alone. Finally, remove-categories can be used to set matching values to NaN.

**Updating categories**

Now that we understand how to create, add, and remove categories in Pandas, let's work on updating and collapsing categories.

**The breed variable**

Take a look at the categories found in the breed column of the dog dataset. The most common category is Unknown Mix. We can rename this category to just be Unknown.



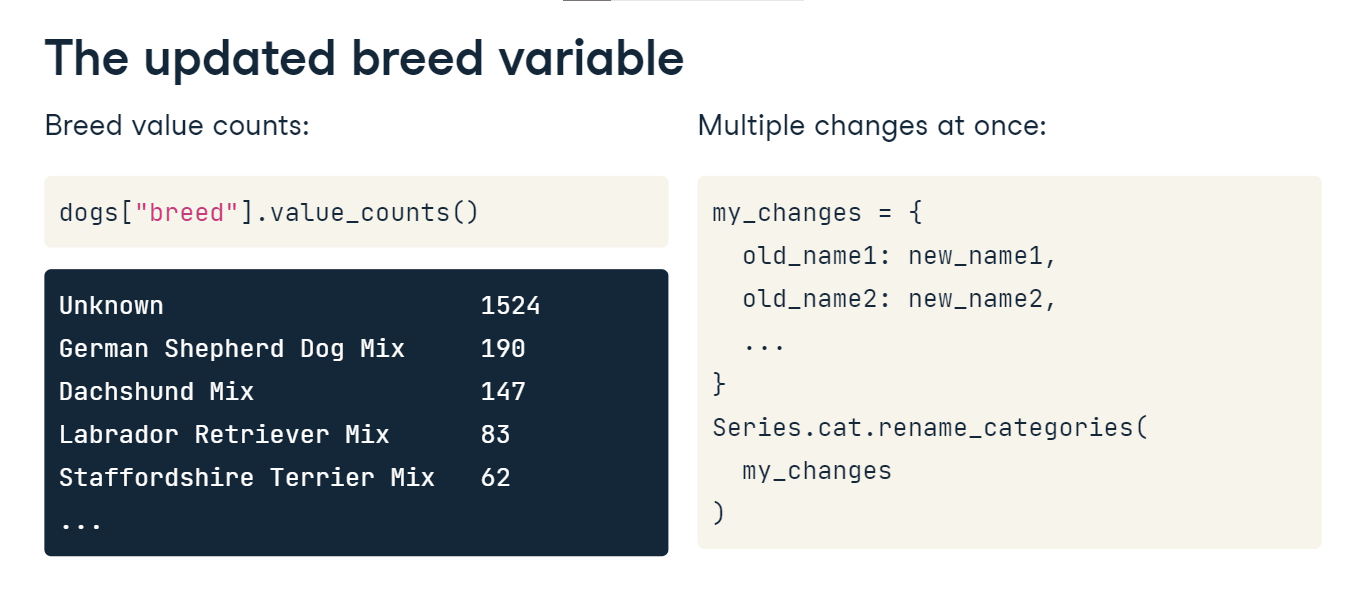
**Renaming categories**

An easy way to do this is with the cat-dot-rename-categories method. If we supply a dictionary of key-value pairs, with the key being the current category, and the value being the desired new category, we can rename categories quickly. Note that this method does not require a dictionary, but we are using one here for clarity. Let's first make a dictionary so we can map the Unknown Mix category to just be Unknown. Next, we update the breed column using cat-dot-rename-categories and passing the dictionary of changes.



**The updated breed variable**

Notice that Unknown Mix has been changed to just be Unknown, but still has 1,524 responses. When using the cat-dot-rename-categories method, you can rename more than one category at a time, just make a bigger dictionary!



