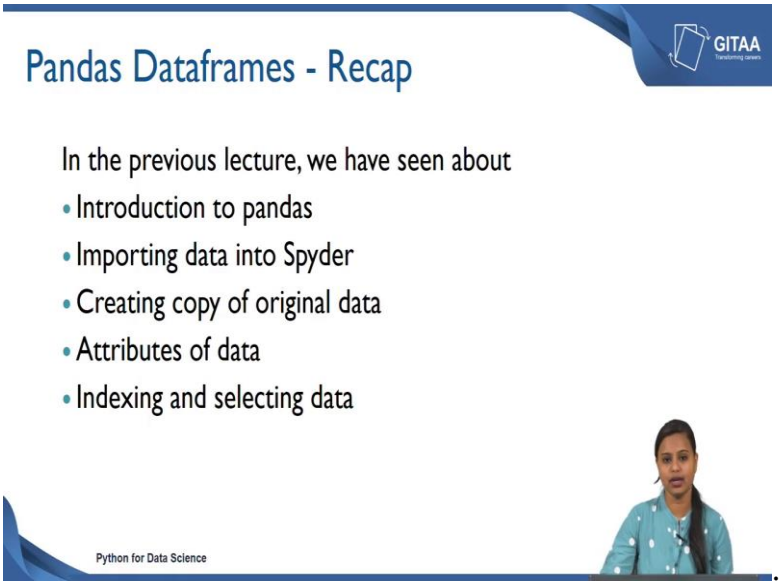
Python for Data Science
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 15
Pandas Dataframes
Part - II

Hello all welcome to the lecture on Pandas Dataframes.

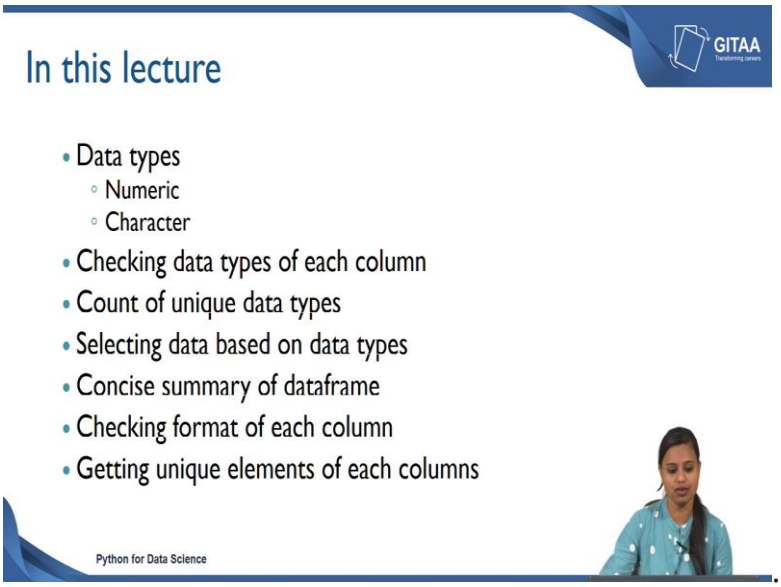
(Refer Slide Time: 00:19)



So, let us have a quick recap on what we have done in the previous lecture, on Pandas Dataframes. So, in the previous lecture we have seen about introduction to Pandas, where we have been introduced to the Pandas library, then we have seen about how to import the data into Spyder.

So, once we imported we have also seen, how to create a copy of original data, we have also seen how to get the attributes of data, followed by that we have also seen indexing and selecting data.

(Refer Slide Time: 00:47)

