Object Oriented Data Science with Python!

Sev Leonard
Portland Data Science Group
11/29/16

@gizm0_0
sev@thedatascout.com

whois Sev

- Portlandian for 12 years
- Circuit designer, software developer, sciencer of data
- Writer, educator
- Usually in the woods

Data Science Bootcamp!

- Early spring 2017 at PDX Code Guild
- Modular! Project based! Evenings!
- Topics include:
    - Python, SQL, and friends
    - Applied stats for data science
    - Machine learning
    - Capstone: Critical thinking, forming data problems

But Tonight..

Play a long at home:

https://github.com/gizm00/blog_code/tree/master/odsc/intro_oods

Juptyer notebook for tonight's talk

Data Science OOD style

- Application of Object Oriented Design principles to data science
- Top down approach to code organization
- Examples based on the Recreation Information Data Base (RIDB)

https://usda.github.io/RIDB

Why?

- Code as building blocks
- Testing
- Sharing and reuse
- Add new functionality without breaking existing code
- “Paper trail” of data manipulation.
- Create data migration robust code base

RIDB

Objects

“An object encapsulates data, attributes, and methods relating to a specific entity.”

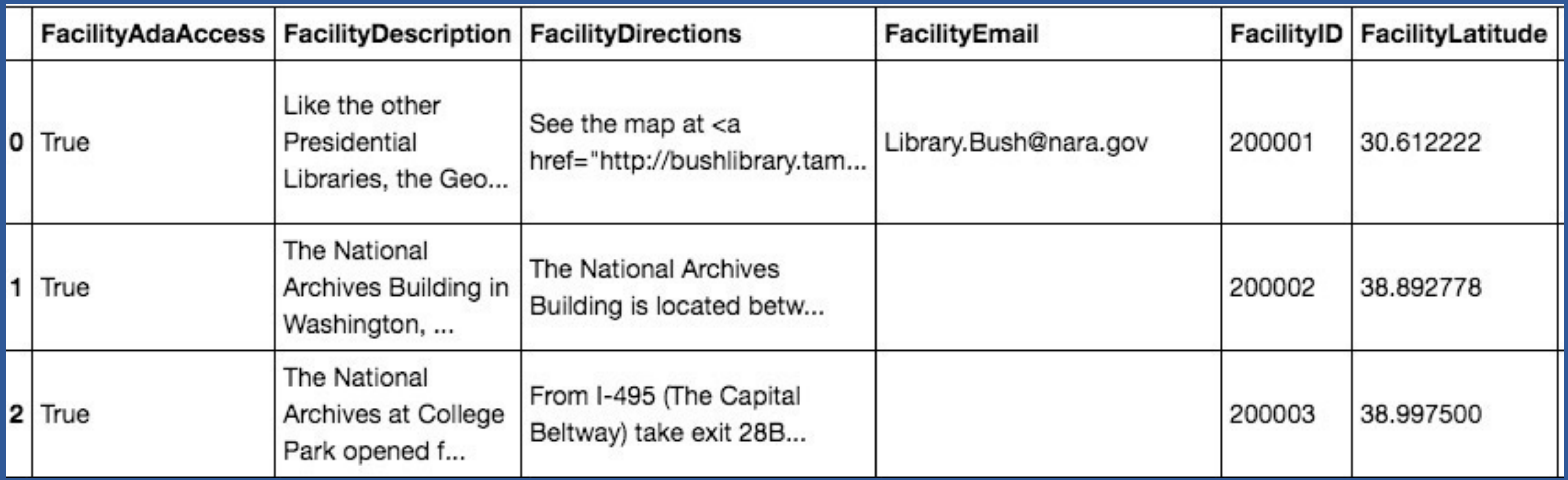
Example: API Object

- Attribute: Endpoint (http://someendpoint.com)
- Attribute: Parameters (key=“Wud22JKu6”)
- Method: Extract()
- Method: Clean()
- Data: DataFrame

