

def

def

for in

yiielldi

3 # Doesn't start the function!

#=><class 'generator'>

=> 0

=> 1

#

=>

2

#raises StopIteration

Simple Generator

```
def generate_ints(n):  
    for i in range(n):  
        yield i
```

The `yield` keyword tells Python to treat the function as a generator

```
g = generate_ints(3)  # Doesn't start the function!  
type(g)  # => <class 'generator'>  
next(g)  # => 0  
next(g)  # => 1  
next(g)  # => 2  
next(g)  # raises StopIteration
```

Another Generator

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

Infinite data stream of Fibonacci numbers