We learn how to generate values on demand.

1. Study session 7-9pm Sunday, exam on Tuesday starting at 5:45pm or 8pm.

2. Need accommodations? Contact Iris & take earlier exam.

3. Questions?

4. A generator is an object that constructs a (possibly infinite) stream of values on demand.

   (a) Whenever we write a function that mentions the yield keyword, the result of the function, when called, is a generator.

   ```python
def countTo(n):
    i = 1
    while i <= n:
        yield i
        i += 1
```

   (b) The generator object, g, can be asked to compute and return the next value in the sequence by calling next(g). This causes the generator to execute the function until a value is returned with yield:

   ```
>>> g = countTo(3)
>>> print(next(g))
1
>>> print(next(g))
2
>>> print(next(g))
3
>>> print(next(g))
Traceback (most recent call last):
  File ``<stdin>``', line 1, in <module>
StopIteration
```

   If you call next to get a value from a generator that has run dry, it raises a StopIteration exception.

   (c) This exception could be caught with a try-except statement, but a more efficient mechanism is to use a for loop:

   ```
>>> for v in countTo(10):
    print(v)
1
(d) Generators have the potential to generate an infinite number of values:

```python
def count(start = 0, step = 1):
    i = start
    while True:  # read: forever!
        yield i
        i += step
```

(e) How would you generate all the Fibonacci numbers? Assume the first two are 0 and 1.

```python
def fibo(a = 0, b = 1):
    yield a
    yield b
    while True:
        a, b = b, a+b
        yield b
```

(f) We can zip two or more iterable objects together, and construct a new iterable that generates tuples. The zipped stream runs dry when any of the component streams run dry:

```python
# print the first 10 fibonacci numbers:
for (i, f) in zip(range(10),fibo()):
    print("{}: {}").format(i,f))
```

(g) How would you generate all prime numbers? Here's a start:

```python
def isPrime(n):
    if n < 2:
        return False
    f = 2
    while f*f <= n:
        if n%f == 0:
            return False
        f += 1
    return True
```