Class.py

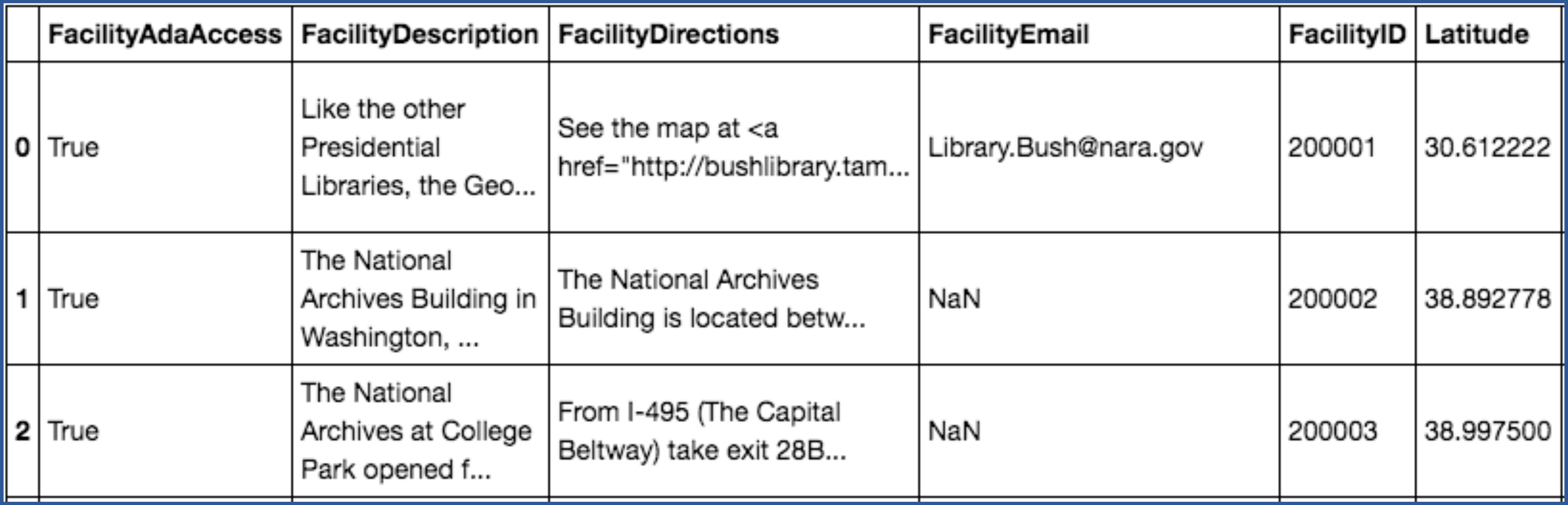
```
class RidbData():

   def __init__(self, endpoint, url_params):

      self.df = pd.DataFrame()

      self.endpoint = endpoint

      self.url_params = url_params

  def extract(self):

      response = requests.get(url=self.endpoint,params=self.url_params)

      data = json.loads(response.text)

      self.df = json_normalize(data['RECDATA'])

   def clean(self) :

      self.df = self.df.replace('', np.nan)

      self.df.columns = self.df.columns.str.replace('.*Latitude', 'Latitude')

      self.df.columns = self.df.columns.str.replace('.*Longitude',
'Longitude')

      self.df = self.df.dropna(subset=['Latitude','Longitude'])
```

```
facilities = RidbData(
    'https://ridb.recreation.gov/api/v1/facilities',
    dict(apiKey = 'MY_RIDB_API_KEY'))
facilities.extract()
facilities.df.head()
```



	FacilityAdaAccess	FacilityDescription	FacilityDirections	FacilityEmail	FacilityID	FacilityLatitude
0	True	Like the other Presidential Libraries, the Geo...	See the map at <a href="http://bushlibrary.tam...	Library.Bush@nara.gov	200001	30.612222
1	True	The National Archives Building in Washington, ...	The National Archives Building is located betw...		200002	38.892778
2	True	The National Archives at College Park opened f...	From I-495 (The Capital Beltway) take exit 28B...		200003	38.997500

```
facilities.clean()

facilities.df.head()
```



	FacilityAdaAccess	FacilityDescription	FacilityDirections	FacilityEmail	FacilityID	Latitude
0	True	Like the other Presidential Libraries, the Geo...	See the map at <a href="http://bushlibrary.tam...	Library.Bush@nara.gov	200001	30.612222
1	True	The National Archives Building in Washington, ...	The National Archives Building is located betw...	NaN	200002	38.892778
2	True	The National Archives at College Park opened f...	From I-495 (The Capital Beltway) take exit 28B...	NaN	200003	38.997500

That’s a lot of work!

```
def get_ridb_data(endpoint,url_params):
    response = requests.get(url = endpoint, params = url_params)
    data = json.loads(response.text)
    df = json_normalize(data['RECDATA'])
    df = df.replace('', np.nan)
    df.columns = df.columns.str.replace('.*Latitude', 'Latitude')
    df.columns = df.columns.str.replace('.*Longitude', 'Longitude')
    df = df.dropna(subset=['Latitude','Longitude'])
    return df
```

Function Example

```
df_cg = get_ridb_data(activities_endpoint,
camping_params)

df_np = get_ridb_data(facilities_endpoint,
nat_parks_params)
```

Same response/extraction, same data cleaning

Different endpoint URLs and parameters.

