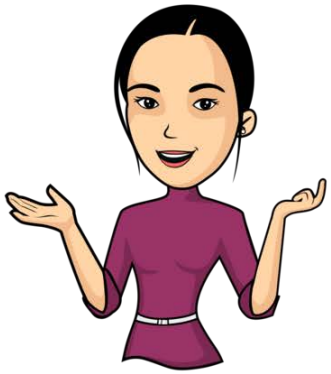
How to be Pythonic? Design a Query Language in Python

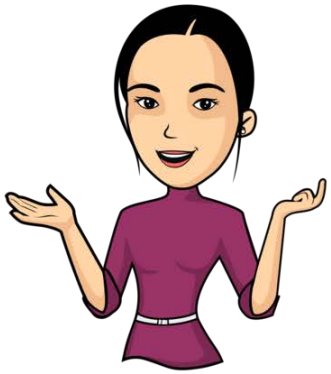
Cheuk Ting Ho

https://cheuk.dev

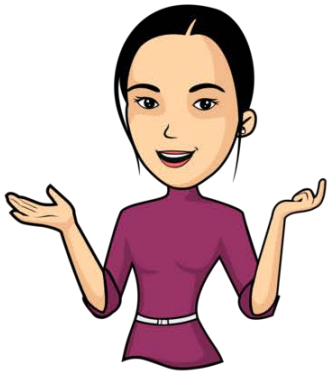
cheukting_ho @cheukting_ho Cheukting



Cheuk Ting Ho



Cheuk Ting Ho



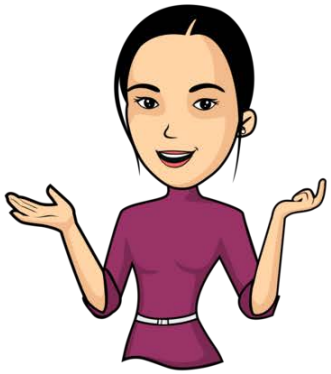
Cheuk Ting Ho

Pick a Mix

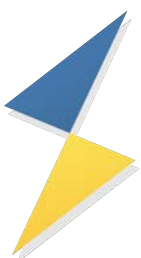
TerminusDB

PYTHON SPRINTS

PyData Global



Cheuk Ting Ho



PYTHON SPRINTS

PyData Global

https://www.twitch.tv/cheukting_ho

“ Pythonic means code that doesn't just get the syntax right but that follows the conventions of the Python community and uses the language in the way it is intended to be used.

“ Pythonic means code that doesn't just get the syntax right but that follows the conventions of the Python community and uses the language in the way it is intended to be used.

- Stackoverflow

“ Pythonic means code that doesn't just get the syntax right but that follows the conventions of the Python community and uses the language in the way it is intended to be used.

- Stackoverflow

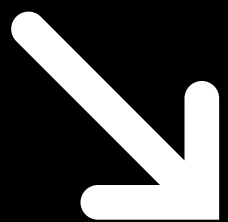
“ Pythonic means code that doesn't just get the syntax right but that follows the conventions of the Python community and uses the language in the way it is intended to be used.

- Stackoverflow

Why can't I just do it in a for-loop?

```
1 for i in (i; i < items.length ; i++)
2  {
3   n = items[i];
4  ... now do something
5  }
```

```
1 for i in (i; i < items.length ; i++)
2  {
3   n = items[i];
4  ... now do something
5  }
```



```
1 for i in items:
2   i.perform_action()
```

```
1 for i in (i; i < items.length ; i++)
2  {
3   n = items[i];
4  ... now do something
5  }
```

```
1 for i in items:
2   i.perform_action()
```

```
1 (i.some_attribute for i in items)
```

```
1 for i in (i; i < items.length ; i++)
2  {
3   n = items[i];
4  ... now do something
5  }
```

```
1 for i in items:
2   i.perform_action()
```

