FUNCTIONAL PROGRAMMING IN PYTHON

ABOUT ME

DATA SCIENTIST @ IDALAB (MAINLY PYTHON)
USED RUBY, JS, PYTHON, HASKELL FOR NONTRIVIAL PROJECTS
PLAYED WITH CLOJURE, SCALA, ERLANG, ELIXIR

HTTP://KIRELABS.ORG/FUN-JS

ABOUT THIS TALK

- NOT A MOTIVATION OF FUNCTIONAL PROGRAMMING
- HOW CAN FP BY USED IN PYTHON

DISCLAIMER

THERE SHOULD BE ONE
– AND PREFERABLY ONLY ONE –
OBVIOUS WAY TO DO IT.
– PEP 20 – THE ZEN OF PYTHON

DISCLAIMER (CONT)

THE FATE OF `reduce()` IN PYTHON 3000

NOT HAVING THE CHOICE STREAMLINES THE THOUGHT PROCESS
- GUIDO VAN ROSSUM

FUNCTIONAL PROGRAMMING (IN PYTHON)

- FIRST CLASS FUNCTIONS
- HIGHER ORDER FUNCTIONS
- PURITY
- IMMUTABILITY
- COMPOSITION
- PARTIAL APPLICATION & CURRYING
- RECURSION

FUNCTIONAL PROGRAMMING (IN PYTHON)

- FIRST CLASS FUNCTIONS
- HIGHER ORDER FUNCTIONS
- PURITY
- ~~IMMUTABILITY~~ (NOT TODAY)
- COMPOSITION
- PARTIAL APPLICATION & CURRYING
- ~~RECURSION~~ (NEITHER)

PURITY

FUNCTIONS WITHOUT SIDE-EFFECTS

```
def add(a, b):
    return a + b

additions_made = 0
def add(a, b):
    additions_made += 1
    return a + b
```

FIRST CLASS FUNCTIONS

```
def add(a, b):
    return a + b

add_function = add

add = lambda a,b: a + b
```

HIGHER ORDER FUNCTIONS

```
def timer(fn):
    def timed(*args, **kwargs):
        t = time()
        fn(*args, *kwargs)
        print "took {time}".format(time=time()-t)

    return timed

def compute():
    #…

timed_compute = timer(compute)
timed_compute()
```

DECORATORS

```
@timer
def compute():
    sleep(1)

compute()
```

PARTIAL FUNCTION APPLICATION

```
def add1(num):
    return add(1, num)
add1(1)

# simpler
from functools import partial
add1 = partial(add, 1)
add1(1)
```

CURRYING

[...] TRANSFORMING A FUNCTION THAT TAKES MULTIPLE ARGUMENTS IN SUCH A WAY THAT IT CAN BE CALLED AS A CHAIN OF FUNCTIONS, EACH WITH A SINGLE ARGUMENT (PARTIAL APPLICATION)

– WIKIPEDIA

CURRYING

```
def curried_add(a):
    def inner(b):
        return add(a,b)

add(1)     # => <function …>
add(1)(1) # => 2
```

INTERLUDE: CLOSURES

```
def curried_add(a):
    def inner(b):
        return add(a,b)

add(1)    # => <function …>
add(1)(1) # => 2
```

CURRYING EXAMPLE FROM THE STDLIB

```
 from operator import itemgetter, attrgetter, methodcaller

obj.method()

from operator import methodcaller
methodcaller("method")(obj)
```

