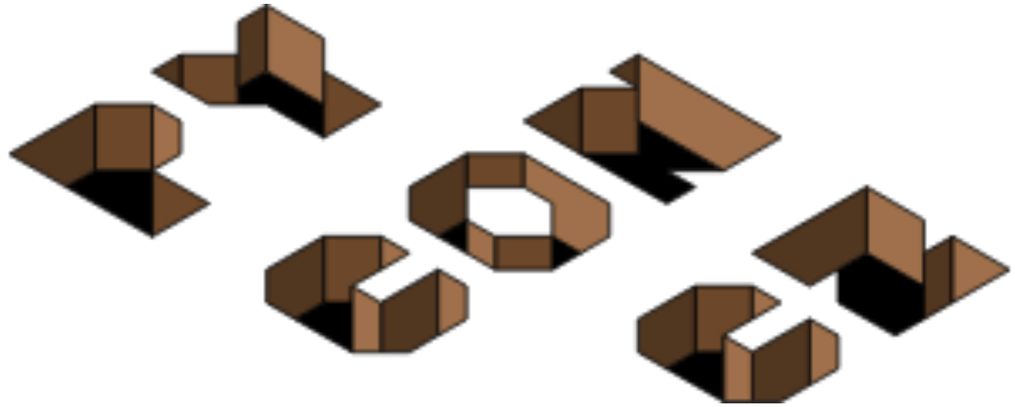
Iterators for Curious Minds

Friday, 14 June, 15:40
in Ballroom

Miloslav Pojman
@MiloslavPojman



Iterators for Curious Minds

Miloslav Pojman
@MiloslavPojman

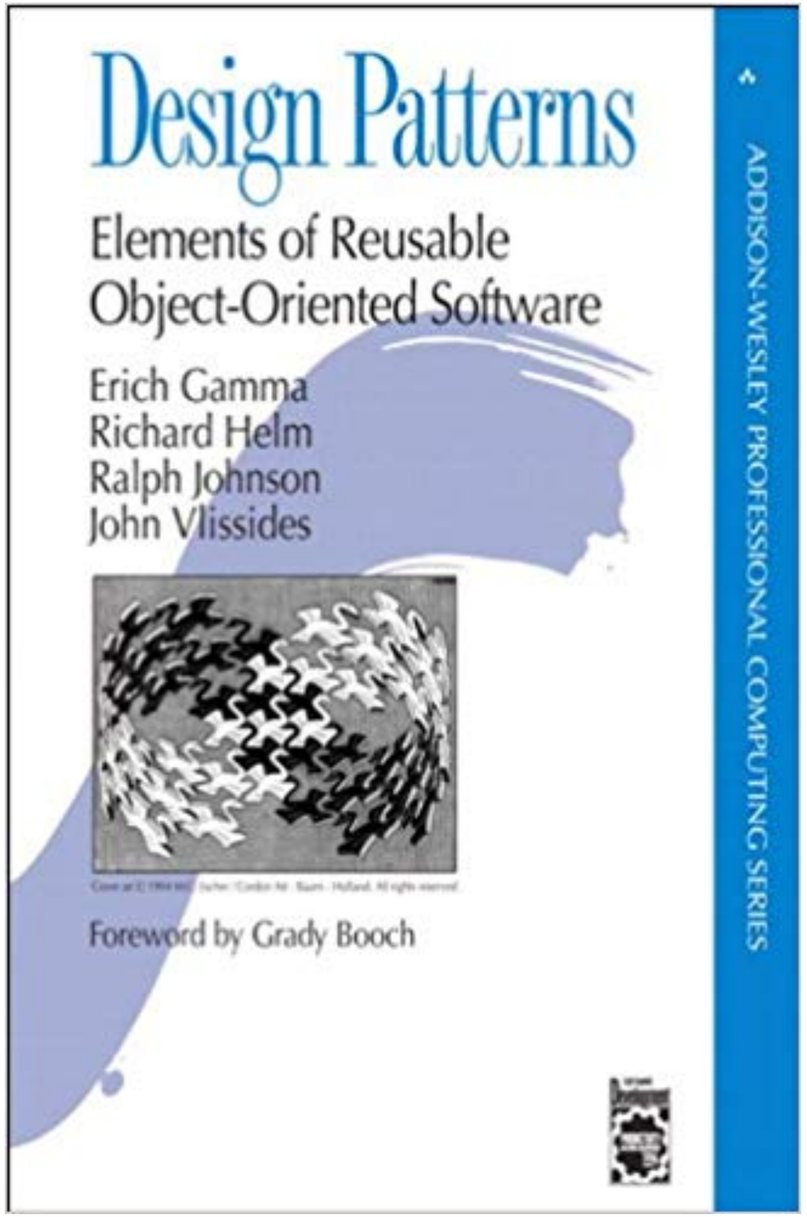
```
for i in range(10):
    print(i)
```

```
iter(v)
```

```
if iter(v) is v:
    ...
```

