CS167: Machine Learning

Pandas Tutorial

Wednesday, September 4th, 2024

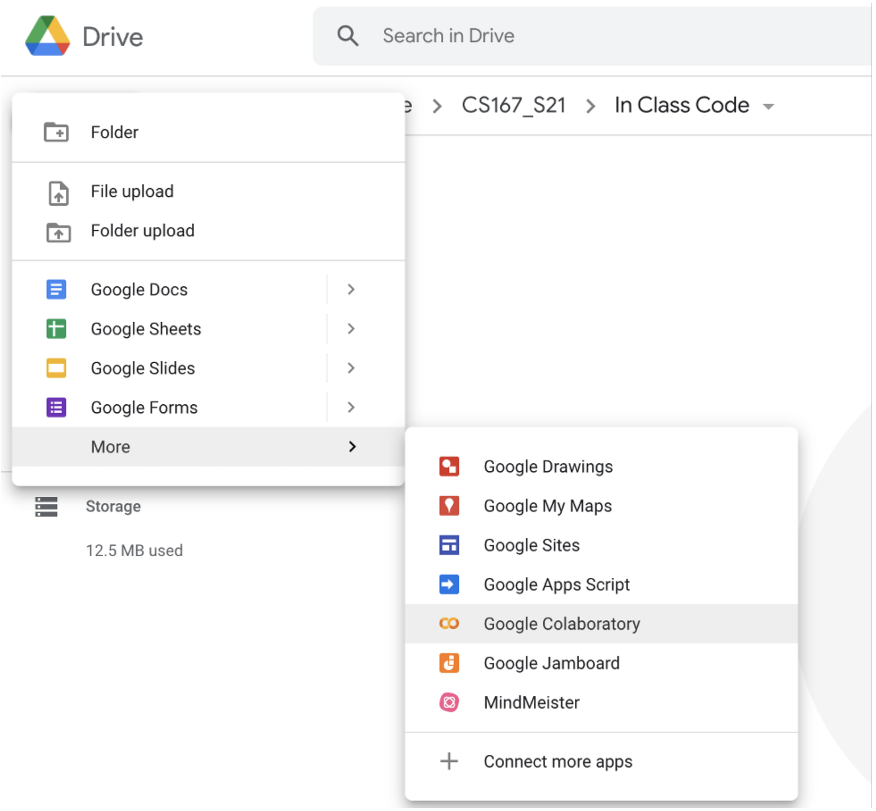
DRAKE UNIVERSITY

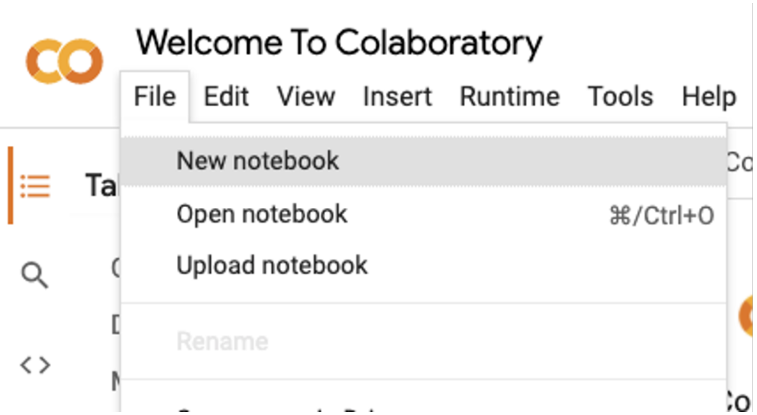
Recap

- Introduction to Google Colab
- Python Lab
- Accessing Data

Recap: Create a new notebook

- There are two ways to do this:
    - From Google Drive: https://drive.google.com/
    - From Colab: https://colab.research.google.com/





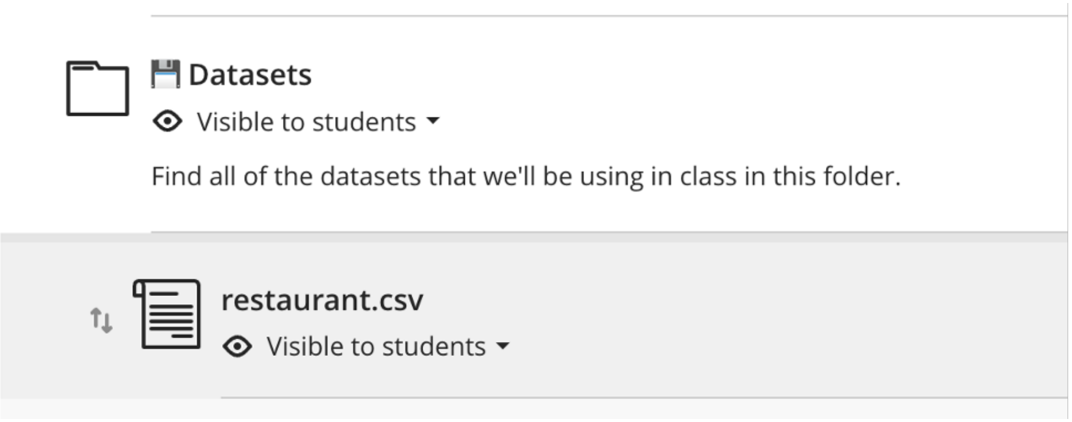
Recap: Python Lab

- Make sure you give your notebook a name (maybe `Day01_notes.ipynb`), and save it to your CS167-Notes Github repository. Your workflow for the rest of class should look something like this:
  - you should have the `Day01_Notes.txt` file open, as well as your Colab Notebook.
  - Copy a section of text from the .txt file and paste it into a new cell in your Colab Notebook.
  - Take a minute and look over the code and predict what will happen. Some cells have specific instructions as to what you should be trying to predict.
  - Run the cell, and see if your prediction was correct.
    - If so, great! Move on.
    - If not, even better--you get to dig into why your expectations were different than how it actually worked, which is a great opportunity to learn something new :)
  - Move on to the next cell and repeat!

Recap: Accessing Data

- Google Colab is a cloud-based tool, which means that we need to store our data in the cloud as well. We cannot simply reference our local data and expect it to work.

- Go ahead and download the `restaurant.csv` file from Blackboard. It is in the Datasets folder.



Datasets

Visible to students

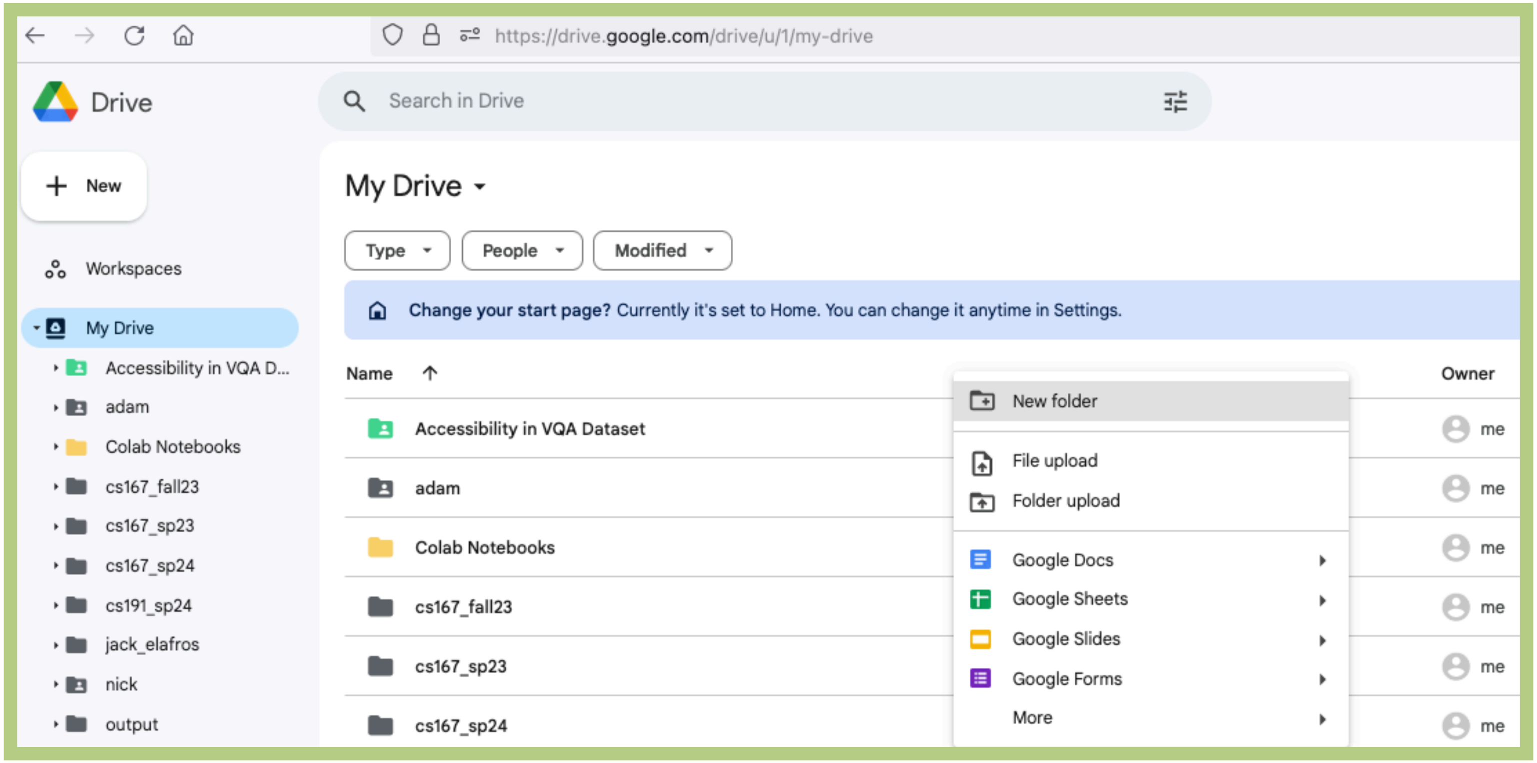
Find all of the datasets that we'll be using in class in this folder.

