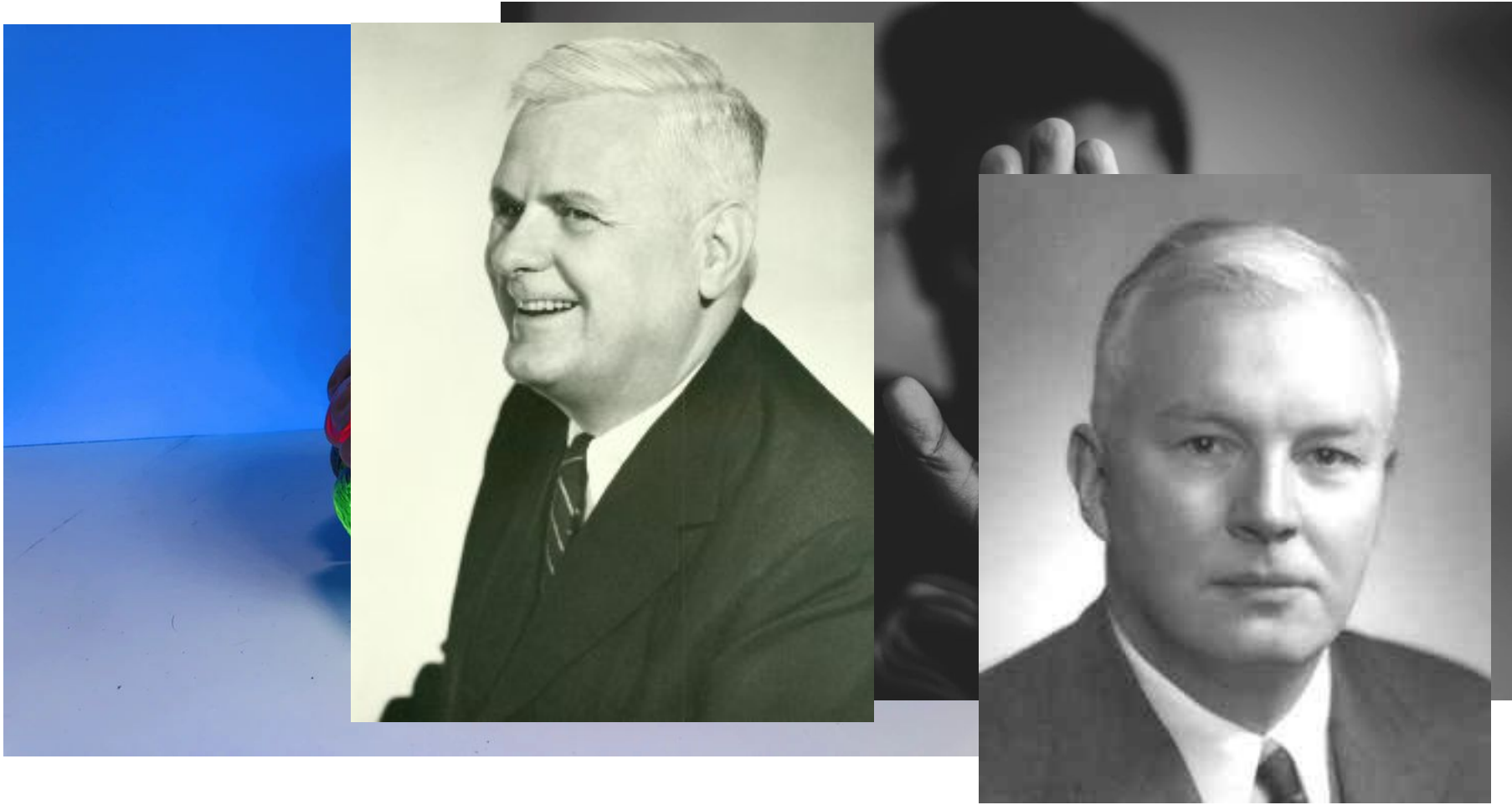
Functional Design in Python

Jerzy Kowalski

What you'll get:

- Fundamental concepts of functional programming
- Patterns and tricks
- Practice example

Functional?



```
map(foo, [“some”, “random”, “list”])
```
~~`[foo(item) for item in [“some”, “random”, “list”]]`~~

```
filter(is_prime, [3, 0, 11])
```
~~`[number for number in [3, 0, 11] if is_prime(number)]`~~

```
lambda x: x + 1
```


QUALITY
BULLSHIT
$2 A BAG

1. **Functions are first-class citizens**
2. Write pure functions

Functions are `objects`...

...so you can pass them to other functions...

```
>>> sorted([“sort”, “me”, “nicely”], key=len)
[“me”, “sort”, “nicely”]

>>> sorted([“too”, “much”, “coffee”], key=lambda word: word.count(“o”)])
[“much”, “coffee”, “too”]
```

Functions are objects...

...and return them from other functions...

```
@staticmethod
def foo():
  return “Foo!”


def exchange(rate):
  return lambda value: value * rate

dollar_to_euro = exchange(0.91)
dollar_to_yuan = exchange(7.12)
```

Functions are objects...

