

## Chapter 14

A collection of hand tools including a hammer, a wrench, pliers, a screwdriver, and a utility knife. The tools are arranged in a cluster, with the hammer and wrench being the most prominent. The wrench is silver and has "R. 77-10" ALLOY" and "GROOV" "POWERS" engraved on it. The pliers have blue and red handles. The screwdriver has a wooden handle and a black grip. The utility knife is black and has a silver blade.

# Selecting and Combining Tools

F. Duveau

02/03/12

# Your toolkit

- **Regular expressions** To search and replace
- **Shell commands** To interact with your computer at the command line
- **Shell scripts** To combine / automate command-line operations
- **Python programs** For more advanced processing

How to choose the convenient tools for a particular task?

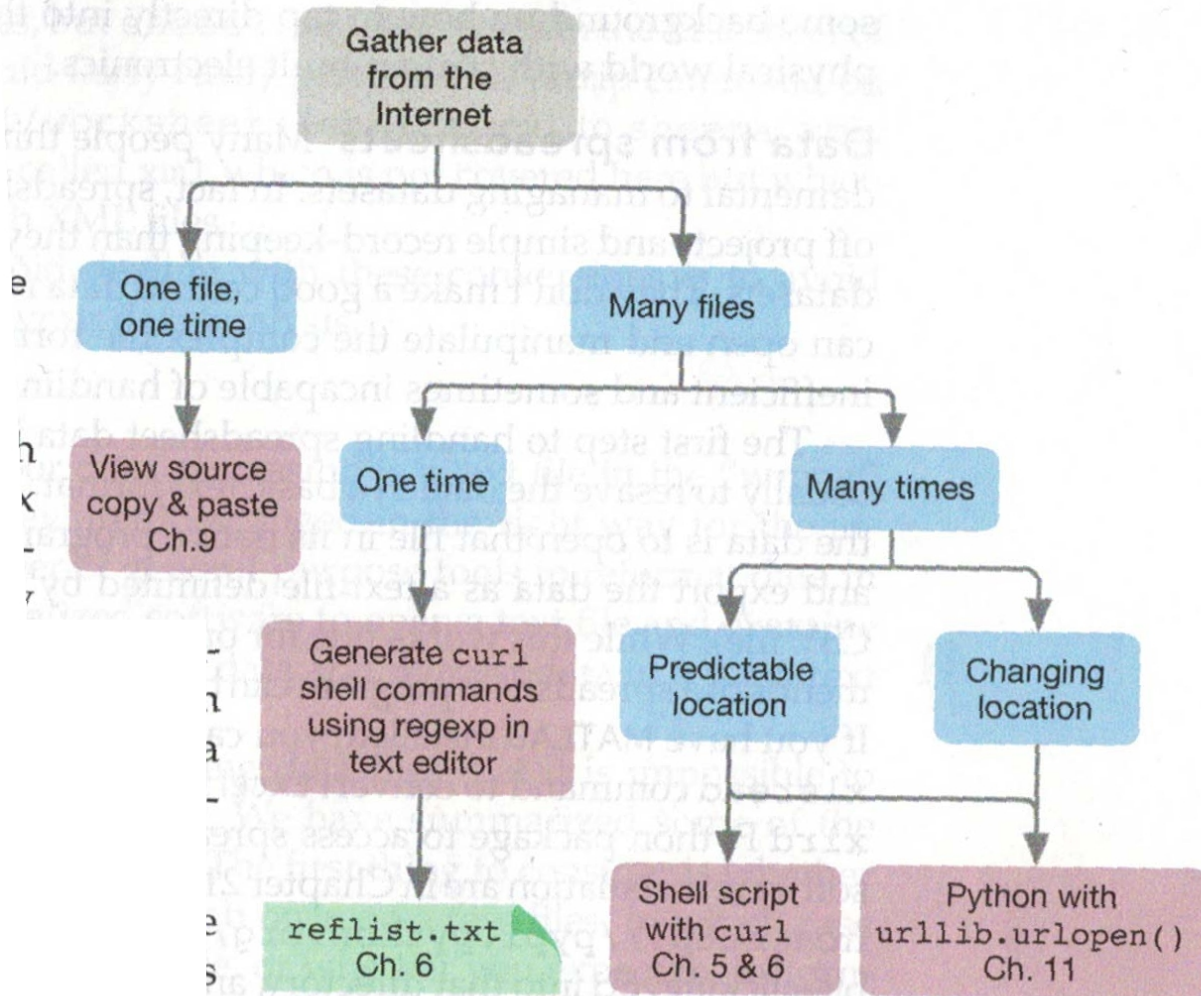


It depends on the kind and number of input data and the amount and frequency of work to perform

# Type of input data

- **Data from user input** `raw-input()` `sys.argv[]`
- **Data from the Internet** `curl()` or `wget()` commands, `urllib2` module
- **Data from other programs** Output of other programs can be captured using redirection (`>` or `>>`) or the pipe operator (`|`)
- **Data from hardware** Chapter 22 gives background on how to interact directly with the physical world
- **Data from spreadsheets** Not suitable to manage large and complex datasets