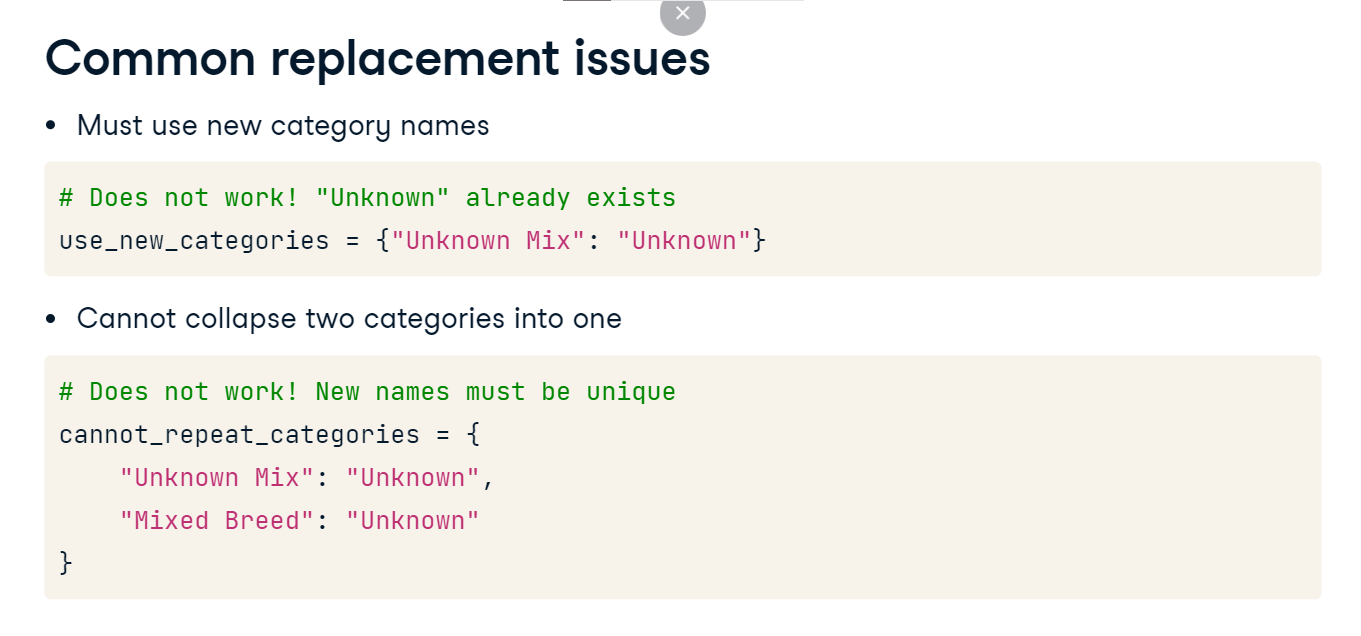
**Renaming categories with a function**

Another nice feature of the cat-dot-rename-categories method is that you can also use a lambda function to update categories. We won't cover lamba functions in this course, but we will show a couple of examples. We are using it here just to show an example of it in action. Let's convert both male and female to be in title case. Using the title method, we convert each category in the sex variable to title case. We now have Female and Male as categories and both are in title case.



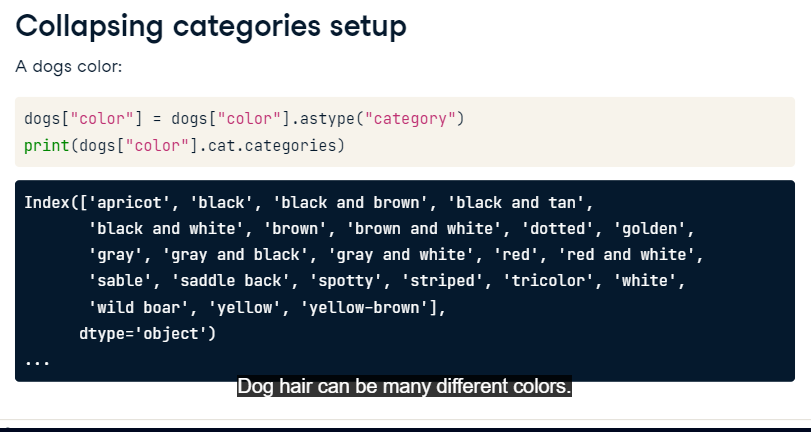
**Common replacement issues**

This method does come with two key issues. First, the new category must not currently be in the list of categories. If Unknown was already a category, we would not be able to rename Unknown Mix to be Unknown. And second, we can't use this method to collapse categories. If we wanted both Unknown Mix and Mixed Breed to be the Unknown category, we can't use this method.



