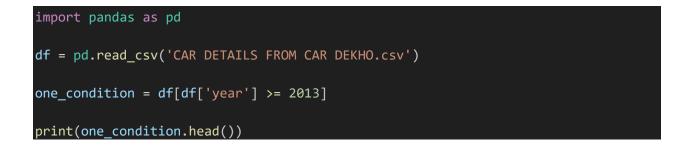
Five Ways to do Conditional Filtering in Pandas

	А	В	С	D	E	F	G	н	1
1	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	
2	Maruti 800 AC	2007	60000	70000	Petrol	Individual	Manual	First Owner	r -
3	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Individual	Manual	First Owner	r
4	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Individual	Manual	First Owner	c i
5	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner	r -
6	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Own	ner
7	Maruti Alto LX BSIII	2007	140000	125000	Petrol	Individual	Manual	First Owner	r
8	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner	r -
9	Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Own	ner
10	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner	r
11	Maruti Celerio Green VXI	2017	365000	78000	CNG	Individual	Manual	First Owner	r i
12	Chevrolet Sail 1.2 Base	2015	260000	35000	Petrol	Individual	Manual	First Owner	r

Filtering Method 1: Selection Brackets

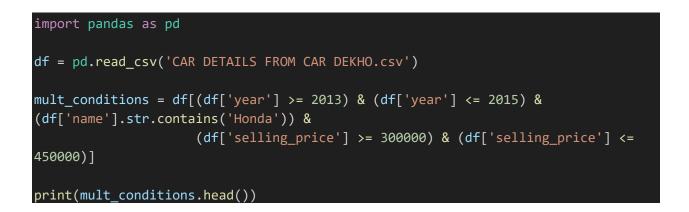
Finding all the vehicles that have a year of 2013 or newer is a fairly standard Pandas filtering task: select the column of the dataset to filter on, tell it what value to filter against, and plug that condition into brackets for the entire dataframe.



output:-

13	D. (al clitvez & C./Osel s/DI	8 Data	Appraca/ Locar/	riograms/ry	chon/ i y	monstry by the	u./arc	nitve/ car dekno.p
	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
6	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
7	Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
8	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner
DS	D:\archive>							

If we want to make our multi-condition search, we can put each individual filters inside parentheses () separated by our Boolean search criteria (& for and, | for or, and ~ for not).



Output:-

		name	year	selling_price	km_driven	fuel	seller_type	transmission		owner
4	1	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second	Owner
1	17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second	Owner
8	37	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First	Owner
2	235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First	Owner
2	245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First	Owner

These multiple conditions technically work, but the readability of this code is not great. There are brackets and parentheses all over the place. To clean up the code and use fewer conditions, pandas has various methods that we can apply for the same results, one of which we just used in the code chunk above, called str.contains().

Filtering Method 2: Selection Brackets with Series Functions

The reason we look at series methods as we filter is because each column of our Pandas.DataFrame individually is a Pandas.Series element, so we can apply Pandas.Series methods and functionality to it.

There are numerous methods we could use with the vehicles dataset, but to filter the data with our multiple condition example, we will use:

- isin() check to see if the series values are in a given list
- str.contains() check to see if a string is in the series
- between() find series value that are between two values

We will use isin() to check which vehicles meet our years of interest,

str.contains() to find which vehicles have Honda in the name, and between() to find vehicles in our price range.

```
import pandas as pd

df = pd.read_csv('CAR DETAILS FROM CAR DEKHO.csv')

years = [2013, 2014, 2015]

mult_condition_filters_methods = df[
   (df["year"].isin(years)) &
   (df["name"].str.contains("Honda")) &
   (df["selling_price"].between(300000, 450000))
]

print(mult_condition_filters_methods.head())
```

Output:-

	name	year	selling_price	km_driven	fuel	<pre>seller_type</pre>	transmission	owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
87	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First Owner
235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
PS D	·\archive>							

This cleans up the code somewhat, and takes advantage of a few Pandas.Series methods, but the code still isn't exactly readable. To make this look better, we can drop our code across multiple lines, one line per filtering action. The way to do that is by putting regular parentheses just inside our initial dataframe selection brackets, then inserting all conditions inside these parentheses.

output:-

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
87	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First Owner
235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner

Filtering Method 3: Selection Brackets with External Filters and Series Methods

A blend of the two methods above, we can define filters outside of our selection brackets as variables and then call each variable inside the selection brackets. This is a clean way to write each filter on its own individual line and then call all filters in one line of code. It means less overall parentheses and line breaks throughout the code.