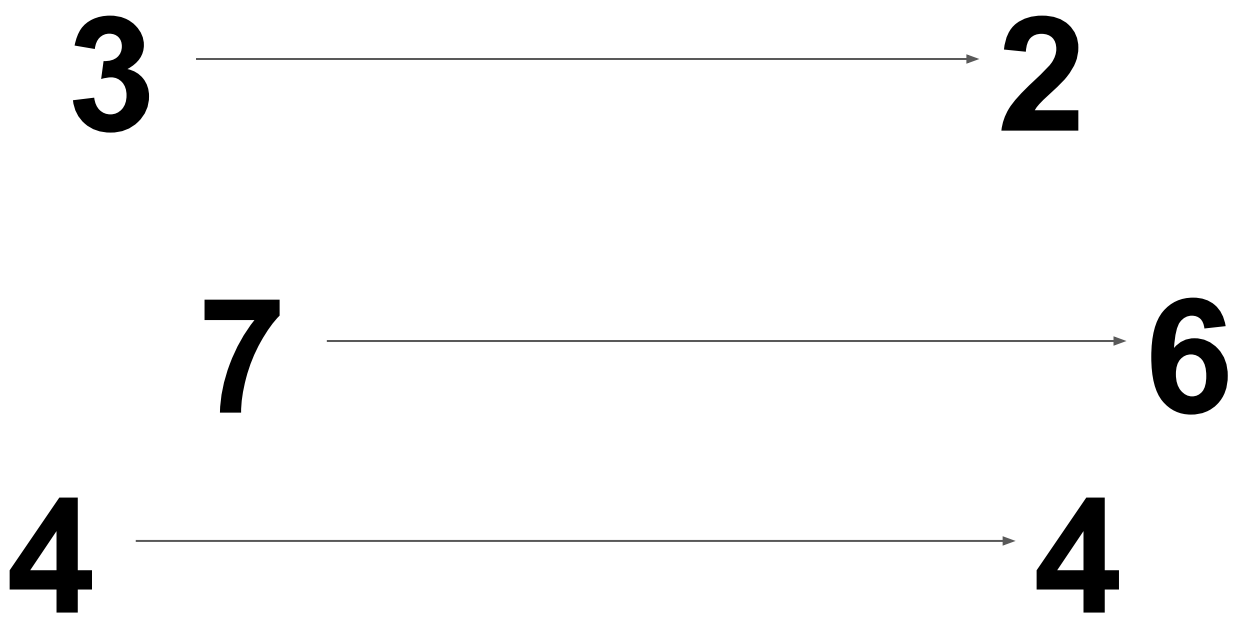
...and assign them

```
def foo(): pass
def bar(): pass

actions = {
  “foo”: foo
  “bar”: bar
}

actions[“bar”]()
```

1. Functions are first-class citizens
2. **Write pure functions**

Pure functions

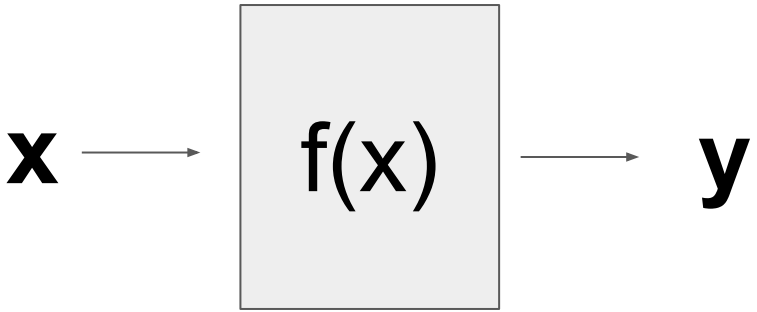
$$f(x) = x - 1$$



Pure functions

Look at input

Produce output



No mutating!

No side effects!

Pure functions

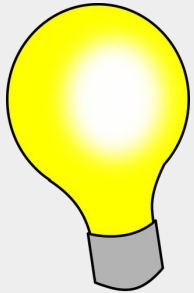
Pure functions

Testability!

```
def test_foo():
    result = foo()
    expected = “foo”
    assert result == expected
```

