Reactive Python for Data Science

- https://learning.oreilly.com/videos/reactive-python-for/9781491979006

Part 1: Introduction

Part 2: Why Reactive Programming?

Reactive programming is a radically effective approach to compose data as queryable, live-time streams

Not only can you concisely wrangle and analyze static snapshots of data, but also real-time infinite data (e.g. stock quotes, Twitter streams)

Part 3: Thinking Reactively

There is growing interest in wrangling and analyzing live data streams, and RxPy is a lightweight library that achieves this

Data does not have to be modeled as something static, but rather something that is constantly in motion

Data analysis professionals should strive to create code that can plug into existing systems easily, as well as be reused and evolve with the business

Part 4: The Observable

Part 5: Operators

Reduce and Scan Using RxPy

Part 6: Combining Observables

snippet.python

```
import rx
from rx import operators as ops

s1 = rx.from_(['A', 'B', 'C', 'D', 'E', 'F'])
s2 = rx.range(1,7)
```

```
stream = rx.zip(s1, s2)
stream.subscribe(lambda s: print(s[0], s[1]))
```

snippet.python

```
letters = rx.from_(['Alpha', 'Beta', 'Gamma', 'Delta', 'Epsilon'])
intervals = rx.interval(1)

stream = rx.zip(letters, intervals)
stream.subscribe(lambda s: print(s))

input('Press any key to quit')
```

Using Group By in RxPy

snippet.python

```
items = ['Alpha', 'Beta', 'Gamma', 'Delta', 'Epsilon']

source = rx.from_(items)
stream = source.pipe(
    ops.group_by(lambda s: len(s)),
    ops.flat_map(lambda g: g.pipe(
        ops.to_list()
    )),
)

stream.subscribe(lambda s: print(s))
```

Part 7: Reading and Analyzing Data

Part 8: Hot Observables

Part 9: Concurrency

Part 10: Wrap-up

관련 문서

- Python