restaurant.csv

Visible to students

Recap: Uploading File to Google Drive

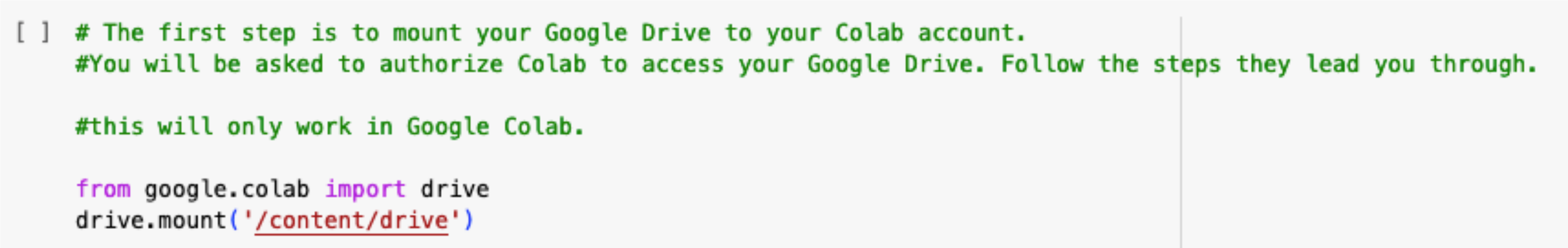
- Upload the `restaurant.csv` to your Google Drive.
  - First go to: drive.google.com
  - Then, create a directory/folder (by right-clicking your mouse) as shown below:



Name	Owner
Accessibility in VQA Dataset	me
adam	me
Colab Notebooks	me
cs167_fall23	me
cs167_sp23	me
cs167_sp24	me

Recap: Accessing Data

- To access this file in Google Colab, you'll need a little bit of code.



```
# The first step is to mount your Google Drive to your Colab account.
#You will be asked to authorize Colab to access your Google Drive. Follow the steps they lead you through.

#this will only work in Google Colab.

from google.colab import drive
drive.mount('/content/drive')
```

- Do a demonstration …

Today's Agenda

- Topics:
  - Introduction to Pandas (a library in Python)
  - Subsetting (Columns, Rows, or both) in a DataFrame
    - Select Columns in a DataFrame
    - Select Rows in a DataFrame
    - Select **subsets** of the DataFrame (both rows and columns)

Accessing Data

- **Pandas** is a super powerful Python data analysis library.
  - it's built on top of another powerful library called `numpy`

- Using Google Colab, `pandas` should already be installed. If you see `In [*]` next to a cell, it means your computer is working on the task

Pandas Datatypes: `DataFrame` and `Series`

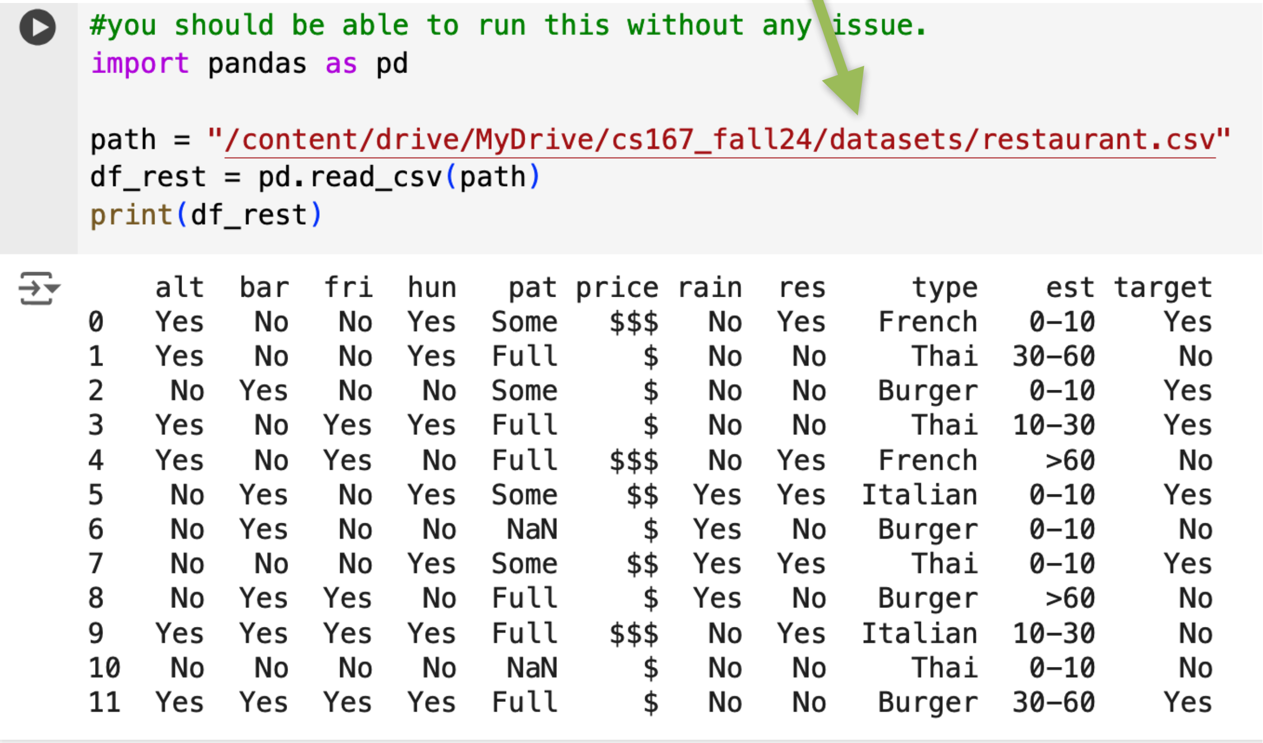
- In `pandas`, there are two main datatypes
  - `DataFrame`
  - `Series`

Pandas Datatypes: `DataFrame`

- Pandas Documentation defines `DataFrames` as:
  - *‘Two-dimensional, size-mutable, potentially heterogeneous tabular data’*
  - basically, think of `DataFrames` as our excel sheets--two dimensional, tabular data
  - Each column has a name, and you can use these names to filter and create subsets of data
  - often, you'll see `DataFrames` abbreviated to `df`

Creating `DataFrame` by reading from a file

- You will be able to show the path of `restaurant.csv` on your Google Drive as follows:



