**1. Introducing DataFrames**

00:00 - 00:05

Hi, I'm Richie. I'll be your tour guide through the world of pandas.

**2. What's the point of pandas?**

00:05 - 00:14

pandas is a Python package for data manipulation. It can also be used for data visualization; we'll get to that in Chapter 4.

**3. Course outline**

00:14 - 00:39

We'll start by talking about DataFrames, which form the core of pandas. In chapter 2, we'll discuss aggregating data to gather insights. In chapter 3, you'll learn all about slicing and indexing to subset DataFrames. Finally, you'll visualize your data, deal with missing data, and read data into a DataFrame. Let's dive in.

**4. pandas is built on NumPy and Matplotlib**

00:39 - 01:03

pandas is built on top of two essential Python packages, NumPy and Matplotlib. Numpy provides multidimensional array objects for easy data manipulation that pandas uses to store data, and Matplotlib has powerful data visualization capabilities that pandas takes advantage of.

**5. pandas is popular**

01:03 - 01:16

pandas has millions of users, with PyPi recording about 14 million downloads in December 2019. This represents almost the entire Python data science community!

1. 1 https://pypistats.org/packages/pandas

**6. Rectangular data**

01:16 - 01:42

There are several ways to store data for analysis, but rectangular data, sometimes called "tabular data" is the most common form. In this example, with dogs, each observation, or each dog, is a row, and each variable, or each dog property, is a column. pandas is designed to work with rectangular data like this.

**7. pandas DataFrames**

01:42 - 02:12

In pandas, rectangular data is represented as a DataFrame object. Every programming language used for data analysis has something similar to this. R also has DataFrames, while SQL has database tables. Every value within a column has the same data type, either text or numeric, but different columns can contain different data types.

**8. Exploring a DataFrame: .head()**

02:12 - 02:38

When you first receive a new dataset, you want to quickly explore it and get a sense of its contents. pandas has several methods for this. The first is head, which returns the first few rows of the DataFrame. We only had seven rows to begin with, so it's not super exciting, but this becomes very useful if you have many rows.

**9. Exploring a DataFrame: .info()**

02:38 - 02:46

The info method displays the names of columns, the data types they contain, and whether they have any missing values.

**10. Exploring a DataFrame: .shape**

02:46 - 02:59

A DataFrame's shape attribute contains a tuple that holds the number of rows followed by the number of columns. Since this is an attribute instead of a method, you write it without parentheses.

**11. Exploring a DataFrame: .describe()**

02:59 - 03:22

The describe method computes some summary statistics for numerical columns, like mean and median. "count" is the number of non-missing values in each column. describe is good for a quick overview of numeric variables, but if you want more control, you'll see how to perform more specific calculations later in the course.

**12. Components of a DataFrame: .values**

03:22 - 03:36

DataFrames consist of three different components, accessible using attributes. The values attribute, as you might expect, contains the data values in a 2-dimensional NumPy array.

**13. Components of a DataFrame: .columns and .index**

03:36 - 04:12

The other two components of a DataFrame are labels for columns and rows. The columns attribute contains column names, and the index attribute contains row numbers or row names. Be careful, since row labels are stored in dot-index, not in dot-rows. Notice that these are Index objects, which we'll cover in Chapter 3. This allows for flexibility in labels. For example, the dogs data uses row numbers, but row names are also possible.

**14. pandas Philosophy**

04:12 - 04:56

Python has a semi-official philosophy on how to write good code called The Zen of Python. One suggestion is that given a programming problem, there should only be one obvious solution. As you go through this course, bear in mind that pandas deliberately doesn't follow this philosophy. Instead, there are often multiple ways to solve a problem, leaving you to choose the best. In this respect, pandas is like a Swiss Army Knife, giving you a variety of tools, making it incredibly powerful, but more difficult to learn. In this course, we aim for a more streamlined approach to pandas, only covering the most important ways of doing things.

**. Sorting and subsetting**

00:00 - 00:08

In this video, we'll cover the two simplest and possibly most important ways to find interesting parts of your DataFrame.

**2. Sorting**

00:08 - 00:41

The first thing you can do is change the order of the rows by sorting them so that the most interesting data is at the top of the DataFrame. You can sort rows using the sort\_values method, passing in a column name that you want to sort by. For example, when we apply sort\_values on the weight\_kg column of the dogs DataFrame, we get the lightest dog at the top, Stella the Chihuahua, and the heaviest dog at the bottom, Bernie the Saint Bernard.

