Saving Puppies with Pandas

DataScienceGo 2019

Alison Peebles Madigan, alison.peeblesmadigan@flatironschool.com

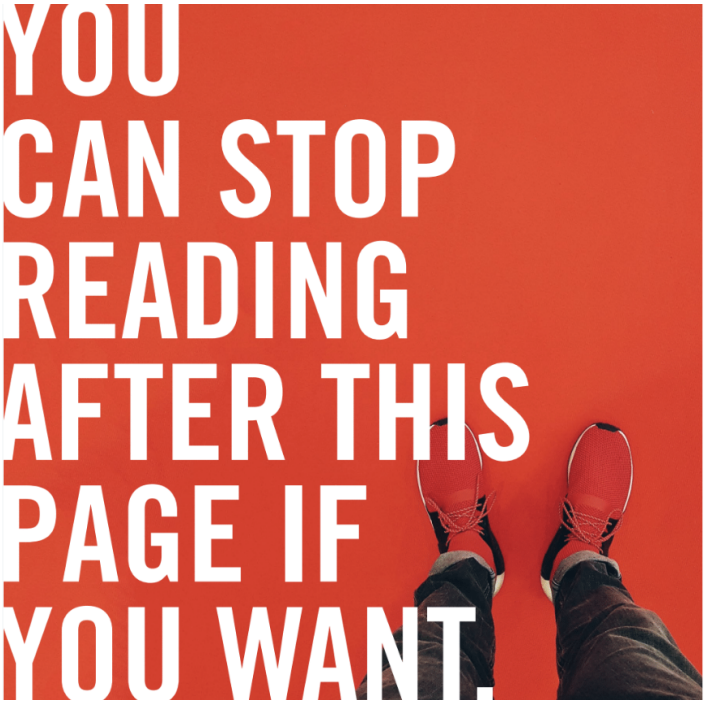
Why save puppies with pandas?

Meet Leo

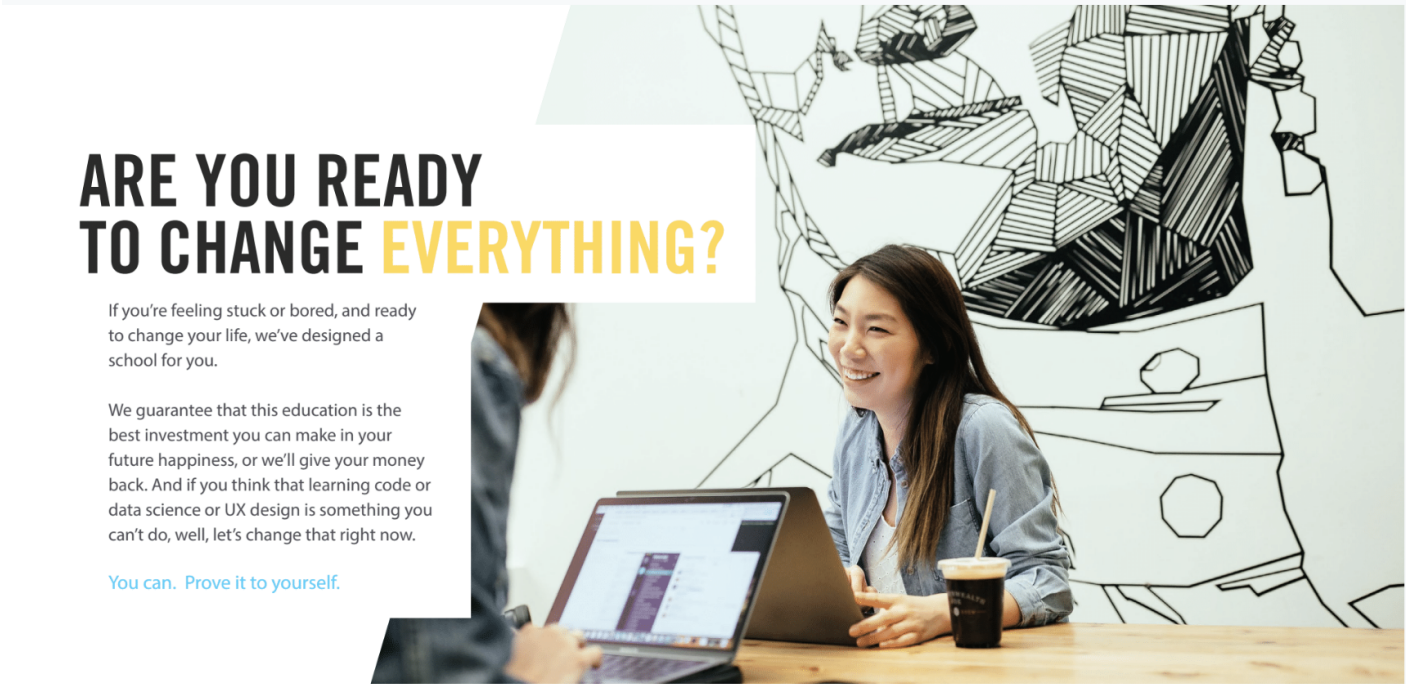
Huge fan of Flatiron School


WELCOME TO FLATIRON SCHOOL
THE SCHOOL OF THE FUTURE IS HERE.
We've redesigned school from scratch with the goal of getting you a new career in the tech industry and giving you the skills you need to change your future, along with a community of other curious, interesting humans.
ALAN TURING
We teach change.


