Introduction to Reactive Programming with RxPY

R. Picard

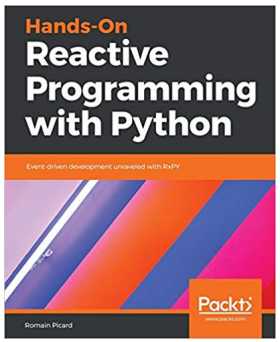
2020/02/01

Agenda

- Reactive Programming, ReactiveX, RxPY
- Hello World
- Error Management
- Concurrency
- RxPY in Real Life

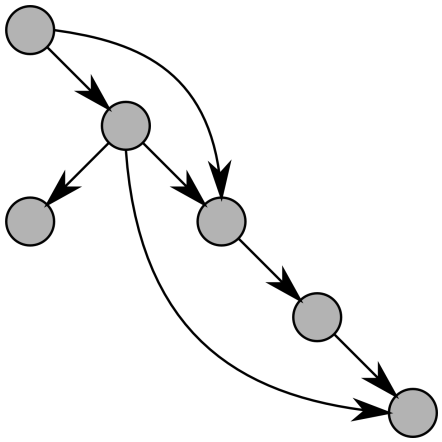
About Me

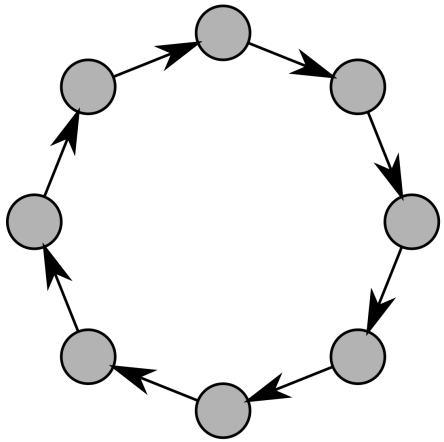
- Data scientist
- Core developer of RxPY
- Author



Reactive Programming, ReactiveX

- Reactive Programming:
  - Easy way to write event driven code
  - Write computation graph
- ReactiveX:
  - From Microsoft
  - Available in 20+ languages





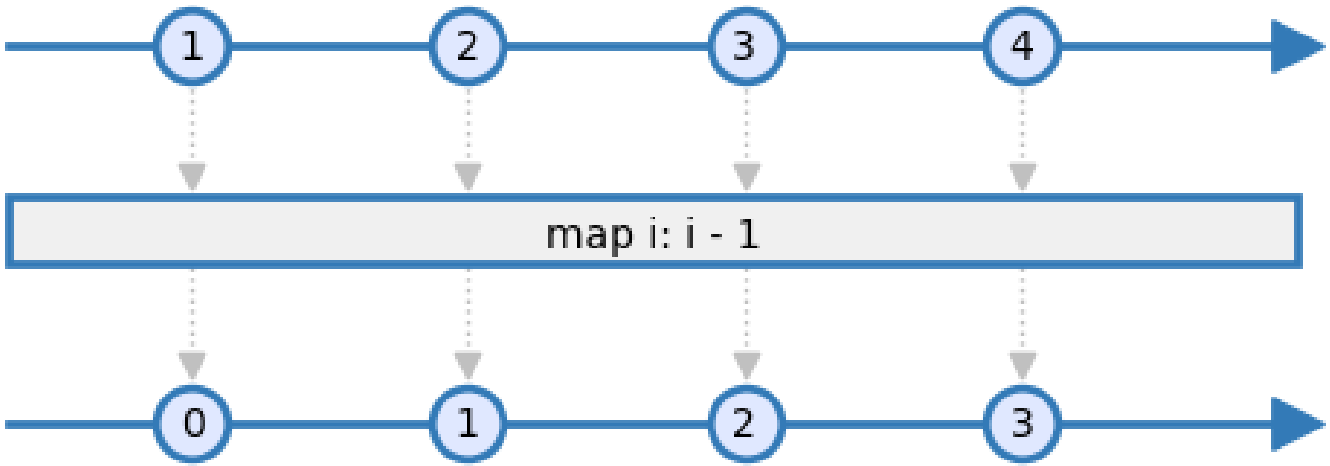
RxPY

- The Python implementation of ReactiveX
- v3 released in summer 2019:
  - Pipe operator
  - Documentation !
  - Python 3 only

