



After downloading Anaconda, launch Jupyter Notebook from its Navigator. Then, from the "New" menu, choose "Notebook" and select the desired data from your computer. Now you are ready to start your journey!

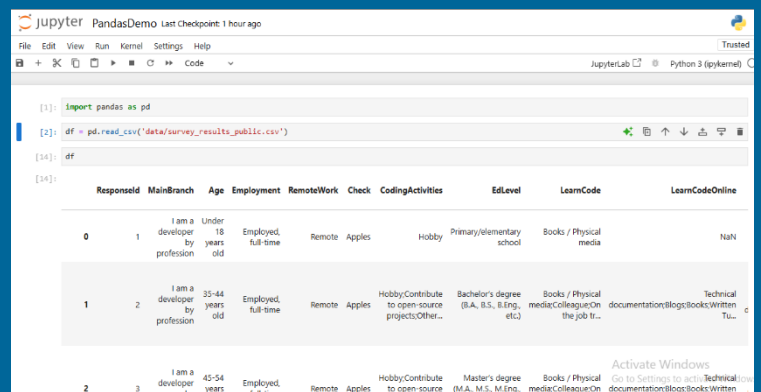
In the first cell you will need to import pandas

```
import pandas as pd
```

1. To load a CSV file into a DataFrame using pandas :

```
df = pd.read_csv('folder name/the desired file name.csv')
```

It looks like this:



The screenshot shows a Jupyter Notebook window titled "PandasDemo" with the following code cells and output:

```
[1]: import pandas as pd
```

```
[2]: df = pd.read_csv('data/survey_results_public.csv')
```

```
[3]: df
```

The output displays the first few rows of the DataFrame:

	ResponseId	MainBranch	Age	Employment	RemoteWork	Check	CodingActivities	EdLevel	LearnCode	LearnCodeOnline
0	1	I am a developer by profession	Under 18 years old	Employed, full-time	Remote	Applies	Hobby	Primary/elementary school	Books / Physical media	NaN
1	2	I am a developer by profession	35-44 years old	Employed, full-time	Remote	Applies	Hobby/Contribute to open-source projects/Other...	Bachelor's degree (B.A., B.S., B.Eng., etc.)	Books / Physical media/Colleagues/On the job tr...	Technical documentation/Blogs/Books/Written Tu...
2	3	I am a developer by profession	45-54 years old	Employed, full-time	Remote	Applies	Hobby/Contribute to open-source projects/Other...	Master's degree (M.A., M.S., M.Eng., etc.)	Books / Physical media/Colleagues/On the job tr...	Technical documentation/Blogs/Books/Written Tu...

2. To Inspect the shape of the DataFrame:

```
df.shape
```

This returns a tuple representing the number of rows and columns in df.

```
[4]: df.shape  
  
[4]: (65437, 114)
```

3. To Get information about the DataFrame:

This provides a concise summary of the DataFrame including the index dtype, column dtypes, non-null counts, and memory usage.

```
df.info()
```

4. To Set display options

```
pd.set_option('display.max_columns', Your desired maximum columns number)  
pd.set_option('display.max_rows' , Your desired maximum rows number)
```

These commands set the maximum number of columns and rows pandas will display when printing a DataFrame. Adjust these numbers based on your needs.

5. To View the first few rows of the DataFrame

This displays the first 10 rows of df. Adjust this number based on your needs.

```
df.head(10)
```

6.View the last few rows of the DataFrame:

This displays the last 5 rows of df . You can pass a number to see more rows if needed.

```
df.tail(5)
```

.iloc (Integer Location)

Purpose: Accesses data by integer position (i.e., row and column indices).

Usage: Works with integer-based indexing. You use integer indices to specify the row and column positions.

Syntax: df.iloc[rows, columns]

Rows: Integer index or list of integers to select specific rows.

Columns: Integer index or list of integers to select specific columns.

.loc (Label Location)

Purpose: Accesses data by label (i.e., row and column names).

Usage: Works with label-based indexing. You use the labels of rows and columns to specify which data to access.

Syntax: df.loc[rows, columns]

Rows: Labels or list of labels to select specific rows.

Columns: Labels or list of labels to select specific columns.

Key Differences

Indexing Type:

.iloc uses integer-based indexing (positions).

.loc uses label-based indexing (names).

Indexing Behavior:

.iloc does not include the end index in slices (exclusive end).

.loc includes the end index in slices (inclusive end).

Error Handling:

.iloc will raise an IndexError if integer positions are out of range.

.loc will raise a KeyError if labels do not exist.

To filter and manipulate the data based on certain conditions

Filtering rows where a certain row contains a desired condition :

```
filt = df['LanguageWorkedWith'].str.contains('Python', na=False)
```

The na=False argument ensures that any NaN values in 'LanguageWorkedWith' are treated as False.

Filtering rows where 'last' is 'Schafer' or 'first' is 'John':

```
filt = (df['last'] == 'Schafer') | (df['first'] == 'John')
```

This creates another boolean Series (filt) where each entry is True if either 'last' equals 'Schafer' or 'first' equals 'John'.

Selecting 'email' column values for rows not matching the filter:

```
df.loc[~filt, 'email']
```

sort a DataFrame :

```
df.sort_values(by=['Country', 'ConvertedComp'], ascending=[True, False],  
inplace=True)
```

you sort the DataFrame df by the 'Country' column in ascending order and by the 'ConvertedComp' column in descending order.

Retrieve the 10 large / smallest st values from the 'ConvertedComp' column.



```
df['ConvertedComp'].nlargest(10)
```



```
df.nsmallest(10, 'ConvertedComp')
```

After sorting by columns, sort the DataFrame by its index :



```
df.sort_index()
```