BE HAPPY :)
CHANGE
THINGS
WE TEACH PEOPLE HOW TO CHANGE THINGS: THEIR CAREERS,THEIR LIVES, AND THEIR WORLD.
STARTING WITH A NEW JOB, A PROMOTION OR A BETTER SALARY IN THE RAPIDLY GROWING TECH FIELD:
Code / Data Science / UX / Design


YOU
CAN STOP
READING
AFTER THIS
PAGE IF
YOU WANT.


OUTCOMES-DRIVEN
BECAUSE
OUR MONEY'S
WHERE OUR
MOUTH IS.*
We can't call out higher education and then not back up all that talk with results. The numbers speak for themselves.
Job-seeking Full Stack Web Development Immersive (now called the Immersive Software Engineering Bootcamp) students in the most recent NYC Outcomes Report who took full-time salaried roles, paid apprenticeships, and part-time roles during reporting period.
99%
JOB PLACEMENT RATE
*And if you don't find a job in the tech or data field in six months after you graduate, we'll refund your tuition.



Learning goals: use pandas to provide real analysis of shelter trends and needs.

Questions:

- Age of animals in shelter
- Average animal length of stay
- Medical staff needed

Practice grouping, organizing, merging, and summarizing data using the pandas' library.

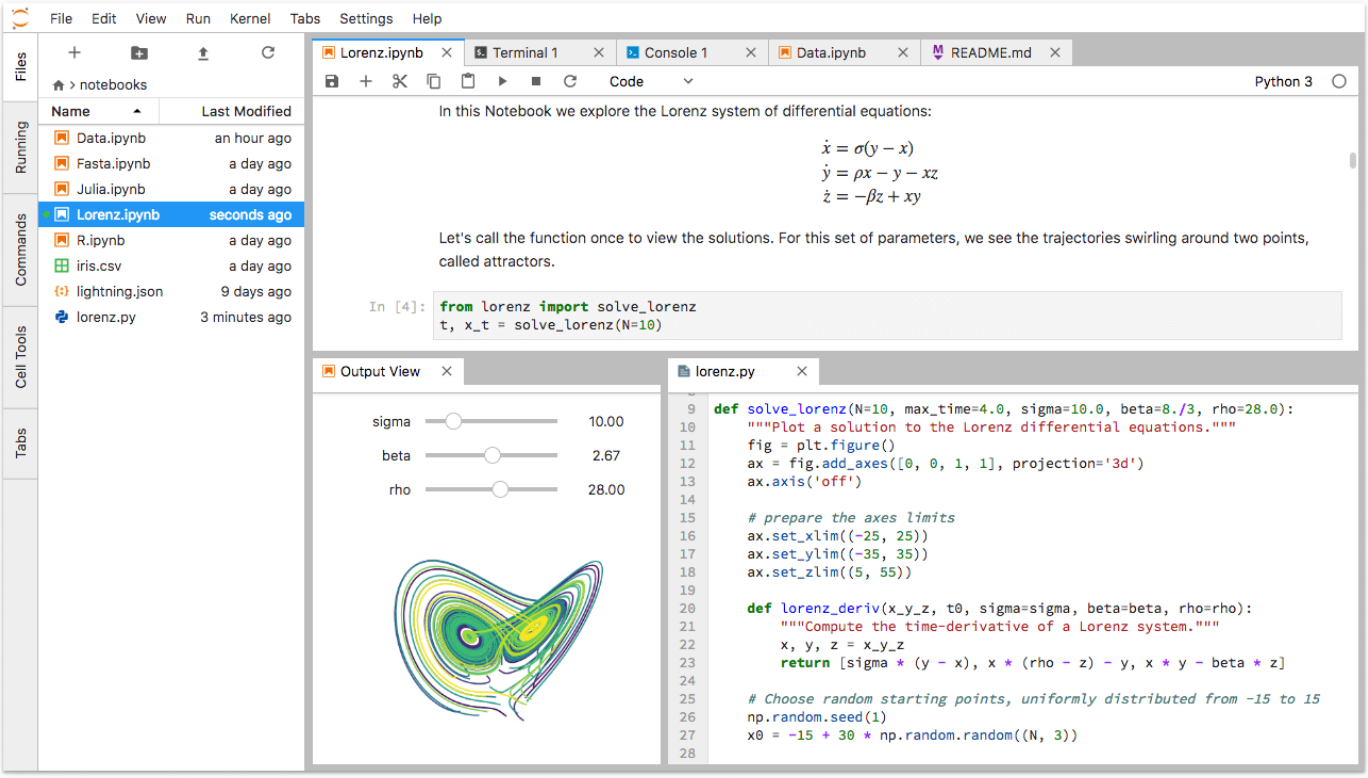
Agenda:

- Familiarize us with Jupyter Lab and our coding environment
- Contextualize `pandas` within the Python ecosystem
- Get and inspect our data
- Clean our data
- Merging and joining with pandas
- Creating new variables with pandas
- Answer our questions!