Lose data transformation “paper trail”

```
def get_ridb_data(endpoint,url_params):
    response = requests.get(url = endpoint, params = url_params)

    data = json.loads(response.text)

    df = json_normalize(data['RECDATA'])

    df = df.replace('', np.nan)

    df.columns = df.columns.str.replace('.*Latitude', 'Latitude')

    df.columns = df.columns.str.replace('.*Longitude', 'Longitude')

    df = df.dropna(subset=['Latitude','Longitude'])

    return df
```

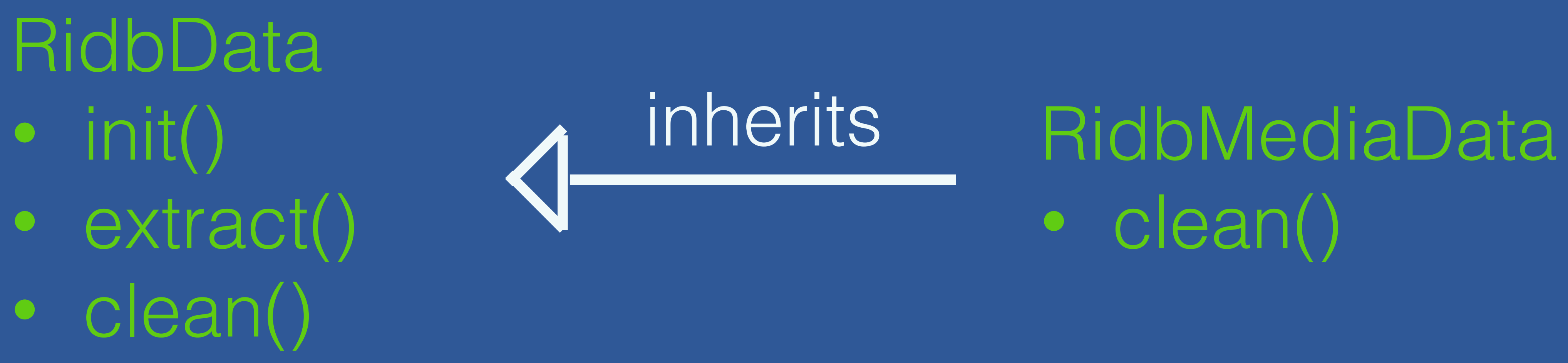
extract()

clean()

```
def get_ridb_facility_media(endpoint, url_params):
    response = requests.get(url = endpoint, params = url_params)

    data = json.loads(response.text)

    df = json_normalize(data['RECDATA'])

    df = df[df['MediaType'] == 'Image']

    return df
```

Open/Closed Principle

Classes are open for extension, but closed for modification.



RidbData

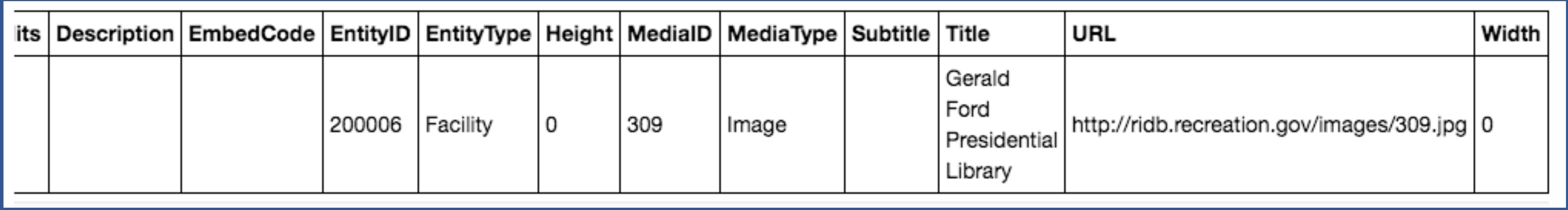
- init()
- extract()
- clean()

inherits

RidbMediaData

- clean()

```
class RidbMediaData(RidbData):
    def clean(self) :
        self.df = self.df[self.df['MediaType'] == ‘Image']

facility_media = RidbMediaData(
    'https://ridb.recreation.gov/api/v1/facilities/200006/media',
    dict(apiKey = 'MY_RIDB_API_KEY'))
facility_media.extract()
facility_media.clean()
facility_media.df
```