**3. Sorting in descending order**

00:41 - 00:49

Setting the ascending argument to False will sort the data the other way around, from heaviest dog to lightest dog.

**4. Sorting by multiple variables**

00:49 - 01:08

We can sort by multiple variables by passing a list of column names to sort\_values. Here, we sort first by weight, then by height. Now, Charlie, Lucy, and Bella are ordered from shortest to tallest, even though they all weigh the same.

**5. Sorting by multiple variables**

01:08 - 01:24

To change the direction values are sorted in, pass a list to the ascending argument to specify which direction sorting should be done for each variable. Now, Charlie, Lucy, and Bella are ordered from tallest to shortest.

**6. Subsetting columns**

01:24 - 01:39

We may want to zoom in on just one column. We can do this using the name of the DataFrame, followed by square brackets with a column name inside. Here, we can look at just the name column.

**7. Subsetting multiple columns**

01:39 - 02:11

To select multiple columns, you need two pairs of square brackets. In this code, the inner and outer square brackets are performing different tasks. The outer square brackets are responsible for subsetting the DataFrame, and the inner square brackets are creating a list of column names to subset. This means you could provide a separate list of column names as a variable and then use that list to perform the same subsetting. Usually, it's easier to do in one line.

**8. Subsetting rows**

02:11 - 02:32

There are lots of different ways to subset rows. The most common way to do this is by creating a logical condition to filter against. For example, let's find all the dogs whose height is greater than 50 centimeters. Now we have a True or False value for every row.

**9. Subsetting rows**

02:32 - 02:42

We can use the logical condition inside of square brackets to subset the rows we're interested in to get all of the dogs taller than 50 centimeters.

**10. Subsetting based on text data**

02:42 - 02:54

We can also subset rows based on text data. Here, we use the double equal sign in the logical condition to filter the dogs that are Labradors.

**11. Subsetting based on dates**

02:54 - 03:12

We can also subset based on dates. Here, we filter all the dogs born before 2015. Notice that the dates are in quotes and are written as year then month, then day. This is the international standard date format.

**12. Subsetting based on multiple conditions**

03:12 - 03:36

To subset the rows that meet multiple conditions, you can combine conditions using logical operators, such as the "and" operator seen here. This means that only rows that meet both of these conditions will be subsetted. You could also do this in one line of code, but you'll also need to add parentheses around each condition.

**13. Subsetting using .isin()**

03:36 - 03:56

If you want to filter on multiple values of a categorical variable, the easiest way is to use the isin method. This takes in a list of values to filter for. Here, we check if the color of a dog is black or brown, and use this condition to subset the data.

**1. New columns**

00:00 - 00:17

In the last lesson, you saw how to subset and sort a DataFrame to extract interesting bits. However, often when you first receive a DataFrame, the contents aren't exactly what you want. You may have to add new columns derived from existing columns.

**2. Adding a new column**

00:17 - 00:53

Creating and adding new columns can go by many names, including mutating a DataFrame, transforming a DataFrame, and feature engineering. Let's say we want to add a new column to our DataFrame that has each dog's height in meters instead of centimeters. On the left-hand side of the equals, we use square brackets with the name of the new column we want to create. On the right-hand side, we have the calculation. Notice that both the existing column and the new column we just created are in the DataFrame.

**3. Doggy mass index**

00:53 - 01:21

Let's see what the results are if we calculate the body mass index, or BMI, of these dogs. BMI is usually calculated by taking a person's weight in kilograms and dividing it by their height in meters, squared. Instead of doing this with people, we'll try it out with dogs. Again, the new column is on the left-hand side of the equals, but this time, our calculation involves two columns.

**4. Multiple manipulations**

01:21 - 01:56

The real power of pandas comes in when you combine all the skills you've learned so far. Let's figure out the names of skinny, tall dogs. First, to define the skinny dogs, we take the subset of the dogs who have a BMI of under 100. Next, we sort the result in descending order of height to get the tallest skinny dogs at the top. Finally, we keep only the columns we're interested in. Here, you can see that Max is the tallest dog with a BMI of under 100.

**5. Let's practice!**

01:56 - 02:01