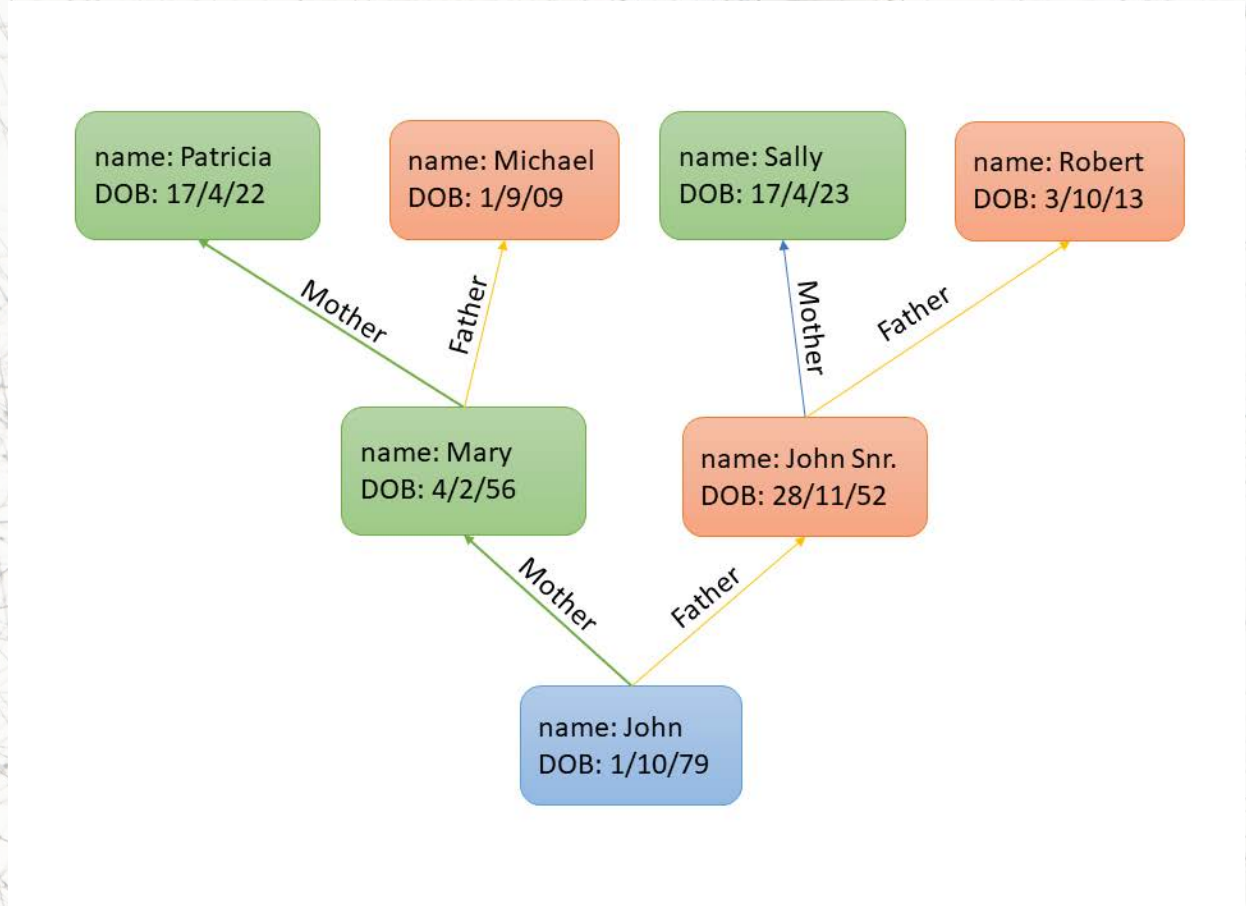
Pythonic!

```
1 (i.some_attribute for i in items)
```

It all started...

Developer Advocate of

Person_ID	Name	DOB	mother	father
1	John	1/10/79	2	3
2	Mary	4/2/56	4	5
3	John snr	28/11/52	6	7
4	Patricia	17/4/22	null	null
5	Michael	1/9/09	null	null
6	Sally	17/4/23	null	null
7	Robert	3/10/13	null	null


name: Patricia
DOB: 17/4/22
name: Michael
DOB: 1/9/09
name: Sally
DOB: 17/4/23
name: Robert
DOB: 3/10/13
Mother
Father
Mother
Father
name: Mary
DOB: 4/2/56
name: John Snr.
DOB: 28/11/52
Mother
Father
name: John
DOB: 1/10/79

```
1  SELECT Name from TABLE where Person_ID =
   (SELECT mother from TABLE where Name="John")
2  SELECT Name from TABLE where Person_ID =
   (SELECT mother from TABLE WHERE Person_ID =
   (SELECT mother from TABLE where Name="John"))
```