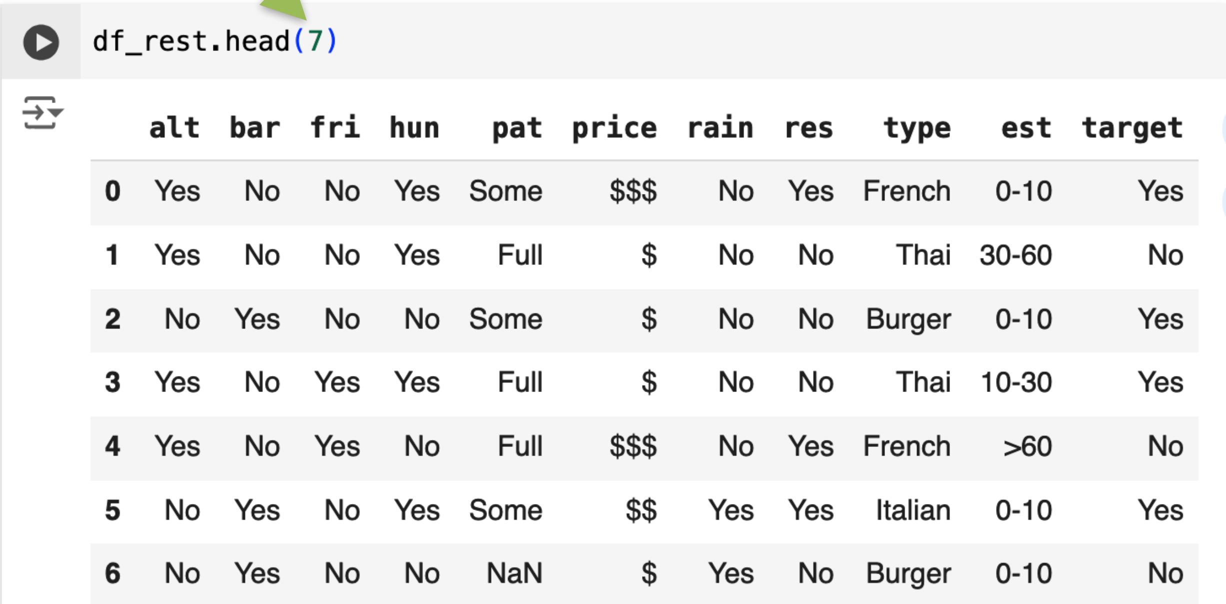
```python
#you should be able to run this without any issue.
import pandas as pd

path = "/content/drive/MyDrive/cs167_fall24/datasets/restaurant.csv"
df_rest = pd.read_csv(path)
print(df_rest)
```

```
    alt  bar  fri  hun   pat price rain  res     type    est target
0   Yes   No   No  Yes  Some   $$$   No  Yes   French   0-10    Yes
1   Yes   No   No  Yes  Full     $   No   No     Thai  30-60     No
2    No  Yes   No   No  Some     $   No   No   Burger   0-10    Yes
3   Yes   No  Yes  Yes  Full     $   No   No     Thai  10-30    Yes
4   Yes   No  Yes   No  Full   $$$   No  Yes   French    >60     No
5    No  Yes   No  Yes  Some    $$  Yes  Yes  Italian   0-10    Yes
6    No  Yes   No   No   NaN     $  Yes   No   Burger   0-10     No
7    No   No   No  Yes  Some    $$  Yes  Yes     Thai   0-10    Yes
8    No  Yes  Yes   No  Full     $  Yes   No   Burger    >60     No
9   Yes  Yes  Yes  Yes  Full   $$$   No  Yes  Italian  10-30     No
10   No   No   No   No   NaN     $   No   No     Thai   0-10     No
11  Yes  Yes  Yes  Yes  Full     $   No   No   Burger  30-60    Yes
```

- We did this last week

Helpful Method Alert: `df.head()`

- The `.head()` method can be called on any `DataFrame`, and by default will display the first 5 lines/rows of the data, as well as the names of the columns.
    - if you want it to display more than 5 rows, you can provide a number as an argument to the method.

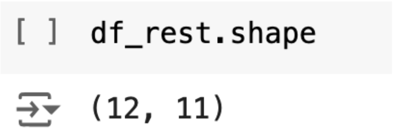


```
df_rest.head(7)
```

	alt	bar	fri	hun	pat	price	rain	res	type	est	target
0	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	Yes
1	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	No
2	No	Yes	No	No	Some	$	No	No	Burger	0-10	Yes
3	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	Yes
4	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
5	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	Yes
6	No	Yes	No	No	NaN	$	Yes	No	Burger	0-10	No

Helpful Method Alert: `df.shape()`

- The `.shape()` method can be called on any `DataFrame`, and it will show the dimensions ie, *number of rows* and *number of columns*



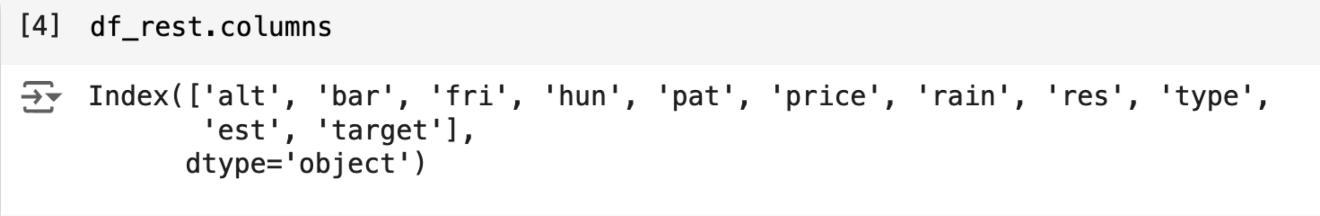
```
[ ]  df_rest.shape
```

```
(12, 11)
```

There are 12 rows
And 11 columns

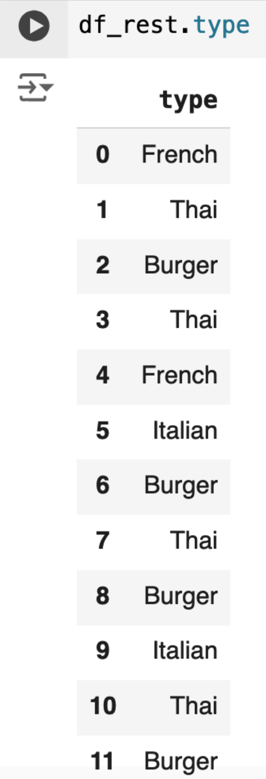
Column Names

- Want to see a list of all of the column names in your dataset? Try using `df.columns`



```
[4]  df_rest.columns

     Index(['alt', 'bar', 'fri', 'hun', 'pat', 'price', 'rain', 'res', 'type',
            'est', 'target'],
           dtype='object')
```

- If there are no spaces in the name of a column, you can also reference it using dot notation like so: `df.type`



```
df_rest.type
```

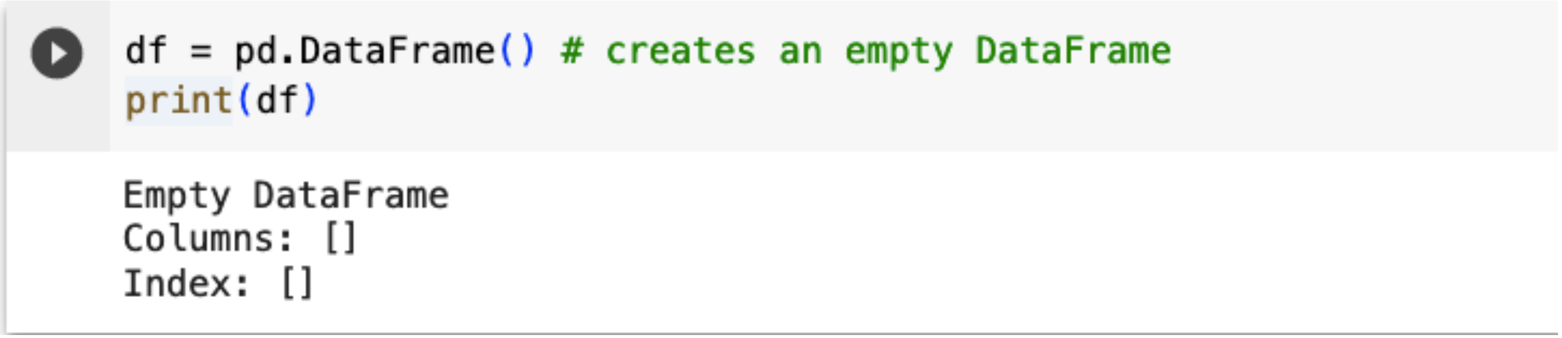
	type
0	French
1	Thai
2	Burger
3	Thai
4	French
5	Italian
6	Burger
7	Thai
8	Burger
9	Italian
10	Thai
11	Burger

Practice Time: Your Turn

- Give these helpful DataFrame methods a try on Google Colab for the next few minutes!

