The Sieve of Eratosthenes

Ethan Bolker

Spring 2015

The Sieve of Eratosthenes is an ancient method for finding all the primes less than or equal to some number specified in advance. You can read about it all over the web.

Please learn how to spell "Eratosthenes" and "sieve".

The `eratosthenes` module contains just one function, with that name, along with code to test it.

Here is the output from the hard coded tests:

```
PS C:\eb\python\factoring> python .\eratosthenes.py
test at a composite, a square and a prime
eratosthenes(15)
[2, 3, 5, 7, 11, 13]
eratosthenes(16)
[2, 3, 5, 7, 11, 13]
eratosthenes(17)
[2, 3, 5, 7, 11, 13, 17]
number of primes <=100000 is len(eratosthenes(100000): 9592
```

```
# The sieve of Eratosthenes
#
# There are many python implementations of the web. I built my
# solution from those at
# http://stackoverflow.com/questions/3939660/sieve-of-eratosthenes-finding-primes-python
#
# It's not the most efficient, nor the most idiomatic,
# I hope it's good for learning elementary python.
#
# Idea:
#  create possibleprimelist of of length n+1
#  mark positions in the list True for primes,
#    False for composites

def eratosthenes(n):
    # create a list with n+1 copies of the one element list [True]
    possibleprimelist = [True] * (n+1)
    # neither 0 nor 1 is a prime - note multiple assignments
    possibleprimelist[0] = possibleprimelist[1] = False
    # nextp holds the next prime found. starts with 2
    nextp = 2
    primelist = []
    while (nextp < len(possibleprimelist)):
        # add nextp to the growing list of primes
        primelist.append(nextp)
        # kill all multiples of nextp, starting at nextp^2
        i = nextp*nextp
        while (i < len(possibleprimelist)):
            possibleprimelist[i] = False
            i += nextp
        # the next True entry is the next prime
        nextp = find_next(possibleprimelist,nextp+1, True) # could be nextp+2
    return primelist

# This is a common list operation.
# You wouldn't normally separate it out as a function but I
# want to discuss several ways to do it.
def find_next(alist, i, thing):
```

```
39      "find index of next occurrence of thing after position i"
40      try:
41          return i+alist[i:].index(thing)
42      except ValueError:
43          return 1+len(alist) # no thing in the tail of the list
44
45  # Here's the naive straightforward way
46  #      while (i < len(alist)):
47  #          if (alist[i] == thing):
48  #              return i
49  #          i += 1
50  #      return 1+len(alist) # no thing in the tail of the list
51
52  # Fake the default implicit main method that tells python
53  # where to start execution - the right trick for testing inside a module.
54  #
55  if __name__ == '__main__':
56      print("test at a composite, a square and a prime")
57      for n in [15, 16, 17]:
58          print( "eratosthenes(" + str(n) + ")")
59          print( str(eratosthenes(n)))
60      print("number of primes <=100000 is len(eratosthenes(100000): "
61            + str(len(eratosthenes(100000))))
```

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.*

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% eratosthenes.tex
%
\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,
  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{hyperref}

\begin{document}

\pythonstyle{}
\setcounter{section}{1} % start with section 2
\begin{center}
\Large{
The Sieve of Eratosthenes \\
Ethan Bolker \\
Spring 2015
}
\end{center}

The Sieve of Eratosthenes is an ancient method for finding all the
primes less than or equal to some number specified in advance. You can
read about it all over the web.

\emph{Please} learn how to spell ``Eratosthenes'' and ``sieve''.
```

```
The \lstinline!eratosthenes! module contains just one function, with
that name, along with code to test it.

Here is the output from the hard coded tests:

\begin{verbatim}
PS C:\eb\python\factoring> python .\eratosthenes.py
test at a composite, a square and a prime
eratosthenes(15)
[2, 3, 5, 7, 11, 13]
eratosthenes(16)
[2, 3, 5, 7, 11, 13]
eratosthenes(17)
[2, 3, 5, 7, 11, 13, 17]
number of primes <=100000 is len(eratosthenes(100000): 9592
\end{verbatim}

\lstinputlisting{eratosthenes.py}

\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}
```