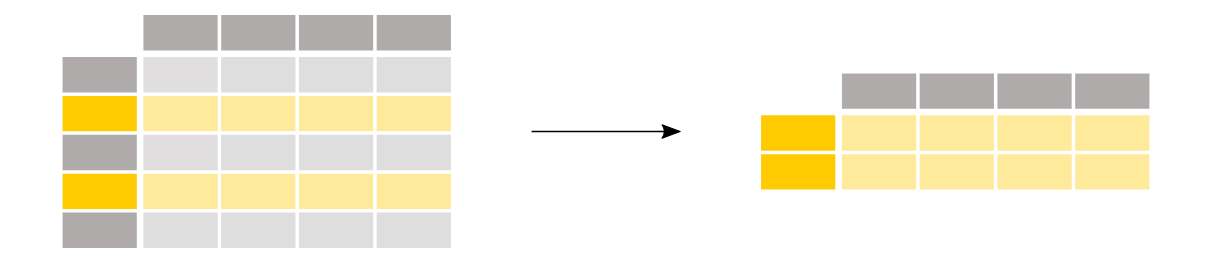
How do I filter specific rows from a DataFrame?

Use (isin() function)



I’m interested in the passengers older than 35 years.

above\_35 = titanic[titanic["Age"] > 35]

In [13]: above\_35.head()

To select rows based on a conditional expression, use a condition inside the selection brackets [].

The condition inside the selection brackets titanic["Age"] > 35 checks for which rows the Age column has a value larger than 35:

titanic["Age"] > 35

it display results in the true and false format

The output of the conditional expression (>, but also ==, !=, <, <=,… would work) is actually a pandas Series of boolean values (either True or False) with the same number of rows as the original DataFrame. Such a Series of boolean values can be used to filter the DataFrame by putting it in between the selection brackets []. Only rows for which the value is True will be selected.

We know from before that the original Titanic DataFrame consists of 891 rows. Let’s have a look at the number of rows which satisfy the condition by checking the shape attribute of the resulting DataFrame above\_35:

above\_35.shape

Out[15]: (217, 12)

I’m interested in the Titanic passengers from cabin class 2 and 3.

class\_23 = titanic[titanic["Pclass"].isin([2, 3])]

In [17]: class\_23.head()

Similar to the conditional expression, the **[isin()](https://pandas.pydata.org/docs/reference/api/pandas.Series.isin.html" \l "pandas.Series.isin" \o "pandas.Series.isin)** conditional function returns a True for each row the values are in the provided list. To filter the rows based on such a function, use the conditional function inside the selection brackets []. In this case, the condition inside the selection brackets titanic["Pclass"].isin([2, 3]) checks for which rows the Pclass column is either 2 or 3.

The above is equivalent to filtering by rows for which the class is either 2 or 3 and combining the two statements with an | (or) operator:

class\_23 = titanic[(titanic["Pclass"] == 2) | (titanic["Pclass"] == 3)]

class\_23.head()

**Note**

When combining multiple conditional statements, each condition must be surrounded by parentheses (). Moreover, you can not use or/and but need to use the or operator | and the and operator &.

I want to work with passenger data for which the age is known.

Use notna() function

age\_no\_na = titanic[titanic["Age"].notna()]

In [21]: age\_no\_na.head()

The **[notna()](https://pandas.pydata.org/docs/reference/api/pandas.Series.notna.html" \l "pandas.Series.notna" \o "pandas.Series.notna)** conditional function returns a True for each row the values are not a Null value. As such, this can be combined with the selection brackets [] to filter the data table.

It display only true values whose not null we vcan see another example

print('use notna() function return those rows who have not null values')

df1=pd.DataFrame([['anu','manu','rita'],['yes','no',None]])

df1

pd.notna(df1)#now it displays all values true and wher is none it shows false

output

| **0** | **1** | **2** |
| --- | --- | --- |
| 0 | True | True | True |
| 1 | True | True | False |

You might wonder what actually changed, as the first 5 lines are still the same values. One way to verify is to check if the shape has changed:

age\_no\_na.shape

Out[22]: (714, 12)

## How do I select specific rows and columns from a DataFrame?

Use loc and iloc function



I’m interested in the names of the passengers older than 35 years.

adult\_names = titanic.loc[titanic["Age"] > 35, "Name"]

In [24]: adult\_names.head()

In this case, a subset of both rows and columns is made in one go and just using selection brackets [] is not sufficient anymore. The loc/iloc operators are required in front of the selection brackets []. When using loc/iloc, the part before the comma is the rows you want, and the part after the comma is the columns you want to select.

When using the column names, row labels or a condition expression, use the loc operator in front of the selection brackets []. For both the part before and after the comma, you can use a single label, a list of labels, a slice of labels, a conditional expression or a colon. Using a colon specifies you want to select all rows or columns.

I’m interested in rows 10 till 25 and columns 3 to 5.

titanic.iloc[9:25, 2:5]

Again, a subset of both rows and columns is made in one go and just using selection brackets [] is not sufficient anymore. When specifically interested in certain rows and/or columns based on their position in the table, use the iloc operator in front of the selection brackets [].

When selecting specific rows and/or columns with loc or iloc, new values can be assigned to the selected data. For example, to assign the name anonymous to the first 3 elements of the fourth column:

titanic.iloc[0:3, 3] = "anonymous"

#### REMEMBER

* When selecting subsets of data, square brackets [] are used.
* Inside these brackets, you can use a single column/row label, a list of column/row labels, a slice of labels, a conditional expression or a colon.
* Select specific rows and/or columns using loc when using the row and column names.
* Select specific rows and/or columns using iloc when using the positions in the table.
* You can assign new values to a selection based on loc/iloc.