Pandas for Beginners - Part 2 - Filtering Rows with 'iloc[]', 'loc[]', and 'query()'

July 11, 2025

1 Introduction

This is my Part 2 of Pandas guide. In this note I will show how to filter rows in the library of Python named Pandas. This guide includes using functions such as: loc; iloc and in the end we shall use a query function which is also popular in SQL.

1.1 Importing Pandas

As we made in the previous guide (watch Pandas guide from Part 1), we should import Pandas to start coding in Python:

[1]:
```
import pandas as pd # Importing Pandas
import matplotlib.pyplot as plt # Importing matplotlib for visualisation
```

1.2 Reading CSV Files

Also do not forget to read our CSV file which we used in the previous guide as well

[2]:
```
df = pd.read_csv('pokemon_data.csv')
```

1.3 Filtering Rows

When we see a whole DataFrame, we do not have time to look the dataframe and try to understand "Where is the rows, which i need urgently to analyse and get the required information from the row/rows?". And in this moment at the scene is coming loc[] and iloc[] functions, which we use to filter data frame based on rows and do not waste too much time to find and analyse the required information. iloc[] function is used to filter rows by index location, meanwhile loc[] function is used to filter rows by names. And in most often cases which meets Data Scienticts or Data Analysts you will be required to filter information by using loc[] function. About either functions, I will explain below.

2 iloc[] function

As I explained before this function is used to filter rows by their index in the DataFrame to see the information of specific row regardless of what information it contains. For example:

[3]:
```
df.iloc[2, 4] # Here we can see the information from second row and fourth column
```

[3]:
```
80
```

2.1 Multiple filtering rows

If we want to see multiple information of rows with required columns in the DataFrame, we can subsetting both rows and columns.

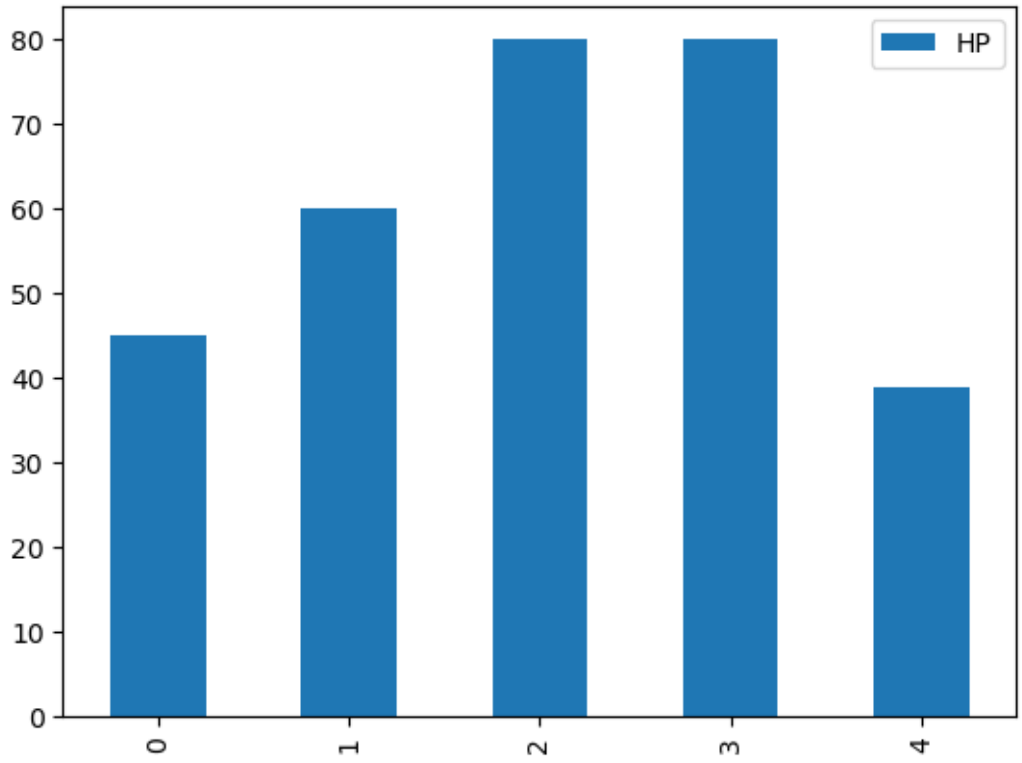
[4]:
```
df.iloc[:5, :5] # Here is the information of first five rows with first five columns
```

[4]:

	#	Name	Type 1	Type 2	HP
0	1	Bulbasaur	Grass	Poison	45
1	2	Ivysaur	Grass	Poison	60
2	3	Venusaur	Grass	Poison	80
3	3	VenusaurMega Venusaur	Grass	Poison	80
4	4	Charmander	Fire	NaN	39

[5]:
```
df.iloc[:5, 1:5].plot.bar() # Bar char of the DataFrame provided above
```

[5]:
```
<Axes: >
```



2.2 Filtering a single row

We can also provide a single value index of the row in the function which will filter a single row.