Gang of Four

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

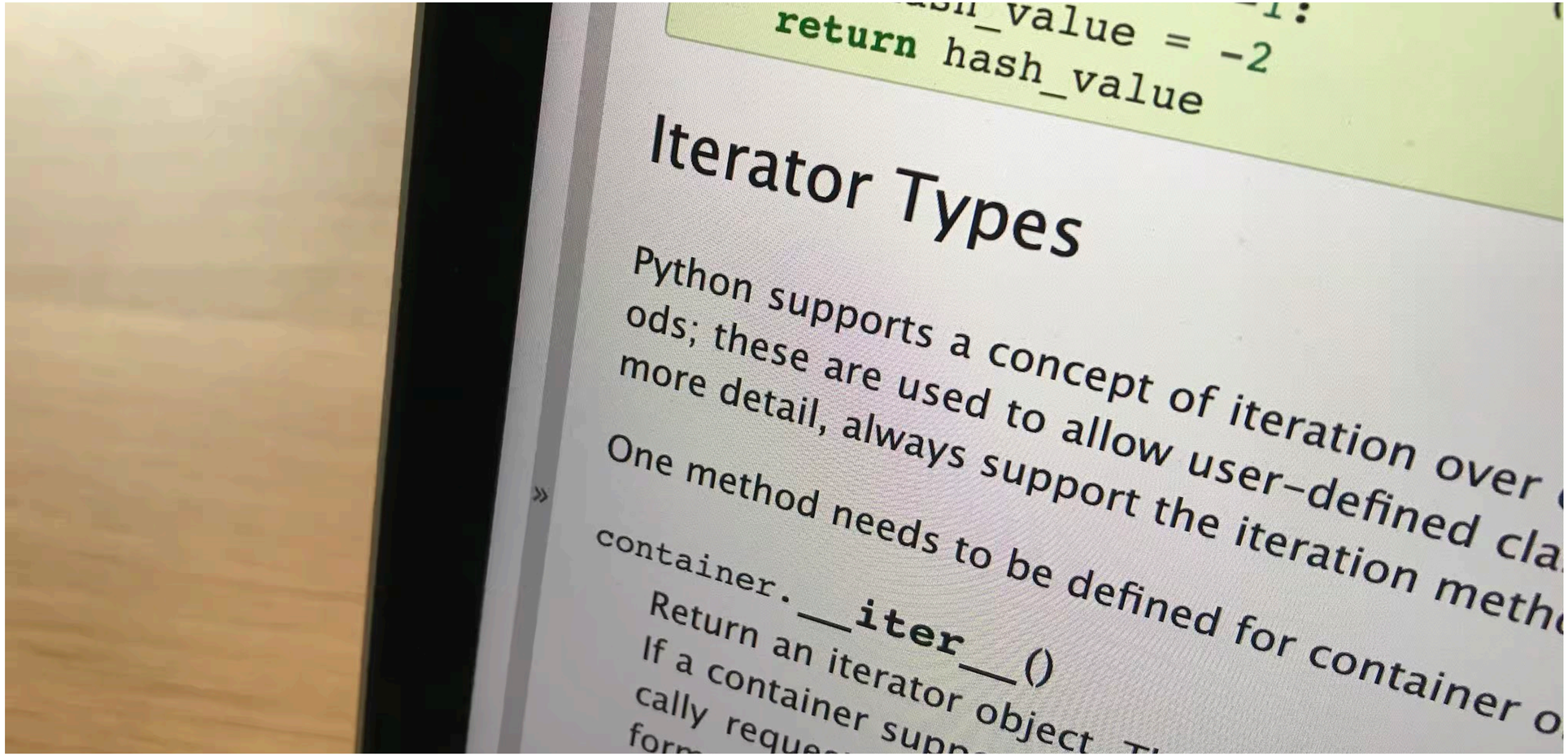


GangOfFour

Martin Fowler

In my view the Gang of Four is the best book ever written on object-oriented design - possibly of any style of design. This book has been enormously influential on the software industry - just look at the Java and .NET libraries which are crawling with GOF patterns.