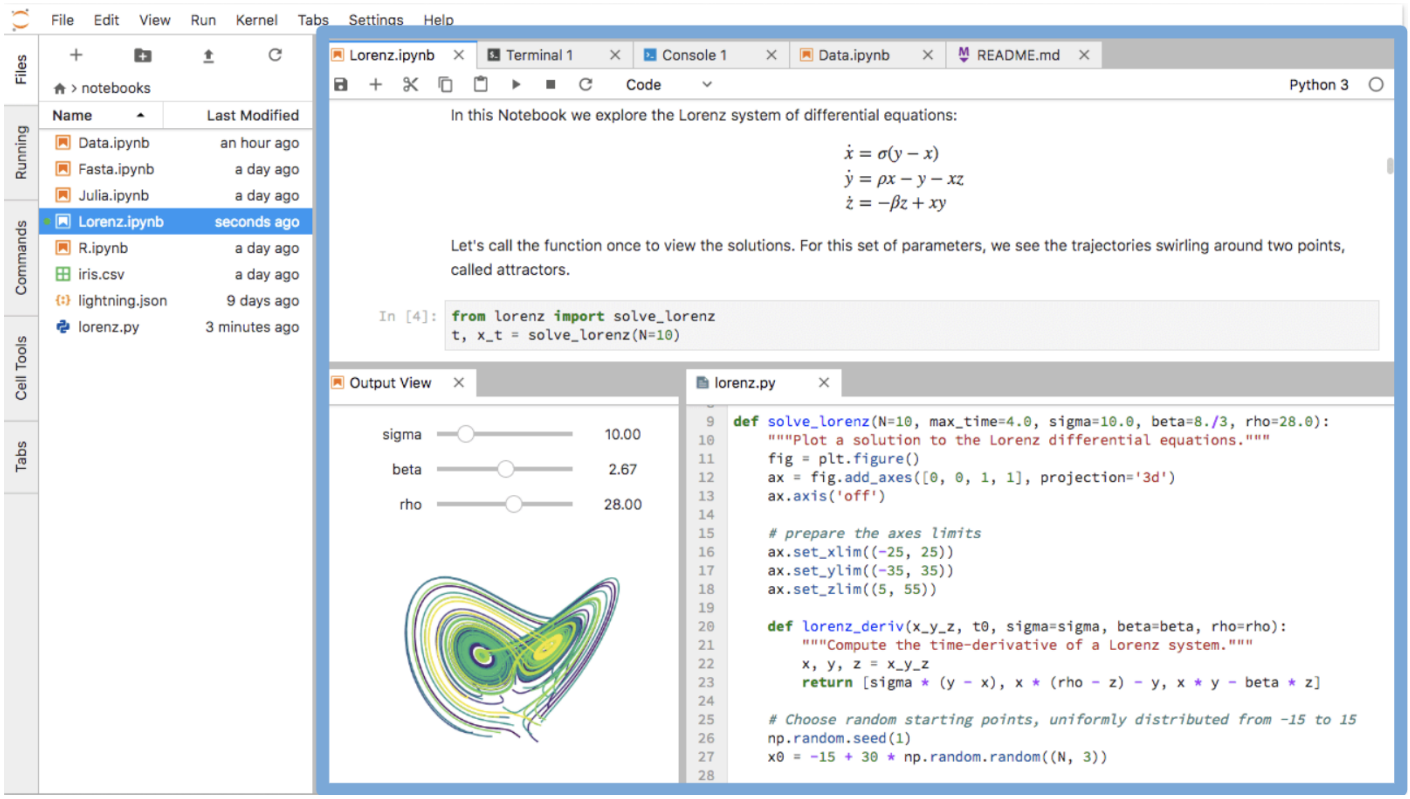
Project Jupyter (https://jupyter.org/)

