GENERATOR EXPRESSIONS

Comprehension Syntax

We already covered comprehension syntax when we studied list comprehensions

```
l = [i ** 2 for i in range(5)]
```

As well as more complicated syntax:

- if statements
- multiple nested loops
- nested comprehensions

```
[(i, j)
  for i in range(1, 6) if i%2==0
  for j in range(1, 6) if j%3==0]
```

```
[[i * j for j in range(5)] for i in range(5)]
```

Generator Expressions

Generator expressions use the same comprehension syntax → including nesting, if

but instead of using `[]`

we use `()`

```
[i ** 2 for i in range(5)]
```

```
(i ** 2 for i in range(5))
```

a `list` is returned

a `generator` is returned

evaluation is eager

evaluation is lazy

has local scope

has local scope

can access nonlocal
and global scopes

can access nonlocal
and global scopes

iterable

iterator

Resource Utilization

List comprehensions are eager

all objects are created right away

→ takes longer to create/return the list

→ iteration is faster (objects already created)

Generators are lazy

object creation is delayed until requested by `next()`

→ generator is created/returned immediately

→ iteration is slower (objects need to be created)

if you iterate through all the elements → time performance is about the same

if you do not iterate through all the elements → generator more efficient

→ entire collections is loaded into memory

→ only a single item is loaded at a time

in general, generators tend to have less memory overhead

Code Exercises