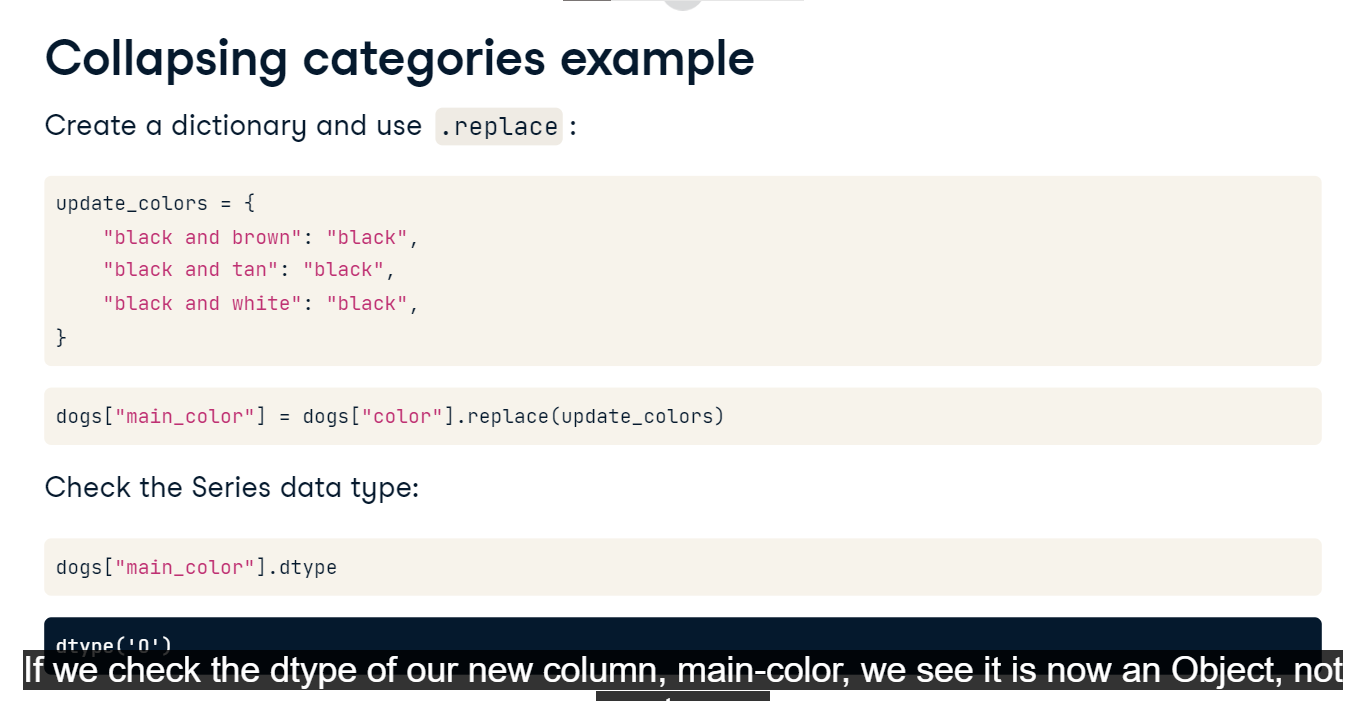
**Collapsing categories setup**

So how do we collapse categories? Let's look at the dogs hair color Series as an example. Dog hair can be many different colors. It might make sense for us to make a new categorical column that just has a dog's main or primary color, instead of all of the combinations of colors.



