Part I

MAKE

Make

- Make is a versatile tool which can run commands

- Some examples
  - Compile source code into executable programs or libraries
  - Run scripts on data files to get summaries or plots
  - Parse and combine text files and plots to create paper
- Make is designed so as to only update files when they need to be. It tracks the dependencies between files it creates and files used to create them.

Setup

- Make is already available in Unix machines
- On Macs, make comes with Xcode
- For Windows, I have no clue
  - https://github.com/swcarpentry/windows-installer
- Open a terminal and check whether you have it

```
make -v
```

```
(base) DPSITs-iMac:~ ergunsimsek$ make -v
GNU Make 3.81
Copyright (C) 2006  Free Software Foundation, Inc.
This is free software; see the source for copying conditions.
There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A
PARTICULAR PURPOSE.

This program built for i386-apple-darwin11.3.0
(base) DPSITs-iMac:~ ergunsimsek$
```

Example

Let’s look at our folder

```
(base) DPSITs-iMac:make-lesson ergunsimsek$ ls -R
books                   makefile_advanced       plotcounts.py
countwords.py           makefile_simple         testzipf.py

./books:
LICENSE_TEXTS.md        isles.txt               sierra.txt
abyss.txt               last.txt
```

We have 3 *.py codes

We have a subfolder called “books” which has some *.txt files

countwords.py

```
import sys
# bunch of functions
# bunch of functions
# …
# at the end, we have the following lines. EXAMINE!!

if __name__ == '__main__':
    input_file = sys.argv[1]
    output_file = sys.argv[2]
    min_length = 1
    if len(sys.argv) > 3:
        min_length = int(sys.argv[3])
    word_count(input_file, output_file, min_length)
```

testzipf.py

Warning

```
from countwords import load_word_counts
import sys

def top_two_word(counts):
    """
    Given a list of (word, count, percentage) tuples,
    return the top two word counts.
    """
    limited_counts = counts[0:2]
    count_data = [count for (_, count, _) in limited_counts]
    return count_data


if __name__ == '__main__':
    input_files = sys.argv[1:]
    print("Book\tFirst\tSecond\tRatio")
    for input_file in input_files:
        counts = load_word_counts(input_file)
        [first, second] = top_two_word(counts)
        bookname = input_file[:-4]
        print("%s\t%i\t%i\t%.2f" %(bookname, first, second, float(first)/second))
```

Example

The Python program “countwords.py” counts the frequency of each word in a text file.

We want to out the frequency of words in “x.txt” to a new data file “x.dat”

Then we want to use the “testzipf.py” to test the “Zipf’s Law” on the “x.dat”

In a terminal, we can achieve this in 3 steps

```
$ python countwords.py books/isles.txt isles.dat
$ python countwords.py books/abyss.txt abyss.dat
$ python testzipf.py isles.dat abyss.dat > results.txt
$ ls -R
__pycache__             countwords.py           makefile_simple         testzipf.py
abyss.dat               isles.dat               plotcounts.py
books                   makefile_advanced       results.txt

./__pycache__:
countwords.cpython-38.pyc

./books:
LICENSE_TEXTS.md        isles.txt               sierra.txt
abyss.txt               last.txt
```

WHAT IF YOUR TASK IS RUNNING THESE CODES FOR THOUSANDS OF BOOKS?

Example Cont.

- We can create a Makefile to automate processing of the texts.
- Let us create a new file called "**makefile**" (no extension) with the following content:

target and dependency are separated by :

"#" denotes a comment

dependency

target

```
# Count words.
isles.dat : books/isles.txt
        python countwords.py books/isles.txt isles.dat
```

ACTION

tab
(not 8 spaces)

dependency

target

To take action, simply enter "make" in the terminal

Phony Targets

- Suppose we only care about the “results.txt”, hence we want to remove all our data files *.dat, when we are done with counting
- We can introduce a new target “clean,” and associated cleaning rule, to the end of our Makefile.

```
.PHONY : clean
clean :
        rm -f *.dat
```

- Here, the target “clean” is “.PHONY” in the sense that it doesn’t refer to a name of file. it is just a name for a recipe to be executed when you make an explicit request.

- -f: ignore non-existing files

```
$ make                                                    # input
python ../countwords.py ../books/abyss.txt abyss.dat

$ ls                                                      # input
abyss.dat isles.dat makefile

$ make clean                                              # input
rm -f *.dat

$ ls                                                      # input
makefile
```

Multiple Targets

As the Makefile gets larger, there may be a variety of data file targets that we would like to be able to build all at once.