https://martinfowler.com/bliki/GangOfFour.html



https://docs.python.org/3/library/stdtypes.html#iterator-types

Language Comparison

Subjective, biased, and unfair

C++

```
map<std::string, std::string>::iterator it;
for (it = obj.begin(); it != obj.end(); it++) {
    std::cout << it->first  // key
              << ": "
              << it->second // value
              << std::endl;
}
```

JavaScript

```
for (var key in obj) {
    if (obj.hasOwnProperty(key)) {
        alert(key + ": " + obj[key])
    }
}
```

Python

```python
for key, value in obj.items():
    print(f"{key}: {value}")
```

Iterations of Evolution: The Unauthorized Biography of the For-Loop

4,5 tis. zhlédnutí 132 0 SDÍLET ULOŽIT •••

David Beazley
Publikováno 16. 12. 2017

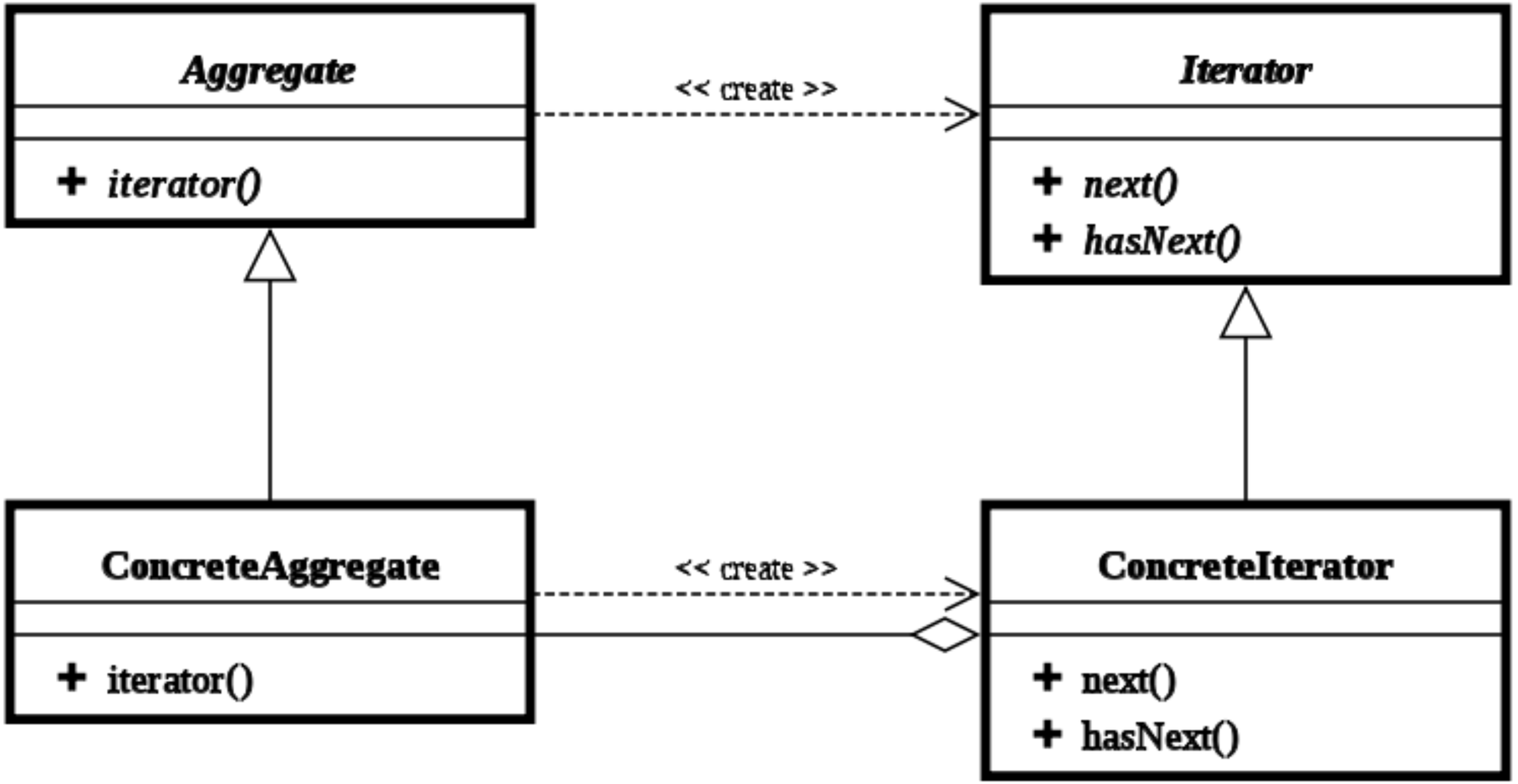
ODEBÍRAT 7,3 TIS.