[6]:
```
df.iloc[5]
```

[6]:
```
#                     5
Name          Charmeleon
Type 1              Fire
Type 2               NaN
HP                    58
Attack                64
Defense               58
Sp. Atk               80
Sp. Def               65
Speed                 80
Generation             1
Legendary          False
Name: 5, dtype: object
```

If we want to filter a single row as DataFrame, we should put the single value index into the list.

[7]:
```
df.iloc[[5]]
```

[7]:
```
   #        Name Type 1 Type 2  HP  Attack  Defense  Sp. Atk  Sp. Def  Speed  \
5  5  Charmeleon   Fire    NaN  58      64       58       80       65     80

   Generation  Legendary
5           1      False
```

2.3 Filtering all rows

If we want to filter all rows and requied column as Series we should do this way:

[8]:
```
df.iloc[:, 1] # Filters all rows and "Name" column as Series
```

[8]:
```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                  Charmander
                ...
795                   Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                 Volcanion
Name: Name, Length: 800, dtype: object
```

But if we want to filter all rows and required column or columns as DataFrame we should put index of columns in the list:

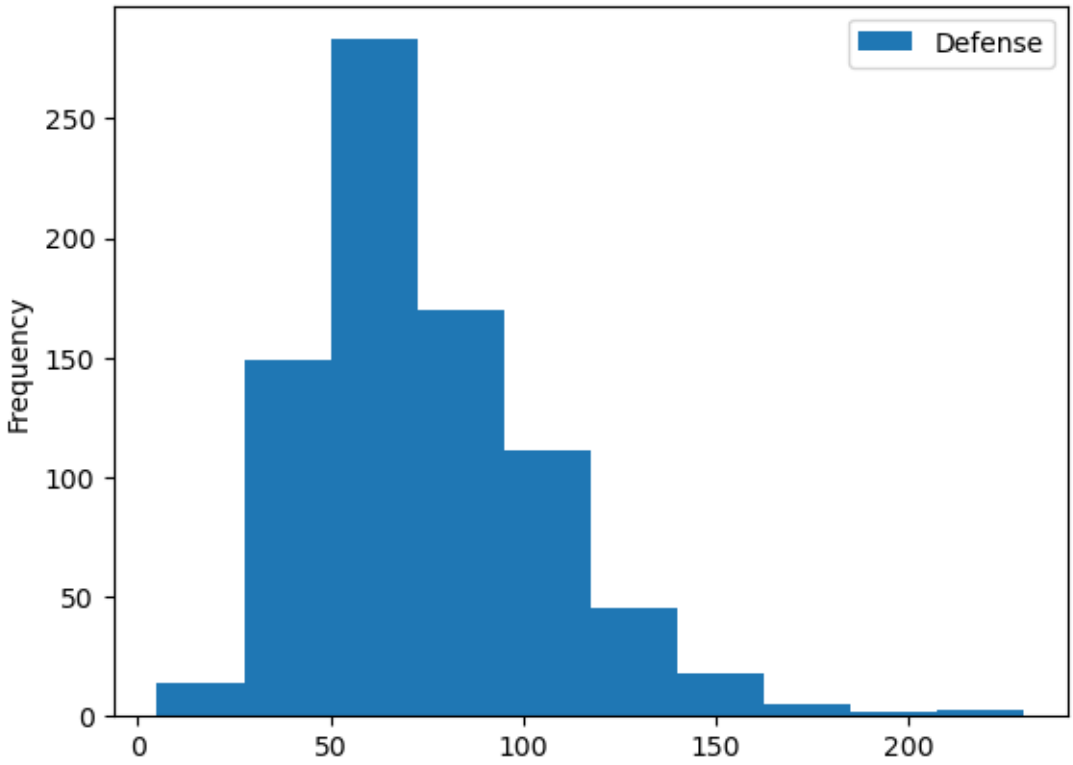
[9]:
```
df.iloc[:, [1, 6]] # Filters all rows and "Name" and "Defense" columns as␣
↪DataFrame
```

