Catchup (Say It Again)
Mathematical Programming with Python

MATH 2604: Advanced Scientific Computing 4
Spring 2025
Monday/Wednesday/Friday, 1:00-1:50pm

https://people.sc.fsu.edu/∼jburkardt/classes/python_2025/catchup/catchup.pdf

1 Overview

Today we will pause, and go back over a few topics from last week which can use a litte more discussion, involving prime numbers, and reading information from a file into a list. We will concentrate on writing and testing some simple programs.

2 Eratosthenes

Last time, we outlined the sieve of Eratosthenes, which can produce all the prime numbers in a given range $[2, N]$. In Python, it is natural to write this as the transfer of some elements of a list of integers `L` to an initially empty list `P`. This example illustrates the use of the `append()` and `remove()` operations, how to start with an empty list, or stop when a list becomes empty, and how to print just the last element of a list.

```
nmax = 100
L = list ( range ( 2, nmax + 1 ) )
P = []

while ( L != [] ):
  p = L[0]
  P.append ( p )              # How to add element p to list P
  for l in L:
    if ( l % p == 0 ):
      L.remove ( l )          # How to remove element l from list L

print ( P )
```

Listing 1: eratosthenes.py

3 Eratosthenes timed

Eratosthenes works fine for $N = 100$, but we have to wonder how well it works as N increases. This is a huge concern to computer scientists and people who design algorithms. Python offers the `perf_counter()` tool to time such operations, and now we can ask what happens to the elapsed time as we double N several times.

```
def eratosthenes_timing ( nmax ):

  from time import perf_counter

  L = list ( range ( 2, nmax + 1 ) )
  P = []

  tic = perf_counter ( )       # Start the clock.

  while ( L != [] ):
    p = L[0]
    P.append ( p )
    for l in L:
      if ( l % p == 0 ):
        L.remove ( l )

  toc = perf_counter ( )      # Stop the clock.

  print ( 'eratosthenes ( ', nmax, ' ) required ', toc - tic, ' seconds' )
  print ( '  Number of primes found was ', len ( P ) )
  print ( '  Biggest prime found was ', P[-1] )          # Last list element
```

Listing 2: eratosthenes_timing.py

4 Is N Prime?

The `is_prime1()` function checks a single number N for primality, rather than the range $[2, N]$.

```
def is_prime1 ( n ):
  value = True
  for d in range ( 2, n ):
    if ( n % d == 0 ):
      value = False

  return value
```

Listing 3: is_prime1.py

5 is_prime1() timed

We are interested in how much faster this calculation is, and so we want to check 199,999 and 6,700,417 and 2,147,483,647 using Python's timer. We also want to estimate how fast the time grows with N.

```
def is_prime1_timing ( n ):

  from is_prime1 import is_prime1
  from time import perf_counter

  tic = perf_counter ( )      # Start the clock.

  is_prime = is_prime1 ( n )

  toc = perf_counter ( )     # Stop the clock.

  print ( 'is_prime1(): is ', n, ' prime? ', is_prime )
  print ( '  Computing seconds was ', toc - tic )
```

Listing 4: is_prime1_timing.py

As a comparison, we can do the same calculations using the `isprime()` function from the `sympy()` library.

6 Display the lines of a text file

There are several ways for Python to work with a file. Once the file is available, Python sees it as a sequence of lines which can easily be accesse.

The `with` statement is coomonly used to open files, and does not require you to close the file afterwards.

```
  with open ( 'grook.txt', 'r' ) as file:  # Open
    for line in file:
      print ( line )
```

Listing 5: grook_reader.py

I prefer using a plain `open()` statement, but this does require that you close the file afterwards, especially if you want to read from the file again.

```
  file = open ( 'grook.txt', 'r' )  #  Open file
  for line in file:
    print ( line )
  file.close ( )                    #  Close file
```

Listing 6: grook_reader.py

Instead of simply printing the file line by line, we can use a similar approach to copy each line into a list. Often, we want to process these lines as they come in, to remove unwanted characters such as the carriage control.