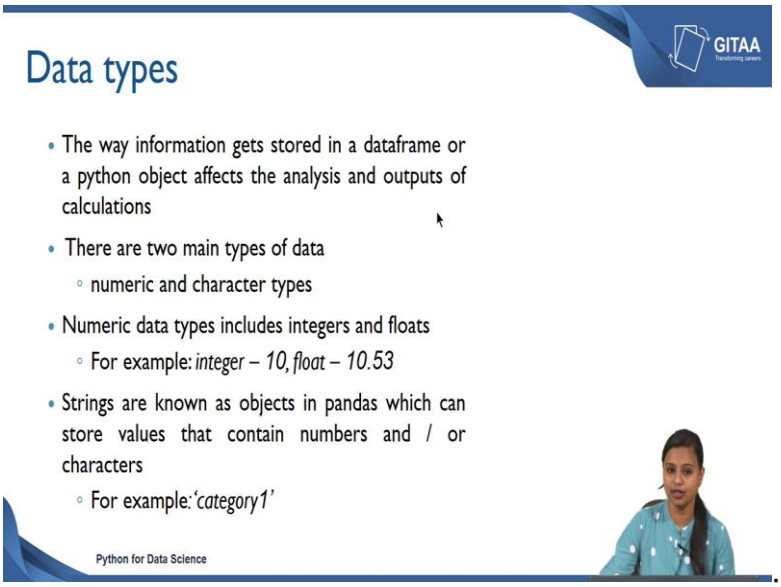


So, in this lecture we are going to see about the data types of variables in a dataframe. We are going to look about numeric data types and character data types, which we are going to use often in our analysis. Once, we know about the two data types we are going to see, how to check the data types of each column in your dataframe.

Followed by that we are going to look at, how to get the count of unique data type? After that we will also see how to select the data based on particular data types. And, then we are going to look at the concise summary of the dataframe, followed by that we are going to check the format of each column just to cross verify whether the data type is of desired data type or not.

After, that we are going to get the unique elements of each column, so, in this lecture we will be looking at different topics as we have mentioned in detail. So, first we will look about data types.

(Refer Slide Time: 01:45)



So, the way information gets stored in a dataframe or in any python object basically affects, whatever analysis that you are going to perform on your dataframe. As well as that results in a different form of outputs from the calculations that you have made. For example, you cannot perform any numerical operations on a string; similarly you cannot do any string related operations on a numerical data. So, we have to be sure of what data type you are handling in a dataframe.

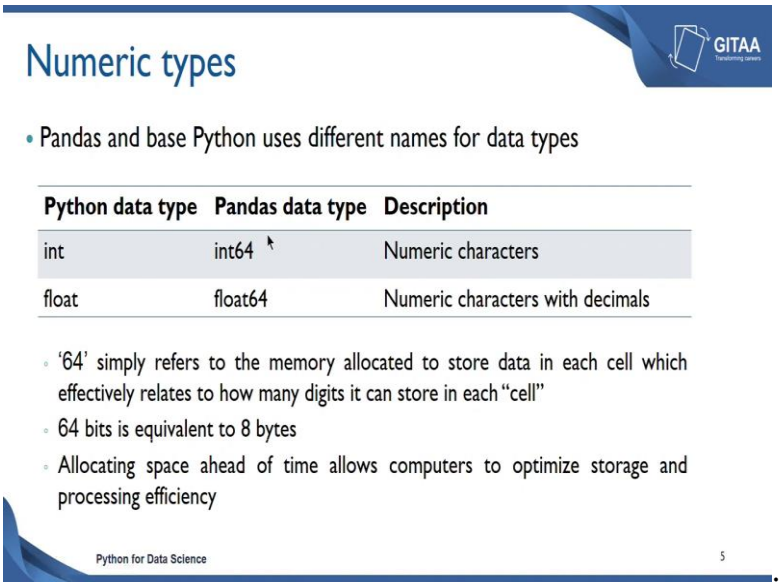
So, basically there are two main types of data as discussed one is being numeric and another one is being character types. And, numeric data types basically includes all the numerical values, which are in terms of integers and floats for example, integer value being represented as 10 and float values are called whenever it has a numerical value with the decimal. So, here if we have decimal values after 10, then it will be represented as float for example, as 10.53.

This is about the numeric data types that the python can handle. Next, we are going to look about the character types. So, character data types are nothing, but all the strings are known as objects in pandas, which can store values that contains numbers as well as characters.

So, whatever value that is been enclosed inside a single or double quotes will be considered as a string and that is being represented as object in pandas. And, for example, if you have a string which is enclosed inside the single or double quotes like,

category1 that becomes a string value or an object. Even, though it has both strings and numbers whatever has been enclosed within single or double quotes will be considered as string in python. We will look in deep about, what string values represents and what numerical values represents?

(Refer Slide Time: 03:45)



So, first we will look about the numeric types whenever we deal with the Pandas library using the python tool. It is not necessary that both the pandas and python uses the same names for data types, because pandas and base python uses different names for different data types. For example, here is a table to illustrate you, how the data type is named in python and how the data type is being named in Pandas library and followed by that we have the description as well.

