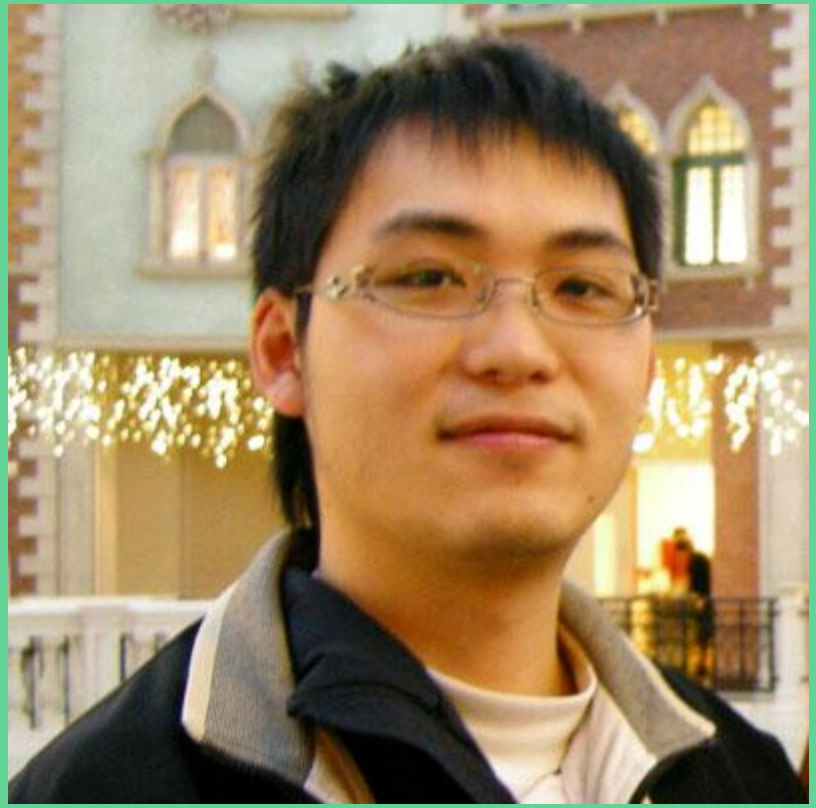
Python Metaclass in Practice

Data Piping

Dboy Liao
< GitHub >



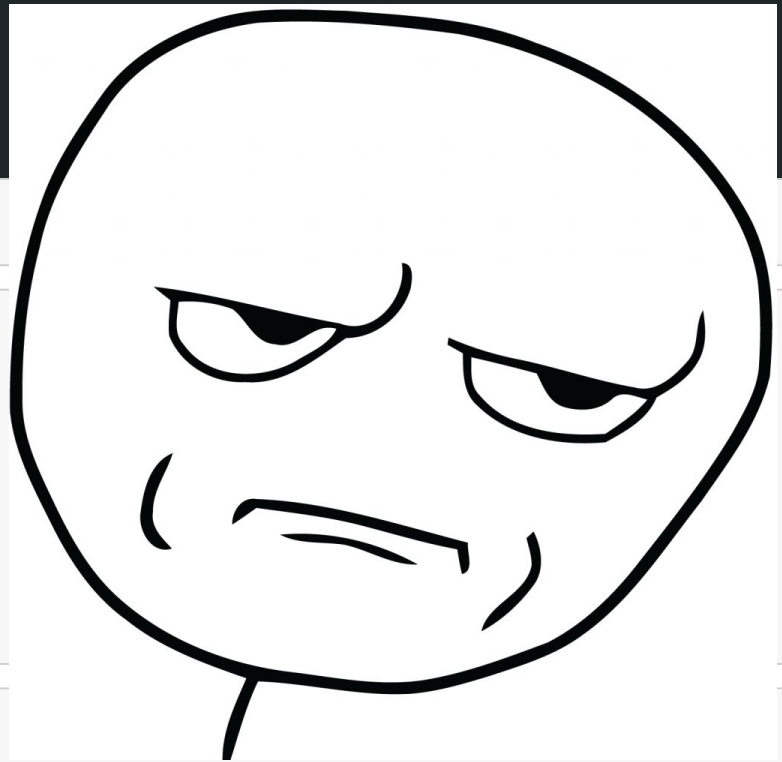
Metaclass is not easy.....