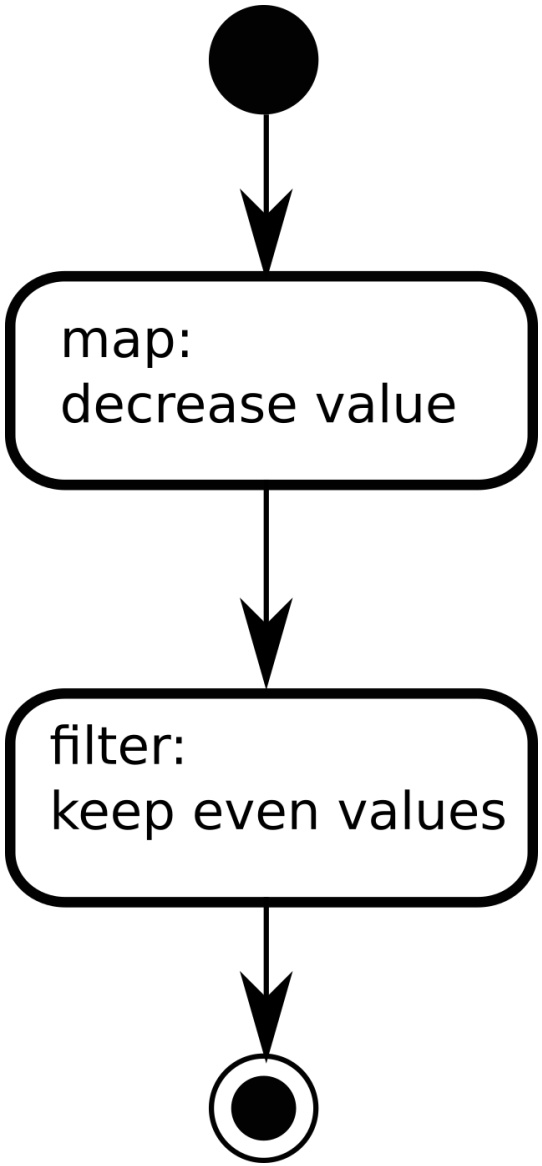
ReactiveX: Observable, Observer, Operator

- Observable: A stream of events
- Observer: Subscribes to Observable
- Operator: Applies transformations on items

ReactiveX: Marble Diagram



Hello World: Reactivity Diagram



Hello World: The Code

```
import rx
import rx.operators as ops
```

Hello World: The Code

```
import rx
import rx.operators as ops

source = rx.from_iterable([1, 2, 3, 4])
```

Hello World: The Code

```
import rx
import rx.operators as ops

source = rx.from_iterable([1, 2, 3, 4])

source.pipe(
    ops.map(lambda i: i - 1),
    ops.filter(lambda i: i % 2 == 0),
)
```

Hello World: The Code

```
import rx
import rx.operators as ops

source = rx.from_iterable([1, 2, 3, 4])

source.pipe(
    ops.map(lambda i: i - 1),
    ops.filter(lambda i: i % 2 == 0),
).subscribe(
    on_next=lambda i: print("on_next: {}".format(i)),
    on_completed=lambda: print("on_completed"),
    on_error=lambda e: print("on_error: {}".format(e))
)
```