Ways of creating DataFrame

- The syntax for creating a DataFrame from scratch looks like this:
  - `pandas.DataFrame(data, index, columns)`

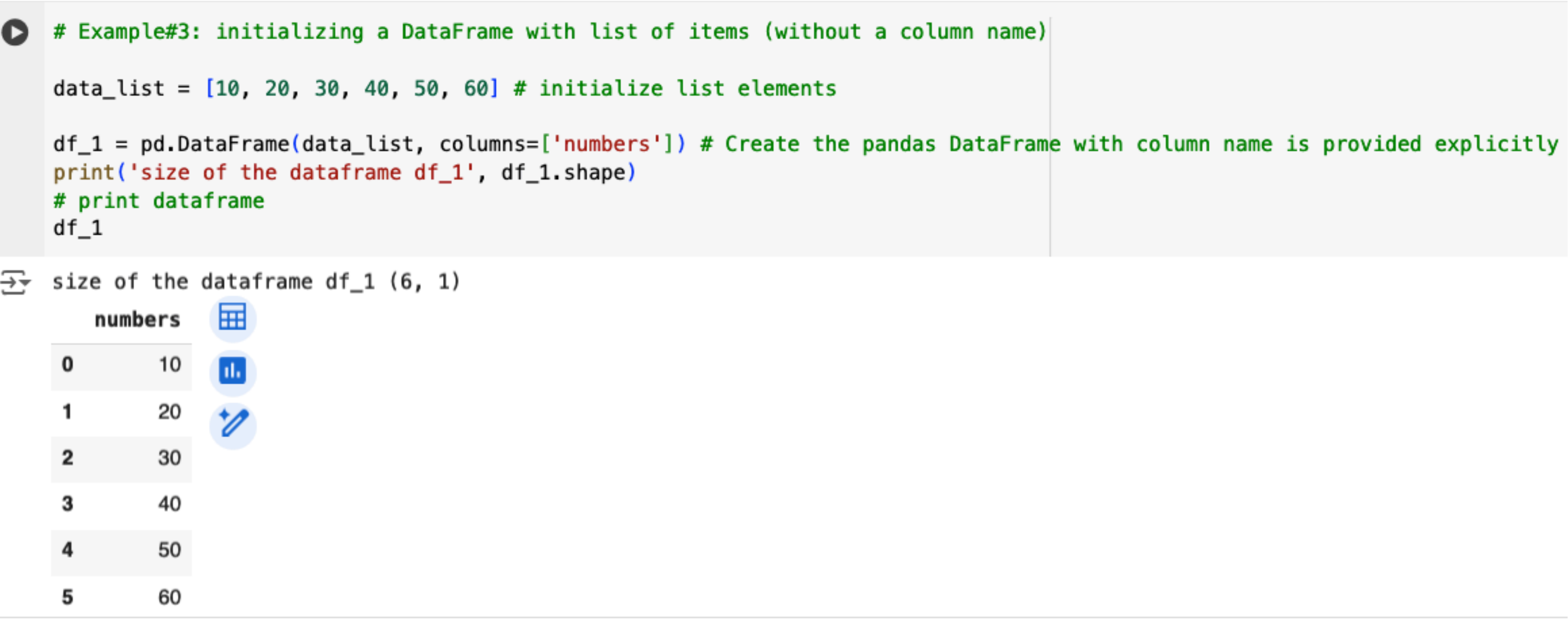


```
df = pd.DataFrame() # creates an empty DataFrame
print(df)
```

```
Empty DataFrame
Columns: []
Index: []
```

Creating DataFrame: 1D List

- The syntax for creating a DataFrame from scratch looks like this:



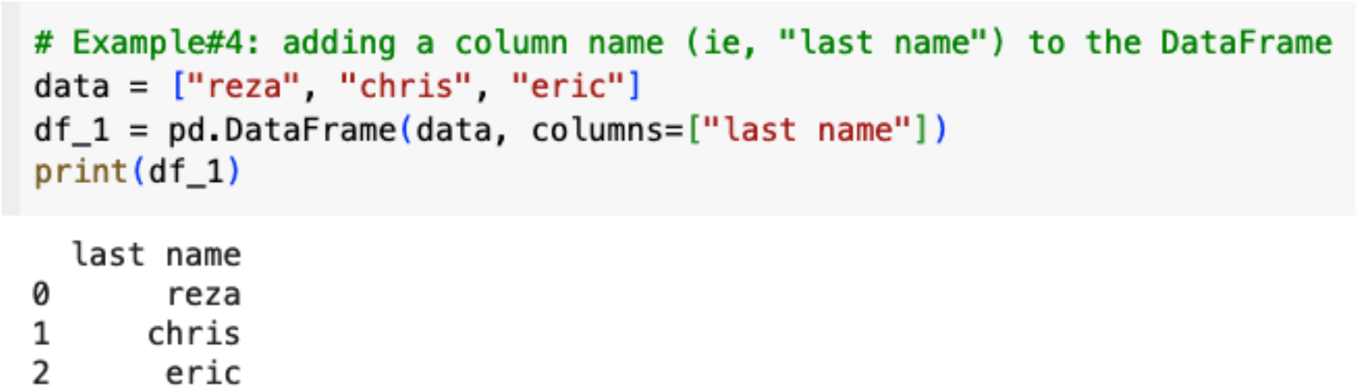
```
# Example#3: initializing a DataFrame with list of items (without a column name)

data_list = [10, 20, 30, 40, 50, 60] # initialize list elements

df_1 = pd.DataFrame(data_list, columns=['numbers']) # Create the pandas DataFrame with column name is provided explicitly
print('size of the dataframe df_1', df_1.shape)
# print dataframe
df_1
```

```
size of the dataframe df_1 (6, 1)
```

	numbers
0	10
1	20
2	30
3	40
4	50
5	60

Creating DataFrame: 1D List

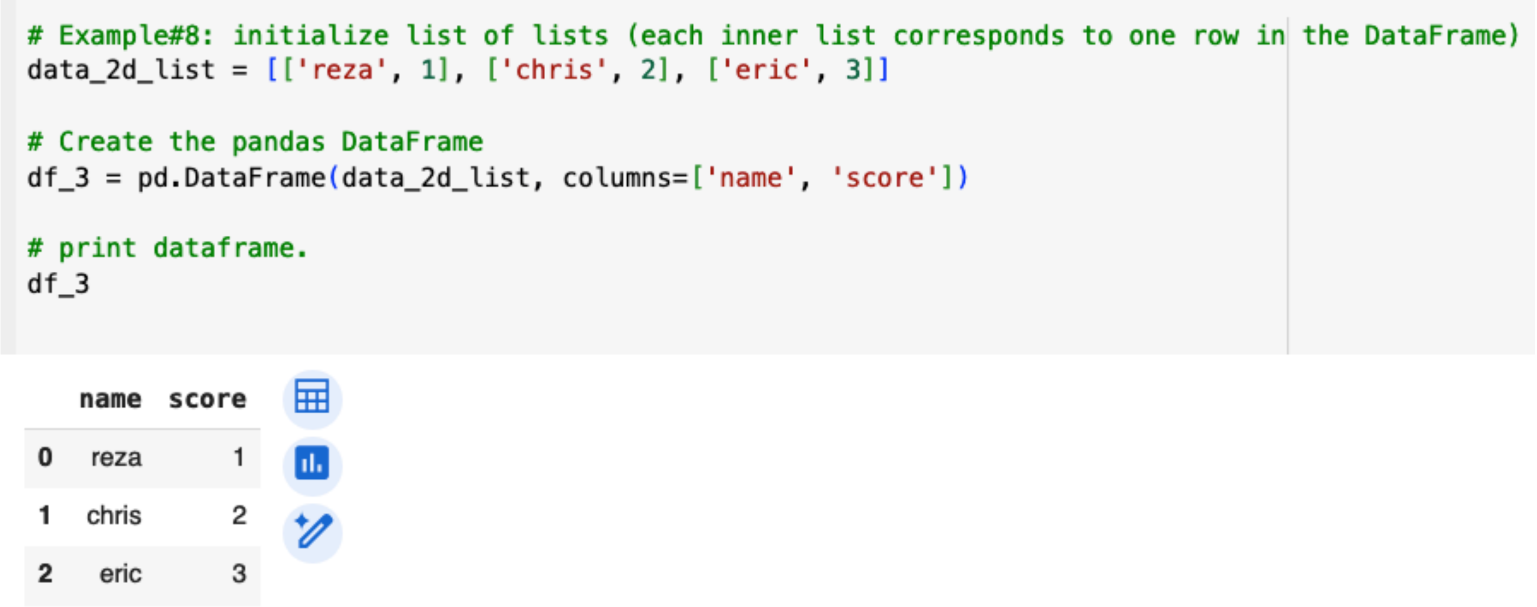
- The syntax for creating a DataFrame from scratch looks like this:



```
# Example#4: adding a column name (ie, "last name") to the DataFrame
data = ["reza", "chris", "eric"]
df_1 = pd.DataFrame(data, columns=["last name"])
print(df_1)
```

```
  last name
0      reza
1     chris
2      eric
```

