**Using iloc, loc, & ix to select rows and columns in Pandas DataFrames**

[76 Comments](https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/#comments) / [blog](https://www.shanelynn.ie/category/blog/), [data science](https://www.shanelynn.ie/category/blog/data-science/), [Pandas](https://www.shanelynn.ie/category/pandas/), [python](https://www.shanelynn.ie/category/blog/python/), [Tutorials](https://www.shanelynn.ie/category/tutorials/) / By [shanelynn](https://www.shanelynn.ie/author/shanelynn/)



**Pandas Data Selection**

There are [multiple ways to select](http://pandas.pydata.org/pandas-docs/stable/indexing.html#different-choices-for-indexing) and index rows and columns from [Pandas](http://pandas.pydata.org/) [DataFrames](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html). I find tutorials online focusing on advanced selections of row and column choices a little complex for my requirements.

**Selection Options**

There’s three main options to achieve the selection and indexing activities in Pandas, which can be confusing. The three selection cases and methods covered in this post are:

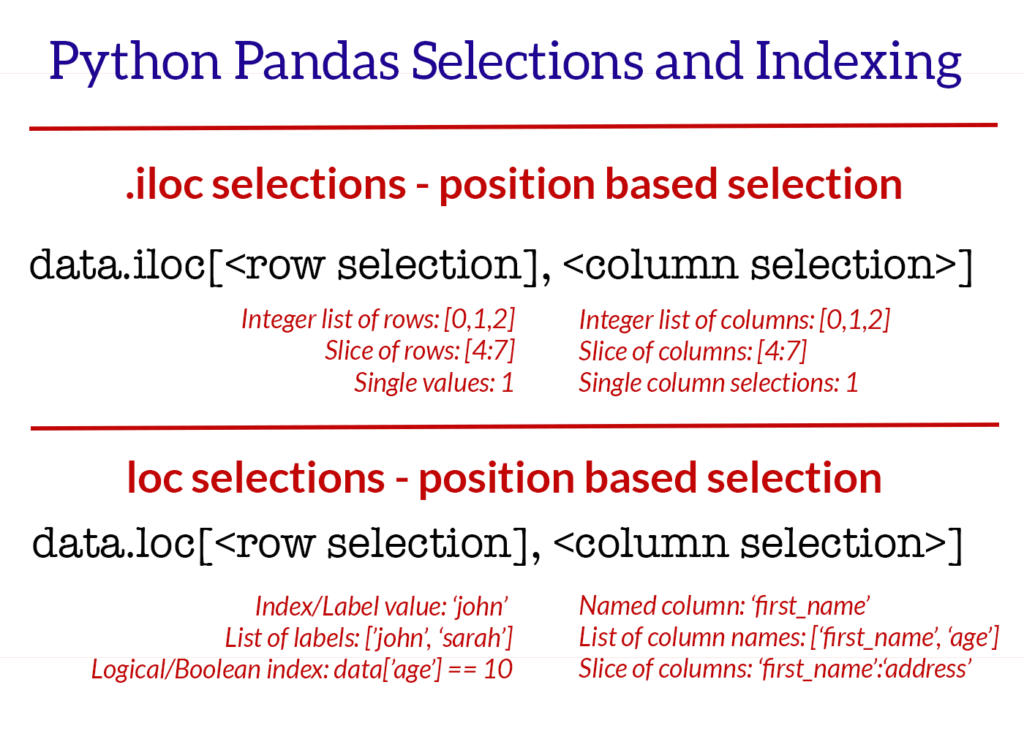
1. [Selecting data by row numbers (.iloc)](https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/#iloc-selection)
2. [Selecting data by label or by a conditional statment (.loc)](https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/#loc-selection)
3. [Selecting in a hybrid approach (.ix) (now Deprecated in Pandas 0.20.1)](https://www.shanelynn.ie/select-pandas-dataframe-rows-and-columns-using-iloc-loc-and-ix/#ix-selection)

**Data Setup**

This blog post, [inspired by other tutorials](http://chrisalbon.com), describes selection activities with these operations. The tutorial is suited for the general data science situation where, typically I find myself:

1. Each row in your data frame represents a data sample.
2. Each column is a variable, and is usually named. I rarely select columns without their names.
3. I need to quickly and often select relevant rows from the data frame for modelling and visualisation activities.