[9]:
```
                      Name  Defense
0                Bulbasaur       49
1                  Ivysaur       63
2                 Venusaur       83
3    VenusaurMega Venusaur      123
4               Charmander       43
..                     …       …
795               Diancie      150
796    DiancieMega Diancie      110
797    HoopaHoopa Confined       60
798     HoopaHoopa Unbound       60
799              Volcanion      120

[800 rows x 2 columns]
```

[10]:
```
df.iloc[:, [1, 6]].plot.hist() # Histogram of defence data
```

[10]:
```
<Axes: ylabel='Frequency'>
```



2.4 loc[] function

This function is used to filter rows as a DataFrame by using names. And one of the most useful things is that we are able to filter also boolean expressions for rows.

[11]:
```
df.loc[df['Legendary'] == False] # Filter rows where value of "Legendary"␣
↪column is False
```

[11]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	\
0	1	Bulbasaur	Grass	Poison	45	49	49	65	
1	2	Ivysaur	Grass	Poison	60	62	63	80	
2	3	Venusaur	Grass	Poison	80	82	83	100	
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	
4	4	Charmander	Fire	NaN	39	52	43	60	
..	…	…	…	…	..	…	…	…	
787	711	GourgeistSuper Size	Ghost	Grass	85	100	122	58	
788	712	Bergmite	Ice	NaN	55	69	85	32	
789	713	Avalugg	Ice	NaN	95	117	184	44	
790	714	Noibat	Flying	Dragon	40	30	35	45	
791	715	Noivern	Flying	Dragon	85	70	80	97	

	Sp. Def	Speed	Generation	Legendary
0	65	45	1	False
1	80	60	1	False
2	100	80	1	False
3	120	80	1	False
4	50	65	1	False
..	…	…	…	…
787	75	54	6	False
788	35	28	6	False
789	46	28	6	False
790	40	55	6	False
791	80	123	6	False

[735 rows x 12 columns]

2.5 Multiple filtering

We can combine all our boolean expressions with AND ('&' symbol) or OR ('|' symbol) statement in action of one loc[] function to filter on multiple things or columns.

[12]:
```
df.loc[
    (df['Legendary'] == False)
    & (df['Type 1'] == 'Grass')
]
```

[12]:
```
        #                   Name Type 1    Type 2   HP  Attack  Defense  \
0       1              Bulbasaur  Grass    Poison   45      49       49
1       2                Ivysaur  Grass    Poison   60      62       63
2       3               Venusaur  Grass    Poison   80      82       83
3       3  VenusaurMega Venusaur  Grass    Poison   80     100      123
48     43                 Oddish  Grass    Poison   45      50       55
..    ...                    ...    ...       ...  ...     ...      ...
718   650                Chespin  Grass       NaN   56      61       65
719   651              Quilladin  Grass       NaN   61      78       95
720   652             Chesnaught  Grass  Fighting   88     107      122
740   672                 Skiddo  Grass       NaN   66      65       48
741   673                 Gogoat  Grass       NaN  123     100       62

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
0         65       65     45           1      False
1         80       80     60           1      False
2        100      100     80           1      False
3        122      120     80           1      False
48        75       65     30           1      False
..       ...      ...    ...         ...        ...
718       48       45     38           6      False
719       56       58     57           6      False
720       74       75     64           6      False
740       62       57     52           6      False
741       97       81     68           6      False

[67 rows x 12 columns]
```

In the example provided above, we can see the filtered rows where value of "Legendary" column is equal to False and the value of "Type 1" column is equal to "Grass".

Sometimes you will have situations where needs to exclude unnecessary information of rows. In that case, we can use '~' symbol inside of the function.