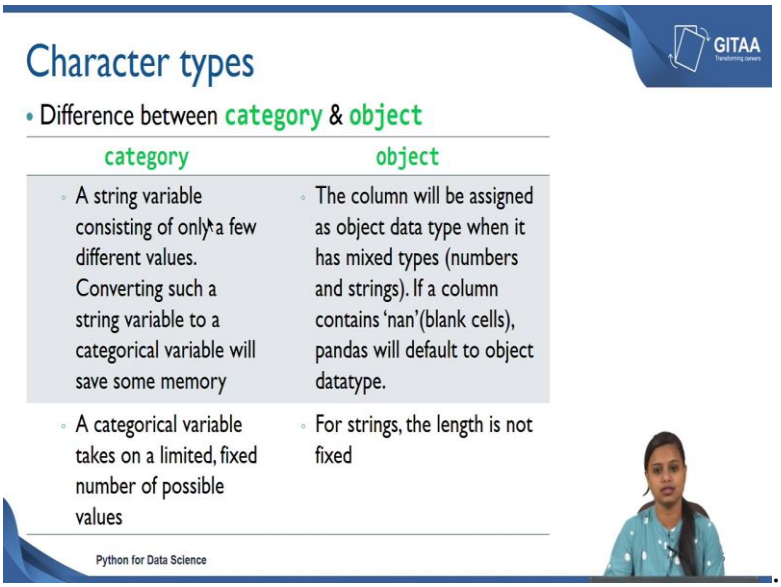
First we will look at integer that is being represented as int in terms of base python and in if you look at the Pandas library; the integers will be represented in terms of int 64. And, int64 corresponds to all numeric characters; it can contain all the numeric characters. Next is a float; float is being represented as float in python whereas; in pandas it is being represented as float64.

So, float64 basically corresponds to all the numeric characters with decimal values. And, these 64 simply refers to the memory allocated to store the data in each cell, which effectively relates to how many digits it can store in each cell?

So, at the max it can store up to 64 bits which is equivalent to 8 bytes. So, why we are really concerned about the memory allocation in a each cell, because allocating space ahead of time allows computers, to optimize storage and processing efficiency. Because, whenever you read any data into Spyder or into any IDE of python, it basically gets read with the data type for each and every variable according to the values that it has.

So, in that case it always allocates memory to store the data in each cell just to optimize the storage and processing efficiency. So, now we have seen about the numeric types, where we have seen about integer and float data types.

(Refer Slide Time: 05:53)



Next, we are going to see about the character types. In python there are two data types that can handle character types; one is category and another one is being object. So, now, there are two different data types that are available for character types. So, let us look at the difference between category and object.

So, when you look at the first point; the first point describes what the category data type would be basically any string value consisting of only a few different values, then that becomes a category. And, we have to convert such string variable to a categorical variable, which can save us some memory. Instead of keeping too many string values in a form of same representation we can always convert them to category data type to have it as a categories.

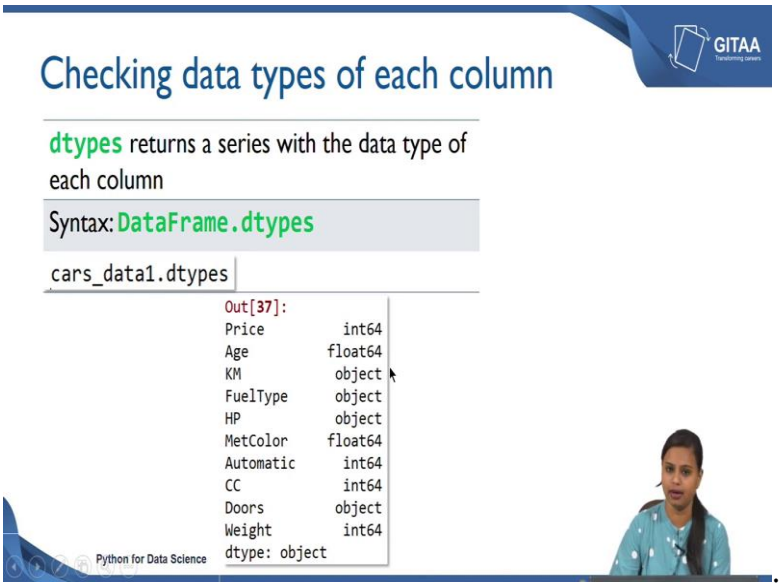
A categorical variable takes on a limited fixed number of possible values, because there is limits to the length, that is being fixed and it can always accommodate only fixed number of possible values; you cannot have 15 to 20 different values, which forms categories. So, in that case we always if you want to have too many categories in your column, then you can go for object, because, the column will be assigned as object data type where it has mixedtypes.