Quick Jupyter Lab Tour

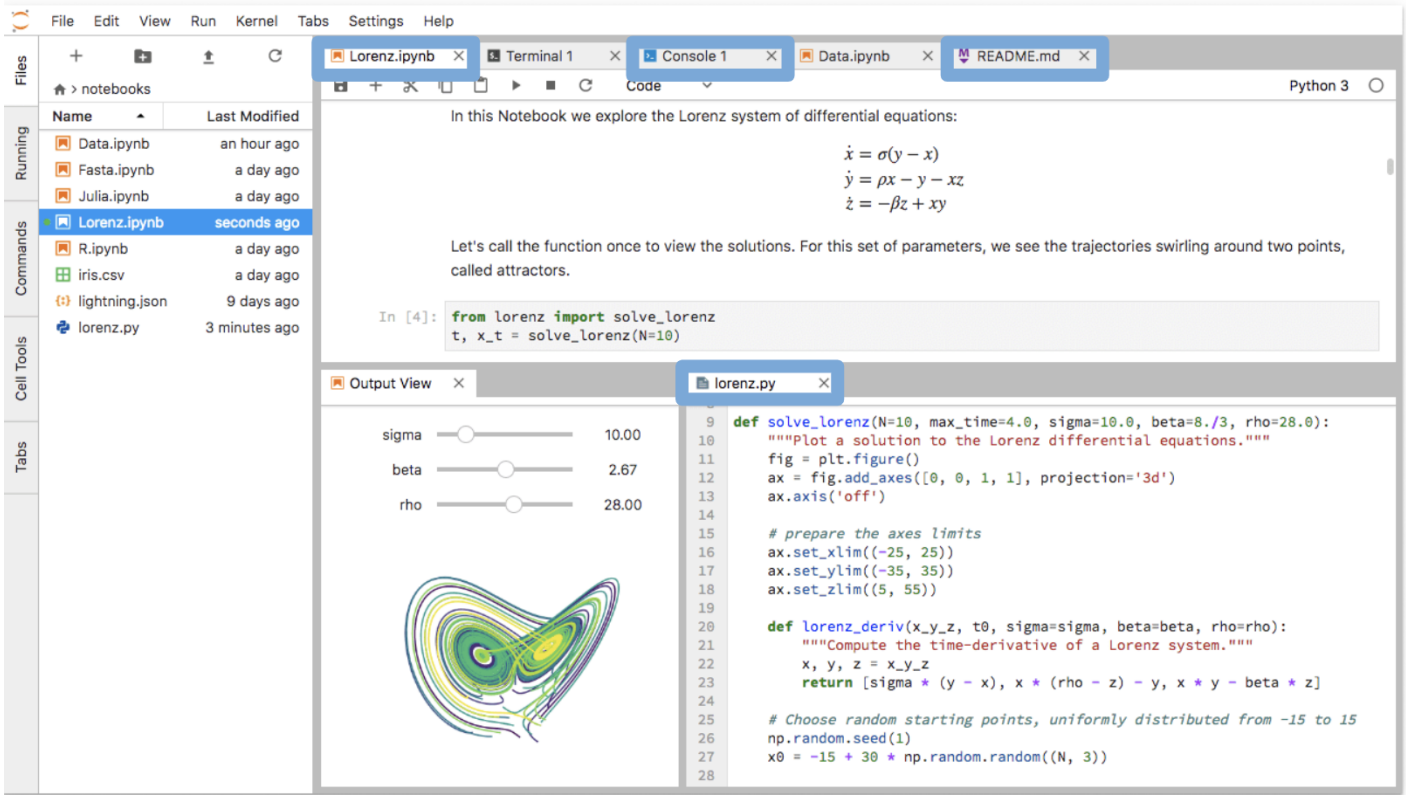
Jupyter Lab interface (https://jupyterlab.readthedocs.io/en/stable/user/interface.html)



