There are **three main options** to achieve the selection and indexing activities although there are multiple ways to select and index rows and columns from Pandas Dataframes.

1.selecting data by row numbers(.iloc)

2.selecting data by label or by a conditional statement (.loc)

3.selecting in a hybrid approach(.ix)

I’m going to write about the iloc indexer here because this indexing is suitable for 100-days-Of-ML challenge Day1 and Day2 problems.

**1.selecting data by row numbers(.iloc)**

- According to its name “iloc”,it is used for interger-location based indexing/selection by position.

-The syntax of iloc indexer is data.iloc[<row selection>, <column selection>].

-You can simply imagine that each row has a row number from 0 to the total rows(data.shape[0]) .

-Based on these numbers, iloc[] allows selection .

-The same applies for columns (from 0 to data.shape[1])… The different indexes of 0 and 1 in data.shape[] means that the first horizontal row can be denoted as “axis 1 ” and the first vertical column can be denoted as axis 0.

**Single selections using iloc and dataframe**

# Single selections using iloc and DataFrame

# Rows:

data.iloc[0] # first row of data frame

data.iloc[1] # second row of data frame

data.iloc[-1] # last row of data frame (

# 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)

**Multiple columns and rows selection**

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