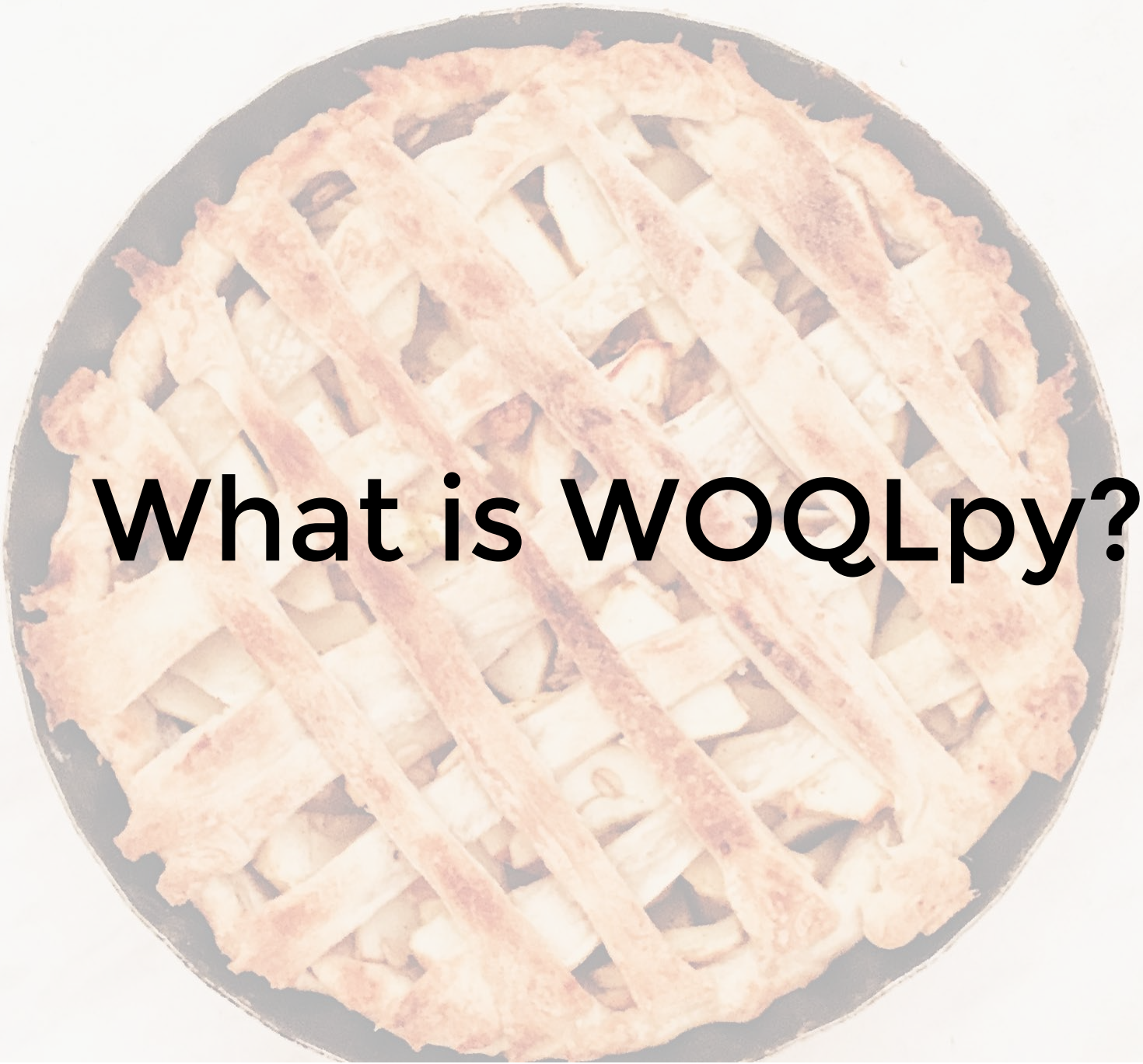
```
WOQL.and(
    WOQL.triple("v:Person", "mother", "v:MotherID"),
    WOQL.triple("v:MotherID", "name", "v:MotherName"),
    WOQL.triple("v:MotherID", "mother", "v:GrandmotherID"),
    WOQL.triple("v:GrandmotherID", "name", "v:GrandmotherName"),
)
```

WOQLpy

a Query Language Client for Pythonistas
and Data Scientists

WOQLpy

a Query Language Client for Pythonistas
and Data Scientists



What is WOQLpy?