0. Nested Function

```
from __future__ import print_function
import math
```

```
def add_by(data, by = 1):

    return list(map(lambda x: x + by, data))

def mul_by(data, by = 1):

    return list(map(lambda x: by*x, data))

def apply_fun(data, fun = math.sin):

    return list(map(fun, data))
```

```
def my_awesome_algorithm(data):

    # Urrr.....
    result = mul_by(apply_fun(mul_by(add_by(data, 1), math.pi), math.cosh), math.sin(3.232489))

    return result
```



0. Nested Function

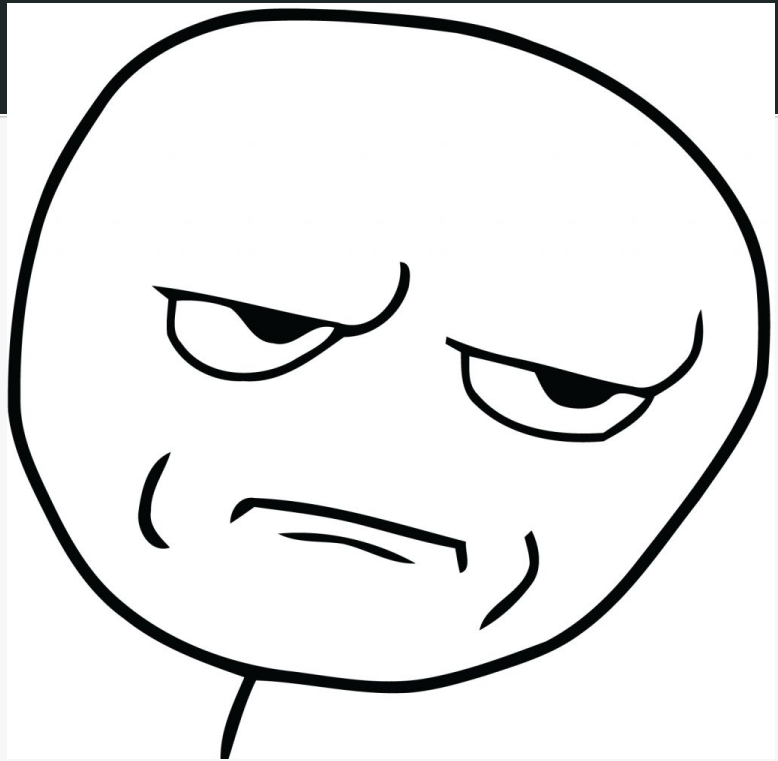
```
def my_awesome_algorithm(data):

    # Seriously?
    result = mul_by(apply_fun(
                        mul_by(
                |               add_by(data, 1),
                                math.pi),
                        math.cosh),
                math.sin(3.232489))

    return result
```

```
print(my_awesome_algorithm([1, 2, 3]))
```

```
[-24.303703304709376, -562.4025703848465, -13014.384932563977]
```

1. How About Class?

```
class MyAwesomeClass(object):

    def add_by(self, data, by = 1):
        return list(map(lambda x: x + by, data))

    def mul_by(self, data, by = 1):
        return list(map(lambda x: by*x, data))

    def apply_fun(self, data, fun = math.sin):
        return list(map(fun, data))

    def awesome_algorithm(self, data):
        # You're not helping....Really!
        result = self.mul_by(
                    self.apply_fun(
                                self.mul_by(
                                        self.add_by(data, 1),
                                        math.pi),
                                math.cosh),
                    math.sin(3.232489))

        return result
```



```
my_awesome_obj = MyAwesomeClass()
my_awesome_obj.awesome_algorithm([1, 2, 3])
```

```
[-24.303703304709376, -562.4025703848465, -13014.384932563977]
```

I Want This!

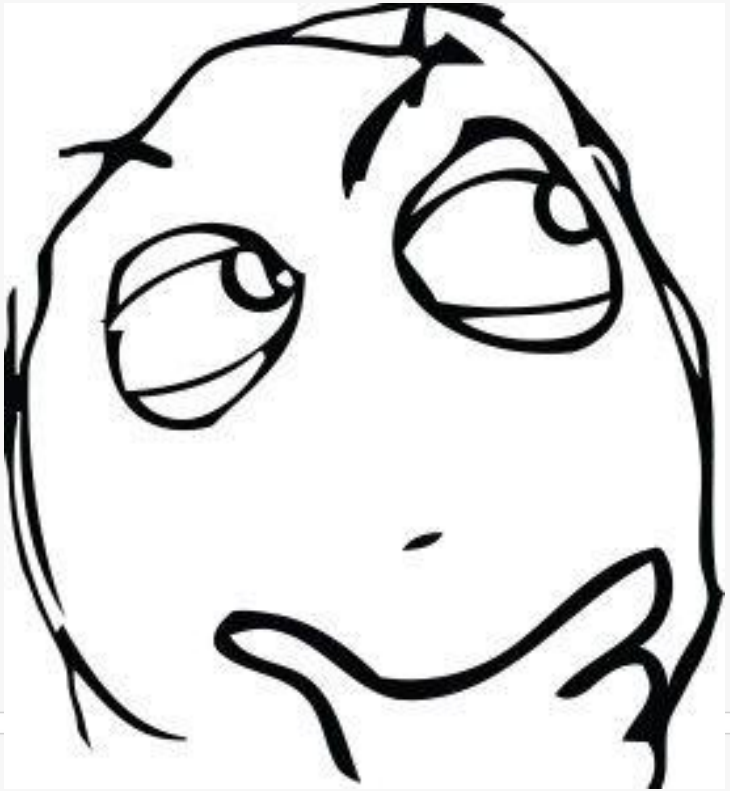
```
foo_foo %>%
  hop(through = forest) %>%
  scoop(up = field_mouse) %>%
  bop(on = head)
```

….in Python

Just Write in R!!