Pure functions

Decide to write pure function



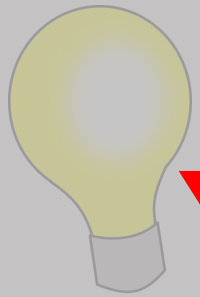
Turn requirements to unit tests

Write function

Celebrate your reliable code

Pure functions

Decide to write pure function



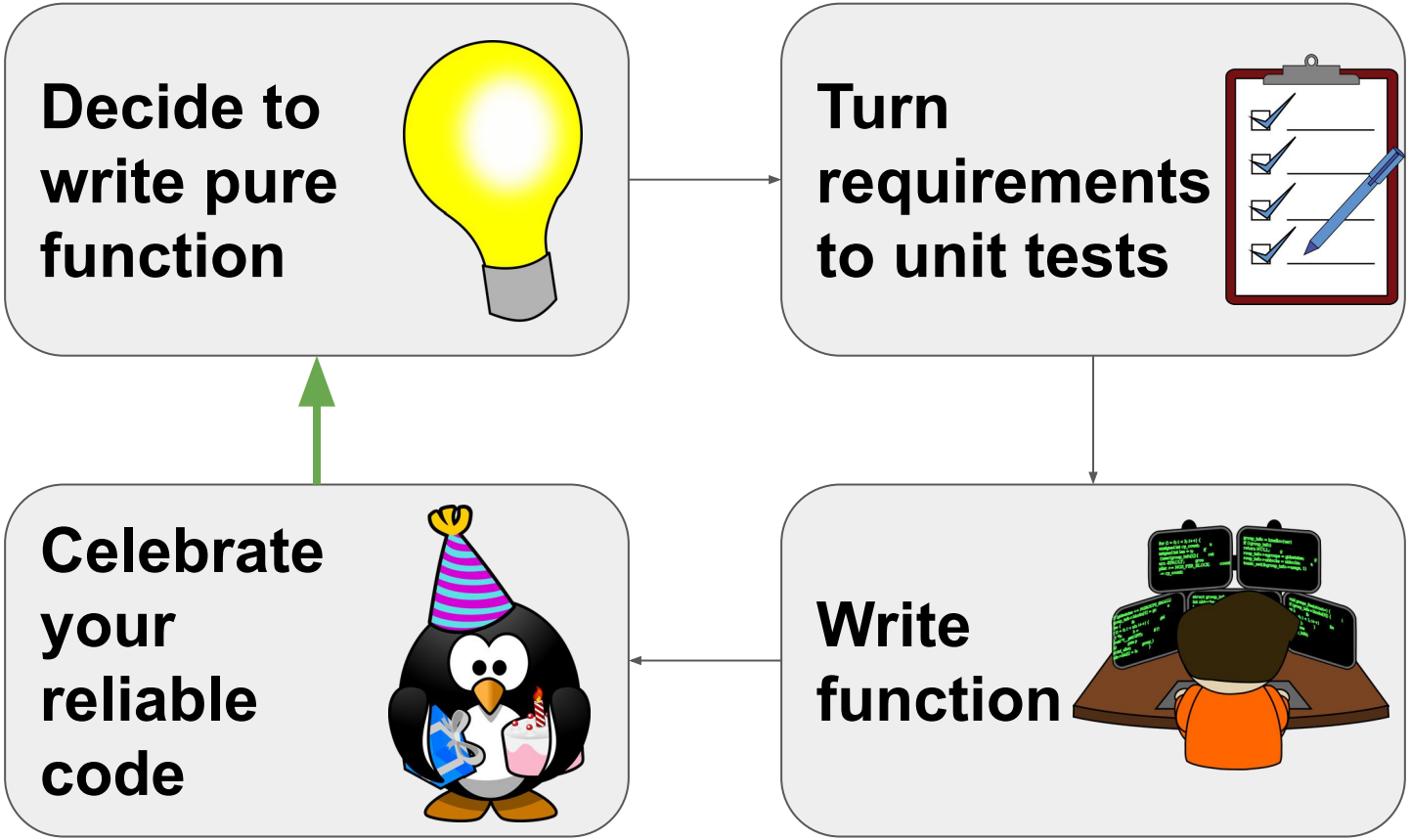
Turn requirements to unit tests

Write function

Celebrate your reliable code

YOU WANT MORE!!!

Pure functions



Pure functions

Predictability

```
foo = Foo()
bar = do_some_magic(foo)
# foo ???
```

Pure functions

OK, but seriously what about side effects?

Isolate them!

Pure functions

```
HEADER = “FOO”

with open(“foo”) as foo_file:
  foo_data = foo_file.read()

if not foo_data.startswith(HEADER):
  foo_data = f”{HEADER_START} {header}
```

Pure functions

```
HEADER = “FOO”
```

```
with open(“foo”) as foo_file:
  foo_data = foo_file.read()
```

Two separate functions!

```
if not foo_data.startswith(HEADER):
  foo_data = f”{HEADER_START} {header}
```

“We’re doing it in hope that it will make our programs easier to write, easier to understand”

Russ Olsen

Functional Programming in 40 minutes

Let’s code!

Our task

Your client wants online shop with potatoes.

His potato suppliers provide APIs to check availability, price etc.

These APIs are inconsistent and messy.

We want to make our front-end colleagues life easier.

Our task

We will use functional attitude we've just learned.

+ as a bonus we will learn about **functional composition**

Let's get our hands dirty!

github.com/jurekkow/functional-potatoes

More stuff:

https://www.youtube.com/watch?v=0if71HOyVjY

https://medium.com/javascript-scene/mocking-is-a-code-smell-944a70c90a6a

https://docs.python.org/3/howto/functional.html

QA

Thank you!

Jerzy Kowalski