Basically, whenever you have numbers which is being represented as 0 1, that can also be an object, if it is being enclosed within single or double quote or even if your column has mixed numbers that is category 1, category 2, category 3, that can also be an object. And, the other point is if a column contains nan values, basically in python all the blank cells will be filled with the default nan values.

And, the next point would be, if a column contains nan that is just a blank cell. Then, the pandas will default to object data type, because by default whenever you read any blank cell in Spyder, all the blank cell will be read as nan values and by default it will default to object data type. Because all the nans will be considered as a different value so, it becomes an object data types itself.

And, here has an advantage is that for strings the length is not fixed, how many ever number of elements you can have as a string and there is no limit to the number of possible values that you can have as an object. So, this is the difference between the category and the object character types. So, now, we have seen the difference between the category and object data types.

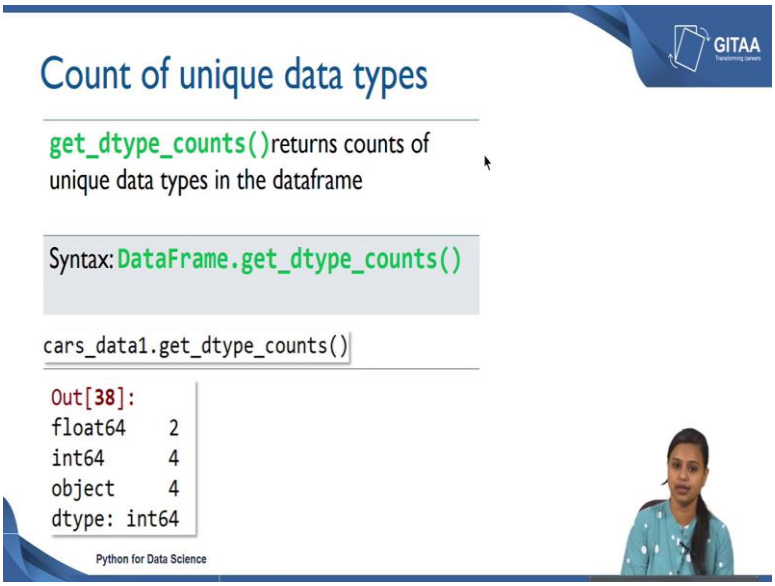
(Refer Slide Time: 08:35)



So, now we know about the basic data type that we are going to often work with. So, now, it is time to check the data types of each columns of the data frame that we are working with. So, basically if you want to check the data type of each column, because whenever you have been given a data, you want to really check what is the structure of the data; that means, which variable has which data type? In, that case you can use dtypes, because that returns a series with the data type of each column and the syntax would be you use dtypes along with the Data Frame name. So, Data Frame.dtypes will give you a series with the data type of each column.

So, the input would be cars_data1 dtypes, where cars the_data1 is the Data Frame, that we are looking at and the output is shown below where you have multiple variables correspondingly you have the data type of each variables. When we look at the first variable that is price, the price data type is of integer 64, which is the desired one and the age being float 64 and the kilometer has been read as object. Similarly, you will be able to check the data type for each variables using the dtypes commander.

(Refer Slide Time: 09:59)

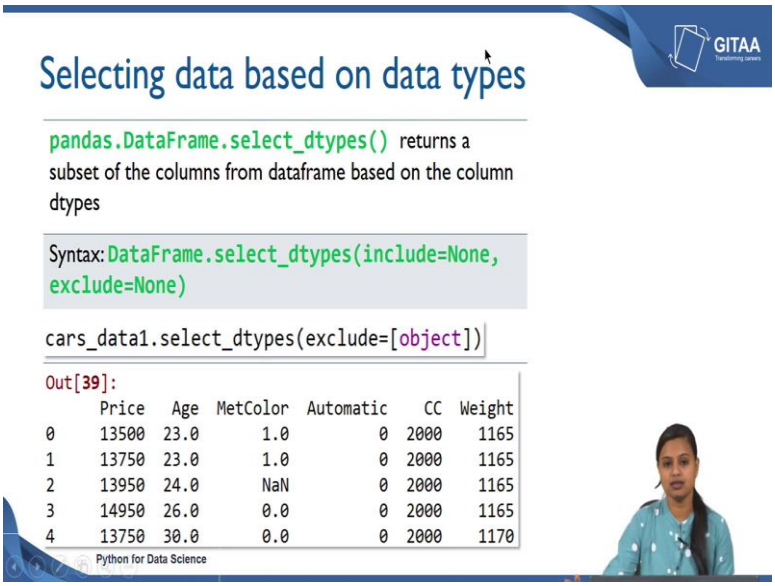


So, now we have an overall idea about what are the data types that we are going to work with using the cars_data. There is also an option where you can get the count of unique data types available in your Data Frame. So, in that case get_dtype_counts, returns the counts of unique data types in the data frame.

