Generator Workshop

In this session you'll learn to write generators that produce infinite sequences, and generators that wrap other generators. If you're capable of using Freenode and gist/some-random-pastebin, please join `#pythonnw` on Freenode so we can easily exchange gist/pastebin links to share solutions with everyone else.

Task 1: sample

Write a function to return a sample of a generator (infinite or otherwise). It should return a list of the first *n* items from *iterable*, and should have the following prototype:

```
def sample(iterable, n=10):
    # Your code goes here
```

Task 2: squares

Write a generator function which yields the square numbers (1, 4, 9, 16, etc.). It should have the prototype:

```
def squares():
    # Your code goes here
```

Task 3: triangles

Write a generator function which yields the triangle numbers (1, 3, 6, 10, 15, etc.). It should have the prototype:

```
def triangles():
    # Your code goes here
```

Task 4: fibonacci

Write a generator function which yields the Fibonacci sequence (1, 1, 2, 3, 5, 8, 13, etc.). It should have the prototype:

```
def fib():
    # Your code goes here
```

Task 5: scaled

Write a generator function which, given an *iterable*, yields each element of iterable multiplied by a scaling factor *n*. It should have the prototype:

```
def scaled(iterable, n):
    # Your code goes here
```

Task 6: sliding_window

Write a generator function which, given an *iterable* and a sequence length *n*, yields each n-sized adjacent subsequent in order. In other words, given the sequence ABCDEF it should yield ABC BCD CDE DEF. It should have the prototype:

```
def sliding_window(iterable, n=2):
    # Your code goes here
```

Task 7: differences

Write a generator function which returns the differences between consecutive values of *iterable*:

```
def differences(iterable):
    # Your code goes here
```

Hint: you can use `sliding_window` for this.

Task 8: ratios

Write a generator function which returns the ratio between values of *iterable*:

```
def ratios(iterable):
    # Your code goes here
```

Hint: this should be *easy* after `differences`

Task 9: Phi!

Find the value that ratios of the Fibonacci sequence converge upon:

```
>>> sample(ratios(fib()))
[1.0,
 2.0,
 1.5,
 1.6666666666666667,
 1.6,
 1.625,
 1.6153846153846154,
 1.619047619047619,
 1.6176470588235294,
 1.6181818181818182]
```

Extra credit: can you make functions which determine whether a sequence converges and what number it converges upon if it does?

Task 10: market_scraper

At the following URL you will find a playback of the historic state of (some of) the FTSE 100 since the late 1980s:

http://mrkrabs.waveform.org.uk/presentations/generators/stocks/market.html

Build a generator function which, on each iteration, yields a new state of the market as a dict mapping stock symbols to their prices (and/or changes, etc.):

```
def market_scraper(url="http://mrkrabs..."):
    while True:
        # Hints:
        # use requests.get to grab the page
        # use bs4.BeautifulSoup to extract data
        # yield the data (if it's changed state)
```

Task 11: portfolio_manager

Build a generator function which, given an iterable of market states (which you now have), yields buy and sell instructions.

```
def portfolio_manager(states, money=1000.0, portfolio=None):
    if portfolio is None:
        portfolio = {}
    # Your code goes here...
```

Hint: use `sliding_window` to look at market states over time, and split out the buy/sell algorithms into their own generators.