For the uninitiated, the [Pandas](http://pandas.pydata.org) library for Python provides high-performance, easy-to-use data structures and data analysis tools for handling tabular data in “series” and in “data frames”. It’s brilliant at making your data processing easier and I’ve written before about [grouping and summarising data](http://104.236.88.249/summarising-aggregation-and-grouping-data-in-python-pandas/) with Pandas.

[](http://104.236.88.249/wp-content/uploads/2016/10/Pandas-selections-and-indexing.png)Summary of iloc and loc methods discussed in this blog post. iloc and loc are operations for retrieving data from Pandas dataframes.

**Selection and Indexing Methods for Pandas DataFrames**

For these explorations we’ll need some sample data – I downloaded the uk-500 sample data set from [www.briandunning.com](https://www.briandunning.com/sample-data/). This data contains artificial names, addresses, companies and phone numbers for fictitious UK characters. To follow along, you can download the .csv file [here](https://s3-eu-west-1.amazonaws.com/shanebucket/downloads/uk-500.csv). Load the data as follows (the diagrams here come from a [Jupyter notebook](http://jupyter.org/) in the [Anaconda Python](https://www.continuum.io/downloads) install):

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| import pandas as pd |
|  |

|  |
| --- |
| import random |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # read the data from the downloaded CSV file. |
|  |

|  |
| --- |
| data = pd.read\_csv('https://s3-eu-west-1.amazonaws.com/shanebucket/downloads/uk-500.csv') |
|  |

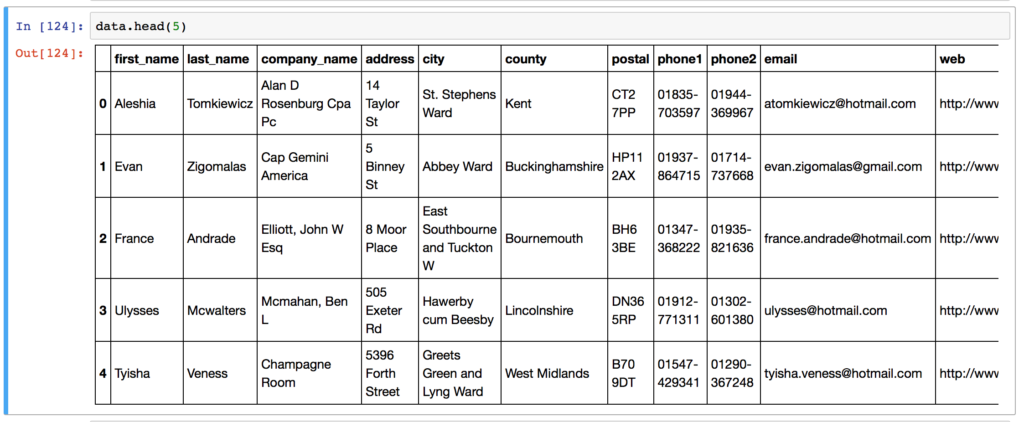
|  |
| --- |
| # set a numeric id for use as an index for examples. |
|  |

|  |
| --- |
| data['id'] = [random.randint(0,1000) for x in range(data.shape[0])] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| data.head(5) |

[view raw](https://gist.github.com/shanealynn/3324456b1a22eae86fce40bdd744f102/raw/2dd4ceaff99b0118fb72ac54fbc5630f66132b76/Pandas%20Index%20-%20Loading%20Data.py) [Pandas Index - Loading Data.py](https://gist.github.com/shanealynn/3324456b1a22eae86fce40bdd744f102#file-pandas-index-loading-data-py) hosted with by [GitHub](https://github.com)

[](http://104.236.88.249/wp-content/uploads/2016/09/pandas_indexing_example-1.png)Example data loaded from CSV file.

**1. Selecting pandas data using “iloc”**

The [iloc](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.iloc.html) indexer for Pandas Dataframe is used for [integer-location based indexing / selection](http://pandas.pydata.org/pandas-docs/stable/indexing.html#selection-by-position) by position.

