



# Pandas for Beginners – Part 1



## DataFrame Basics

---



July 2025



by Ernest Kuderskyi

This notebook is part of an open learning series for beginners who want to understand how to work with tabular data using Python and Pandas library.



# Pandas for Beginners – Part 1

<Importing & Exploring DataFrames>

Ernest Kuderskyi

July 2025

## Introduction

This is my Part 1 of Pandas guide. In this note I will show how to use the library of Python named Pandas from importing Pandas to Subsetting columns. Pandas is the library for Python, which commonly used by Data Analysts in working and filtering DataFrames or

files which contains DataFrame. I originally created this note for myself, but I believe it could also help others who want to learn or improve their Pandas skills. Also not less important, this is my first note, so I will be happy If you would support me and do not be so critical to me.

## Importing Pandas

So, If we would like to start coding with Pandas, we have to import the library Pandas from our system in Python. To do this, we write this code in the line provided below:

```
In [4]: import pandas as pd # Importing pandas  
import matplotlib.pyplot as plt # Importing matplotlib for visualization
```

We can also print "as" keyword and "pd" alias in the line, to refer Pandas library instead printing "pandas".

## Reading CSV Files

So that the system could see, store and filter big data sets, we use csv files (Comma-Separated Values), which stores data in a tabular format and has the ability to separate data record in each table by commas. This format is widely used in data analysis and especially useful when working with the Pandas library. To read csv file we should store it in variable "df", which stands for DataFrame and call the function "read\_csv()" in the code provided below:

```
In [5]: df = pd.read_csv('pokemon_data.csv')
```

Then we just need to show our file as DataFrame:

```
In [20]: df # Shows only first and last five rows of the entire DataFrame
```

Out[20]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Gene
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	
...	...	...	...	...	...	...	...	...	...	...	
795	719	Diancie	Rock	Fairy	50	100	150	100	150	50	
796	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	
797	720	HoopaHoopa Confined	Psychic	Ghost	80	110	60	150	130	70	
798	720	HoopaHoopa Unbound	Psychic	Dark	80	160	60	170	130	80	
799	721	Volcanion	Fire	Water	80	110	120	130	90	70	

800 rows × 12 columns



But if we would like to print entire data frame, we should write the function "to\_string()" which can return all data records on the Terminal.

## DataFrame Basics

Most important basic methods, which returns head, tail, columns. But first we should talk about heads.

### Head() Method

The head() method returns the number of rows that are located at the top of the DataFrame. This method can be useful if we want to see only the number of rows that are located at the top of the DataFrame. For example:

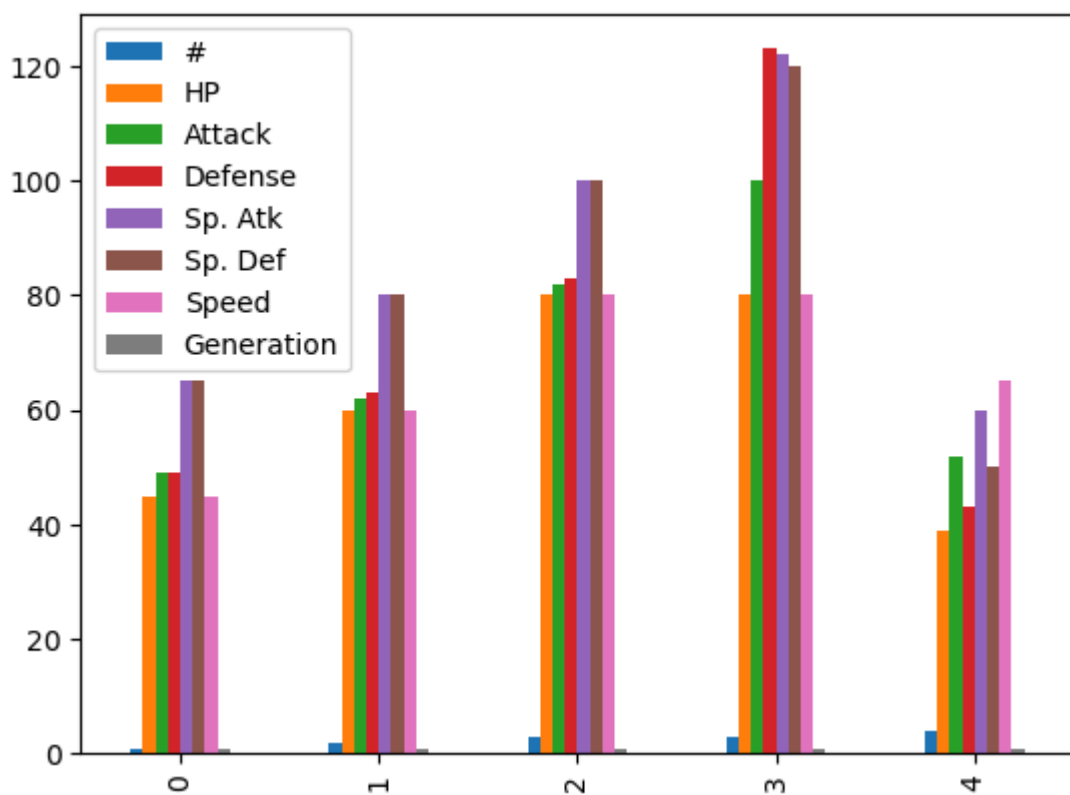
```
In [42]: df.head() # by default the command returns first five rows of the DataFrame
```