# Gathering data from the Internet



or >>), or else send the output to

# Data from spreadsheets

More suited to small one-off projects and simple record-keeping

The first step is usually to resave the data in a basic text format (eg CSV)

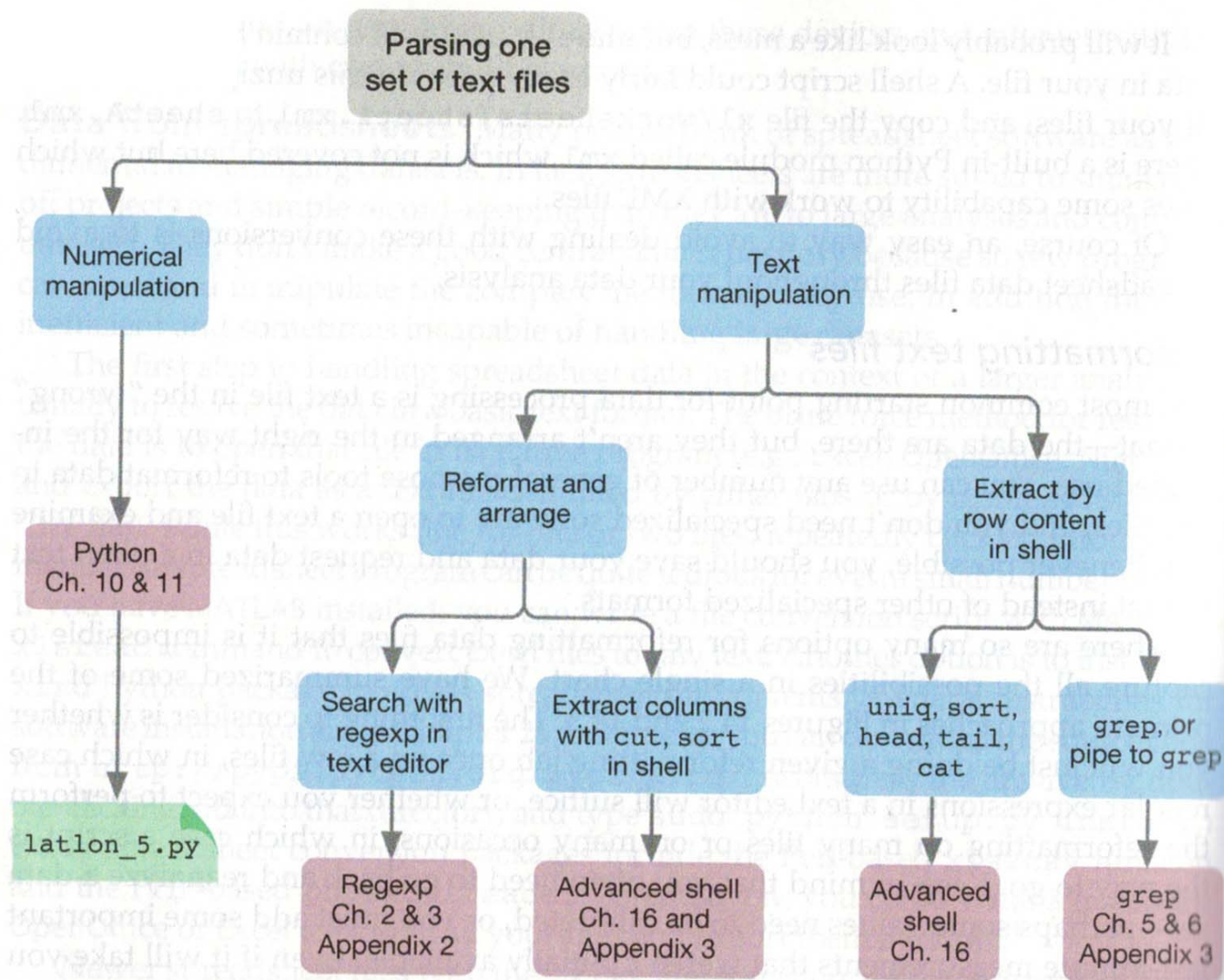
`xlrd` Python package to access spreadsheet file contents

`.xlsx` files are compressed archives containing XML files

```
xl/worksheet directory:  
host:~ lucy$ unzip SpreadsheetDataA.xlsx  
  
  inflating: [Content_Types].xml  
  inflating: _rels/.rels  
  inflating: xl/_rels/workbook.xml.rels  
  inflating: xl/workbook.xml  
  ...etc...  
host:~ lucy$ cd xl/worksheets  
host:worksheets lucy$ ls  
  
sheet2.xml  sheet4.xml  sheet6.xml  sheet8.xml
```