Keynote talk from PyCon Pakistan. December 16, 2017. Remote presentation from Chicago. I discuss the early evolution of the for-loop, the iteration protocol, and developments leading to generator functions and

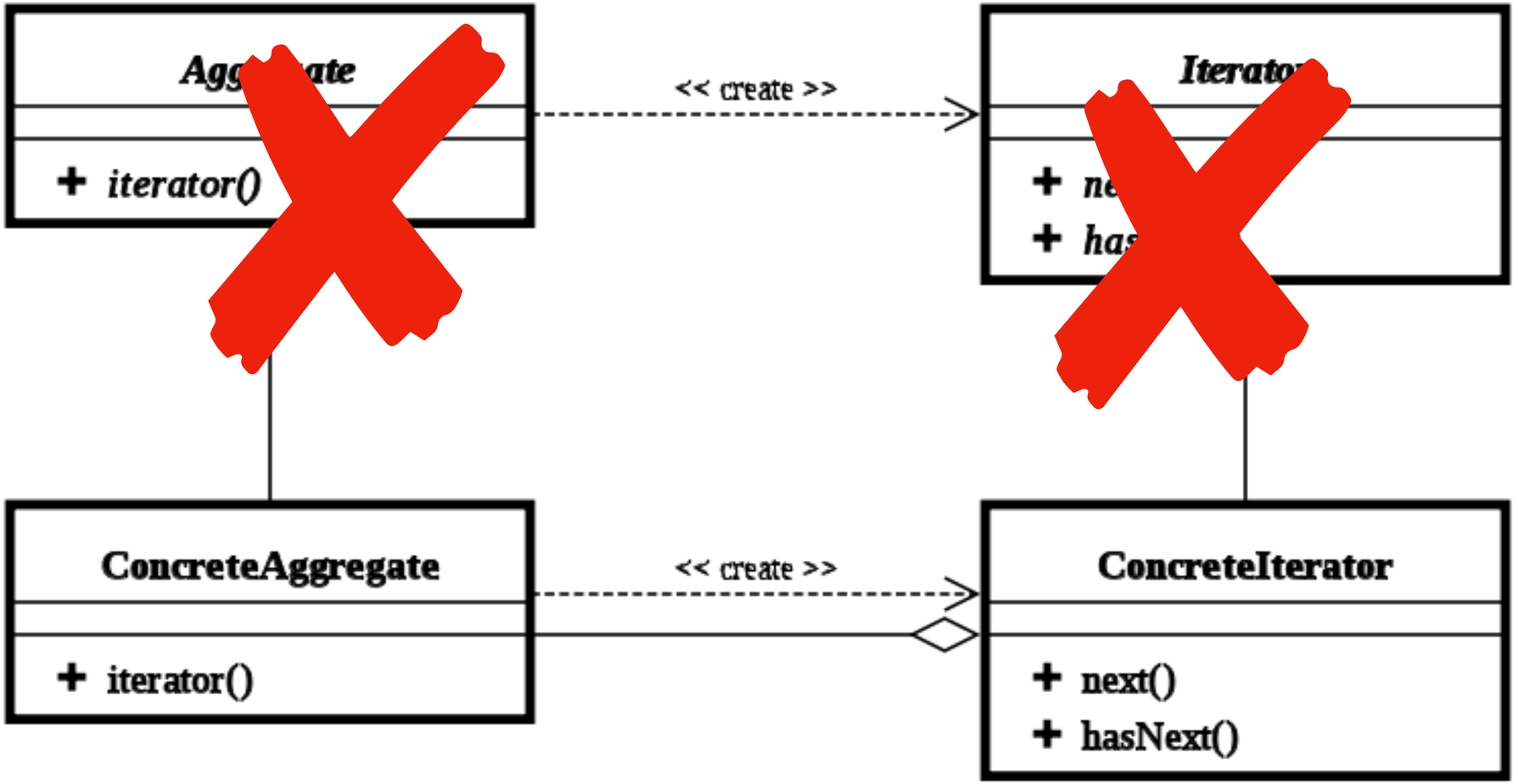
https://youtu.be/2AXuhgid7E4

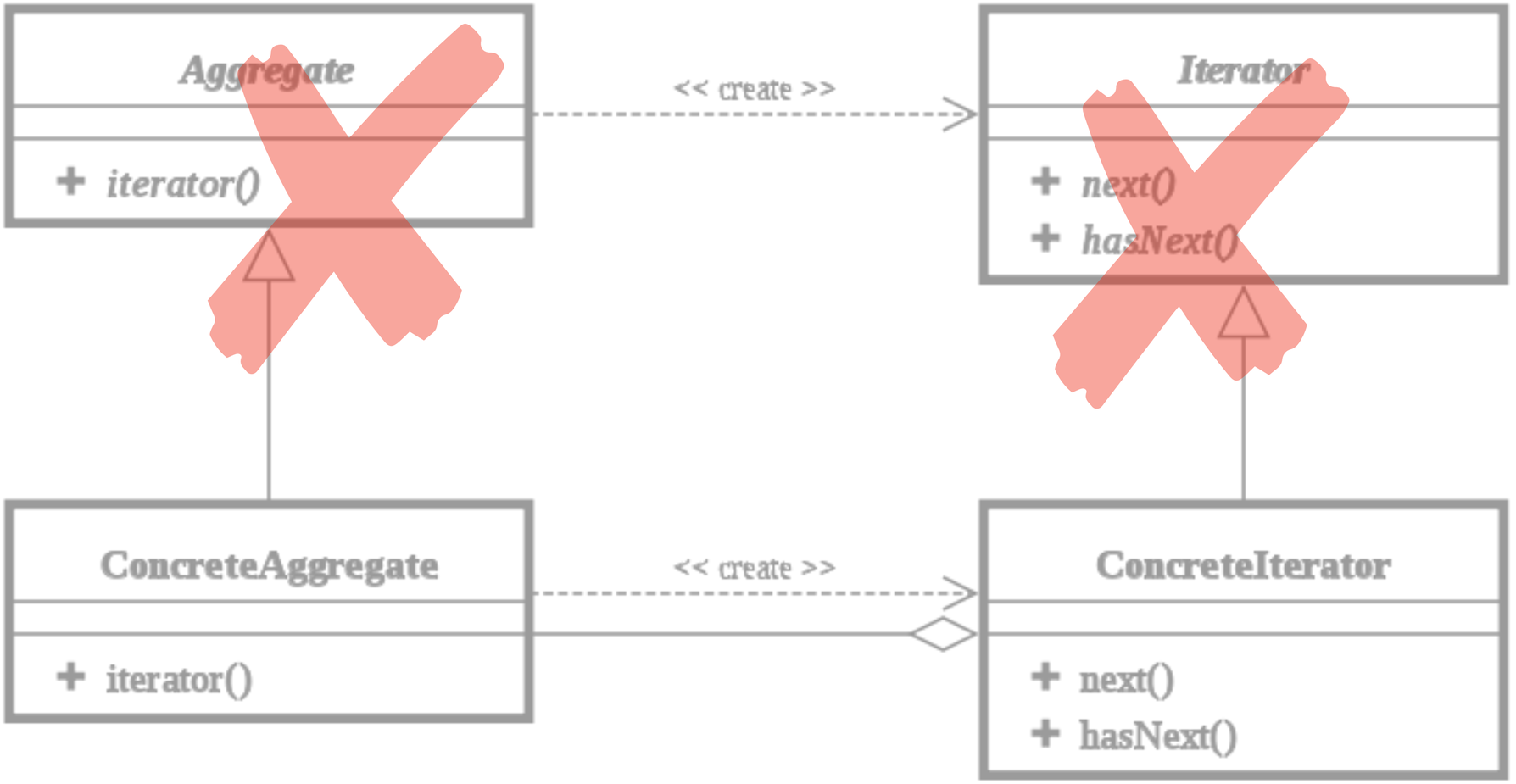
Iterator Pattern

Lesson from software engineering



https://commons.wikimedia.org/wiki/File:Iterator_UML_class_diagram.svg


+ iterator()
<< create >>
+ next()
+ has
ConcreteAggregate
+ iterator()
<< create >>
ConcreteIterator
+ next()
+ hasNext()



Container

Iterator

```
>>> it = iter(["Hi", "🐍"])
>>> next(it)
'Hi'
>>> next(it)
'🐍'
>>> next(it)
Traceback (most recent call last):
...
StopIteration
```

EAFP

“Easier to ask for forgiveness than permission.”