1. How About Class?

```
# Ok...I got it
class MyDataProcessor(object):

    def __init__(self, data):
        self.data = data


    def add_data_by(self, by = 1):
        self.data = list(map(lambda x: x + by, self.data))
        return self

    def mul_data_by(self, by = 1):
        self.data = list(map(lambda x: by*x, self.data))
        return self

    def apply_fun(self, fun = math.sin):
        self.data = list(map(fun, self.data))
        return self

    def awesome_algorithm(self):
        # Much bettr now
        self.add_data_by(1)\
            .mul_data_by(math.pi)\
            .apply_fun(math.cosh)\
            .mul_data_by(math.sin(3.232489))

        return self  # <--- this line is repetitive....
```



```
data_processor = MyDataProcessor([1, 2, 3])
print(data_processor.awesome_algorithm().data)
```

```
[-24.303703304709376, -562.4025703848465, -13014.384932563977]
```

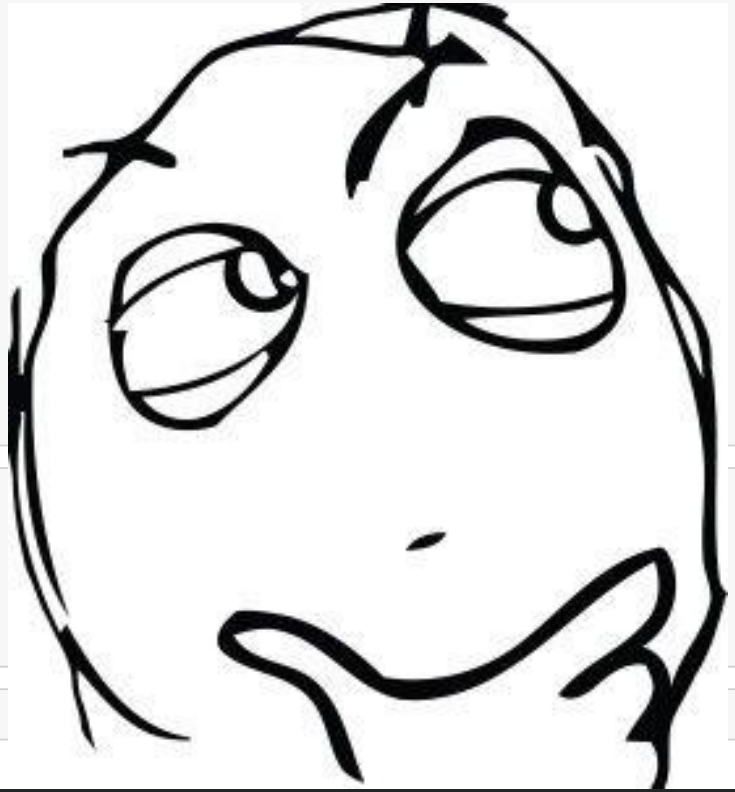
2. Decorate Your Own Code

```
class MyDataProcessor(object):

    def __init__(self, data):
        self.data = data

    @return_self
    def add_data_by(self, by = 1):
        return list(map(lambda x: x + by, self.data))

    @return_self
    def mul_data_by(self, by = 1):
        return list(map(lambda x: by*x, self.data))

    @return_self
    def apply_fun(self, fun = math.sin):
        return list(map(fun, self.data))
```

```
data_processor = MyDataProcessor([1, 2, 3])
# Nice!
result = data_processor.add_data_by(1)\
                       .mul_data_by(math.pi)\
                       .apply_fun(math.cosh)\
                       .mul_data_by(math.sin(3.232489)).data
```

```
print(result)
```

```
[-24.303703304709376, -562.4025703848465, -13014.384932563977]
```



3. Metaclass Save Your Code

```
from pypeup import DataPipe

class MyDataProcessor(DataPipe):

    def add_data_by(self, by = 1):
        return list(map(lambda x: x + by, self.data))

    def mul_data_by(self, by = 1):
        return list(map(lambda x: by*x, self.data))

    def apply_fun(self, fun = math.sin):
        return list(map(fun, self.data))
```

```
data_processor = MyDataProcessor([1, 2, 3])
result = data_processor.add_data_by(1)\
                       .mul_data_by(math.pi)\
                       .apply_fun(math.cosh)\
                       .mul_data_by(math.sin(3.232489)).data
```

```
print(result)
```

```
[-24.303703304709376, -562.4025703848465, -13014.384932563977]
```

Thanks for Your Attention

References

- PyCon Keynote by David Beazley
- pandas.DataFrame.pipe
- GitHub - dboyliao/pypeup
- Wiki - Method Chaining

Other good python packages available

- Toolz
- CyToolz
- PyFunctional
- dask
- dill
- PyMonad (<-- this one is crazy!)

Special thanks to Joseph Yen

We are hiring!