The iloc indexer syntax is data.iloc[<row selection>, <column selection>], which is sure to be a source of confusion for R users. “iloc” in pandas is used to **select rows and columns by number**, in the order that they appear in the data frame. You can imagine that each row has a row number from 0 to the total rows (data.shape[0])  and iloc[] allows selections based on these numbers. The same applies for columns (ranging from 0 to data.shape[1] )

There are two “arguments” to iloc – a row selector, and a column selector.  For example:

|  |
| --- |
|  |
| # Single selections using iloc and DataFrame | |
|  | |

|  |
| --- |
| # Rows: |
|  |

|  |
| --- |
| data.iloc[0] # first row of data frame (Aleshia Tomkiewicz) - Note a Series data type output. |
|  |

|  |
| --- |
| data.iloc[1] # second row of data frame (Evan Zigomalas) |
|  |

|  |
| --- |
| data.iloc[-1] # last row of data frame (Mi Richan) |
|  |

|  |
| --- |
| # Columns: |
|  |

|  |
| --- |
| data.iloc[:,0] # first column of data frame (first\_name) |
|  |

|  |
| --- |
| data.iloc[:,1] # second column of data frame (last\_name) |
|  |

|  |
| --- |
| data.iloc[:,-1] # last column of data frame (id) |

[view raw](https://gist.github.com/shanealynn/1efd0555a0a668b5f0e3f5fa5593c673/raw/c68229b34a17f2179b28a08a51d0b1fce9b80738/Pandas%20Index%20-%20Single%20iloc%20selections.py) [Pandas Index - Single iloc selections.py](https://gist.github.com/shanealynn/1efd0555a0a668b5f0e3f5fa5593c673#file-pandas-index-single-iloc-selections-py) hosted with by [GitHub](https://github.com)

Multiple columns and rows can be selected together using the .iloc indexer.

|  |
| --- |
|  |
| # Multiple row and column selections using iloc and DataFrame | |
|  | |

|  |
| --- |
| data.iloc[0:5] # first five rows of dataframe |
|  |

|  |
| --- |
| data.iloc[:, 0:2] # first two columns of data frame with all rows |
|  |

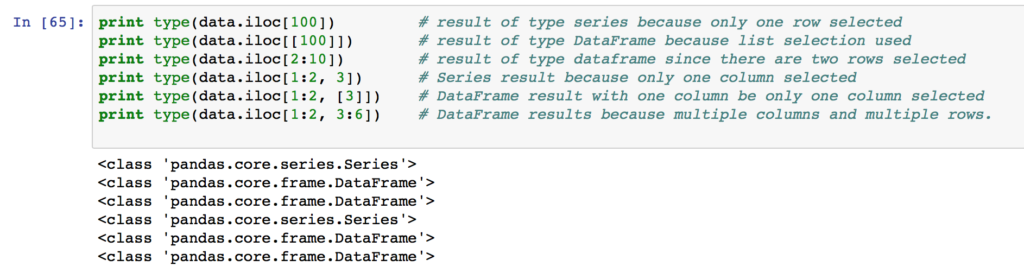
|  |
| --- |
| data.iloc[[0,3,6,24], [0,5,6]] # 1st, 4th, 7th, 25th row + 1st 6th 7th columns. |
|  |

|  |
| --- |
| data.iloc[0:5, 5:8] # first 5 rows and 5th, 6th, 7th columns of data frame (county -> phone1). |

[view raw](https://gist.github.com/shanealynn/3cd4c410ca1514dd9575443ab1b08a06/raw/ef0b34bed66d27ad60cb3960cd688b88069849d9/Pandas%20Index%20-%20Multi%20iloc%20selections.py) [Pandas Index - Multi iloc selections.py](https://gist.github.com/shanealynn/3cd4c410ca1514dd9575443ab1b08a06#file-pandas-index-multi-iloc-selections-py) hosted with by [GitHub](https://github.com)

There’s two gotchas to remember when using iloc in this manner:

1. Note that .iloc returns a Pandas Series when one row is selected, and a Pandas DataFrame when multiple rows are selected, or if any column in full is selected. To counter this, pass a single-valued list if you require DataFrame output.