So, if you want to summarize how many in 64 variables are there and how many float 64 variables are there, then in that case you can use get_dtype_counts command. And, the syntax will be you will use the command along with the data frame name. So, let us see how to do that? So, here is an input where I have given cars_data1.get_dtype_counts. So, that will give me an output which is shown here. So, I have the summarization here, where I have different types of data type as well as I have the corresponding count also.

So, basically on the whole I have 2 variables of float64 data type and I have 4 variables of int 64 data type. As well as I have 4 variables, which consist of object data type? And dtype int 64 represents the output data types since it has the values it is being represented as int64.

(Refer Slide Time: 11:25)



So, now we also have an overall idea about the count of unique data types that we are going to handle with. So, now, we know about how to get the data type of each variables. So, there might be cases where you want to perform the operations only on a numerical data type. Similarly, there can be cases where you are going to work with only categorical data type.

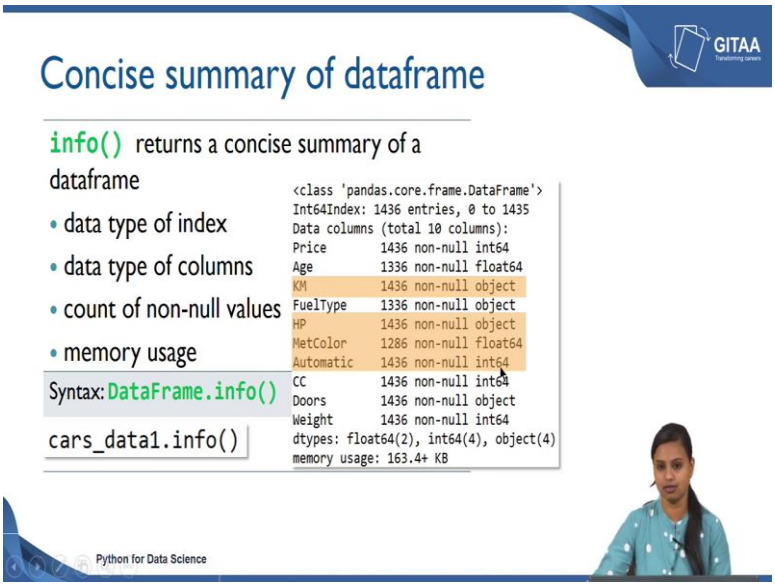
In that case, there is also a platform where you can select the data based on the data types available in your DataFrame. So, let us see how to do that? So, select_dtypes is the command that we are going to use, to select the data based on the data types. And, along with that you have to give the DataFrame name and since that is from the Pandas library this it is being represented as Pandas.DataFrame.select_dtypes. And, this command returns a subset of the columns from the DataFrame, based on the column types you have specified inside a function.

So, let us see how to use the function, here DataFrame would be the DataFrame name and here is a command that the data select_dtypes inside the function there are two arguments. One is include and another one is exclude both being nan by default, if you want to select only those columns, which are of object data type, then you can just use object inside the include argument and if you want to exclude any particular data type from your analysis. In that case you can use the data types under the exclude argument.

So, let us see an example how we are going to do that? So, here is the input the DataFrame that we are working with is cars_data one. And, the command being select_dtypes and inside the function, I have given exclude is equal to object. And, whatever data type that you are giving that should be given inside the square braces, because you can also give multiple data types object comma int 64. Basically, in that case you will be excluding all the columns which are of object and integer 64 data types.

Here, I just want to include the object data type. So, I have just given object. So, the output would be a DataFrame with the variables, which are not of object data type. So, here is the output we have price age metcolor automatic CC and weight with the data type integer and float. For example, the FuelType the doors all those have been excluded. So, this is how we basically select the data based on the data types.