Out[42]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	1
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	1
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	1

In [41]: `df.head().plot.bar()` *# the bar chart of heads of the DataFrame*


Out[41]: <Axes: >



In [43]: `df.head(10)` *# the function returns first ten rows of the DataFrame*

Out[43]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generatio
0	1	Bulbasaur	Grass	Poison	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	
4	4	Charmander	Fire	NaN	39	52	43	60	50	65	
5	5	Charmeleon	Fire	NaN	58	64	58	80	65	80	
6	6	Charizard	Fire	Flying	78	84	78	109	85	100	
7	6	CharizardMega Charizard X	Fire	Dragon	78	130	111	130	85	100	
8	6	CharizardMega Charizard Y	Fire	Flying	78	104	78	159	115	100	
9	7	Squirtle	Water	NaN	44	48	65	50	64	43	



## Tail() Method

Unlike the head() Method, tail() Method returns the number of rows, which are located at the bottom of the DataFrame.

```
In [8]: df.tail(10) # the function returns last ten rows of the DataFrame
```

Out[8]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Gene
<b>790</b>	714	Noibat	Flying	Dragon	40	30	35	45	40	55	
<b>791</b>	715	Noivern	Flying	Dragon	85	70	80	97	80	123	
<b>792</b>	716	Xerneas	Fairy	NaN	126	131	95	131	98	99	
<b>793</b>	717	Yveltal	Dark	Flying	126	131	95	131	98	99	
<b>794</b>	718	Zygarde50% Forme	Dragon	Ground	108	100	121	81	95	95	
<b>795</b>	719	Diancie	Rock	Fairy	50	100	150	100	150	50	
<b>796</b>	719	DiancieMega Diancie	Rock	Fairy	50	160	110	160	110	110	
<b>797</b>	720	HoopaaHoopaa Confined	Psychic	Ghost	80	110	60	150	130	70	
<b>798</b>	720	HoopaaHoopaa Unbound	Psychic	Dark	80	160	60	170	130	80	
<b>799</b>	721	Volcanion	Fire	Water	80	110	120	130	90	70	

## Info() Method

Sometimes data analysts has situations, when they are need to get a short summary of the data set e.g a value of columns, rows, memory usage etc. In that case, we can use info() method, which is provide the information that user requires.

In [29]: `df.info()` # provides a full information of the entire DataFrame

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 12 columns):
#   Column      Non-Null Count  Dtype
---  -
0   #           800 non-null   int64
1   Name        800 non-null   object
2   Type 1      800 non-null   object
3   Type 2      414 non-null   object
4   HP          800 non-null   int64
5   Attack      800 non-null   int64
6   Defense     800 non-null   int64
7   Sp. Atk     800 non-null   int64
8   Sp. Def     800 non-null   int64
9   Speed       800 non-null   int64
10  Generation  800 non-null   int64
11  Legendary   800 non-null   bool
dtypes: bool(1), int64(8), object(3)
memory usage: 69.7+ KB
```

As we can see, we have 800 rows and 12 columns. The memory usage is more than 69.7 KB.

```
In [28]: df.info(verbose=False) # provides a short information of the entire DataFrame
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 800 entries, 0 to 799  
Columns: 12 entries, # to Legendary  
dtypes: bool(1), int64(8), object(3)  
memory usage: 69.7+ KB
```


## Sample() Method

If we want to return a random subset of rows from the entire data set, we can use a sample method, which will return an amount or a fraction of required random subset of rows provided into the parentheses. We can use this method to see a random subset of rows from the entire data set in fast way.