Photo by Paul Hanaoka on Unsplash


>>> help(iter)
>>> help(next)

Python data model

```
str(v); v.__str__()
len(v); v.__len__()
bool(v); v.__bool__()

iter(v); v.__iter__()
next(v); v.__next__()
```

dunder

An **alias** for double **underscore**, "__". Specially useful in **the Python** computer language, in which names like "__init__" are common. Coined as a speech shorthand.

*Eric: So **the problem** is initdata? (**the method** initdata())*
*John: Nope, dunder **init** dunder is where things start to go wrong... (__init__)*

#ampersand #at #python #underscore #punctuation #programming #computer #computer language

by ajaksu May 22, 2006

https://www.urbandictionary.com/define.php?term=dunder

Containers vs. Iterators

The tricky part

Separation of concerns

```
>>> items = ["🥚", "🐔"]
>>> iter(items)
<list_iterator object at ...>
>>> reversed(items)
<list_reverseiterator object at ...>
>>> itertools.cycle(items)
<itertools.cycle object at ...>
```

Statefulness

```
teams = "ABCD"

for t1 in teams:
    for t2 in teams:
        print(f"{t1}:{t2}")
```

Iterables

```
for item in range(10):
    ...
for item in iter(range(10)):
    ...
for item in reversed(range(10)):
    ...
```

Why does Java not allow foreach on iterators (only on iterables)? [duplicate]

67

Possible Duplicate:
Why is Java's Iterator not an Iterable?

Idiomatic way to use for-each loop given an iterator?

Can we use for-each loop for iterating the objects of Iterator type?

https://stackoverflow.com/questions/11216994/

```
>>> counter = itertools.count()
>>> iter(counter) is counter
True
>>> iter(iter(counter)) is counter
True
```

```
countdown = reversed(range(10))
for i in countdown:
    print(f"Dry run {i}")
for i in countdown:
    print(f"Live run {i}")
```

```
if iter(v) is v:
    v = list(v)
```

Files

```
>>> f = open('/etc/hosts')
>>> iter(f) is f
True
>>> iter(sys.stdin) is sys.stdin
True
```

Custom Iterables

Homework assignment

```
for date in DateRange(
    dt.date(2019, 6, 14),
    dt.date(2019, 6, 17),
):
    print(date)
```

```
class DateRange:
    def __init__(self, start, stop):
        self.start = start
        self.stop = stop

    def __iter__(self):
        return DateRangeIterator(self)
```

```
class DateRangeIterator:
    def __init__(self, date_range):
        self.date = date_range.start
        self.stop = date_range.stop

    def __iter__(self):
        return self

    def __next__(self):
        ...
```

```
def __next__(self):
    if self.date >= self.stop:
        raise StopIteration
    rv = self.date
    self.date += dt.timedelta(days=1)
    return rv
```

Generators

Do you have three hours for a workshop?

```
class DateRange:
    ...

    def __iter__(self):
        date = self.start
        while date < self.stop:
            yield date
            date += self.step
```

```
def date_range(start, stop):
    date = start
    while date < stop:
        yield date
        date += dt.timedelta(days=1)
```

```
def parse_log(f):
    rv = []
    for line in f:
        parts = line.strip().split("\t")
        _, level, message = parts
        if level in ("E", "W"):
            rv.append(message)
    return rv
```