One way of doing this would be with a phony target “dats” which depends on all of the targets we want to build.

Let’s put the following at the beginning of the Makefile so that it is the default target.

```
# Count words.
.PHONY : dats
dats : isles.dat abyss.dat

isles.dat : books/isles.txt
        python countwords.py books/isles.txt isles.dat

abyss.dat : books/abyss.txt
        python countwords.py books/abyss.txt abyss.dat

.PHONY : clean
clean :
        rm -f *.dat
```

Automation (Step 1)

Automatic variables are reserved expressions for targets and dependencies to be used within rules.

- `$@` is the target of the current rule.
- `$<` is the first dependency of the current rule.
- `$^` is all of the dependencies of the current rule.

```
# Count words.
.PHONY : dats
dats : isles.dat abyss.dat

isles.dat : books/isles.txt
        python countwords.py $< $@

abyss.dat : books/abyss.txt
        python countwords.py $< $@

.PHONY : clean
clean :
        rm -f *.dat
```

Automation (Step 2: Pattern Rules)

Our Makefile still has repeated content. Specifically, the rules for each data file are identical except for the book name.

We can replace these rules with a single pattern rule.

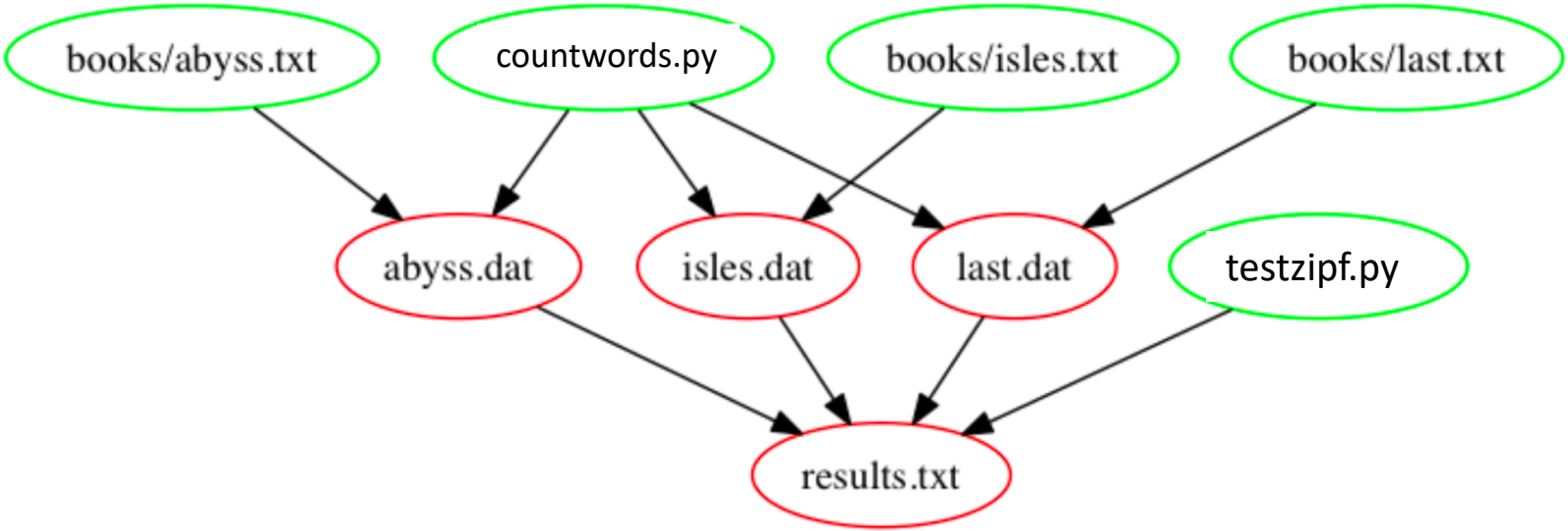
This rule specifies how to build any data file from a text file in the books directory:

```
# Count words.
.PHONY : dats
dats : isles.dat abyss.dat

%.dat : books/%.txt
	python countwords.py $< $*.dat

.PHONY : clean
clean :
	rm -f *.dat
```

Process, Build Results, Clean the Mess!



Make: Build Results

```
1   # Count words.
2   .PHONY : dats
3   dats : isles.dat abyss.dat
4
5   %.dat : books/%.txt
6       python countwords.py $< $*.dat
7
8   # Generate summary table.
9   results.txt : testzipf.py isles.dat abyss.dat
10      python $< *.dat > $@
11
12  .PHONY : clean
13  clean :
14      rm -f *.dat
```