It comes with the Python Client, which you can pip install:

```
pip install terminusdb-client
```

It comes with the Python Client, which you can pip install:

```
pip install terminusdb-client
```

Output to DataFrames

```
pip install terminusdb-client[dataframe]
```

Change the result returned form your query into pandas DataFrame

```
WOQLDataFrame.result_to_df(result)
```

It let's you to "talk" to TerminusDB like this:

```
import terminusdb_client as woql
from terminusdb_client import WOQLQuery

db_id = "pybike"
client = woql.WOQLClient(server_url = "http://localhost:6363")
client.connect(key="root", account="admin", user="admin")
client.create_database(db_id, accountid="admin", label = "Bike Graph",
                       description = "Create a graph with bike data")

station_dt = WOQLQuery().doctype("Station",
                                 label="Bike Station",
                                 description="A station where bikes are deposited")
bicycle_dt = WOQLQuery().doctype("Bicycle", label="Bicycle")
journey_dt = (
  WOQLQuery().doctype("Journey", label="Journey").
  property("start_station", "Station", label="Start Station").
  property("end_station", "Station", label="End Station").
  property("duration", "integer", label="Journey Duration").
  property("start_time", "dateTime", label="Time Started").
  property("end_time", "dateTime", label="Time Ended").
  property("journey_bicycle", "Bicycle", label="Bicycle Used")
)
schema = station_dt + bicycle_dt + journey_dt
schema.execute(client)
```

It let's you to "talk" to TerminusDB like this:

```
import terminusdb_client as woql
from terminusdb_client import WOQLQuery

db_id = "pybike"
client = woql.WOQLClient(server_url = "http://localhost:6363")
client.connect(key="root", account="admin", user="admin")
client.create_database(db_id, accountid="admin", label = "Bike Graph",
                       description = "Create a graph with bike data")

station_dt = WOQLQuery().doctype("Station",
                                 label="Bike Station",
                                 description="A station where bikes are deposited")
bicycle_dt = WOQLQuery().doctype("Bicycle", label="Bicycle")
journey_dt = (
  WOQLQuery().doctype("Journey", label="Journey").
  property("start_station", "Station", label="Start Station").
  property("end_station", "Station", label="End Station").
  property("duration", "integer", label="Journey Duration").
  property("start_time", "dateTime", label="Time Started").
  property("end_time", "dateTime", label="Time Ended").
  property("journey_bicycle", "Bicycle", label="Bicycle Used")
)
schema = station_dt + bicycle_dt + journey_dt
schema.execute(client)
```

Instead of this:

```
{
  "when": [
    {
      "true": []
    },
    {
      "and": [
        {
          "add_quad": [
            "scm:Station",
            "rdf:type",
            "owl:Class",
            "db:schema"
          ]
        },
        {
          "add_quad": [
            "scm:Station",
            "rdfs:subClassOf",
            "tcs:Document",
            "db:schema"
          ]
        },
        {
```

You can do both

```
1  (WOQLQuery().doctype("Station")
2              .label("Bike Station")
3              .description("A station where bikes are deposited")
4  )
```

or

```
1  WOQLQuery().doctype("Station",
2                      label="Bike Station",
3                      escription="A station where bikes are deposited")
```

Design challanges

JavaScript: WOQL.and()

Python: WOQLQuery().and() ?

Design challanges

JavaScript: WOQL.and()

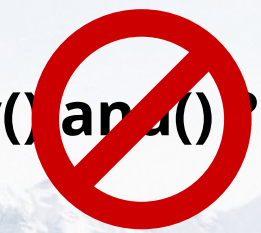
Python: WOQLQuery().and()

"and" is a key word, you dummy!

Design challanges

JavaScript: WOQL.and()

Python: WOQLQuery().and()



"and" is a key word, you dummy!

OK, woql_and then....😓
WOQLQuery().woql_and()

*** actually you can use the + operator**
thanks for the overload ability in Python

also happened to: or, not, as, from...

