Pandas Dataframes
Part II

Pandas Dataframes - Recap

In the previous lecture, we have seen about

- Introduction to pandas
- Importing data into Spyder
- Creating copy of original data
- Attributes of data
- Indexing and selecting data

In this lecture

- Data types
  - Numeric
  - Character
- Checking data types of each column
- Count of unique data types
- Selecting data based on data types
- Concise summary of dataframe
- Checking format of each column
- Getting unique elements of each column

Data types

- The way information gets stored in a dataframe or a python object affects the analysis and outputs of calculations
- There are two main types of data
  - numeric and character types
- Numeric data types includes integers and floats
  - For example: *integer – 10, float – 10.53*
- Strings are known as objects in pandas which can store values that contain numbers and / or characters
  - For example: *'category1'*

Numeric types

- Pandas and base Python uses different names for data types

Python data type	Pandas data type	Description
int	int64	Numeric characters
float	float64	Numeric characters with decimals

- ‘64’ simply refers to the memory allocated to store data in each cell which effectively relates to how many digits it can store in each “cell”
- 64 bits is equivalent to 8 bytes
- Allocating space ahead of time allows computers to optimize storage and processing efficiency

Character types

- Difference between `category` & `object`

`category`	`object`
◦ A string variable consisting of only a few different values. Converting such a string variable to a categorical variable will save some memory	◦ The column will be assigned as object data type when it has mixed types (numbers and strings). If a column contains 'nan'(blank cells), pandas will default to object datatype.
◦ A categorical variable takes on a limited, fixed number of possible values	◦ For strings, the length is not fixed

Checking data types of each column

dtypes returns a series with the data type of each column

Syntax: `DataFrame.dtypes`

```
cars_data1.dtypes
```

```
Out[37]:
Price          int64
Age          float64
KM            object
FuelType      object
HP            object
MetColor     float64
Automatic      int64
CC             int64
Doors         object
Weight         int64
dtype: object
```

Count of unique data types

`get_dtype_counts()` returns counts of unique data types in the dataframe

Syntax: `DataFrame.get_dtype_counts()`

```
cars_data1.get_dtype_counts()
```

```
Out[38]:
float64    2
int64      4
object     4
dtype: int64
```

Selecting data based on data types

`pandas.DataFrame.select_dtypes()` returns a subset of the columns from dataframe based on the column dtypes

Syntax: `DataFrame.select_dtypes(include=None, exclude=None)`

```
cars_data1.select_dtypes(exclude=[object])
```

```
Out[39]:
   Price   Age  MetColor  Automatic    CC  Weight
0  13500  23.0       1.0          0  2000    1165
1  13750  23.0       1.0          0  2000    1165
2  13950  24.0       NaN          0  2000    1165
3  14950  26.0       0.0          0  2000    1165
4  13750  30.0       0.0          0  2000    1170
```

Concise summary of dataframe

`info()` returns a concise summary of a dataframe

- data type of index
- data type of columns
- count of non-null values
- memory usage

Syntax: `DataFrame.info()`

```
cars_data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1436 entries, 0 to 1435
Data columns (total 10 columns):
Price        1436 non-null int64
Age          1336 non-null float64
KM           1436 non-null object
FuelType     1336 non-null object
HP           1436 non-null object
MetColor     1286 non-null float64
Automatic    1436 non-null int64
CC           1436 non-null int64
Doors        1436 non-null object
Weight       1436 non-null int64
dtypes: float64(2), int64(4), object(4)
memory usage: 163.4+ KB
```

Checking format of each column

By using `info()`, we can see

- 'KM' has been read as object instead of integer
- 'HP' has been read as object instead of integer
- 'MetColor' and 'Automatic' have been read as float64 and int64 respectively since it has values 0/1
- Ideally, 'Doors' should've been read as int64 since it has values 2, 3, 4, 5. But it has been read as object
- Missing values present in few variables

Let's encounter the reason !

Unique elements of columns

`unique()` is used to find the unique elements of a column

Syntax: `numpy.unique(array)`

```
print(np.unique(cars_data1['KM']))
```

```
['1' '10000' '100123' ... '99865' '99971' '??']
```

- 'KM' has special character to it - `'??'`
- Hence, it has been read as object instead of int64

Unique elements of columns

Variable 'HP' :

```
print(np.unique(cars_data1['HP']))
```

```
['107' '110' '116' '192' '69' '71' '72'
'73' '86' '90' '97' '98' '????']
```

- 'HP' has special character to it - `'????'`
- Hence, it has been read as object instead of int64

Variable 'MetColor' :

```
print(np.unique(cars_data1['MetColor']))
```

```
[ 0.  1. nan nan nan nan nan nan nan nan
nan nan nan nan nan nan nan nan
```

- 'MetColor' have been read as float64 since it has values 0. & 1.

Unique elements of columns

Variable 'Automatic' :

```
print(np.unique(cars_data1['Automatic']))
[0 1]
```

- 'Automatic' has been read as int64 since it has values 0 & 1

Variable 'Doors' :

```
print(np.unique(cars_data1['Doors']))
['2' '3' '4' '5' 'five' 'four' 'three']
```

- 'Doors' has been read as object instead of int64 because of values 'five' 'four' 'three' which are strings

Summary

- Data types
  - Numeric
  - Character
- Checked data types of each column
- Count of unique data types
- Selected data based on data types
- Concise summary of dataframe
- Checked format of each column
- Got unique elements of each column


operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
_operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
_operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True
selection at the end -add
_ob.select= 1
ier_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
irror_ob.select = 0
= bpy.context.selected_ob
ata.objects[one.name].se
int("please select exactly

THANK YOU