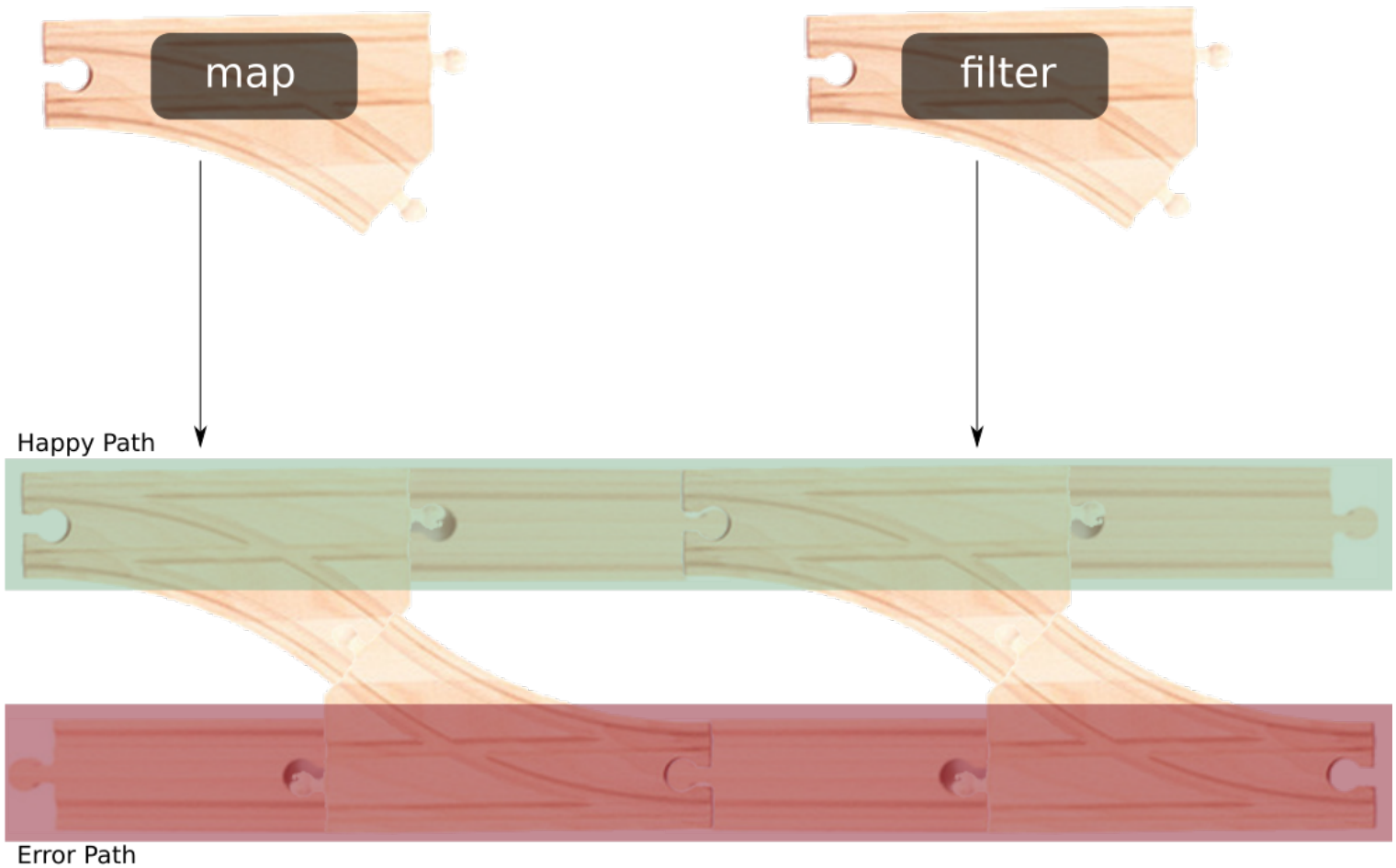
Hello World: The Code

```
import rx
import rx.operators as ops

source = rx.from_iterable([1, 2, 3, 4])

disposable = source.pipe(
    ops.map(lambda i: i - 1),
    ops.filter(lambda i: i % 2 == 0),
).subscribe(
    on_next=lambda i: print("on_next: {}".format(i)),
    on_completed=lambda: print("on_completed"),
    on_error=lambda e: print("on_error: {}".format(e))
)

disposable.dispose()
print("Done!")
```

Hello World: Result

```
$ python demo1.py
on_next: 0
on_next: 2
on_completed
Done!
```