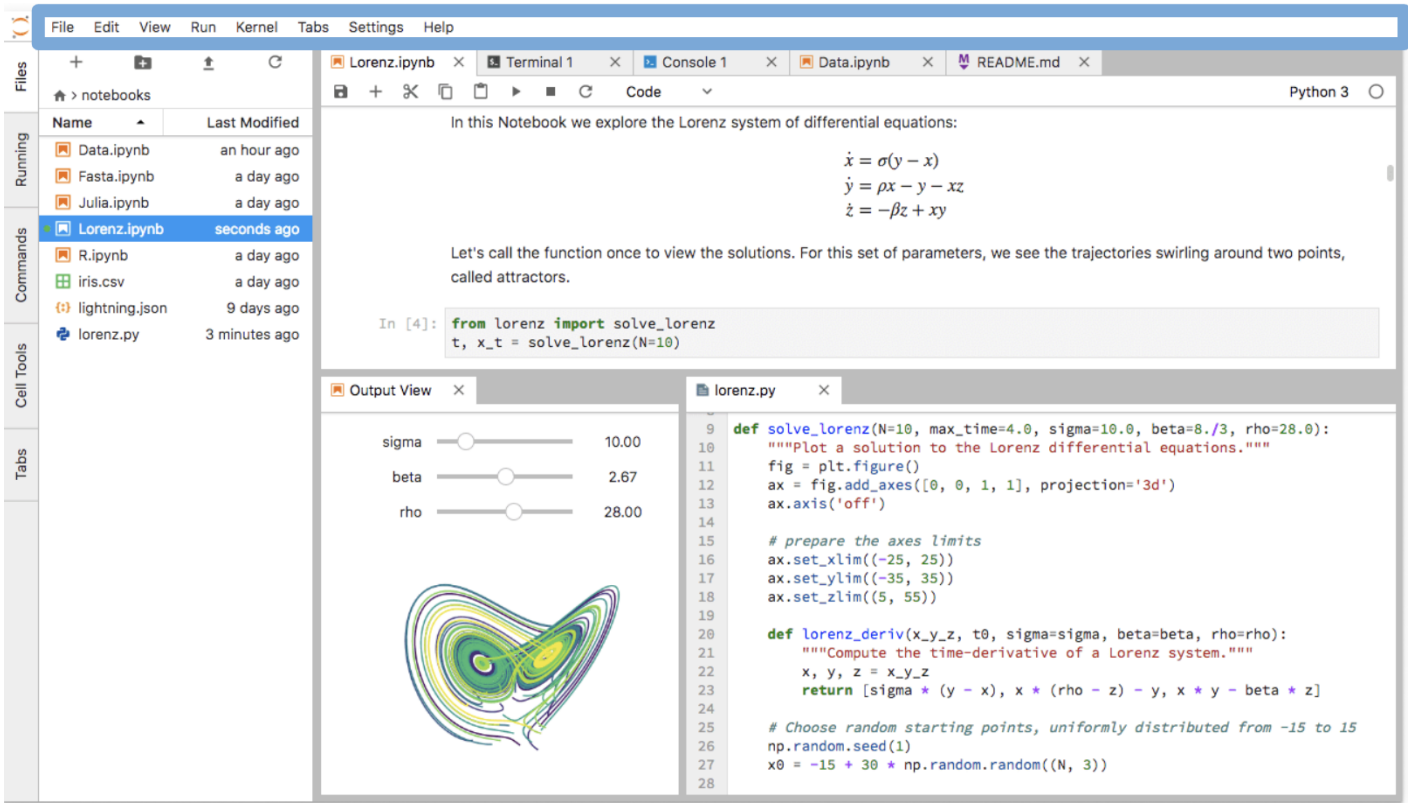
Jupyter Lab main area



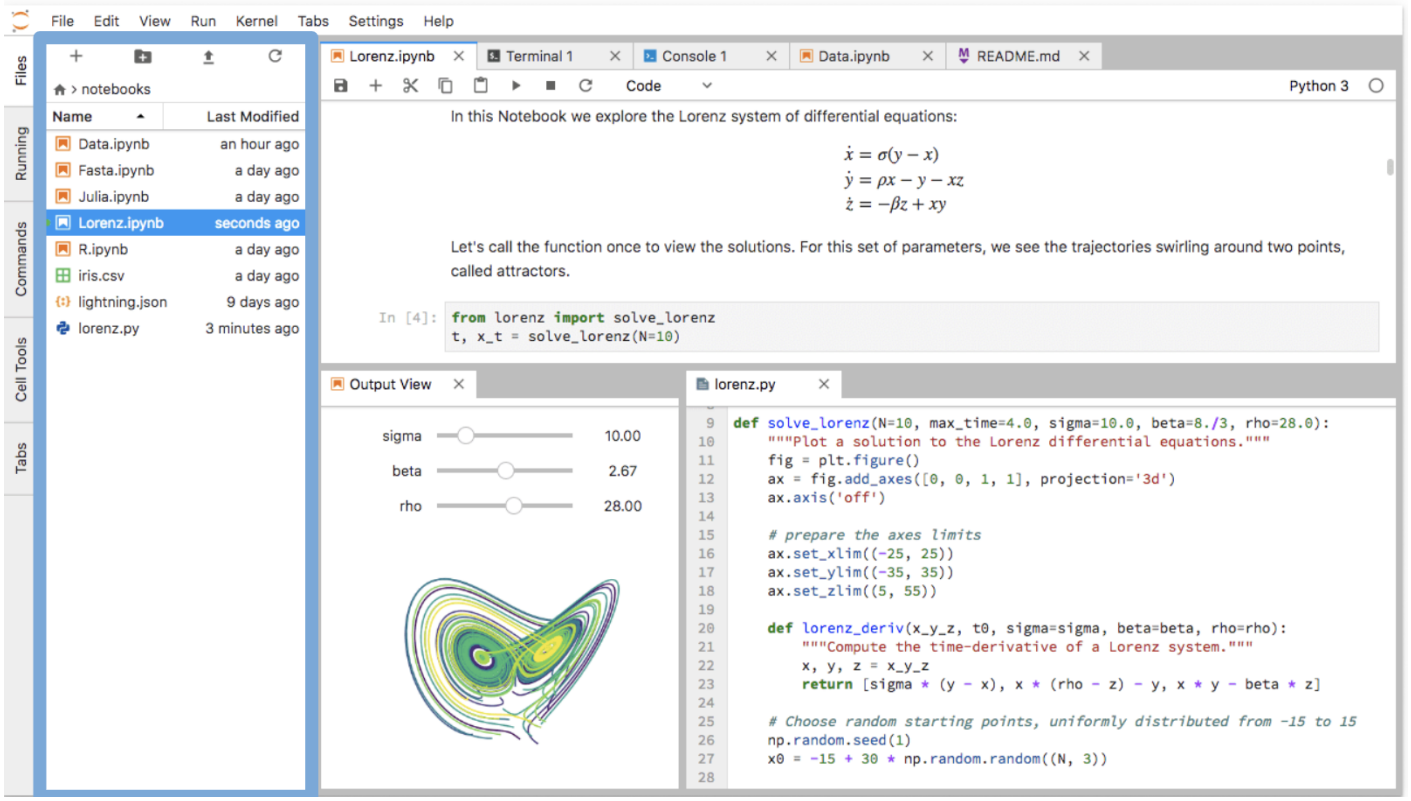
Jupyter Lab: many file types and panes



Jupyter Lab Menu

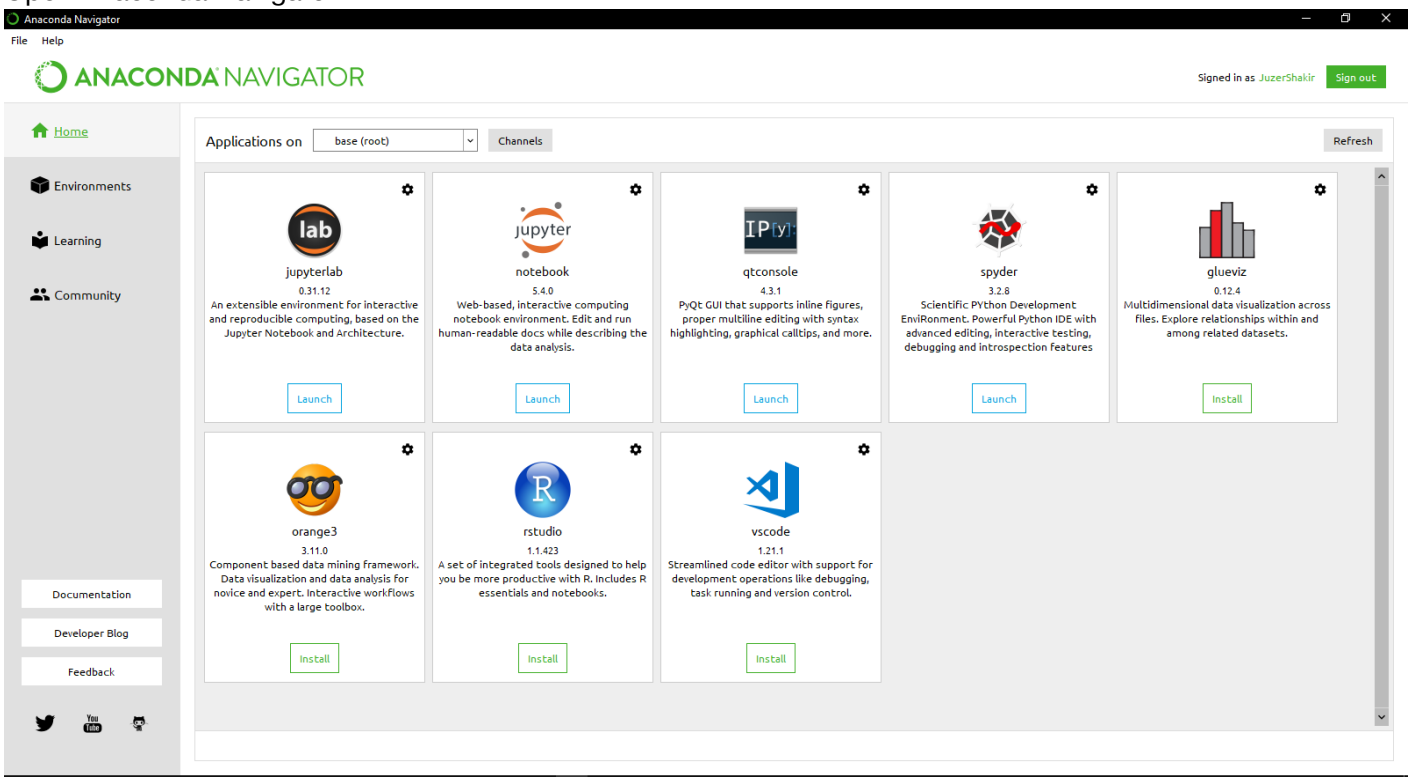


Jupyter Lab navigation



Let's open Jupyter lab!

Open Anaconda Navigator



Find the files for today on your computer!

Find this notebook and get to this cell

Quick and easy essential commands

Executing code

`command/ctrl + enter/return` to run a cell

```
In [ ]: print("I am excited to learn pandas with Alison")
```

Creating new cells

`b` to get a new code cell below where you are
`a` to get a new code cell above

Try it!