7 Copy the lines of a text file as elements of a list

Once a file has been opened, a `for()` loop can be used to copy each line of the file as an element of a list. The `.strip()` method removes the carriage control or newline character, and any leading or trailing blank spaces.

```
def grook_reader_strip ( ):
  file = open ( 'grook.txt', 'r' )
  lines = [ line.strip() for line in file ]
  file.close ( )
```

Listing 7: grook_reader_strip.py

The object `lines` is now a list of 4 strings.

```
  print ( lines )

  ['The road to wisdom?  Well, it is plain',
   'And simple to express:',
   'Err and err and err again,',
   'But less and less and less.']
```

8 Read the lines into a list with words as elements

If we want to study the words in the file, not the sentences, then we need to split each line into its words.

```
def grook_reader_split ( ):

  file = open ( 'grook.txt', r' )
  words = [ line.split() for line in file ]
  file.close ( )
```

Listing 8: grook_reader_split.py

results in a list of lists. Each sentence in the previous output is now replaced by a list of its words:

```
  print ( words )

[ ['The', 'road', 'to', 'wisdom?', 'Well,', 'it', 'is', 'plain'],
  ['And', 'simple', 'to', 'express:'],
  ['Err', 'and', 'err', 'and', 'err', 'again,'],
  ['But', 'less', 'and', 'less', 'and', 'less.'] ]
```

9 sum() turns a list of lists into a plain list

It might be more convenient to have a plain list of words. We will see in a moment how to do this using a different way of reading the file. However, if you have a list of lists, you can convert it to a plain list with a command like this:

```
  words = [ line.split() for line in file ]
  words2 = sum ( words, [] )
```

What's happening is essentially a concatenation:

```
  words2 = words[0] + words[1] + ... + words[-1]   # This works too!
```

and the result is

```
print ( words2 )

['The', 'road', 'to', 'wisdom?', 'Well,', 'it', 'is', 'plain', 'And', 'simple', 'to', '
    express:', 'Err', 'and', 'err', 'and', 'err', 'again,', 'But', 'less', 'and', 'less', '
    and', 'less.']
```

For text analysis, notice that we still have some capital letters, and punctuation marks. If we want to count the occurrences of a word, those items would have to be dealt with.

10 Read all the words of a file into a single simple list

If we want a simple list of words, we can use a double loop, which grabs each line, splits it into words, and copies each word into our list:

```
def list_words_in_file ( filename ):

  words = []

  file = open ( filename, 'r' )
  for line in file:
    for word in line.lower().split():
      words.append ( word )
  file.close ( )

  return words
```

11 Some list operations

Once we have gotten a list of words, there are a number of things we might do with it, using standard list operations.

We can ask for the number of items in a list using `len()`. If we are looking at a list of lists, then we will simply get the number of lists inside the main list.

```
  len ( [ 1, 2, 3 ] ) = 3
  len ( [ [ 1, 2, 3 ], [ 4, 5, 6, 7 ] ] ) = 2
```

We can sort the items in a list:

```
  words = sorted ( words )
```

One of the important uses of word lists is to discover the most frequently used words. This is why it becomes important to clean the text, removing the distinction between upper and lower case, and handling punctuation somehow. I tried to sketch a program called `word_frequency.py`, but in our last experiment, we will see that I didn't do enought.

12 Alice in Wonderland

Surprisingly, to read a large text like Alice in Wonderland into a list of words can be done surprisingly quickly. Unfortunately, the file contains lots of "surprising" characters that Python doesn't know how to ignore, so it incorrectly records words like Alice" (double quote) or thus- (long dash), and this makes it hard to find all the matching words. More work is needed.

```
def alice_word_list ( ):

  from list_words_in_file import list_words_in_file
  from word_frequency import word_frequency

  filename = 'alice_in_wonderland.txt'
  words = list_words_in_file ( filename )
  print ( '' )
  print ( '  First 50 words' )
  print ( words[0:50] )

  counts = word_frequency ( words )

  print ( '' )
```

```
print('Word frequencies:')
print(counts)
```

Listing 9: alice_word_list.py