[](http://104.236.88.249/wp-content/uploads/2016/10/returning_dataframe_or_series_from_pandas_selections.png)When using .loc, or .iloc, you can control the output format by passing lists or single values to the selectors.

1. When selecting multiple columns or multiple rows in this manner, remember that in your selection e.g.[1:5], the rows/columns selected will run from the first number to *one minus* the second number. e.g. [1:5] will go 1,2,3,4., [x,y] goes from x to y-1.

In practice, I rarely use the iloc indexer, unless I want the first ( .iloc[0] ) or the last ( .iloc[-1] )  row of the data frame.

**2. Selecting pandas data using “loc”**

The Pandas [loc](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.loc.html) indexer can be used with DataFrames for two different use cases:

* a.) Selecting [rows by label/index](http://pandas.pydata.org/pandas-docs/stable/indexing.html#selection-by-label)
* b.) Selecting rows with a [boolean / conditional lookup](http://pandas.pydata.org/pandas-docs/stable/indexing.html#boolean-indexing)

The loc indexer is used with the same syntax as iloc: data.loc[<row selection>, <column selection>] .

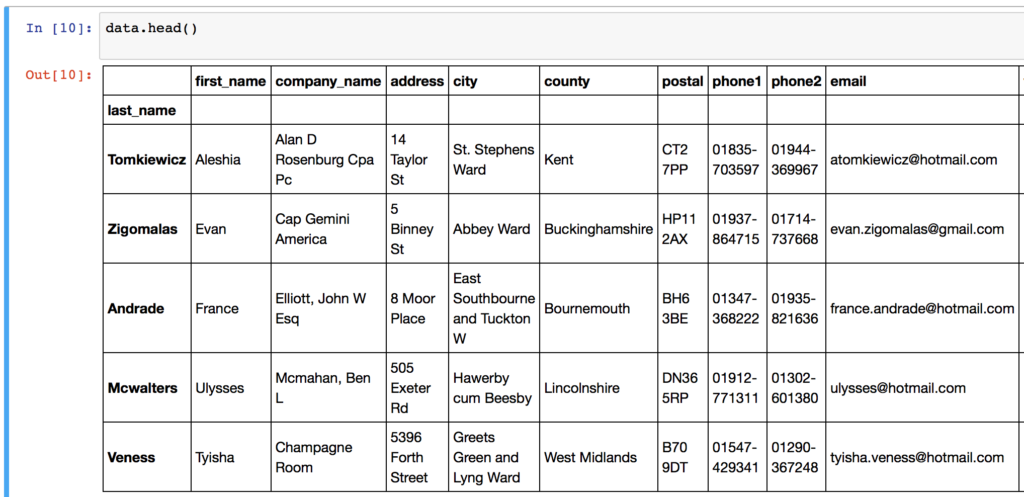
**2a. Label-based / Index-based indexing using .loc**

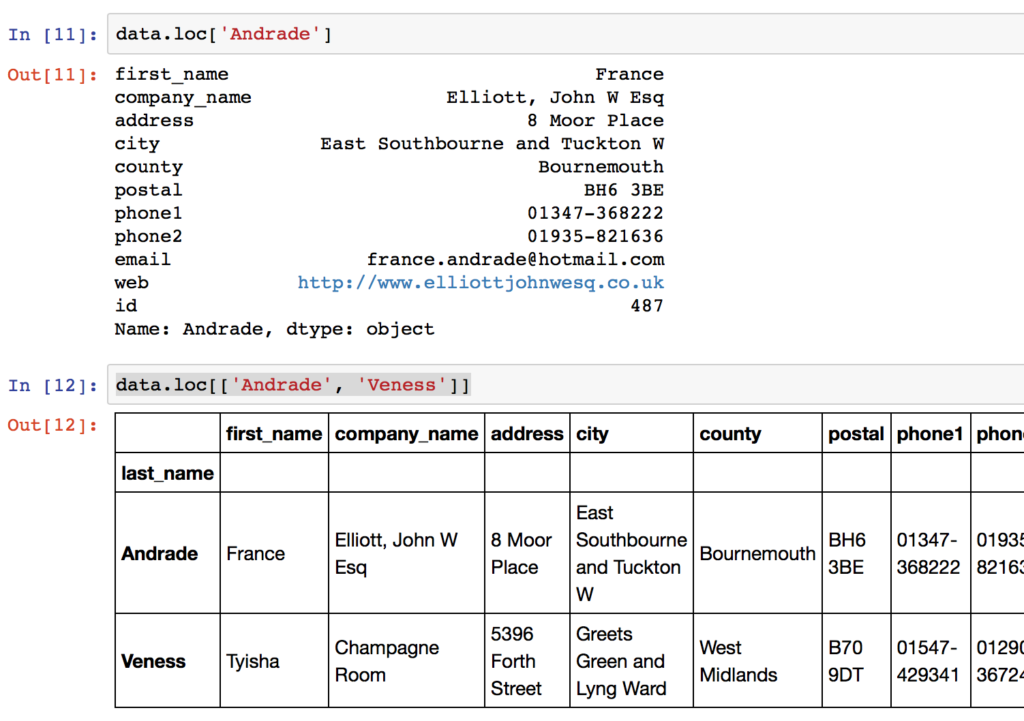
Selections using the loc method are based on the index of the data frame (if any). Where the index is set on a DataFrame, using <code>df.set\_index()</code>, the .loc method directly selects based on index values of any rows. For example, setting the index of our test data frame to the persons “last\_name”:

|  |
| --- |
|  |
| data.set\_index("last\_name", inplace=True) | |
|  | |

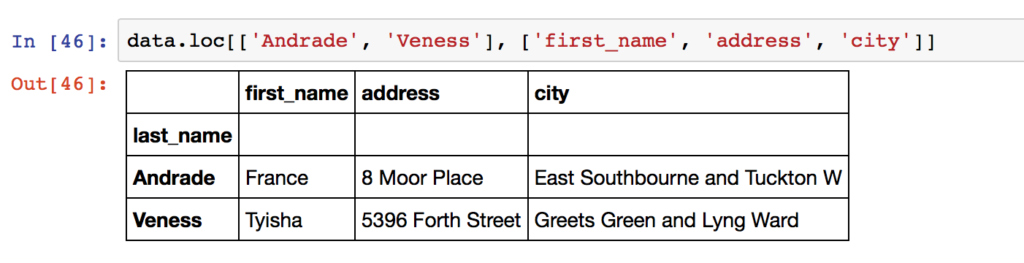
|  |
| --- |
| data.head() |

[view raw](https://gist.github.com/shanealynn/12069e026b30f15d043193bdc5032846/raw/b84289c9399284b130df950aa23713733e027cf2/Pandas%20Index%20-%20Setting%20index%20for%20iloc.py) [Pandas Index - Setting index for iloc.py](https://gist.github.com/shanealynn/12069e026b30f15d043193bdc5032846#file-pandas-index-setting-index-for-iloc-py) hosted with by [GitHub](https://github.com)

[](http://104.236.88.249/wp-content/uploads/2016/09/index_set_dataframe.png)Last Name set as Index set on sample data frameNow with the index set, we can directly select rows for different “last\_name” values using .loc[<label>]  – either singly, or in multiples. For example:

[](http://104.236.88.249/wp-content/uploads/2016/09/pandas-loc-indexing-based-on-index-labels.png)Selecting single or multiple rows using .loc index selections with pandas. Note that the first example returns a series, and the second returns a DataFrame. You can achieve a single-column DataFrame by passing a single-element list to the .loc operation.

Select columns with .loc using the names of the columns. In most of my data work, typically I have named columns, and use these named selections.

[](http://104.236.88.249/wp-content/uploads/2016/10/named_column_selection_loc.png)When using the .loc indexer, columns are referred to by names using lists of strings, or “:” slices.

