Avoiding Unnecessary Work Using Lazy Evaluation



Gerald Britton
IT Solutions Designer

@GeraldBritton www.linkedin.com/in/geraldbritton

Eager vs Lazy Evaluation

Eager	Lazy
Statements evaluated when reached	Evaluation delayed until results needed
Variables assigned results immediately	Variables assigned when needed
Pay up front	Pay as you go
Eager evaluation can be costly	Lazy evaluation can save resources
Can be cheaper when most results used	Can cost more if most results used

Lazy Evaluation in Python

For/While loop

Version 1.4: `xrange()`

Version 1.5: `__getitem__()`

```
for i in xrange(10):
  print x[i]
```

Version 2.2: iterators, generators, yield

Version 3.0: `map` **and** `filter` **are lazy**

Version 3.3: `yield from`

Lazy evaluation by default

Demo

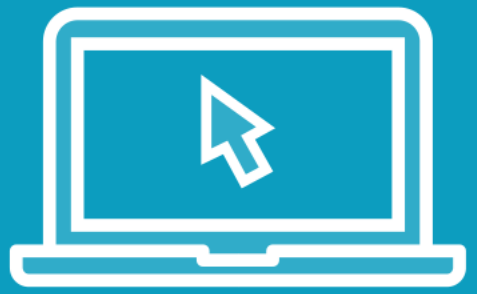


Get a subset of orders and order items

Computation may be expensive

Only requested now and then

Demo



Generator pros and cons

- Computing once can be cheaper
- `tee, islice`

Eager or lazy?

Refactoring helper functions

Summary



Lazy evaluation in Python

Compute results when needed

Saves on expensive operations

Opposite is eager evaluation

Lazy or eager?

Since 3.0 many built-in iterables are lazy

- `map, filter, functools, itertools`

Lazy evaluation fits well with generators