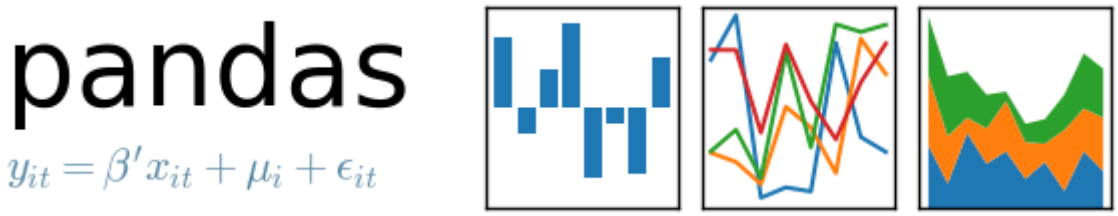
Cell navigation and conversion

`return/enter` to select it
`esc` to get out of cell to type in
`command/ctrl + m` to convert a cell to a markdown cell

Short exercise:

- create new code cell beneath this text
- select it
- type `print("I have successfully completed this step")`
- run the cell
- celebrate seeing the output below

Python for Data Analysis (https://pandas.pydata.org/index.html)



Assumption: all beginners at pandas

Quick aside: Why aren't we doing this in excel? Why code at all?

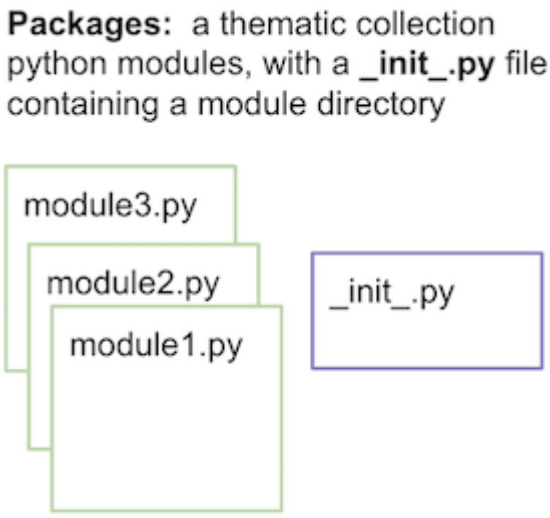
Most people have used Microsoft Excel or Google sheets. But what are the limitations of excel?

