Denis Shevchenko

- ☑ Haskell Developer at IOHK
- ☑ Co-founder of ruHaskell
- ☑ Code since 2005

@dshevchenko_biz

pyerevan

August 2019

A practical introduction to FP in Python

FP?

FP — **rewrite** your code!

OOP's for

```
names = ['David', 'Karen', 'Ruben']
secrets = []
for i in range(len(names)):
    secrets.append(hash(names[i]))
```

FP's map

```
names = ['David', 'Karen', 'Ruben']
secrets = map(hash, names)
```

rethink

FP — ~~rewrite~~ your code!

```
def inc():
    global a
    a += 1

def check():
    if a > 12:
        print("Enough!")
```

```
a = 3

inc()
check()
```

OOP → FP

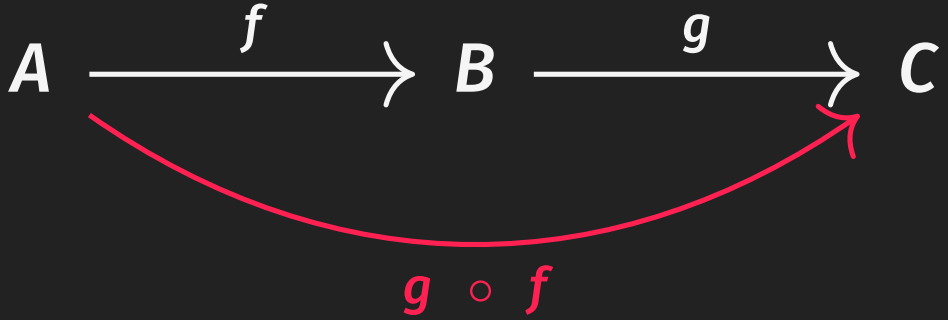
Reduce mutable state!

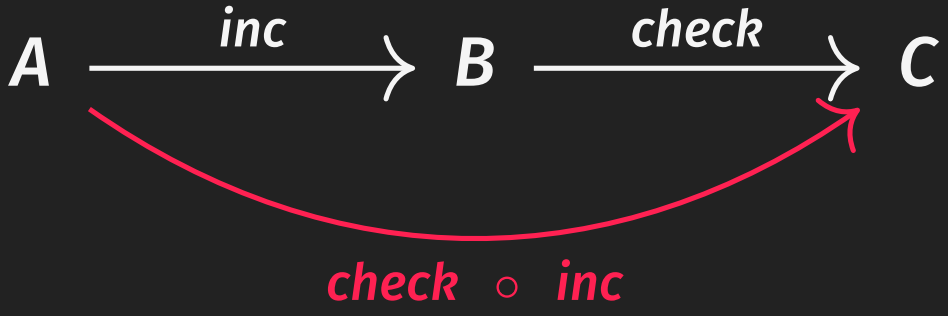
```
def inc(a):
    return(a + 1)

def check(a):
    if a > 12:
        return "Enough!"
```

```
print(check(inc(3)))
```

Compose your functions


A
f
B
g
C
g ∘ f


A
inc
B
check
C
check ∘ inc

```
from infix import shift_infix as infix

@infix
def then(f, g):
    return lambda x: g(f(x))
```

```
do = inc <<then>> check
```

```
do = inc <<then>> check

do_all = do <<then>> print
```

```
do = inc <<then>> check

do_all = do <<then>> print

...

do_all(3)
```

Use fair constants

```
PI = 3.14159265

...

def calcCircLength(r: float) -> float:
    return 2 * PI * r
```

```
import cmath

...

def circle_length(r: float) -> float:
    return 2 * cmath.pi * r
```

```python
import cmath

cmath.pi = 3.129 # Bad joke!

def circle_length(r: float) -> float:
    return 2 * cmath.pi * r
```

```
def pi() -> float:
    return 3.14159265

...

def circle_length(r: float) -> float:
    return 2 * pi() * r
```

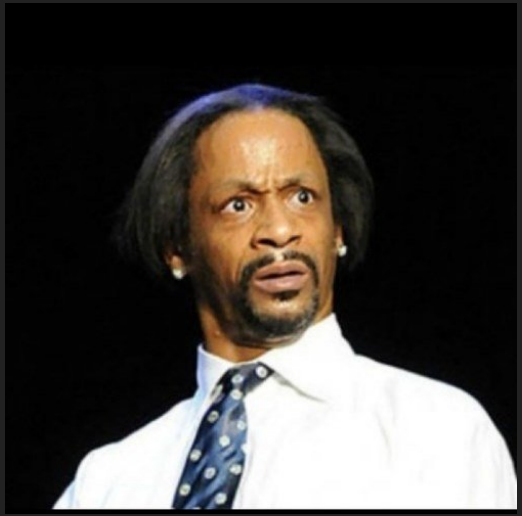
Math?!

$$M = (S, \bullet, e)$$

I $S = \{\ldots\}$

II $\bullet : S \times S \to S$

III $\exists e \in S \mid e \bullet a = a \bullet e = a$



```
https_prefix = "https://"
base_url     = "istc.am"
secure_url   = https_prefix + base_url
```

```
https_prefix = "https://"
base_url     = "istc.am"
secure_url   = https_prefix + base_url
```

Monoid — idea!

```
https_prefix = "https://"
base_url     = "istc.am"
secure_url   = https_prefix + base_url
```

Take **existent**, create **new**.

So...

Reduce mutable state!

I **Compose** your functions

II Use **fair** constants

III Don’t be **afraid** of math

What’s **next**?

«A practical introduction
to functional programming»

bit.do/**fpPython**

«FP in Python: Lessons from Haskell and Closure»

bit.do/**fpLessons**

Thank you!

Questions?