```
In [16]: df.sample(5) # In this sample, the function returns 5 random subsets
```

```
Out[16]:
```

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Ger
245	227	Skarmory	Steel	Flying	65	80	140	40	70	70	
405	370	Luvdisc	Water	NaN	43	30	55	40	65	97	
196	181	AmpharosMega Ampharos	Electric	Dragon	90	95	105	165	110	45	
659	598	Ferrothorn	Grass	Steel	74	94	131	54	116	20	
631	570	Zorua	Dark	NaN	40	65	40	80	40	65	



```
In [22]: df.sample(frac = 0.1) # In this sample, the function returns a fraction of random
```

Out[22]:

	#	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
664	603	Eelektrek	Electric	NaN	65	85	70	75	70	40	
234	216	Teddiursa	Normal	NaN	60	80	50	50	50	40	
279	257	BlazikenMega Blaziken	Fire	Fighting	80	160	80	130	80	100	
88	81	Magnemite	Electric	Steel	25	35	70	95	55	45	
9	7	Squirtle	Water	NaN	44	48	65	50	64	43	
...	...	...	...	...	...	...	...	...	...	...	
558	499	Pignite	Fire	Fighting	90	93	55	70	55	55	
126	117	Seadra	Water	NaN	55	65	95	95	45	85	
538	481	Mesprit	Psychic	NaN	80	105	105	105	105	80	
629	568	Trubbish	Poison	NaN	50	50	62	40	62	65	
504	454	Toxicroak	Poison	Fighting	83	106	65	86	65	85	

80 rows × 12 columns



## Columns Function

If we want to see names of the DataFrame, the columns function can help you to provide it in 1D list. This function can be useful to see all names of columns in the DataFrame in fast way.

```
In [24]: df.columns
```

```
Out[24]: Index(['#', 'Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def', 'Speed', 'Generation', 'Legendary'], dtype='object')
```

## Index() Method

If we want to see the value of rows of the DataFrame in 1D list, here is the index method which shows the row number of the DataFrame.

```
In [26]: df.index
```

```
Out[26]: RangeIndex(start=0, stop=800, step=1)
```

## Describe() Method

This method can be run on any DataFrame to provide some descriptive statistics of columns, e.g. mean, quartiles, standard deviation etc. in fast way.



```
In [35]: df.describe()
```

```
Out[35]:
```

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Spe
count	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000
mean	362.813750	69.258750	79.001250	73.842500	72.820000	71.902500	68.277500
std	208.343798	25.534669	32.457366	31.183501	32.722294	27.828916	29.060400
min	1.000000	1.000000	5.000000	5.000000	10.000000	20.000000	5.000000
25%	184.750000	50.000000	55.000000	50.000000	49.750000	50.000000	45.000000
50%	364.500000	65.000000	75.000000	70.000000	65.000000	70.000000	65.000000
75%	539.250000	80.000000	100.000000	90.000000	95.000000	90.000000	90.000000
max	721.000000	255.000000	190.000000	230.000000	194.000000	230.000000	180.000000



If we want to see the descriptive statistics of unique column, between the end of name of variable and the dot, we should put into double square brackets names of unique columns, which we want to review.

```
In [41]: df[['HP', 'Attack']].describe() # In our case we are considering "HP" and "Attack"
```

```
Out[41]:
```

	HP	Attack
count	800.000000	800.000000
mean	69.258750	79.001250
std	25.534669	32.457366
min	1.000000	5.000000
25%	50.000000	55.000000
50%	65.000000	75.000000
75%	80.000000	100.000000
max	255.000000	190.000000

## Subsetting Columns

We should be ready to demonstrate information of an unique columns. This can help us do not waste too much time to find required information and do not interact with extra columns. We can select columns as Series or as DataFrame.

## Selecting columns as DataFrame

If we want to select columns as DataFrame, after the name of our DataFrame we put in double square brackets names of the columns which we want to review.

Note: In the previous section, we used the describe() function to view descriptive statistics of the dataset.

```
In [30]: df[['HP']] # Shows the DataFrame of "HP" column
```

```
Out[30]:
```

	HP
0	45
1	60
2	80
3	80
4	39
...	...
795	50
796	50
797	80
798	80
799	80

800 rows × 1 columns

## Selecting columns as Series

In that case the terminal returns series's of the columns which we putted into the single square brackets.