You can select ranges of index labels – the selection </code>data.loc[‘Bruch’:’Julio’]</code> will return all rows in the data frame between the index entries for “Bruch” and “Julio”. The following examples should now make sense:

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| # Select rows with index values 'Andrade' and 'Veness', with all columns between 'city' and 'email' |
|  |

|  |
| --- |
| data.loc[['Andrade', 'Veness'], 'city':'email'] |
|  |

|  |
| --- |
| # Select same rows, with just 'first\_name', 'address' and 'city' columns |
|  |

|  |
| --- |
| data.loc['Andrade':'Veness', ['first\_name', 'address', 'city']] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Change the index to be based on the 'id' column |
|  |

|  |
| --- |
| data.set\_index('id', inplace=True) |
|  |

|  |
| --- |
| # select the row with 'id' = 487 |
|  |

|  |
| --- |
| data.loc[487] |

[view raw](https://gist.github.com/shanealynn/b27fbdce4688f06108a32c767b65fc3f/raw/c9020c045d7e2ccf5a6201b537423912b52cf350/Pandas%20Index%20-%20Select%20rows%20with%20loc.py) [Pandas Index - Select rows with loc.py](https://gist.github.com/shanealynn/b27fbdce4688f06108a32c767b65fc3f#file-pandas-index-select-rows-with-loc-py) hosted with by [GitHub](https://github.com)

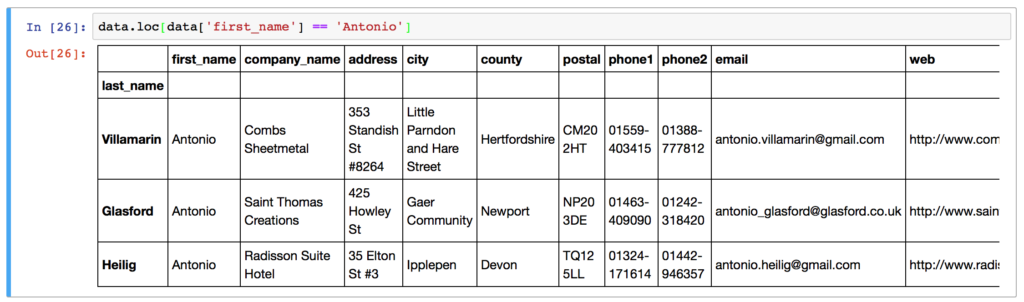
Note that in the last example, data.loc[487]**(the row with index value 487) is not equal to** data.iloc[487]**(the 487th row in the data)**. The index of the DataFrame can be out of numeric order, and/or a string or multi-value.

**2b. Boolean / Logical indexing using .loc**

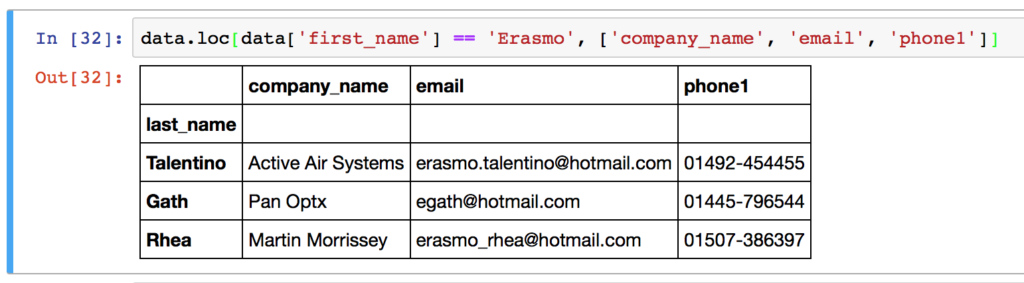
[Conditional selections with boolean arrays](http://pandas.pydata.org/pandas-docs/stable/indexing.html#boolean-indexing) using data.loc[<selection>] is the most common method that I use with Pandas DataFrames. With boolean indexing or logical selection, you pass an array or [Series](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html) of True/False values to the .loc indexer to select the rows where your Series has *True* values.