> Great example of limitations (https://www.bbc.com/news/magazine-22223190)

How is using python different?

- create documentation of processes as you code
- reduces chances for human error
- no "drag and drop"
- repeatable
- transparent

`pandas` is python `package`



Importing packages

All packages need to be imported into your python session to be used.
(If you're ever looking for a good time, go to the Python Package Index(PyPi (https://pypi.org/)) and explore all the other published and maintained python packages out there)

(also, if you're ever curious, the source code for pandas lives here (https://github.com/pandas-dev/pandas/tree/master/pandas/core))

Enough context, let's code!

Get and inspect data

In [ ]:
```
import pandas as pd
```

The data from the Austin Animal Shelter (http://www.austintexas.gov/department/aac) is hosted in these locations:

Intakes: https://data.austintexas.gov/Health-and-Community-Services/Austin-Animal-Center-Intakes/wter-evkm (https://data.austintexas.gov/Health-and-Community-Services/Austin-Animal-Center-Intakes/wter-evkm)
Outcomes: https://data.austintexas.gov/Health-and-Community-Services/Austin-Animal-Center-Outcomes/9t4d-g238 (https://data.austintexas.gov/Health-and-Community-Services/Austin-Animal-Center-Outcomes/9t4d-g238)

We will read it into our notebook using pd.read_csv (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.read_csv.html)

In [ ]:
```
outcomes = pd.read_csv('./data/Austin_Animal_Center_Outcomes.csv')
```

Let's do the same for intakes!

In [ ]:
```
intakes = pd.read_csv('./data/Austin_Animal_Center_Intakes.csv')
```

Inspect data

Check top of dataset

In [ ]:
```
outcomes.head()
```

Now that we can read in data, let's get more comfortable with our Pandas data structures.

In [ ]:
```
type(outcomes)
```

It's important that we know it's a `DataFrame` because now, given the documentation (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html), we can always expect answers on any dataset we load in.

What's the length and width of our dataframe?

In [ ]:
```
outcomes.shape
```

Get column names

In [ ]:
```
outcomes.columns
```

Columns on an individual level are `Series` objects
To access an individual column, the easiest way to to use `.` notation:
`outcomes.Name`

In [ ]:
```
outcomes.Name
```

Check data type of each column

Type of the data (integer, float, Python object, etc.)

In [ ]:
```
outcomes.dtypes
```

Apply to `intakes`

Now, for the `intakes` dataset. How does it compare to `outcomes` ?

- does it have the same number of observations?
- same column names?

Get data type *and* an idea of how many missing values

Which columns have missing data?

In [ ]:
```
outcomes.info()
```

Now, how about for intakes?

In [ ]:
```

```

Revisit our questions

- Age of animals in shelter
- Average animal length of stay
- Medical staff needed

Age of Animals in shelter should be easy, we have 'Age upon Outcome'

In [ ]:
```
outcomes['Age upon Outcome'].mean()
```

Wait! Something went wrong!

What happened? Why?

We are going to need to struggle through some data cleaning

Data Cleaning

First step: make the column names easier to work with

Going to use `str` , `lower` , and `replace` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.str.replace.html) to make our lives easier.

In [ ]:
```
outcomes.columns = outcomes.columns.str.lower()
```

In [ ]:
```
outcomes.columns
```

In [ ]:
```
outcomes.columns = outcomes.columns.str.replace(' ', '_')
```

In [ ]:
```
outcomes.columns
```

Apply to intakes!

In [ ]:
```

```

In [ ]:
```

```

Why care about that?

Because now I can use `tab` to find column names.
Let's now see if I can get the `value_counts()` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html) for the animal types in outcomes.

In [ ]:
```
outcomes
```

Let's see the unique values of age

In [ ]:
```
outcomes.age_upon_outcome.value_counts()
```

What's the challenge with these numbers?

What could we use instead?

Steps needed:

- convert dates to correct date types
- create a new age variable subtracting dates
- drop the original age variable

Converting dypes

Okay, going to use a `apply` (https://pandas.pydata.org/pandas-docs/version/0.18/generated/pandas.Series.apply.html) and a `lambda` (https://www.w3schools.com/python/python_lambda.asp) function.

It's getting exciting, now!

Anonymous Functions (Lambda Abstraction)

Simple functions can be defined right in the function call. This is called 'lambda abstraction'; the function thus defined has no name and hence is "anonymous".

Inspect data

Check top of dataset

In [ ]:
```
shelter_data.head()
```

Now that we can read in data, let's get more comfortable with our Pandas data structures.

In [ ]:
```
type(shelter_data)
```

What's the length and width of our dataframe?

In [ ]:
```
shelter_data.shape
```

Get column names

In [ ]:
```
shelter_data.columns
```

Check data type of each column

In [ ]:
```
shelter_data.dtypes
```

In [ ]:
```
# We can find the type of a particular columns in a data frame in this w
ay.

shelter_data['animal_id'].dtypes
```

Get data type *and* an idea of how many missing values

In [ ]:
```
shelter_data.info()
```

In [ ]:
```
outcomes['date_o'] = outcomes.datetime.apply(lambda x: x[:10])
```

Using `to_datetime` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html)

In [ ]:
```
# convert date formats
outcomes['date_o'] =  pd.to_datetime(outcomes['date_o'], format='%m/%d/%Y')
outcomes['dob'] =  pd.to_datetime(outcomes['date_of_birth'], format='%m/%d/%Y')
```

Check to see if it worked!