[13]:
```
df.loc[
    ~((df['Type 1'] == 'Grass')
    & (df['Legendary'] == False))
]
```

[13]:
```
        #                     Name   Type 1  Type 2  HP  Attack  Defense  \
4       4               Charmander     Fire     NaN  39      52       43
5       5               Charmeleon     Fire     NaN  58      64       58
6       6                Charizard     Fire  Flying  78      84       78
7       6  CharizardMega Charizard X     Fire  Dragon  78     130      111
8       6  CharizardMega Charizard Y     Fire  Flying  78     104       78
..    ...                      ...      ...     ...  ..     ...      ...
795   719                  Diancie     Rock   Fairy  50     100      150
796   719       DiancieMega Diancie     Rock   Fairy  50     160      110
```

```
797  720        HoopaHoopa Confined  Psychic   Ghost  80     110       60
798  720         HoopaHoopa Unbound  Psychic    Dark  80     160       60
799  721                  Volcanion     Fire   Water  80     110      120

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
4         60       50     65           1      False
5         80       65     80           1      False
6        109       85    100           1      False
7        130       85    100           1      False
8        159      115    100           1      False
..       …        …      …           …         …
795      100      150     50           6       True
796      160      110    110           6       True
797      150      130     70           6       True
798      170      130     80           6       True
799      130       90     70           6       True

[733 rows x 12 columns]
```

In the example above we have excluded all rows which contained a value “Grass” and False statement.

3 Query() method

An alternatively way how to filter our DataFrame is using query method which takes in a string representation of the boolean expression you wish to filter on

[14]:
```
df.query('(HP > 40) and (Attack < 100)')
```

[14]:
```
       #                    Name  Type 1  Type 2  HP  Attack  Defense  Sp. Atk  \
0      1               Bulbasaur   Grass  Poison  45      49       49       65
1      2                 Ivysaur   Grass  Poison  60      62       63       80
2      3                Venusaur   Grass  Poison  80      82       83      100
5      5              Charmeleon    Fire     NaN  58      64       58       80
6      6               Charizard    Fire  Flying  78      84       78      109
..   …                       …      …       …  ..      …        …        …
784  711  GourgeistAverage Size   Ghost   Grass  65      90      122       58
785  711    GourgeistSmall Size   Ghost   Grass  55      85      122       58
786  711    GourgeistLarge Size   Ghost   Grass  75      95      122       58
788  712                Bergmite     Ice     NaN  55      69       85       32
791  715                 Noivern  Flying  Dragon  85      70       80       97

     Sp. Def  Speed  Generation  Legendary
0         65     45           1      False
1         80     60           1      False
2        100     80           1      False
5         65     80           1      False
```

```
6          85    100           1      False
..        …     …          …        …
784        75     84           6      False
785        75     99           6      False
786        75     69           6      False
788        35     28           6      False
791        80    123           6      False

[503 rows x 12 columns]
```

In the example above we putted in the query where in "HP" column values more than 40 and in "Attack" column values less than 100

3.1 '@' symbol in query method

Any values in the query are assumed to be columns but you can represent strings by wrapping the value in quotes or your query expression can access an external variable by using '@' symbol before the name in your query string.

[15]:
```
min_hp = 70
df.query('(HP > @min_hp)')
```

[15]:
```
       #                       Name   Type 1  Type 2   HP  Attack  Defense  \
2      3                   Venusaur    Grass  Poison   80      82       83
3      3      VenusaurMega Venusaur    Grass  Poison   80     100      123
6      6                  Charizard     Fire  Flying   78      84       78
7      6  CharizardMega Charizard X     Fire  Dragon   78     130      111
8      6  CharizardMega Charizard Y     Fire  Flying   78     104       78
..   …                          …      …      …   …     …       …
793  717                    Yveltal     Dark  Flying  126     131       95
794  718           Zygarde50% Forme   Dragon  Ground  108     100      121
797  720        HoopaHoopa Confined  Psychic   Ghost   80     110       60
798  720         HoopaHoopa Unbound  Psychic    Dark   80     160       60
799  721                  Volcanion     Fire   Water   80     110      120

     Sp. Atk  Sp. Def  Speed  Generation  Legendary
2        100      100     80           1      False
3        122      120     80           1      False
6        109       85    100           1      False
7        130       85    100           1      False
8        159      115    100           1      False
..       …      …    …          …        …
793      131       98     99           6       True
794       81       95     95           6       True
797      150      130     70           6       True
798      170      130     80           6       True
799      130       90     70           6       True
```

```
[321 rows x 12 columns]
```

In the exabple above we created a variable which indicates our minimum value of HP column, than we called query() method and refered our variable by '@' symbol to filter rows where values of HP column are more than minimum variable.

4 Conclusion

In this guide we have looked in detail the information of Pandas about filtering rows in the DataFrame. If this helped you — feel free to follow or connect. I am just starting to share my journey in data science and teaching. Part 3 coming soon!