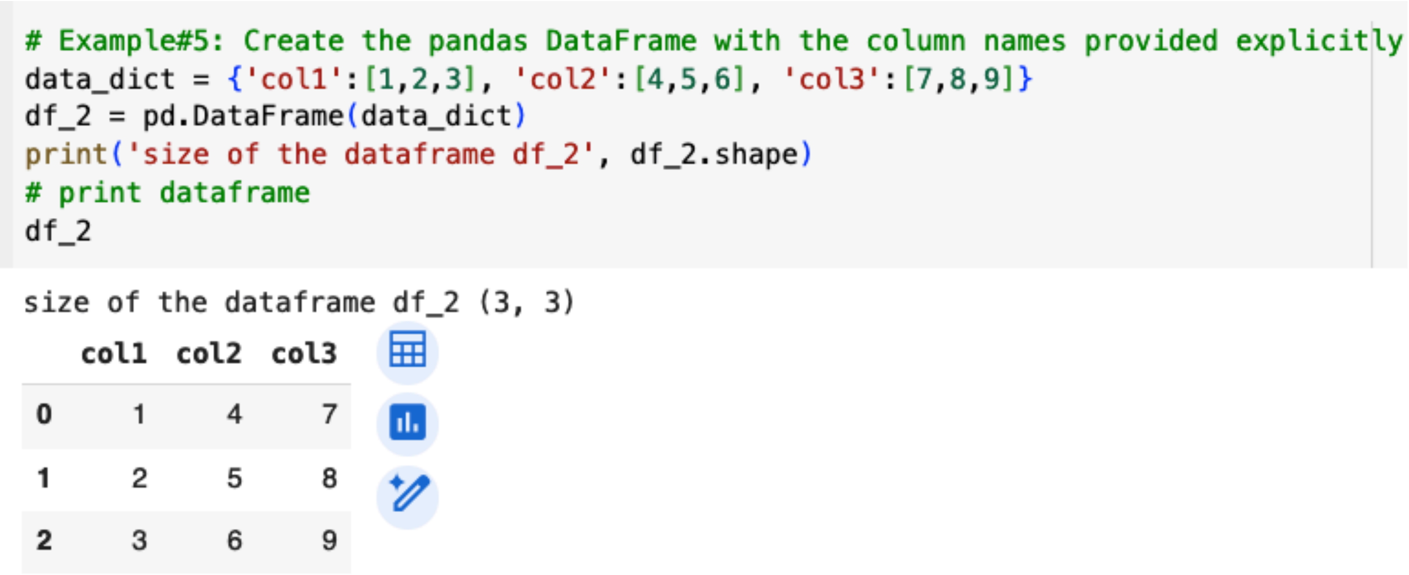
Creating DataFrame: 2D List



```python
# Example#8: initialize list of lists (each inner list corresponds to one row in the DataFrame)
data_2d_list = [['reza', 1], ['chris', 2], ['eric', 3]]

# Create the pandas DataFrame
df_3 = pd.DataFrame(data_2d_list, columns=['name', 'score'])

# print dataframe.
df_3
```

	name	score
0	reza	1
1	chris	2
2	eric	3

Creating DataFrame: dictionary

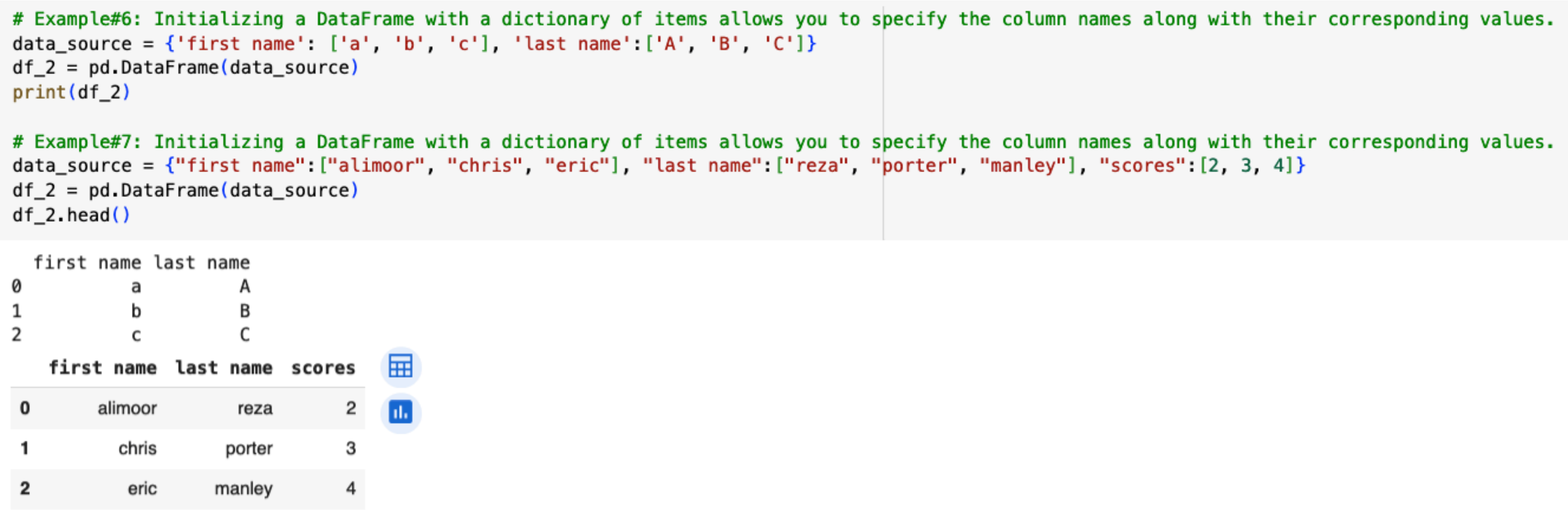


```
# Example#5: Create the pandas DataFrame with the column names provided explicitly
data_dict = {'col1':[1,2,3], 'col2':[4,5,6], 'col3':[7,8,9]}
df_2 = pd.DataFrame(data_dict)
print('size of the dataframe df_2', df_2.shape)
# print dataframe
df_2
```

```
size of the dataframe df_2 (3, 3)
```

	col1	col2	col3
0	1	4	7
1	2	5	8
2	3	6	9

Creating `DataFrame: dictionary`



```python
# Example#6: Initializing a DataFrame with a dictionary of items allows you to specify the column names along with their corresponding values.
data_source = {'first name': ['a', 'b', 'c'], 'last name':['A', 'B', 'C']}
df_2 = pd.DataFrame(data_source)
print(df_2)

# Example#7: Initializing a DataFrame with a dictionary of items allows you to specify the column names along with their corresponding values.
data_source = {"first name":["alimoor", "chris", "eric"], "last name":["reza", "porter", "manley"], "scores":[2, 3, 4]}
df_2 = pd.DataFrame(data_source)
df_2.head()
```

```
  first name last name
0          a         A
1          b         B
2          c         C
```