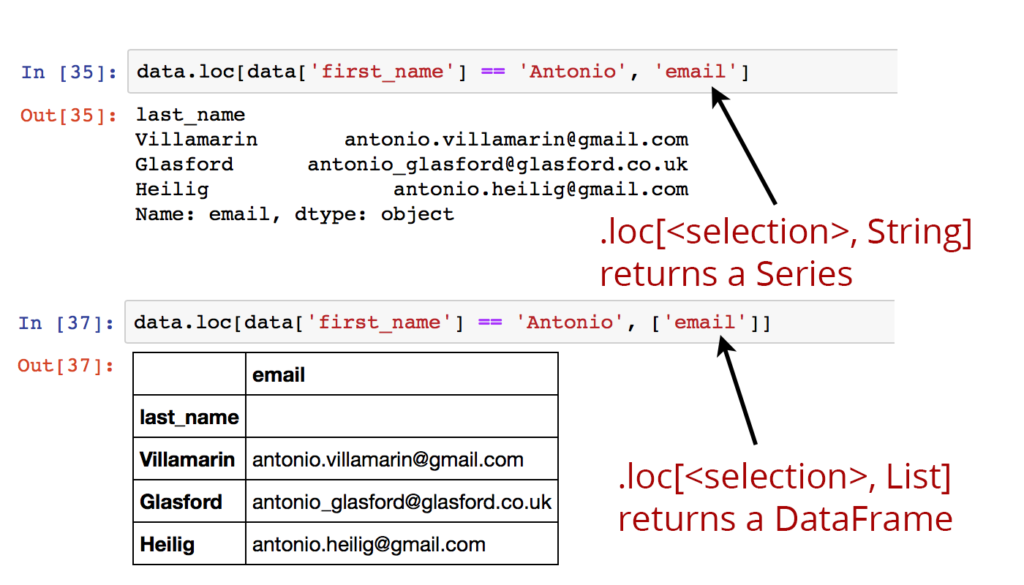
In most use cases, you will make selections based on the values of different columns in your data set.

For example, the statement data[‘first\_name’] == ‘Antonio’] produces a Pandas Series with a True/False value for every row in the ‘data’ DataFrame, where there are “True” values for the rows where the first\_name is “Antonio”. These type of boolean arrays can be passed directly to the .loc indexer as so:

[](http://104.236.88.249/wp-content/uploads/2016/10/Boolean-Selection-with-loc.png)Using a boolean True/False series to select rows in a pandas data frame – all rows with first name of “Antonio” are selected.

As before, a second argument can be passed to .loc to select particular columns out of the data frame. Again, columns are referred to by name for the loc indexer and can be a single string, a list of columns, or a slice “:” operation.

[](http://104.236.88.249/wp-content/uploads/2016/10/multiple_column_select_loc_indexer.png)Selecting multiple columns with loc can be achieved by passing column names to the second argument of .loc[]Note that when selecting columns, if one column only is selected, the .loc operator returns a Series. For a single column DataFrame, use a one-element list to keep the DataFrame format, for example:

[](http://104.236.88.249/wp-content/uploads/2016/10/loc_indexer_returns_series_or_dataframe.png)If selections of a single column are made as a string, a series is returned from .loc. Pass a list to get a DataFrame back.

Make sure you understand the following additional examples of .loc selections for clarity:

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| # Select rows with first name Antonio, # and all columns between 'city' and 'email' |
|  |

|  |
| --- |
| data.loc[data['first\_name'] == 'Antonio', 'city':'email'] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Select rows where the email column ends with 'hotmail.com', include all columns |
|  |

|  |
| --- |
| data.loc[data['email'].str.endswith("hotmail.com")] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Select rows with last\_name equal to some values, all columns |
|  |

|  |
| --- |
| data.loc[data['first\_name'].isin(['France', 'Tyisha', 'Eric'])] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Select rows with first name Antonio AND hotmail email addresses |
|  |

|  |
| --- |
| data.loc[data['email'].str.endswith("gmail.com") & (data['first\_name'] == 'Antonio')] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # select rows with id column between 100 and 200, and just return 'postal' and 'web' columns |
|  |

|  |
| --- |
| data.loc[(data['id'] > 100) & (data['id'] <= 200), ['postal', 'web']] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # A lambda function that yields True/False values can also be used. |
|  |

|  |
| --- |
| # Select rows where the company name has 4 words in it. |
|  |

|  |
| --- |
| data.loc[data['company\_name'].apply(lambda x: len(x.split(' ')) == 4)] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Selections can be achieved outside of the main .loc for clarity: |
|  |

|  |
| --- |
| # Form a separate variable with your selections: |
|  |

|  |
| --- |
| idx = data['company\_name'].apply(lambda x: len(x.split(' ')) == 4) |
|  |

|  |
| --- |
| # Select only the True values in 'idx' and only the 3 columns specified: |
|  |

|  |
| --- |
| data.loc[idx, ['email', 'first\_name', 'company']] |

[view raw](https://gist.github.com/shanealynn/b34acd07fdaba4f220f4e03c6f902a9f/raw/565aa819a584b4dafb0f9d6e9cbf40a228152e05/Pandas%20index%20-%20loc%20selection%20examples.py) [Pandas index - loc selection examples.py](https://gist.github.com/shanealynn/b34acd07fdaba4f220f4e03c6f902a9f#file-pandas-index-loc-selection-examples-py) hosted with by [GitHub](https://github.com)