In [ ]:
```
outcomes.head()
```

In [ ]:
```
outcomes.dtypes
```

We did it!
Let's `drop` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html) the variables we will no longer use.

In [ ]:
```
outcomes.drop( )
```

Make new variable of age and years_old

In [ ]:
```
outcomes['age'] = outcomes.date_o - outcomes.dob
```

In [ ]:
```
outcomes['years_old'] = outcomes.age.apply(lambda x: x.days/365)
```

In [ ]:
```
outcomes.dtypes
```

NOW try `mean`!

In [ ]:
```
outcomes.years_old.mean()
```

But does that tell us anything useful?
No? Why?

In [ ]:
```
outcomes.columns
```

Filtering and sub-setting

Going to use a `groupby` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html) and a `loc` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html) function to help us aggregate and filter.

In [ ]:
```
outcomes[['animal_type', 'years_old']].groupby(['animal_type']).mean()
```

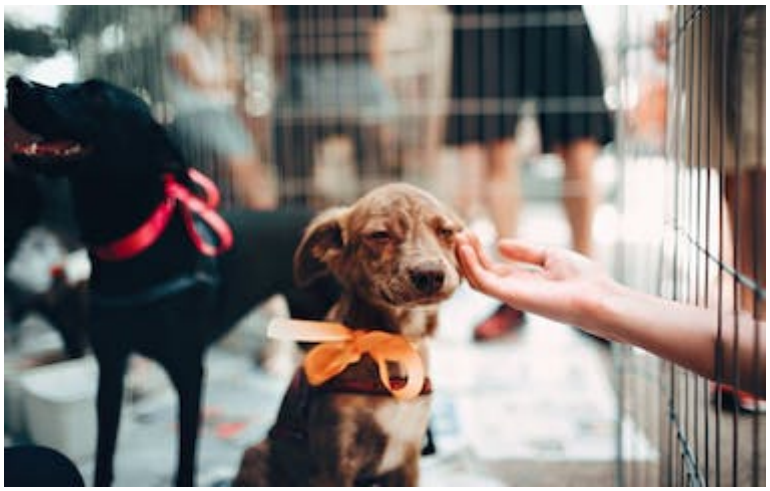
In [ ]:
```
outcomes[['animal_type', 'years_old']].loc[outcomes['date_o'] > '01-01-2019'].groupby(['animal_type']).mean()
```

Now we are rolling!!

Pause

If we haven't take a break yet, let's take a 5-10 minute stretch

Two more questions to go for the puppies!!!



How long is average stay?

- What data do we need to solve this question?
- What columns from which dataset?

In [ ]:
```
# Let's repeat the cleaning process for intake date and create some new
 variables

intakes['date_i'] = intakes.datetime.apply(lambda x: x[:10])

# convert date formats
intakes['date_i'] =  pd.to_datetime(intakes['date_i'], format='%m/%d/%Y'
)

# get more date info
intakes['month_i'] = intakes['date_i'].apply(lambda x: x.month)
intakes['year'] = intakes['date_i'].apply(lambda x: x.year)
intakes['weekday_i'] = intakes['date_i'].apply(lambda x: x.weekday())
```

In [ ]:
```
outcomes['year'] = outcomes['date_o'].apply(lambda x: x.year)
```

Methods for Combining DataFrames: `.join()`, `.merge()`, `.concat()`, `.melt()`

Today we are just using `merge` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html)

In [ ]:
```
animal_shelter_df  = pd.merge(intakes,
                              outcomes,
                              on=['animal_id', 'year'],
                              how='left',
                              suffixes=('_intake', '_outcome'))

animal_shelter_df = animal_shelter_df[(~animal_shelter_df['date_o'].isna
())
                                      & (animal_shelter_df['date_o'] > a
nimal_shelter_df['date_i'])]
```

In [ ]:
```
animal_shelter_df['days_in_shelter'] = (animal_shelter_df['date_o'] - an
imal_shelter_df['date_i']).dt.days
```

In [ ]:
```
pd.set_option('display.max_columns', 500)
animal_shelter_df.head()
```

How long to animals stay in the shelter?

In [ ]:
```
animal_shelter_df.days_in_shelter.mean()
```

Is that the full question and answer?

In [ ]:
```

```

Last question! Medical needs

1. How many animals come in injured? And what happens to them?
2. How many animals come in and over their stay get neutered?

> import numpy and use np.where

In [ ]:

Final reflection:

Wow, we really got somewhere!

What's a question about the animals in this dataset you could now feel confident answering?

Thank you!

Further Resources

- Learn from Wes McKinney himself (https://www.youtube.com/watch?v=_T8LGqJtuGc#action=share) in his "Pandas in 10 minutes video"
- Make the pandas documentation (https://pandas.pydata.org/pandas-docs/stable/reference/index.html) your best friend
- Codecademy
- apply to the Flatiron School!

May you hold on to pandas knowledge like this panda and his ball