Error Management



Error Management

```
source = rx.from_iterable([1, "foo" , 3, 4])
```

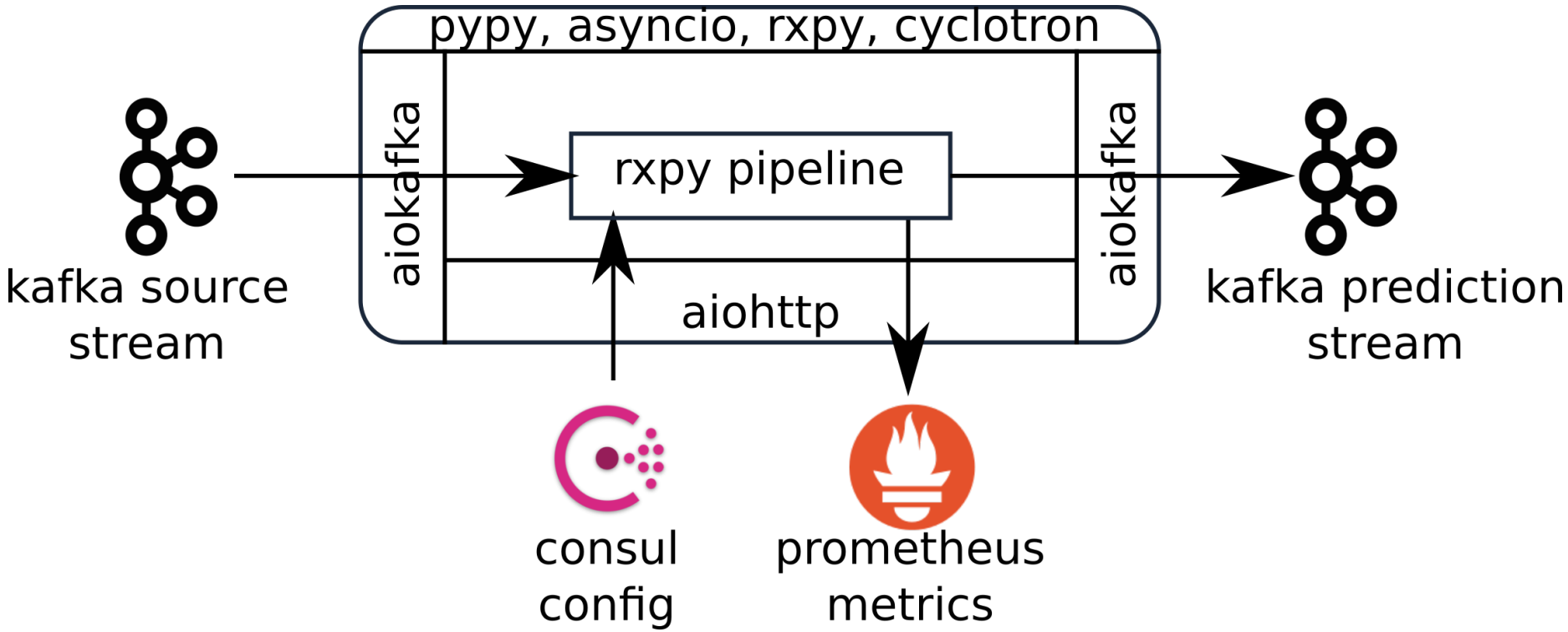
Error Management

```
$ python demo1_error.py
on_next: 0
on_error: unsupported operand type(s) for -:
          'str' and 'int'
Done!
```

Concurrency

- Schedulers for various environments
- CPU Concurrency: Thread pool, new thread
- IO Concurrency: AsyncIO, Twisted, GEvent, Eventlet
- Frameworks mainloops: Qt, GTK, PyGame

RxPY in Real Life



Going Further

- PyPI: pip3 install rx
- Documentation: https://rxpy.readthedocs.io
- ReactiveX: http://reactivex.io
- Cyclotron: `https://github.com/mainro/cyclotron-py`
- Railway Oriented Programming, by Scott Wlaschin

Thank you !

- Blog: `https://blog.oakbits.com`
- GitHub: `https://github.com/mainro`
- Twitter: `https://twitter.com/_mainro_`