	first name	last name	scores
0	alimoor	reza	2
1	chris	porter	3
2	eric	manley	4

Practice Time: Your Turn

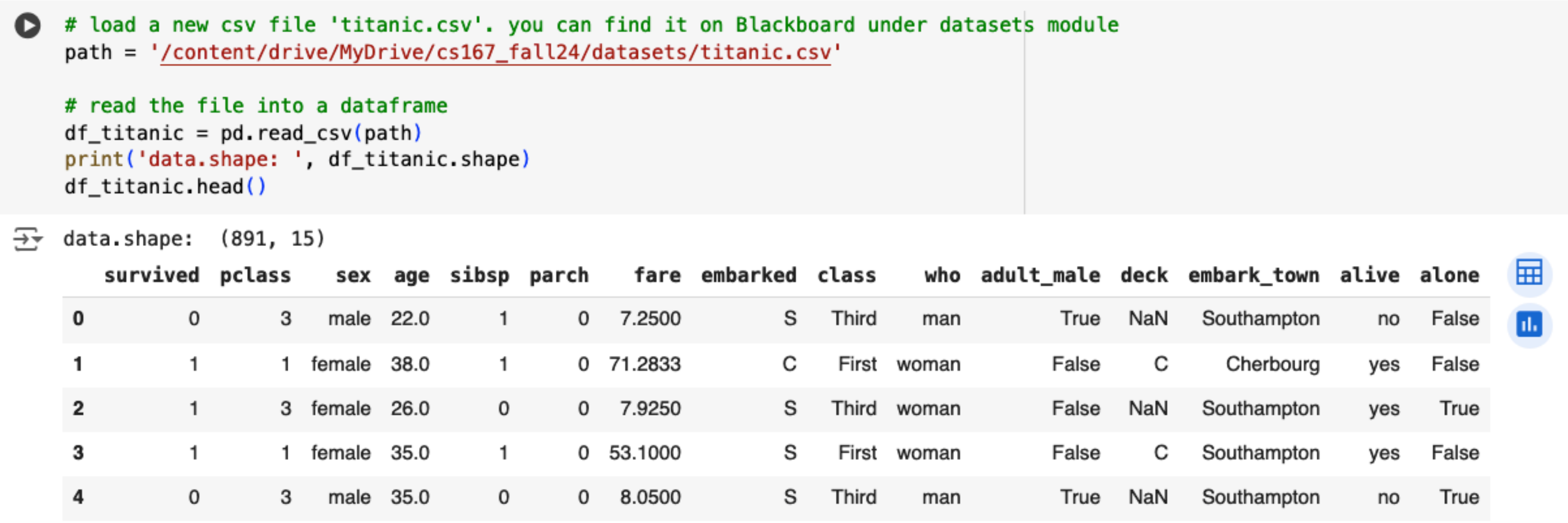
- Give these helpful DataFrame methods a try on Google Colab for the next few minutes!

Selecting Rows in DataFrames using `loc` and `iloc`:

- Simply put:
  - `loc` gets DataFrame rows and columns by **labels/names**
  - `iloc` gets DataFrame rows and columns by **index/ position**

Selecting Rows in DataFrames: `loc`

- Let’s read the dataset and try `loc` to get items by rows **labels/names**



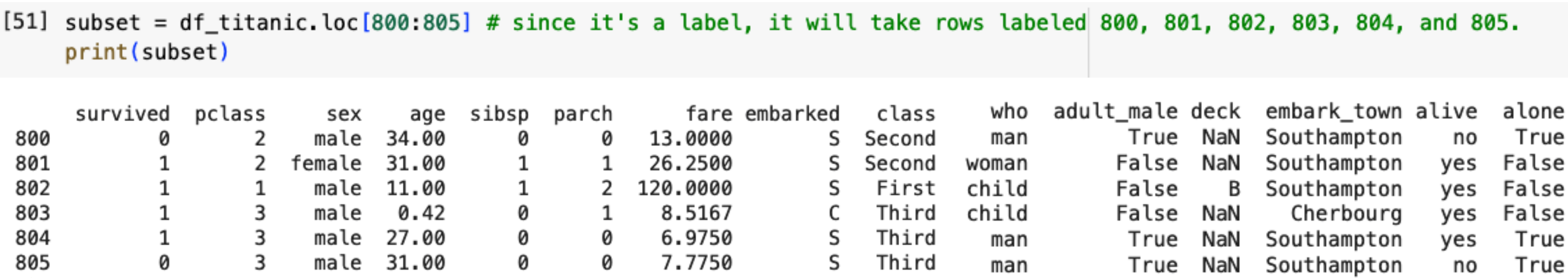
```
# load a new csv file 'titanic.csv'. you can find it on Blackboard under datasets module
path = '/content/drive/MyDrive/cs167_fall24/datasets/titanic.csv'

# read the file into a dataframe
df_titanic = pd.read_csv(path)
print('data.shape: ', df_titanic.shape)
df_titanic.head()
```

```
data.shape:  (891, 15)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**
- Let's take a subset of titanic and try to use `loc`:



```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

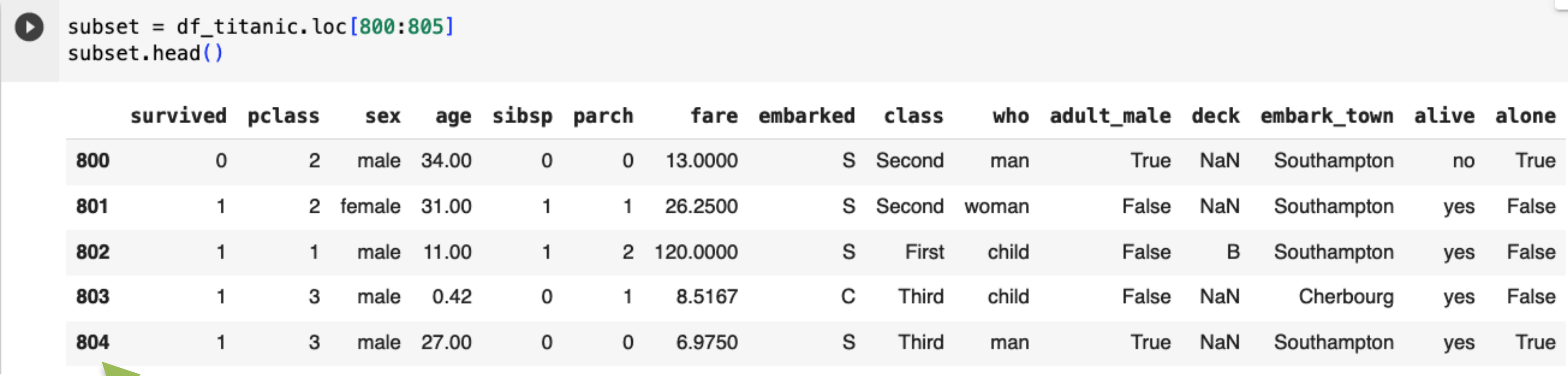
```
     survived  pclass     sex    age  sibsp  parch      fare embarked   class    who adult_male deck embark_town alive  alone
800         0       2    male  34.00      0      0   13.0000        S  Second    man       True  NaN Southampton    no   True
801         1       2  female  31.00      1      1   26.2500        S  Second  woman      False  NaN Southampton   yes  False
802         1       1    male  11.00      1      2  120.0000        S   First  child      False    B Southampton   yes  False
803         1       3    male   0.42      0      1    8.5167        C   Third  child      False  NaN   Cherbourg   yes  False
804         1       3    male  27.00      0      0    6.9750        S   Third    man       True  NaN Southampton   yes   True
805         0       3    male  31.00      0      0    7.7750        S   Third    man       True  NaN Southampton    no   True
```

labels/names (it’s not a number)

ALERT: `print(subset)` shows all 6 rows

Selecting Rows in DataFrames: `loc`

- `loc` gets DataFrame rows and columns by **labels/names**
- Let's take a subset of titanic and try to use `loc` and `iloc`:



```
subset = df_titanic.loc[800:805]
subset.head()
```

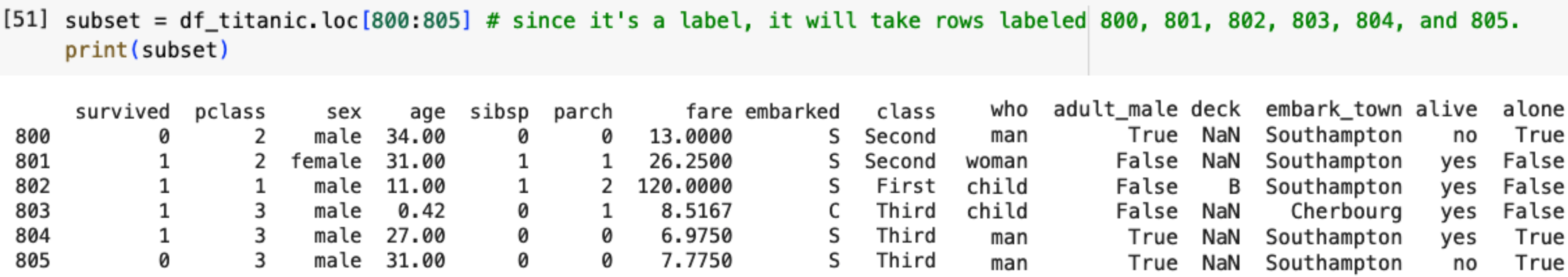
	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
800	0	2	male	34.00	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
801	1	2	female	31.00	1	1	26.2500	S	Second	woman	False	NaN	Southampton	yes	False
802	1	1	male	11.00	1	2	120.0000	S	First	child	False	B	Southampton	yes	False
803	1	3	male	0.42	0	1	8.5167	C	Third	child	False	NaN	Cherbourg	yes	False
804	1	3	male	27.00	0	0	6.9750	S	Third	man	True	NaN	Southampton	yes	True

labels/names (it's not a number)

ALERT: `subset.head()` only shows the first 5 rows.