(Refer Slide Time: 13:57)



So, next we are going to see about how to get the concise summary of DataFrame. So, there is a command called info that returns a concise summary of a DataFrame, the concise summary includes the data type of index; index being the row labels, the data type of row labels is what the output gives as well as it gives the data type of columns, it also gives the count of non-null values. Basically, how many filled values are there in your DataFrame. Also, it gives the memory usage of the DataFrame and the syntax would be you use the info command along with the DataFrame name.

So, let us see how to do that I have given cars_data1.info. So, the output will be similar to this, where the output starting line is Pandas.core.frame.DataFrame. So, it is of pandas core DataFrame and int 64 index being the index are being represented in terms of int 64, where you have 1400 and 36 entries, which are ranging from the 0 to 1435 row labels and totally you have 10 columns in your DataFrame. And, after that you have list of variables along with that, you have non-null values and what is the data type corresponding to that variable.

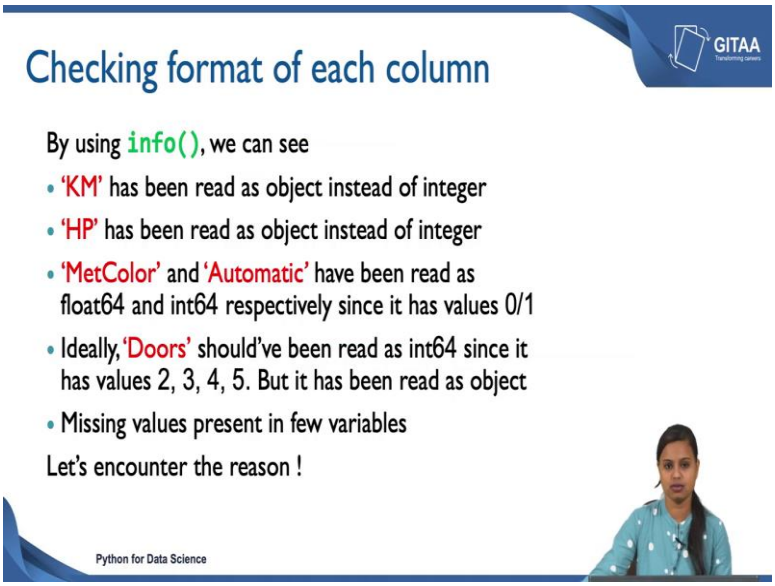
So, for example, under the price variable the total observations are 1436 and there are also 1436 non-null values and the data type of price being integer 64. In this case I have highlighted few rows. So, the purpose to check the concise summary of DataFrame is to verify, whether all the variables have been read with the proper data type or not.

If not we have to go back and convert them back to the desired data type. So, for example, price and age are of expected data type and if you see kilometer, there are no missing values actually 1400 and 56 observations are non-null and it is being read as object kilometer would be ideally be numbers. So, it should have not been read as object, but in this case it have been read as object.

Similarly, fuel type should be of object there is no problem in that and if you see HP, MetColor and Automatic, they have been read as object, float64, int64. Why, because metallic color automatic basically represents categories under that column, metallic color represents the whether the car has metallic color or not. So, it should be ideally it should be having some categories to it.

So, there is some problem that is why it has been read as float 64 and if you look at the automatic column; automatic also represents the type of gear box that the car possess. So, in that case it should ideally be categories and it should ideally be object. In this case it is int 64.

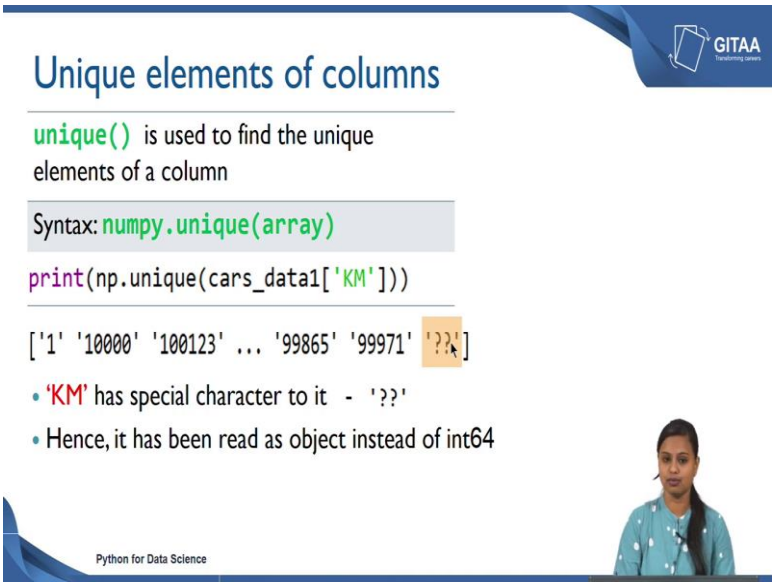
(Refer Slide Time: 16:59)



So, now let us just check the format of each column. So, just to summarize by using info, we can see that kilometer has been read as object instead of integer right. Next, we have horsepower that has been read as object instead of integer as well. Next, being metallic color and automatic both have been read as float 64 and int 64 respectively. Since, it just has values zeros and ones that is the reason that it has been read as float 64 and int 64.