FUNCTIONAL COLLECTION TRANSFORMATIONS

MAP

```
map(f, iter)

[f(el) for el in seq]
```

Filter

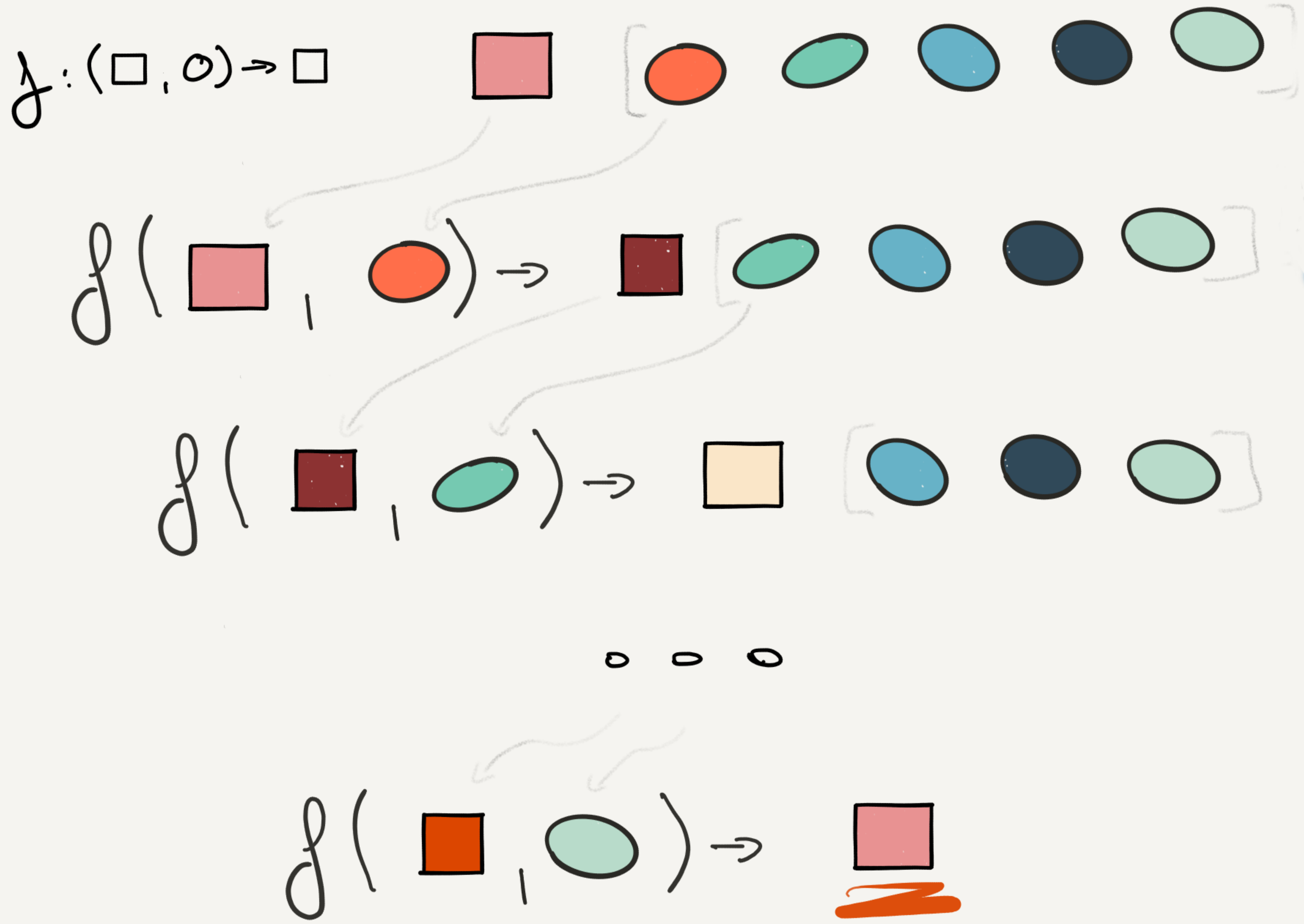
```
filter(p, seq)

[el for el in seq if p(el)]
```

REDUCE

```python
from functools import reduce
reduce(f, seq, initial)

result = initial
for el in seq:
    result = f(result, el)
```


f : (□, ○) → □

FUNCTION COMPOSITION

```
[f(x) for x in seq if p(x)]

map(f, filter(p, seq))

from toolz.curried import compose, map, filter
compute = compose(map(f), filter(p))
compute(seq)
```