Selecting Rows in DataFrames: `loc`

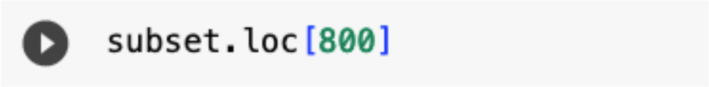
- `loc` gets DataFrame rows and columns by **labels/names**



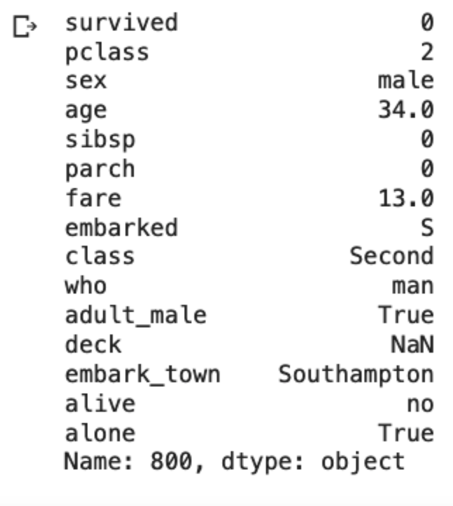
```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

```
     survived  pclass     sex    age  sibsp  parch      fare embarked   class    who  adult_male deck  embark_town alive  alone
800         0       2    male  34.00      0      0   13.0000        S  Second    man        True  NaN  Southampton    no   True
801         1       2  female  31.00      1      1   26.2500        S  Second  woman       False  NaN  Southampton   yes  False
802         1       1    male  11.00      1      2  120.0000        S   First  child       False    B  Southampton   yes  False
803         1       3    male   0.42      0      1    8.5167        C   Third  child       False  NaN    Cherbourg   yes  False
804         1       3    male  27.00      0      0    6.9750        S   Third    man        True  NaN  Southampton   yes   True
805         0       3    male  31.00      0      0    7.7750        S   Third    man        True  NaN  Southampton    no   True
```

- What would happen if I do the following?



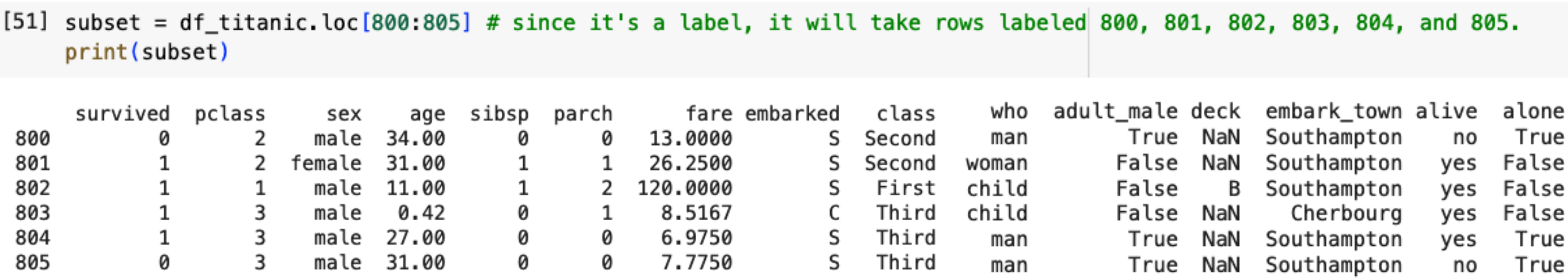
```
subset.loc[800]
```



```
survived                 0
pclass                   2
sex                   male
age                   34.0
sibsp                    0
parch                    0
fare                  13.0
embarked                 S
class               Second
who                    man
adult_male            True
deck                   NaN
embark_town    Southampton
alive                   no
alone                 True
Name: 800, dtype: object
```

Selecting Rows in DataFrames: `loc`

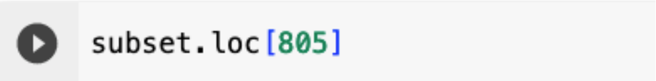
- `loc` gets DataFrame rows and columns by **labels/names**



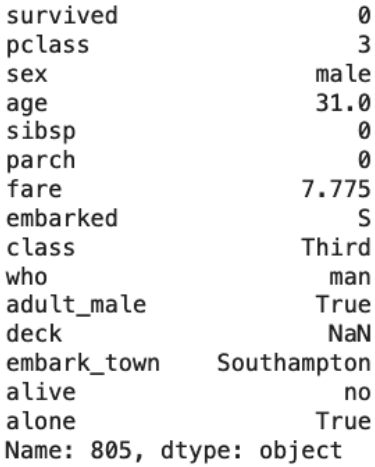
```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

```
     survived  pclass     sex    age  sibsp  parch      fare embarked   class    who  adult_male deck  embark_town alive  alone
800         0       2    male  34.00      0      0   13.0000        S  Second    man        True  NaN  Southampton    no   True
801         1       2  female  31.00      1      1   26.2500        S  Second  woman       False  NaN  Southampton   yes  False
802         1       1    male  11.00      1      2  120.0000        S   First  child       False    B  Southampton   yes  False
803         1       3    male   0.42      0      1    8.5167        C   Third  child       False  NaN    Cherbourg   yes  False
804         1       3    male  27.00      0      0    6.9750        S   Third    man        True  NaN  Southampton   yes   True
805         0       3    male  31.00      0      0    7.7750        S   Third    man        True  NaN  Southampton    no   True
```

- What would happen if I do the following?



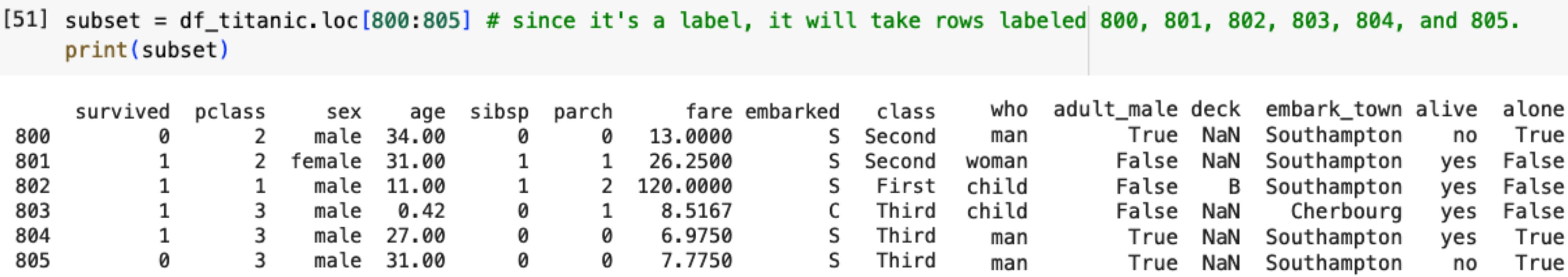
```
subset.loc[805]
```



```
survived                 0
pclass                   3
sex                   male
age                   31.0
sibsp                    0
parch                    0
fare                 7.775
embarked                 S
class                Third
who                    man
adult_male            True
deck                   NaN
embark_town    Southampton
alive                   no
alone                 True
Name: 805, dtype: object
```

Selecting Rows in DataFrames: `loc`

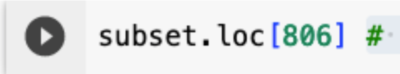
- `loc` gets DataFrame rows and columns by **labels/names**



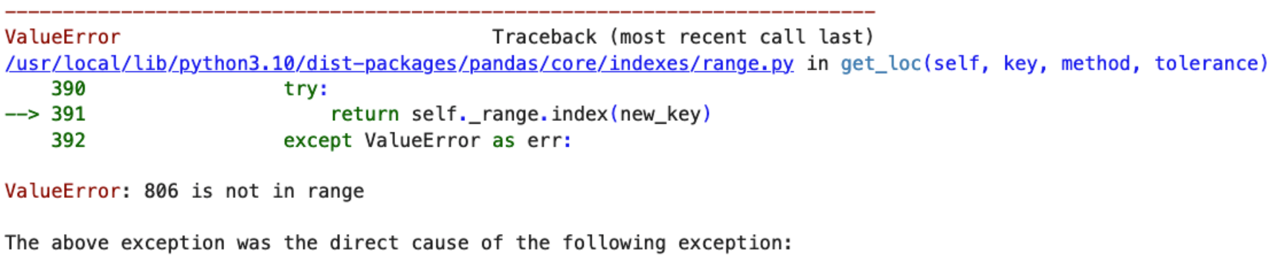
```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