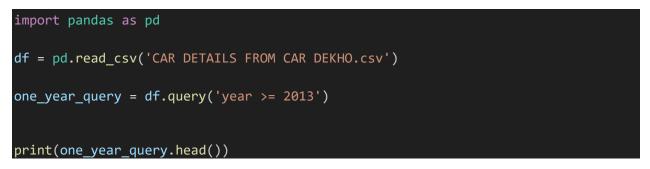
```
import pandas as pd
df = pd.read_csv('CAR DETAILS FROM CAR DEKHO.csv')
years = [2013,2014,2015]
filter_a = df['year'].isin(years)
filter_b = df['name'].str.contains('Honda')
filter_c = df['selling_price'].between(300000,450000)
mult_condition_filters = df[filter_a & filter_b & filter_c]
print(mult_condition_filters.head())
```

Output:-

	name	vear	selling price	km driven	fuel	seller type	transmission	owner
4	Honda Amaze VX i-DTEC	2014	450000		Diesel	Individual	Manual	Second Owner
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
87	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First Owner
235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner

Filtering Method 4: query()

I first heard of pandas.Series.query a year or two ago on a podcast, and I wasn't a fan at first. Over time, it has really grown on me. A query expression is a great way to subset data: they can be basic and easy or complex and powerful. The query expression to subset vehicles with years 2013 and newer is simple. You feed your filtering parameter(s) in as a string.

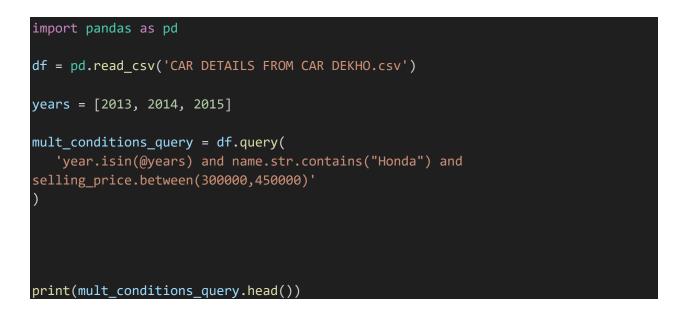


Output:-

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
3	Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
6	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
7	Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
8	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner

As you move on to multi-condition filters, you can make your query string more complex. Instead of typing & or | between your filter parameters, you simply type and or or, respectively. Below is the code to write a query expression for our multi-condition filter. *Note: to call variables that are inside

the environment but outside of the DataFrame/ Series you are querying, you need to use an @ before calling the variable. See the use of @ immediately before calling the list "years."



Output:-

		name	year	selling_price	km_driven	fuel	seller_type	transmission	owner
	1	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
	17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
	37	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First Owner
	235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
	245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
- 1		Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Own

This is a really neat way to subset your data! Yet, the more query parameters you add, the less readable it becomes. To overcome this problem, using query, we can simply add \ at the place where we want a line break and continue the query expression on the next line. If we want, we can maintain the notation of putting one filter condition per line.



Output:-

	name	year	selling_price	km_driven	fuel	seller_type	transmission		owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second	Owner
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second	Owner
87	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First	Owner
235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First	Owner
245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First	Owner

Filtering Method 5: loc[]

I really enjoy the power that comes with using python lambda functions. How can we translate lambda into filtering the vehicles dataset with our conditions? With the simple, single condition filter we have been applying, we call loc off of our dataframe, and with lambda, we can insert our condition.

```
import pandas as pd
df = pd.read_csv('CAR DETAILS FROM CAR DEKHO.csv')
one_year_loc = df.loc[lambda x: x["year"] >= 2013]
print(one_year_loc.head())
```

output:-

name	year	selling_price	km_driven	fuel	seller_type	transmission	owner .
3 Datsun RediGO T Option	2017	250000	46000	Petrol	Individual	Manual	First Owner
4 Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
6 Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Individual	Manual	First Owner
7 Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Individual	Manual	Second Owner
8 Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol	Individual	Manual	First Owner

If we want to add multiple conditions, we can just chain another loc off the results of the previous one. However, if left on one line, there are brackets and periods everywhere! It becomes very difficult to read. To make this more readable, we can wrap the entire right side of our expression in parentheses and then can put each loc filter on its own line.

Output:-

	name	year	selling_price	km_driven	fuel	<pre>seller_type</pre>	transmission	owner
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Individual	Manual	Second Owner
87	Honda Brio S MT	2015	371000	20000	Petrol	Dealer	Manual	First Owner
235	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner
245	Honda Mobilio V i DTEC	2014	300000	150000	Diesel	Individual	Manual	First Owner