EXAMPLE: A BAD CSV PARSER (1/3)

```
csv = """firstName;lastName
Jim;Drake
Ben;James
Tim;Banes"""

target = [{'firstName': 'Jim', 'lastName': 'Drake'},
          {'firstName': 'Ben', 'lastName': 'James'},
          {'firstName': 'Tim', 'lastName': 'Banes'}]
```

EXAMPLE: IMPERATIVE PYTHON (2/3)

```python
lines = csv.split("\n")
matrix = [line.split(';') for line in lines]
header = matrix.pop(0)
records = []
for row in matrix:
    record = {}
    for index, key in enumerate(header):
        record[key] = row[index]
    records.append(record)
```

EXAMPLE: FUNCTIONAL PYTHON (3/3)

```
from toolz.curried import compose, map
from functools import partial
from operator import methodcaller

split = partial(methodcaller, 'split')
split_lines = split("\n")
split_fields = split(';')
dict_from_keys_vals = compose(dict, zip)
csv_to_matrix = compose(map(split_fields), split_lines)

matrix = csv_to_matrix(csv)
keys = next(matrix)
records = map(partial(dict_from_keys_vals, keys), matrix)
```

PYSPARK

```
docker run --rm -v ${PWD}:/home/jovyan/work -p 8888:8888 jupyter/pyspark-notebook
```

```
def sample(p):
    x, y = random(), random()
    return 1 if x*x + y*y < 1 else 0

count = sc.parallelize(range(0, NUM_SAMPLES)) \
    .map(sample) \
    .reduce(lambda a, b: a + b)
print("Pi is roughly %f" % (4.0 * count / NUM_SAMPLES))
```

WHATS MISSING IN PYTHON (OR WHAT I AM MISSING)

- MORE LIST FUNCTIONS
- NICER LAMBDA SYNTAX
- AUTOMATIC CURRYING, COMPOSITION SYNTAX
- ADTS (SUM TYPES)*
- PATTERN MATCHING

* POSSIBLE BUT UGLY HTTP://STUPIDPYTHONIDEAS.BLOGSPOT.DE/2014/08/ADTS-FOR-PYTHON.HTML

FUNCTIONAL LIBRARIES

(MORE LIST FUNCTIONS)

- HTTP://TOOLZ.READTHEDOCS.IO/EN/LATEST/
  - HTTPS://GITHUB.COM/KACHAYEV/FN.PY
- HTTP://PEDRORODRIGUEZ.IO/PYFUNCTIONAL/

NICER LAMBDA SYNTAX

```
map(lambda x: x**2, range(5)) # => [0, 1, 4, 9, 16]

from fn import _
map(_**2, range(5)) # => [0, 1, 4, 9, 16]
```

MAIN TAKEAWAYS

- FP IS POSSIBLE IN PYTHON (TO A DEGREE)
- SMALL COMPOSABLE FUNCTIONS ARE GOOD
- FP == BUILD GENERAL TOOLS AND COMPOSE THEM

OTHER INTERESTING STUFF

- SEPARATION OF PURE CODE AND SIDEEFFECTS: HTTPS://PYPI.PYTHON.ORG/PYPI/EFFECT/
- PERSISTENT IMMUTABLE DATA STRUCTURES HTTPS://PYPI.PYTHON.ORG/PYPI/PYRSISTENT/
- HTTPS://DOCS.PYTHON.ORG/3/HOWTO/FUNCTIONAL.HTML

OTHER TALKS (WHERE I HAVE STOLEN MATERIAL)

- HTTP://KACHAYEV.GITHUB.IO/TALKS/UAPYCON2012/
- HTTPS://VIMEO.COM/80096814
- HTTP://KIRELABS.ORG/FUN-JS

MORE FP?

- SICP (HTTP://DEPTINFO.UNICE.FR/~ROY/SICP.PDF)
- HTTP://LEARNYOUAHASKELL.COM/
- REAL WORLD HASKELL (HTTP://BOOK.REALWORLDHASKELL.ORG/READ/)

THANK YOU

DANIEL KIRSCH
DANIEL.KIRSCH@IDALAB.DE
@KIREL
HTTPS://GITHUB.COM/KIREL/FUNCTIONAL-PYTHON

ida
lab.