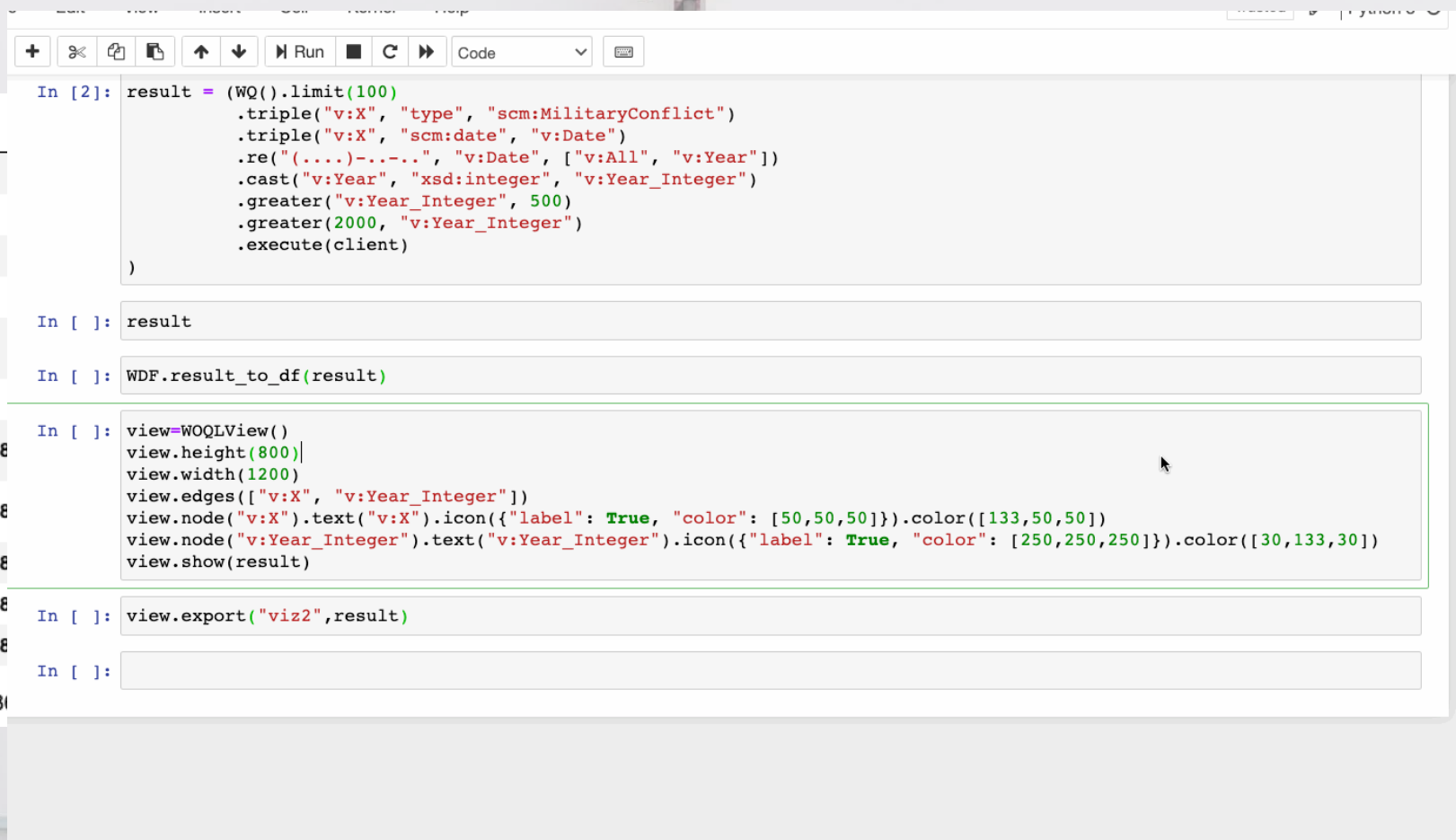
Integration with Jupyter notebook

Integration with Jupyter notebook

	End	End_Label	Start	Start_Label
0	http://localhost:6363/pybike/document/Station3...	S Glebe & Potomac Ave	http://localhost:6363/pybike/document/Station3...	Crystal City Metro / 18th & Bell St
1	http://localhost:6363/pybike/document/Station3...	S Glebe & Potomac Ave	http://localhost:6363/pybike/document/Station3...	Crystal City Metro / 18th & Bell St
2	http://localhost:6363/pybike/document/Station3...	New Hampshire Ave & T St NW	http://localhost:6363/pybike/document/Station3...	14th & V St NW
3	http://localhost:6363/pybike/document/Station3...	New Hampshire Ave & T St NW	http://localhost:6363/pybike/document/Station3...	14th & V St NW
4	http://localhost:6363/pybike/document/Station3...	Georgia Ave and Fairmont St NW	http://localhost:6363/pybike/document/Station3...	14th & V St NW
...	...	...	...	...
1857	http://localhost:6363/pybike/document/Station3...	11th & O St NW	http://localhost:6363/pybike/document/Station3...	New Jersey Ave & N St NW/Dunbar HS
1858	http://localhost:6363/pybike/document/Station3...	11th & O St NW	http://localhost:6363/pybike/document/Station3...	New Jersey Ave & N St NW/Dunbar HS
1859	http://localhost:6363/pybike/document/Station3...	D St & Maryland Ave NE	http://localhost:6363/pybike/document/Station3...	2nd St & Massachusetts Ave NE