```
In [17]: df['HP'] # Shows the DataFrame of "HP" column
```

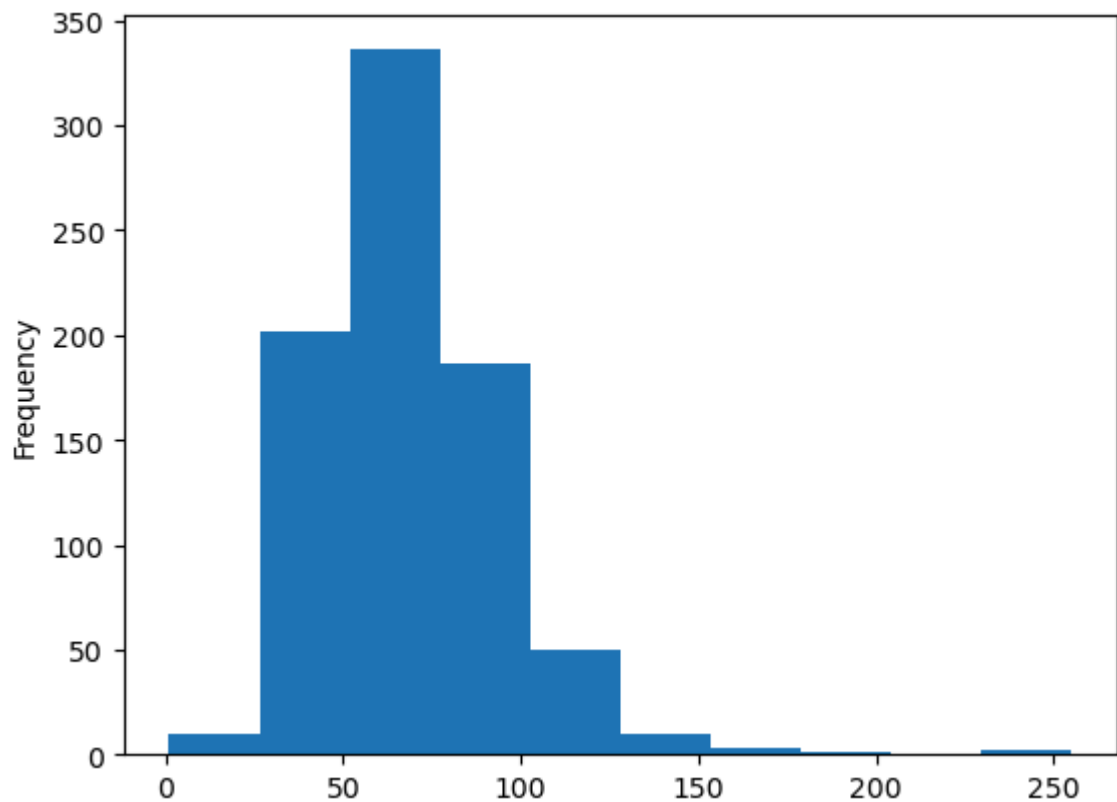
```
Out[17]:
```

0	45
1	60
2	80
3	80
4	39
...	..
795	50
796	50
797	80
798	80
799	80

Name: HP, Length: 800, dtype: int64

```
In [6]: df['HP'].plot.hist() # Shows the histogram of DataFrame in "HP" column
```

```
Out[6]: <Axes: ylabel='Frequency'>
```



## Substituting Columns by Number

If we want to see row information for unique columns and not including their names, we can take a few columns that we would like to see and substitute our DataFrame by the number of columns we took earlier. This feature is very useful in situations where we do not know the exact names of the columns or those names are too long to filter by name.

```
In [10]: df[df.columns[:4]] # Here we can see the information of rows only for the first
```

Out[10]:

	#	Name	Type 1	Type 2
0	1	Bulbasaur	Grass	Poison
1	2	Ivysaur	Grass	Poison
2	3	Venusaur	Grass	Poison
3	3	VenusaurMega Venusaur	Grass	Poison
4	4	Charmander	Fire	NaN
...	...	...	...	...
795	719	Diancie	Rock	Fairy
796	719	DiancieMega Diancie	Rock	Fairy
797	720	HoopaHoopa Confined	Psychic	Ghost
798	720	HoopaHoopa Unbound	Psychic	Dark
799	721	Volcanion	Fire	Water

800 rows × 4 columns

## Filtering columns by data type

Here is the situation when we select columns and showing rows by data type of the columns.

In [20]: `df.select_dtypes('int')` *# here we see rows only with filtering columns as integers*

Out[20]:

	#	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation
0	1	45	49	49	65	65	45	1
1	2	60	62	63	80	80	60	1
2	3	80	82	83	100	100	80	1
3	3	80	100	123	122	120	80	1
4	4	39	52	43	60	50	65	1
...	...	...	...	...	...	...	...	...
795	719	50	100	150	100	150	50	6
796	719	50	160	110	160	110	110	6
797	720	80	110	60	150	130	70	6
798	720	80	160	60	170	130	80	6
799	721	80	110	120	130	90	70	6

800 rows × 8 columns

# Conclusion

In this guide we have detaily looked the information of Pandas from importing to subsetting columns. If this helped you — feel free to follow or connect. I'm just starting to share my journey in data science and teaching. Part 2 coming soon!