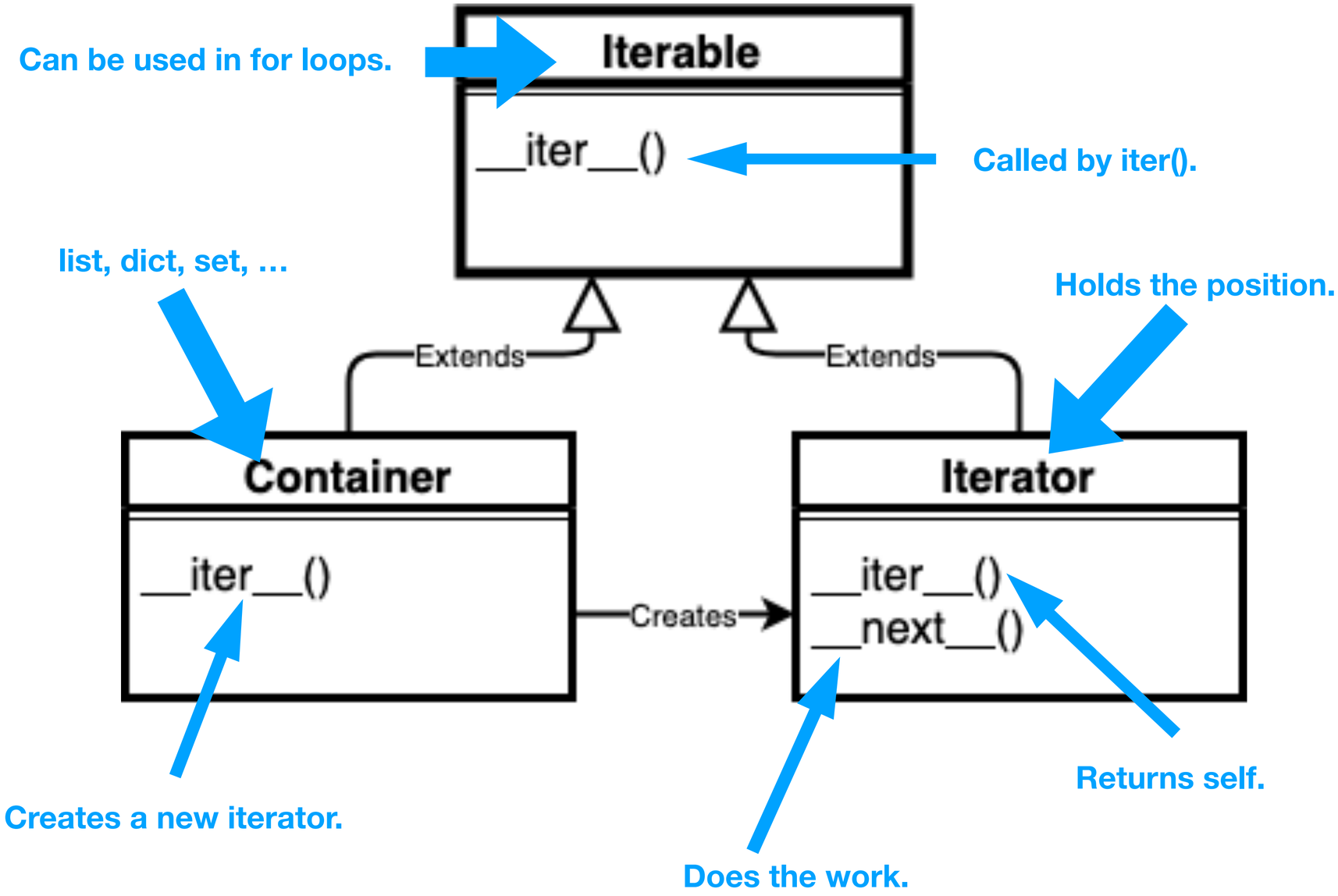
```
def parse_log(f):

    for line in f:
        parts = line.strip().split("\t")
        _, level, message = parts
        if level in ("E", "W"):
            yield message
```

```
def parse_log(f):
    lines = map(str.strip, f)
    parts = (line.split("\t") for line in lines)
    messages = (
        m for _, l, m in parts if l in ("E", "W")
    )
    return messages
```

Photo by tian kuan on Unsplash

Summary

The end!


Can be used in for loops.
Iterable
__iter__()
Called by iter().
list, dict, set, ...
Holds the position.
Extends
Extends
Container
__iter__()
Creates
Iterator
__iter__()
__next__()
Creates a new iterator.
Does the work.
Returns self.

iter(v) is not v

```
list(), tuple(), dict(), set()
str(), bytes()
sorted()
range()
collections.deque()
[f(v) for v in l if p(v)]
```

iter(v) is v

```
iter(items), reversed(items)
enumerate(items), zip(a, b)
map(f, items), filter(f, items)
itertools.chain(a, b)
open(path)
(f(v) for v in l if p(v))
```

Iterators for Curious Minds

Miloslav Pojman
@MiloslavPojman