But, it has been read as object, because since it has numbers only ideally it should have been read as integer 64, but it has been read as object. And, we have also seen there are missing values present in few variables. So, we have to encounter the reason behind all of these points.

(Refer Slide Time: 17:53)



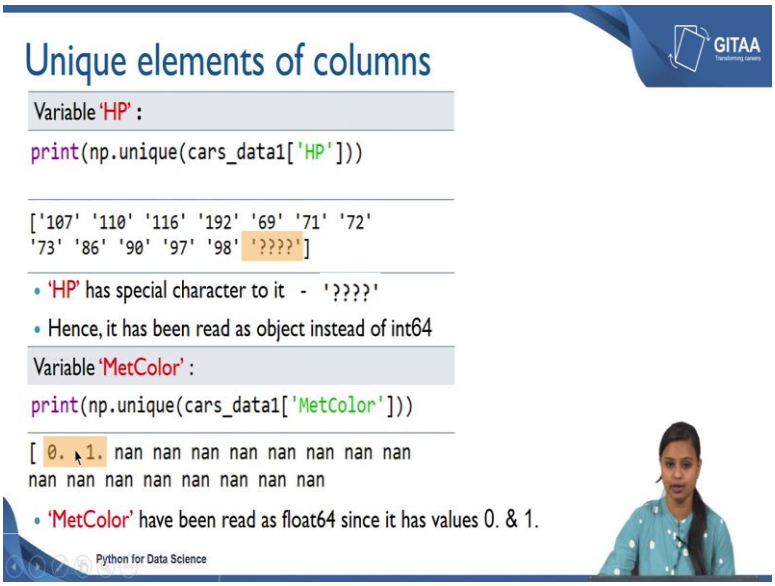
So, unique function is used to get the unique elements of a column the syntax being numpy.unique of array, unique function comes from the numpy library and the input should be an array, you cannot perform the unique operation on a multiple array it can be done on a single array only. So, here I have used np.unique that is why because I have imported numpy as np.

So, the alias is just np if you have not imported the numpy library at this point of time you can import numpy library as np and then you can follow with the code here. So, I am just print np.unique and the input should be an array. So, I am accessing the kilometer variable from the cars_data1.

So, that will give me the unique values of kilometer column. So, there are so, many unique values, but only few being shown here, you have a ?? symbol that is the representation, you are not seeing all of the values. And, the special thing about the kilometer is it has a special character that is double question mark, which has been enclosed with in single quote that is the reason; it has been read as object instead of 64.

So, whenever you have a special character all the values will be converted to character or string data type. So, that is why the kilometer has been read as object instead of int 64.

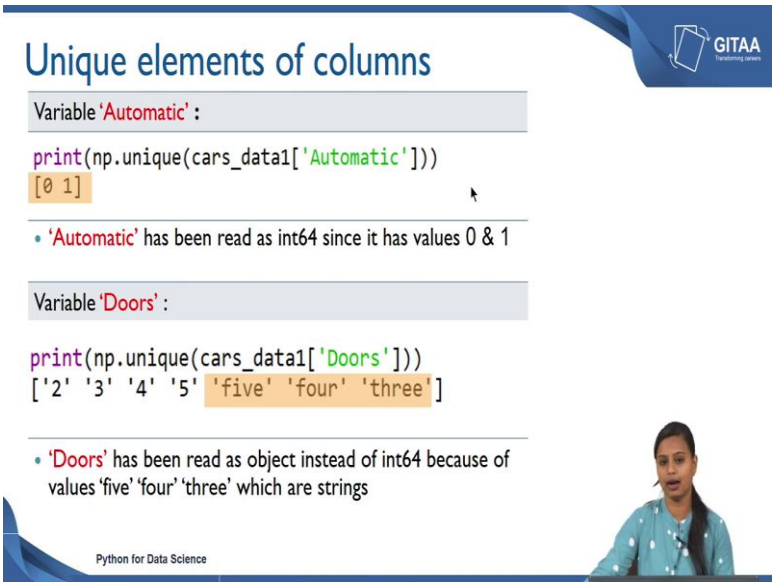
(Refer Slide Time: 19:23)



Similarly, we are going to look at the variable horsepower. So, I am going to get the unique elements of the column horsepower using the same unique command. So, you have different values under the horsepower. And, you also have a special character like 4 question marks that is the reason it has been read as the object instead of int 64.