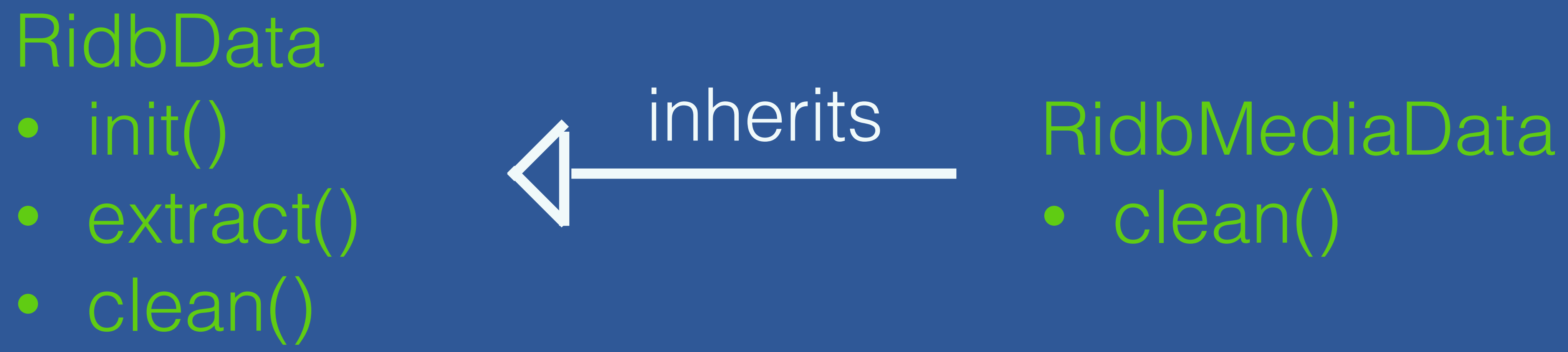
its	Description	EmbedCode	EntityID	EntityType	Height	MediaID	MediaType	Subtitle	Title	URL	Width
			200006	Facility	0	309	Image		Gerald Ford Presidential Library	http://ridb.recreation.gov/images/309.jpg	0

Putting it all together

```
facilities_endpoint = 'https://ridb.recreation.gov/api/v1/facilities/'

recareas_endpoint = 'https://ridb.recreation.gov/api/v1/recareas'

key_dict = dict(apiKey = config.API_KEY)

facilities = RidbData('facilities', facilities_endpoint, key_dict)

recareas = RidbData('recareas', recareas_endpoint, key_dict)

facility_media = RidbMediaData('facilitymedia', facilities_endpoint,
media_params)


ridb_data = [facilities,recareas,facility_media]

list(map(lambda x: x.extract(), ridb_data))

list(map(lambda x: x.clean(), ridb_data))
```

Summary

- Reduce repeated code



RidbData

- init()
- extract()
- clean()

inherits

RidbMediaData

- clean()

- Minimal new code to test

```
def clean(self) :

    self.df = self.df[self.df['MediaType'] == ‘Image']
```

Moar Summary

- React to data migration
  - Extend with new extract(), clean() methods as data changes
- Track data transformations
- Uniform interface

```
ridb_data = [facilities,recareas,facility_media]

list(map(lambda x: x.extract(), ridb_data))

list(map(lambda x: x.clean(), ridb_data))
```

More on OODS

- PyCon Object Oriented Data Pipelineing Tutorial
  - https://github.com/gizm00/pycon2016
  - https://www.youtube.com/watch?v=n4VLLQXF_9Y
- Github for this presentation: https://github.com/gizm00/blog_code/tree/master/odsc/intro_oods
- ODSC article: https://www.opendatascience.com/blog/an-introduction-to-object-oriented-data-science-in-python/

Thanks!

sev@thedatascout.com

@gizm0_0

github.com/gizm00