Logical selections and boolean Series can also be passed to the generic [] indexer of a pandas DataFrame and will give the same results: data.loc[data[‘id’] == 9] == data[data[‘id’] == 9] .

**3. Selecting pandas data using ix**

**Note**: The ix indexer [has been deprecated](http://pandas.pydata.org/pandas-docs/stable/whatsnew.html#whatsnew-0200-api-breaking-deprecate-ix) in recent versions of Pandas, starting with version 0.20.1.

The [ix[] indexer](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.ix.html) is a hybrid of .loc and .iloc. Generally, ix is label based and acts just as the .loc indexer. However, .ix also supports integer type selections (as in .iloc) where passed an integer. This only works where the index of the DataFrame is not integer based. ix will accept any of the inputs of .loc and .iloc.

Slightly more complex, I prefer to explicitly use .iloc and .loc to avoid unexpected results.

As an example:

|  |
| --- |
|  |
|  |
|  |

|  |
| --- |
| # ix indexing works just the same as .loc when passed strings |
|  |

|  |
| --- |
| data.ix[['Andrade']] == data.loc[['Andrade']] |
|  |

|  |
| --- |
| # ix indexing works the same as .iloc when passed integers. |
|  |

|  |
| --- |
| data.ix[[33]] == data.iloc[[33]] |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # ix only works in both modes when the index of the DataFrame is NOT an integer itself. |

[view raw](https://gist.github.com/shanealynn/2c61235a30cbe1dff15d81b90c216cfe/raw/5e31330041dcecfffa4f4552a86aa46b5d427bf0/Pandas%20index%20-%20ix%20selections.py) [Pandas index - ix selections.py](https://gist.github.com/shanealynn/2c61235a30cbe1dff15d81b90c216cfe#file-pandas-index-ix-selections-py) hosted with by [GitHub](https://github.com)

**Setting values in DataFrames using .loc**

With a slight change of syntax, you can actually update your DataFrame in the same statement as you select and filter using .loc indexer. This particular pattern allows you to update values in columns depending on different conditions. The setting operation does not make a copy of the data frame, but edits the original data.

As an example:

|  |
| --- |
|  |
| # Change the first name of all rows with an ID greater than 2000 to "John" | |
|  | |

|  |
| --- |
| data.loc[data['id'] > 2000, "first\_name"] = "John" |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # Change the first name of all rows with an ID greater than 2000 to "John" |
|  |

|  |
| --- |
| data.loc[data['id'] > 2000, "first\_name"] = "John" |

[view raw](https://gist.github.com/shanealynn/1c30da11d88cba2ecdf3936209a981b9/raw/dc82d5f8b119343bfe17849f4ac97aebd82b5eb0/Pandas%20index%20-%20changing%20data%20with%20loc.py) [Pandas index - changing data with loc.py](https://gist.github.com/shanealynn/1c30da11d88cba2ecdf3936209a981b9#file-pandas-index-changing-data-with-loc-py) hosted with by [GitHub](https://github.com)

That’s the basics of indexing and selecting with Pandas. If you’re looking for more, take a look at the [.iat,](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.iat.html) and .[at](http://pandas.pydata.org/pandas-docs/version/0.17.0/generated/pandas.DataFrame.at.html) operations for some more [performance-enhanced value accessors](http://pandas.pydata.org/pandas-docs/stable/indexing.html#fast-scalar-value-getting-and-setting) in the [Pandas Documentation](http://pandas.pydata.org/pandas-docs/stable/) and take a look at [selecting by callable functions](http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-callable) for more iloc and loc fun.