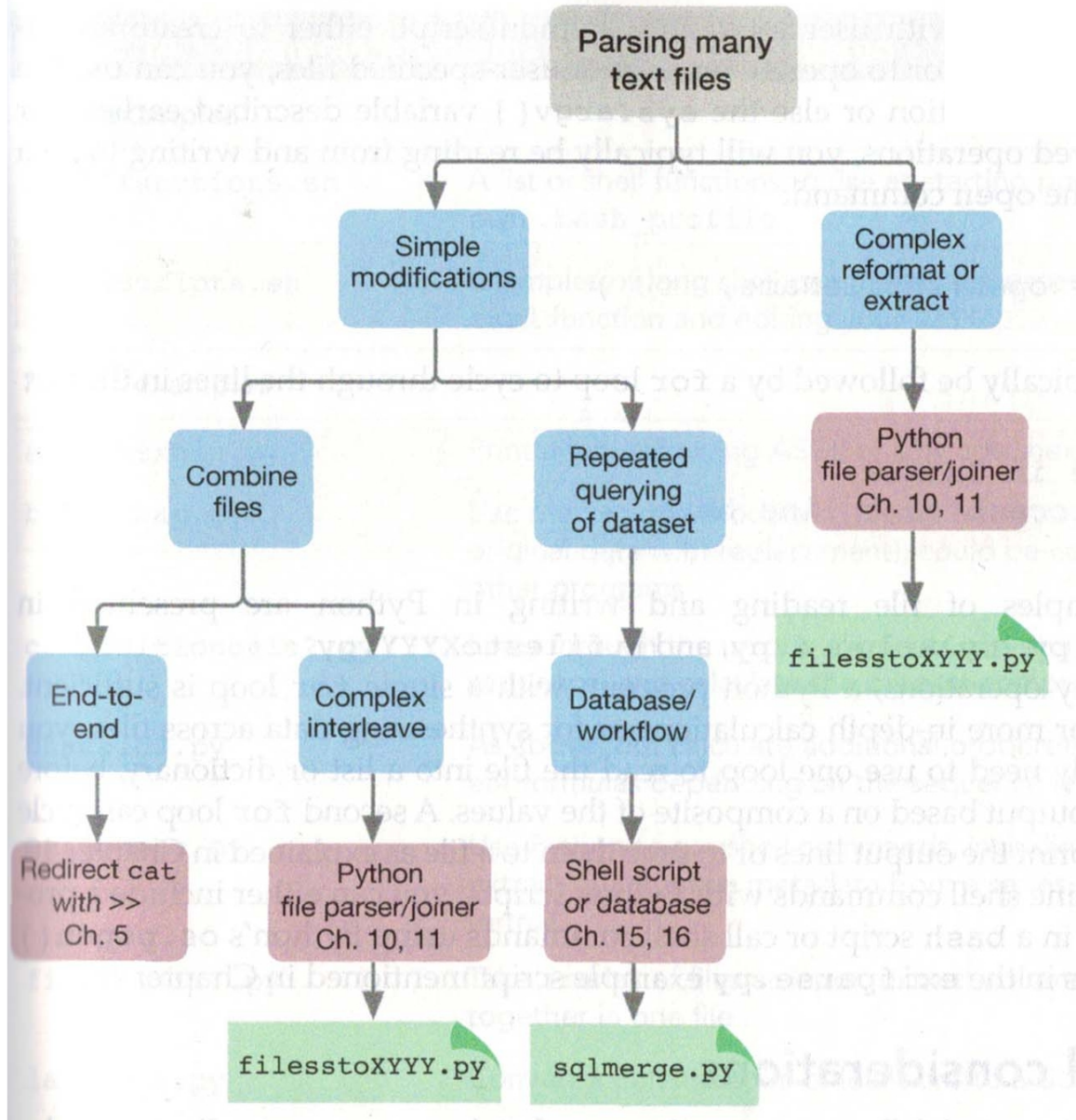
<sup>1</sup>A macro is a script built within another program's own internal scripting environment.

# Parsing one set of text files





# Parsing many text files



# Correction of the exercise

- **Input data:** *html* files obtained from the MWG website
- **Output:** *csv* file containing names, sequences and order numbers

**STEP 1:** Create a list of primer sequences and names from 1 file

**STEP 2:** Extend this list to multiple files (fixed number of files)

**STEP 3:** Retrieve the filenames from the order numbers in *manage-order.html*

Automatically loop over all orders

**STEP 4:** Create the list of order numbers

**STEP 5:** Write the output file



# Collecting data from the Internet

The idea is to adapt the script to find the information directly from webpages instead of manually downloading html files

**Problem:** The website is secured with HTTPS protocol.

`https://ecom.mwgdna.com/services/track/manage-orders.tcl`

**Strategy to solve this problem** (to be implemented in your Python script):

- 1) Change the default User Agent of Python to simulate a classical web browser
- 2) Use a Python module (eg. `httpplib`) that can send HTTP orders to the server
- 3) Send the username and password to the authentication HTTPS address
- 4) The server should send a response allowing your program to access your primer data