



Jirka Lhotka / Most Common Pandas Cheatsheet Questions Published at Apr 12, 2021





#### **Most Common Pandas Cheetsheet Questions**

Let's face it. Pandas API can be pretty confusing. They sometimes use camelCase instead of pascal\_case and the names of the functions are often not the easiest to remember. Is it count\_values or value\_counts or values\_counts ? I never know and that's why I end up searching for the same things over and over again.

I decided to make this notebook to put all the common things I'm googling into one place. These are the most common questions I found useful on StackOverflow. Every answer will link to the original post whose authors deserve all the credit.

# Select rows based on column values from Pandas **DataFrame**

#### Columns that equal value

```
# some_value is scalar (e.g. a number)
df.loc[df['column_name'] == some_value]
# some_values is iterable (e.g. a list)
df.loc[df['column_name'].isin(some_values)]
# Use & to combine multiple conditions. Note the parantheses!
df.loc[(df['column_name'] >= A) & (df['column_name'] <= B)]</pre>
```

#### Columns that do not equal value

```
# Use != for rows which do not equal scalar some_value (e.g. a number)
df.loc[df['column_name'] != some_value]
# Use ~ for rows which do not equal iterable some_value (e.g. a list)
df.loc[~df['column_name'].isin(some_values)]
```

Source

# **Select multiple columns of Pandas DataFrame**

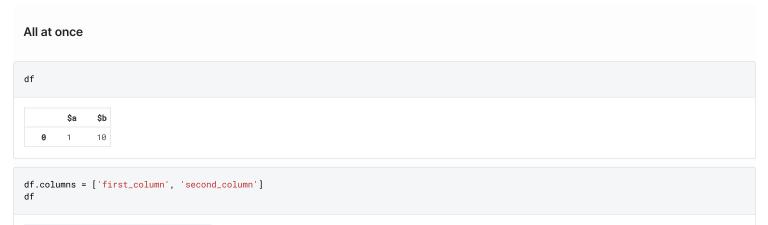
```
# By name
df1 = df[['a',
              'b']] # Note this produces a copy
# By index
df1 = df.iloc[:. 0:2] # Remember that Python does not slice inclusive of the ending index.
```

Source

#### Iterate over rows of Pandas DataFrame

Don't do it! It's not idiomatic. Vectorise your operations instead. Click here for full reasoning

#### Rename columns of Pandas DataFrame



#### Only some

first\_column

second\_column

10

Source

## **Delete columns of Pandas DataFrame**

```
# columns
df.drop(columns=['B','C'])
# rows
df.drop(index=[0,1])
```

**Source** 

# **Get row/column count of Pandas DataFrame**

```
# rows
len(df.index)
# rows
len(df.column)
# both (but slow on big datasets)
rows_count, columns_count = df.shape
```

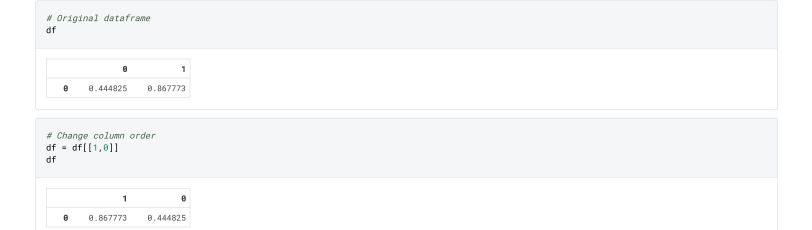
Source

## **Get list of column headers of Pandas DataFrame**

```
# If you hate typing
list(df)
# If you hate not being explicit
list(df.columns.values)
```

Source

# Rearange the order of columns of Pandas DataFrame



Source

### Add new column to Pandas DataFrame

```
# Simple version
df['new_name'] = new_column
# Proper version recommended by Pandas
df = df.assign(new_name=new_column)
```

Source

#### Add new rom to Pandas DataFrame

Don't do it! It's slow and unidiomatic. Gather all the data first and only create the dataframe after.

Source

# Drop rows whose values in a certain column is NaN in Pandas DataFrame

# Subset lists columns you care about

```
df.dropna(subset = ['column1_name', 'column2_name', 'column3_name'])
```

Source

# **Change column type in Pandas DataFrame**

```
# convert column "a" to int64 dtype and "b" to np.float64 type
df = df.astype({"a": int, "b": np.float64})
```

Source

# Delete row based on value of particular column from Pandas DataFrame

See also "how to select rows based on column values" for other options

```
df = df[df.relevant_column != some_value]
```

Source

#### Save Pandas DataFrame to a CSV

```
df.to_csv(file_name, sep='\t', encoding='utf-8', index=False)
```

Source

## Is it count\_values or values\_count or what?

It's value\_counts. If you asked this question, you might wanna try Deepnote which has autocomplete and would tell you.

df.value\_counts()

Source