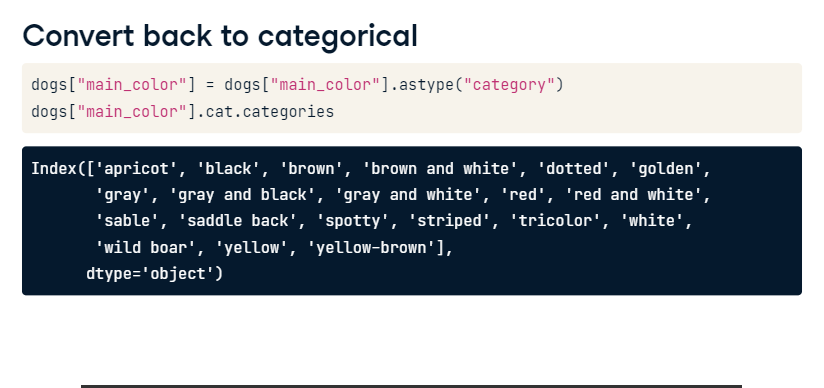
**Collapsing categories example**

We start by making a dictionary of all of the categories we want to collapse. Here we take all black plus one additional color categories and collapse them to the primary color of black. We use the dot-replace method to change each key-value pair listed in the update-colors dictionary. This method, however, does not preserve the categorical data type and does not use the dot-cat accessor object. What it is really doing is replacing every key in our dictionary with every value, but the method is matching strings, not categories. If we check the dtype of our new column, main-color, we see it is now an Object, not a category.



**Convert back to categorical**

Anytime you are updating the underlying string of a category, you will need to convert the Series back to a categorical dtype using the astype method and specifying category. If we check the categories of our new Series, we see that black is one of them, but black and brown, black and tan, and black and white are all gone.



**Reordering categories**

We have talked about the ordering of categories several times already, so let's take a look at how to reorder categories in a Pandas Series.

**Why would you reorder?**

You might reorder variables in pandas for a few different reasons. First, if the variable wasn't set as an ordinal variable upon creation, you might reorder your variables and set the Series as ordinal at this time. You might also set the order so that an analysis is displayed in a specific order, making the results easier to understand. And finally, it's good to remember that converting a Series to categorical can save on memory.

**Reordering example**

Let's look at an example. The coat variable has four values. If the wirehaired value is somewhere between medium and long, we might want to reorder the categories to short, medium, wirehaired, and long. Notice that we also specified that ordered is true here, as these are lengths of a dogs coat, and length has a natural order. As a quick aside, here is an example of using the inplace parameter. By setting this to true, the coat variable is updated without needing to set the variable equal to an updated version of itself. Several functions and methods in pandas have this as a parameter and it is generally used as a way to reduce the amount of typed code.



**Grouping when ordered=True**

Here we have borrowed the reorder categories setup that we used on the previous slide. Now that we have reordered our categories, several methods and visualizations will use this order when printing output. Note that the order of the printout is based on the order of the new categories parameter. Whichever order is specified using this parameter will be used. Take a look at this groupby statement. The average age for each group will be shown in the order of the categories of the coat column.



**Grouping when ordered=False**

Let's use another reorder-categories call. In this example, we want the output of our summary statistics to be short, medium, long, and then wirehaired. However, this isn't the natural order because wirehaired is shorter than long. In this context, we will set the ordered parameter equal to false because we don't want the coat column to be treated as an ordinal categorical variable. This means that you can still reorder categories for display purposes without the category being ordinal. Here is the group by statement followed by the output. We see the order we specified - short, medium, long, and then wirehaired.



**Cleaning and accessing data**

For this next lesson, we will focus on cleaning categorical columns and accessing other data by filtering categorical data.

**Possible issues with categorical data**

Data is messy, especially when you are working with strings or categories. Let's focus on a few of the main issues that may arise. First, categories may be inconsistent, and although you may recognize similar values as the same category, Python does not. Capitalization and white spaces are common culprits here and these issues may occur when appending different data sources or columns. Second, spelling issues can cause big problems. This occurs frequently in surveys or online forms when the field is left to the user to fill out. And finally, if we do make corrections, we need to make sure our column dtype remains category, and is not changed to an object.