1860	http://localhost:6363/pybike/document/Station3...	Maine Ave & 7th St SW	http://localhost:6363/pybike/document/Station3...	Potomac Ave & Half St SW
1861	http://localhost:6363/pybike/document/Station3...	King Farm Blvd & Piccard Dr	http://localhost:6363/pybike/document/Station3...	Shady Grove Metro West

1862 rows × 4 columns

Integration with Jupyter notebook



Schema builder

```
 1 class Coordinate(Object):
 2     x : float
 3     y : float
 4
 5 class Country(Document):
 6     name : str
 7     perimeter : List[Coordinate]
 8
 9 class Address(Object):
10     street : str
11     country : Country
12
13 class Person(Document):
14     name : str
15     age : int
16     friend_of : Set['Person']
17
18 class Employee(Person):
19     address_of : Address
20     contact_number : str
21     managed_by : 'Employee'
```

Look into the future

Mass loading data from DataFrame (and csvs)

Mass loading data from DataFrame (and csvs)

Network graph analysing algorithms

Mass loading data from DataFrame (and csvs)

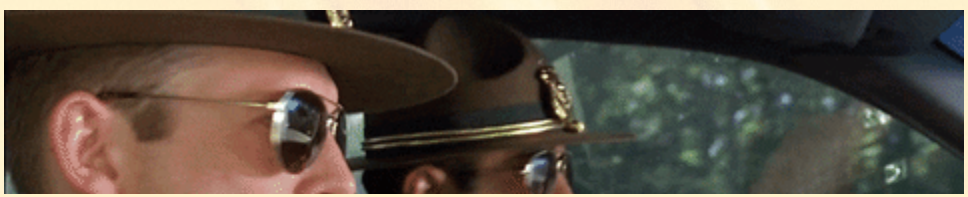
Network graph analysing algorithms

Schema checker

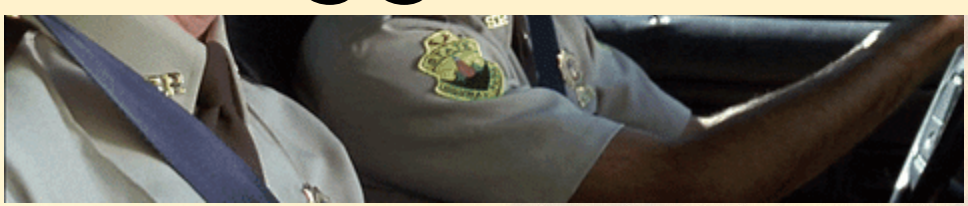
Mass loading data from DataFrame
(and csvs)

Network graph analysing algorithms

Schema checker



Your suggestion here



TerminusDB Academy:
https://academy.terminusdb.com/

To get the newest update👍:
Follow us on Twitter: @TerminusDB
Website: https://terminusdb.com/

Join the community at Discord:
https://discord.gg/Gvdqw97

We want to hear from you 😊