And, when we look at the metallic color, I am using the same function dot unique to get the unique elements under the metallic color column. So, basically it has only the value 0s and 1s 0 point and 1 point. So, that is why it has been read as float 64. Since, it has values 0 and 1. Even, though it has nan values just because the value has decimals to it the whole variable have been read as float 64.

(Refer Slide Time: 20:19)

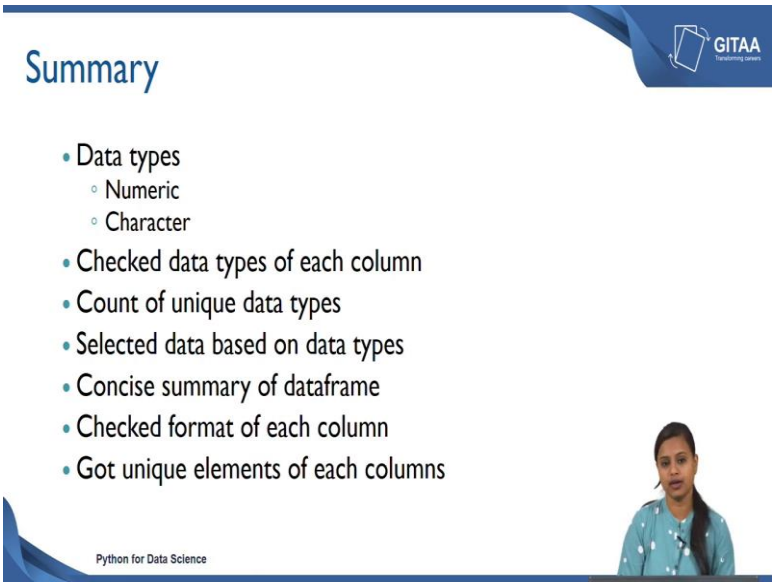


So, next we are going to look at another column that is automatic. I have used the same.unique function to get the unique elements of the column automatic. So, the output being 0 and 1, as we know 0 and 1 represents category part automatic has been read as integer 64. Since, it has value 0 and 1.

Next, we are going to look at the variable door and where we tried to get the unique values out of the doors column, if you see the output there are few values 2 3 4 and 5 also you have values as strings that is being represented as 5 4 3. So, there might be a typo where all the numerical 3 values have been typed as 3 in characters, similarly for 5 and 4.

So, this might be an error while getting the data from the source. So, this is the problem where door has been read as object instead of int 64, that is because of the values 5 4 3, which are in strings data type. So, now we have an overall idea about how do we check the data type of each variable and how to cross verify whether each data type is of expected data type or not. If, not we have seen how to get the unique elements out of columns to cross verify, whether there are some problems to it. So, that we can go back and reconvert them back to the expected data type.

(Refer Slide Time: 21:51)



So, next I am going to summarize whatever we have done in this lecture. Basically, we started with looking into 2 data types that is numeric and character for variables in the data frame. We have also checked the data type of each column, whether each data type is of expected data type or not. And, then we have also seen how to get the count of unique data types to get an overall idea about, what are the data types that we will be working with in our dataframe.

And, next we have seen how to select the data based on data types. For example, if you want to perform any operations that are completely related to numbers, then you would be selecting the data only with respect to numeric data types, which is like integer and float 64 data types. We have seen how to select the data based on data types?

And, we have also looked at the concise summary of dataframe to basically look at the variable and what is the data type of each variable along with that, we have also seen how to get the count of non-null values are there, which basically describes how many filled values are there in your dataframe.

After looking at the concise summary of data frame we had an idea about what each data type represents and after that we have also checked the format of each column just to cross verify, whether it is of expected data type or not, but not all the variables are of expected data type. So, we have to convert them back to the expected data type in the next lecture, we have also seen how to get the unique elements of each column. So, that

we got an idea about what each variables values are that is causing the variable which is not of expected data type.

So, in the next lecture we will be looking at to resolve all the problems that we have encountered in this lecture.