Factoring with Python – continued
Spring 2015

1 Croft sieve

http://www.primesdemystified.com/

```
# http://rosettacode.org/wiki/Prime_decomposition#Python:_Using_Croft_Spiral_sieve
from __future__ import print_function

import sys
from itertools import islice, cycle, count

try:
    from itertools import compress
except ImportError:
    def compress(data, selectors):
        """compress('ABCDEF', [1,0,1,0,1,1]) ---> A C E F"""
        return (d for d, s in zip(data, selectors) if s)


def is_prime(n):
    return list(zip((True, False), decompose(n)))[-1][0]

class IsPrimeCached(dict):
    def __missing__(self, n):
        r = is_prime(n)
        self[n] = r
        return r

is_prime_cached = IsPrimeCached()

def croft():
    """Yield prime integers using the Croft Spiral sieve.

    This is a variant of wheel factorisation modulo 30.
    """
    # Copied from:
    #   https://code.google.com/p/pyprimes/source/browse/src/pyprimes.py
    # Implementation is based on erat3 from here:
    #   http://stackoverflow.com/q/2211990
    # and this website:
    #   http://www.primesdemystified.com/
    # Memory usage increases roughly linearly with the number of primes seen.
    # dict ``roots`` stores an entry x:p for every prime p.
    for p in (2, 3, 5):
        yield p
    roots = {9: 3, 25: 5}  # Map d**2 -> d.
    primeroots = frozenset((1, 7, 11, 13, 17, 19, 23, 29))
    selectors = (1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0)
    for q in compress(
            # Iterate over prime candidates 7, 9, 11, 13, ...
            islice(count(7), 0, None, 2),
            # Mask out those that can't possibly be prime.
            cycle(selectors)
            ):
        # Using dict membership testing instead of pop gives a
        # 5-10% speedup over the first three million primes.
        if q in roots:
```

```
            p = roots[q]
            del roots[q]
            x = q + 2*p
            while x in roots or (x % 30) not in primeroots:
                x += 2*p
            roots[x] = p
        else:
            roots[q*q] = q
            yield q
primes = croft

def decompose(n):
    for p in primes():
        if p*p > n: break
        while n % p == 0:
            yield p
            n //=p
    if n > 1:
        yield n


if __name__ == '__main__':
    # Example: calculate factors of Mersenne numbers to M59 #

    import time

    for m in primes():
        p = 2 ** m - 1
        print( "2**{0:d}-1 = {1:d}, with factors:".format(m, p) )
        start = time.time()
        for factor in decompose(p):
            print(factor, end=' ')
            sys.stdout.flush()

        print( "=> {0:.2f}s".format( time.time()-start ) )
        if m >= 59:
            break
```

Output (from rosetta)

```
2**2-1 = 3, with factors:
3 => 0.00s
2**3-1 = 7, with factors:
7 => 0.01s
2**5-1 = 31, with factors:
31 => 0.00s
2**7-1 = 127, with factors:
127 => 0.00s
2**11-1 = 2047, with factors:
23 89 => 0.00s
2**13-1 = 8191, with factors:
8191 => 0.00s
2**17-1 = 131071, with factors:
131071 => 0.00s
2**19-1 = 524287, with factors:
524287 => 0.00s
2**23-1 = 8388607, with factors:
47 178481 => 0.01s
2**29-1 = 536870911, with factors:
233 1103 2089 => 0.01s
```

```
2**31-1 = 2147483647, with factors:
2147483647 => 0.03s
2**37-1 = 137438953471, with factors:
223 616318177 => 0.02s
2**41-1 = 2199023255551, with factors:
13367 164511353 => 0.01s
2**43-1 = 8796093022207, with factors:
431 9719 2099863 => 0.01s
2**47-1 = 140737488355327, with factors:
2351 4513 13264529 => 0.01s
2**53-1 = 9007199254740991, with factors:
6361 69431 20394401 => 0.04s
2**59-1 = 576460752303423487, with factors:
179951 3203431780337 => 1.22s
```

2 Floating point arithmetic

```
# http://rosettacode.org/wiki/Prime_decomposition#Python:_Using_floating_point

from math import floor, sqrt
try:
    long
except NameError:
    long = int

def fac(n):
    step = lambda x: 1 + (x<<2) - ((x>>1)<<1)
    maxq = long(floor(sqrt(n)))
    d = 1
    q = n % 2 == 0 and 2 or 3
    while q <= maxq and n % q != 0:
        q = step(d)
        d += 1
    return q <= maxq and [q] + fac(n//q) or [n]

if __name__ == '__main__':
    import time
    start = time.time()
    tocalc =  2**59-1
    print("%s = %s" % (tocalc, fac(tocalc)))
    print("Needed %ss" % (time.time() - start))
```

Output (from rosetta)

```
576460752303423487 = [3203431780337, 179951]
Needed 0.9240529537200928s
```

Here is the LaTeX source for this document. You can cut it from the pdf and use it to start your answers. I used the `\jobname` *macro for the source file name, so you can call your file by any name you like.*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
\documentclass[10pt]{article}
\usepackage[textheight=10in]{geometry}

\usepackage{verbatim}
\usepackage{amsmath}
\usepackage{amsfonts} % to get \mathbb letters

\usepackage[utf8]{inputenc}
\DeclareFixedFont{\ttb}{T1}{txtt}{bx}{n}{9} % for bold
\DeclareFixedFont{\ttm}{T1}{txtt}{m}{n}{9}  % for normal
% Defining colors
\usepackage{color}
\definecolor{deepblue}{rgb}{0,0,0.5}
\definecolor{deepred}{rgb}{0.6,0,0}
\definecolor{deepgreen}{rgb}{0,0.5,0}

\usepackage{listings}

%Python style from
%http://tex.stackexchange.com/questions/199375/problem-with-listings-package-for-python-syntax-colo
\newcommand\pythonstyle{\lstset{
  language=Python,
  backgroundcolor=\color{white}, %%%%%%%
  basicstyle=\ttm,
  otherkeywords={self},
  keywordstyle=\ttb\color{deepblue},
  emph={MyClass,__init__},
  emphstyle=\ttb\color{deepred},
  stringstyle=\color{deepgreen},
  commentstyle=\color{red},  %%%%%%%%
  frame=tb,
  showstringspaces=false,
  numbers=left,numberstyle=\tiny,numbersep =5pt
}}
\usepackage{fancyvrb}
\usepackage{hyperref}

\begin{document}

\pythonstyle{}
\begin{center}
\Large{
Factoring with Python -- continued \\
Spring 2015
}
\end{center}


\section{Croft sieve}

\url{http://www.primesdemystified.com/}

\lstinputlisting{croft.py}

Output (from rosetta)
```

```
\VerbatimInput{croft.txt}


\section{Floating point arithmetic}

\lstinputlisting{floating.py}

Output (from rosetta)

\VerbatimInput{floating.txt}

\newpage

\emph{
Here is the \LaTeX{} source for this document. You can cut it from the
pdf and use it to start your answers. I used the} \verb!\jobname!
\emph{macro for the source file name, so you can call your file by any
  name you like.}
\verbatiminput{\jobname}

\end{document}
```