```
     survived  pclass     sex    age  sibsp  parch      fare embarked   class    who  adult_male deck  embark_town alive  alone
800         0       2    male  34.00      0      0   13.0000        S  Second    man        True  NaN  Southampton    no   True
801         1       2  female  31.00      1      1   26.2500        S  Second  woman       False  NaN  Southampton   yes  False
802         1       1    male  11.00      1      2  120.0000        S   First  child       False    B  Southampton   yes  False
803         1       3    male   0.42      0      1    8.5167        C   Third  child       False  NaN    Cherbourg   yes  False
804         1       3    male  27.00      0      0    6.9750        S   Third    man        True  NaN  Southampton   yes   True
805         0       3    male  31.00      0      0    7.7750        S   Third    man        True  NaN  Southampton    no   True
```

- What would happen if I do the following?



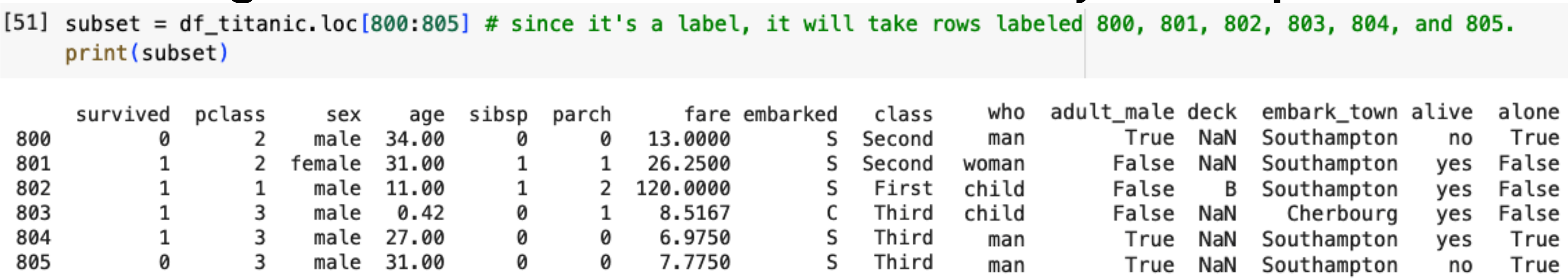
```
subset.loc[806] #
```



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/range.py in get_loc(self, key, method, tolerance)
    390                 try:
--> 391                     return self._range.index(new_key)
    392                 except ValueError as err:

ValueError: 806 is not in range

The above exception was the direct cause of the following exception:
```

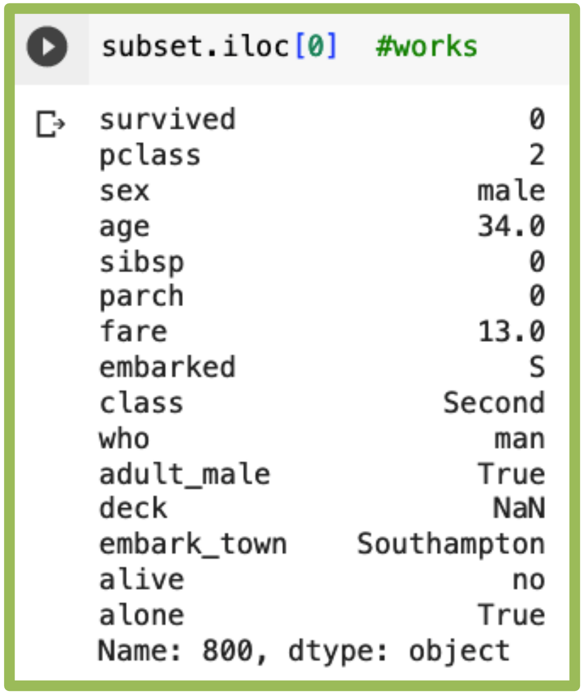
Selecting Rows in DataFrames: iloc

- `iloc` gets DataFrame rows and columns by **index/position**



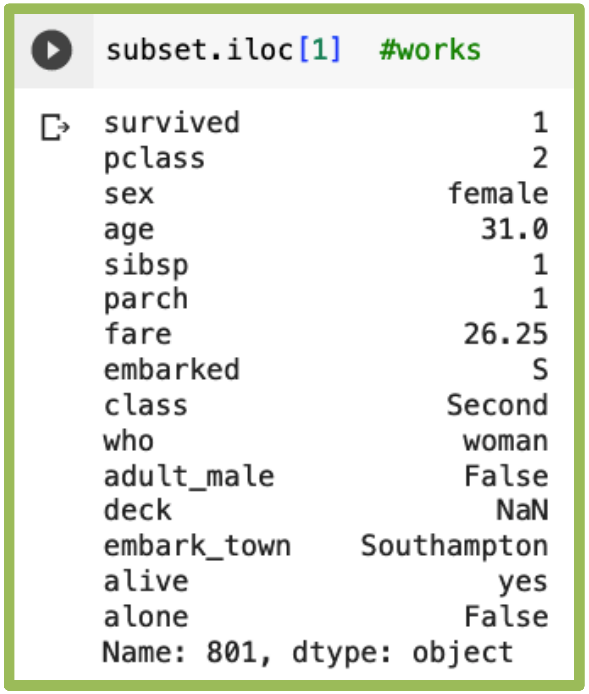
```
[51] subset = df_titanic.loc[800:805] # since it's a label, it will take rows labeled 800, 801, 802, 803, 804, and 805.
     print(subset)
```

```
     survived  pclass     sex    age  sibsp  parch      fare embarked   class    who  adult_male deck  embark_town alive  alone
800         0       2    male  34.00      0      0   13.0000        S  Second    man        True  NaN  Southampton    no   True
801         1       2  female  31.00      1      1   26.2500        S  Second  woman       False  NaN  Southampton   yes  False
802         1       1    male  11.00      1      2  120.0000        S   First  child       False    B  Southampton   yes  False
803         1       3    male   0.42      0      1    8.5167        C   Third  child       False  NaN    Cherbourg   yes  False
804         1       3    male  27.00      0      0    6.9750        S   Third    man        True  NaN  Southampton   yes   True
805         0       3    male  31.00      0      0    7.7750        S   Third    man        True  NaN  Southampton    no   True
```



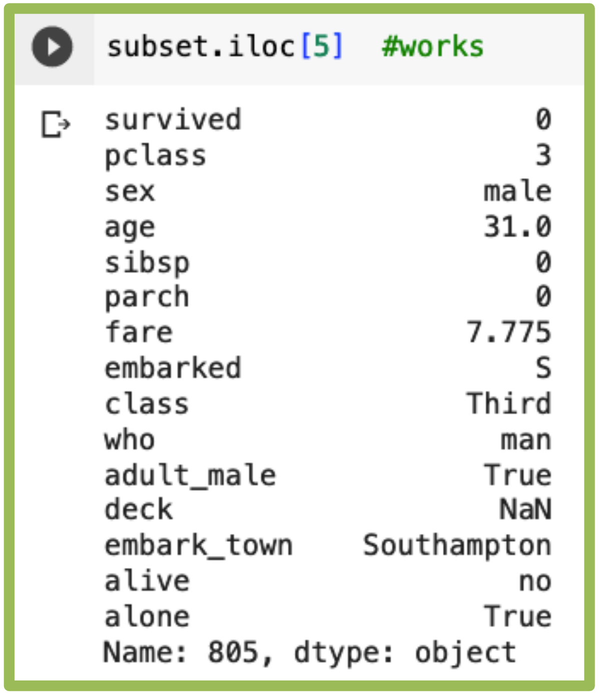
```
subset.iloc[0]   #works
```

```
survived               0
pclass                 2
sex                 male
age                 34.0
sibsp                  0
parch                  0
fare                13.0
embarked               S
class             Second
who                  man
adult_male          True
deck                 NaN
embark_town  Southampton
alive                 no
alone               True
Name: 800, dtype: object
```



```
subset.iloc[1]   #works
```

```
survived               1
pclass                 2
sex               female
age                 31.0
sibsp                  1
parch                  1
fare               26.25
embarked               S
class             Second
who                woman
adult_male         False
deck                 NaN
embark_town  Southampton
alive                yes
alone              False
Name: 801, dtype: object
```



```
subset.iloc[5]   #works
```

```
survived               0
pclass                 3
sex                 male
age                 31.0
sibsp                  0
parch                  0
fare               7.775
embarked               S
class              Third
who                  man
adult_male          True
deck                 NaN
embark_town  Southampton
alive                 no
alone               True
Name: 805, dtype: object
```

Practice Time: Your Turn

- Try `loc/iloc` on your Google Colab notebook using different name or indices!