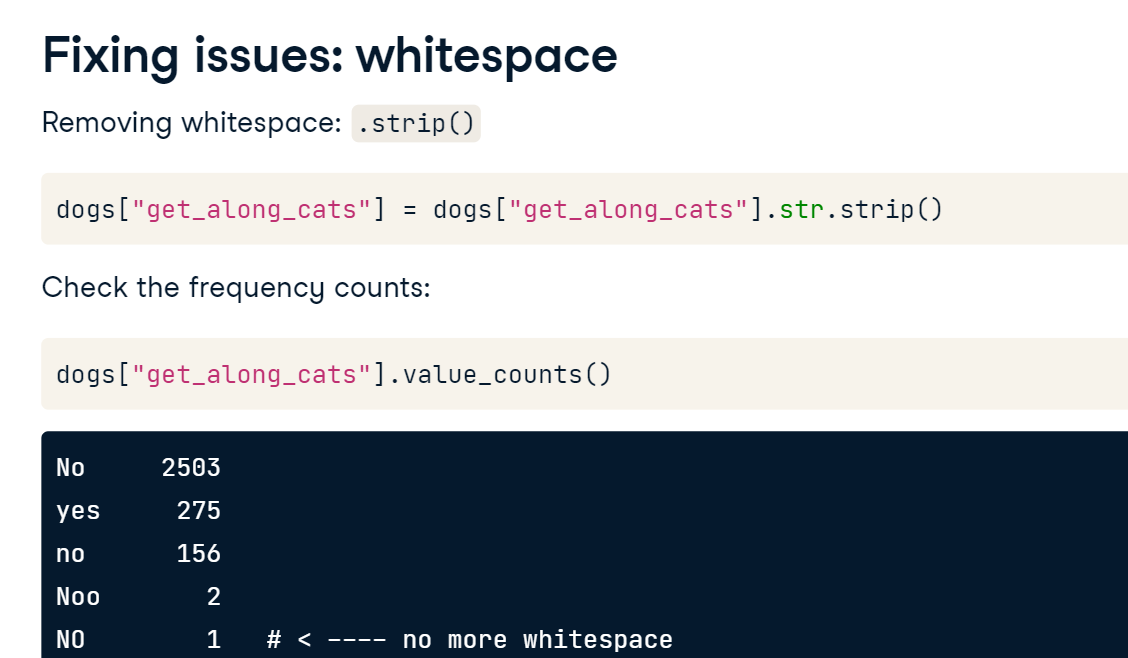
**Identifying issues**

The easiest way to identify issues in our categorical columns is to use either the cat dot categories method to view the categories, or the value counts method to see the counts of each category. Let's use the value counts method on the gets-along-with-cats column of the adoptable dogs dataset. We notice 3 issues. Varying capitalization, leading white spaces, and misspellings.



**Fixing issues: whitespace**

Fortunately, we have the same resources available for fixing categorical values that we do for fixing strings. We can remove whitespace by accessing the string value of the column using str, and then using the strip method to remove leading and trailing whitespace. Checkout the one category at the bottom that no longer has a leading whitespace.



**Fixing issues: capitalization**

We can fix capitalization issues with str again, but this time using either the title, upper, or lower methods, depending on what type of result we want. Here we have used title, and all of the responses have been switched to title case.



**Fixing issues: misspelled words**

Finally, to fix a typo, we can use the same mapping methods we learned in the renaming categories lesson. First we make a mapping, and then we use the replace method to replace the values. This leaves us with only two final categories.



**Checking the data type**

If you do use one of these methods, your column will be converted to an Object data type. Remember, to check the dtype we need to use the dtype property. As always, use the astype method to convert the Series back to categorical.

**Using the str accessor object**

We have already seen that we can update categories using the str accessor object, but we can also filter data using str as well. One way to use str is to look for categories that contain a specific string, such as Shepherd. We can use this filter to see all of the dogs that have some sort of Shepherd in their breed name. We are setting the regex parameter to false in this example so that we use string matching and not a regular expression.



**Accessing data with loc**

One of the great things about using columns that are categories is that the data access methods of loc and iloc work like normal. We can access the size of the dogs that get along with cats by using the loc method, specifying that dogs get along with cats, and selecting the size column. Let's add the value counts method at the end of accessing this data. Note that the value counts method does not automatically use the categorical order when printing results. We can use the sort parameter here so that the output will be ordered by the order of the category.

