ISTM 6212 - Week 3
tidy data & types, filters, parallel

Daniel Chudnov, dchud@gwu.edu

2016-09-13

Agenda

- Exercise 02
- Tidy data
- Data types
- Filters
- Parallel
- Project 01

Exercise 02

My mistakes

- Incorrect use of "{}" -> should be "{{}}"
- Incorrect value in os.listdir('.')
- Incorrect use of "_"
- Did not auto-grade correctly

What you did well

- Posting questions in forum
- Posting answers in forum
- Getting started early: time to report errors, find fixes
- Completed on time

Good answers (part 1)

- 'The symbol ">" tells the shell to redirect the command’s output to a file instead of printing it to the screen. The shell will create the file (in this example: filelist.txt) if it doesn’t exist. If the file exists, it will be silently overwritten, which may lead to data loss. Fortunately, the symbol ">>" avoid the data loss problem. ">>" does not overwrite the file. It appends the file. If the file exists, symbol ">>" tells the shell to redirect the output to the existing file and the file will consist both the output and the previous contents.'

Good answers (part 1)

- "The operator ‘>’ redirects the output of a command on the left side of the operator to the file listed on the right side of the operator. If the file does not exist, then a new one will be automatically created. If the file exists, each time you execute ‘>’, the file will be overwritten. For example, usually, the result of a command will be displayed on the screen, but if you command ‘>’ , let’s say % ls > myfile1.txt , the content of your current directory will not be displayed on the screen. The output will be written in the myfile1.txt. You have to open the myfile.txt then you can see your output. However, the operator ‘>>’ appends the content on the left side of the operator to the right side. It adds new information to the original files and makes it longer instead of overwriting it. When there is no file, the ‘>>’ operator will automatically create a new one."

Good answers (part 1)

- "Both >> and > redirect outputs of a command to a file. If the file does not exists in the path, they create the file. Otherwise, > overwrites the file. On the other hand, >> appends the outputs to the end of the file."

Good answers (part 1)

- "The problem is two-fold. First, we are searching for unique words in a case-sensitive way, and second, we are sorting our list of unique words by alphabetical order and not numerical. To fix these issues, we need to put all words in lowercase and we need to sort the list in reverse numerical order (greatest to lowest)."

Good answers (part 2)

- "Based on the two cells listed below, it appears that trips beginning at Massachusetts Ave & Dupont Circle NW had a longer ride duration in comparison to trips that ended at Massachusetts Ave & Dupont Circle NW. This finding is supported by both the mean and median of Duration (ms)."

- "From the statistics above, trips starting at 'Massachusetts Ave & Dupont Circle NW' has the average trip duration of 946431.07508(ms), and trips ending at 'Massachusetts Ave & Dupont Circle NW' has the average trip duration of 840543.301862(ms). Therefore, the former has a longer average trip duration."

Okay answers (part 2)

- 'Based on the csvstat above we can conclude that trips beginning at "Massachusetts Ave & Dupont Circle NW" had longer average trip durations. See output above. But at the same time, it is interesting to find that the trips ending at "Massachusetts Ave & Dupont Circle NW" had longer total trip duration.'

Common mistakes

- !ls > mydirectory/myfiles.txt
- !rm myfiles.txt
- What's the mistake?

Common mistakes

- ">" vs ">>"

lower|sort|uniq (or) sort|lower|uniq ?

- Of six order permutations, which are correct?

Common mistakes

- Use markdown for prose, not these:
  - "# Here is my text in a code cell with a Python comment"
  - "print('Here is my text in a Python print statement')"

- Clever! But **just use markdown** for text.

Common questions

- Ignore "write error: broken pipe"
- (especially if the output looks correct!)

Common mistakes

- !ls > ~/Downloads/mydirectory/myfiles.txt
  - This is an absolute path; not the same as mine
- Units: in "Duration (ms)", the "ms" refers to _____ ?
- Consider precision: 946431.0750804995ms, 946s, 15.7min?
- Don't re-download data if you already have it
- Use different folders for different classes

Please please please

- Use the answer cells provided for your answers
- Don't add cells unless encouraged
- Don't remove tests and test cells
- Doing these correctly helps grading!

The 15 minute rule

- If you get stuck, stop after 15 minutes
- Take a break: a walk, a snack, a nap, etc.
- Try again for 15 minutes
- If you can't solve the problem after 30 minutes, **ask somebody**

Tidy Data

Tidy data has:

- Every **variable** a column
- Every **observation** a row
- Every table about one **type of observation**

	treatmentA	treatmentB
John Smith	—	5
Jane Doe	1	4
Mary Johnson	2	3

Table 1: Typical presentation dataset.

treatment type is a **variable**

each value is an **observation**

now tidied, each row is an **observation**

name	treatment	result
Jane Doe	a	1
Jane Doe	b	4
John Smith	a	—
John Smith	b	5
Mary Johnson	a	2
Mary Johnson	b	3

Table 2: Tidied data.

images from Wickham H, "Tidy Data," Journal of Statistical Software 59:10 (2014)

Tidy data notes

- Widely useful format
- Not the only useful format
- Like Codd's third normal form in RDBMS theory
- Python, R, Excel, SAS, SPSS, Stata all read it easily

Goals for wrangling

- Are columns variables, or values?
- Are rows or columns filled with blanks for a good reason?
- Is the table just one kind of observation?

Data types

Data types in Python

- Primitives: strings, integers, floats
- Sequences and mutables: tuples, lists, dicts, sets

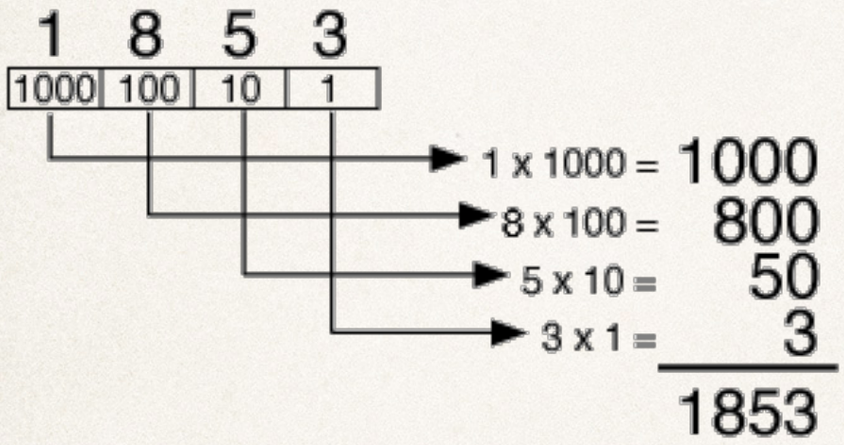
Primitive data types

- Characters and strings
- Booleans
- Small integers, integers, long integers
- Signed and unsigned integers
- Floats
- Dates

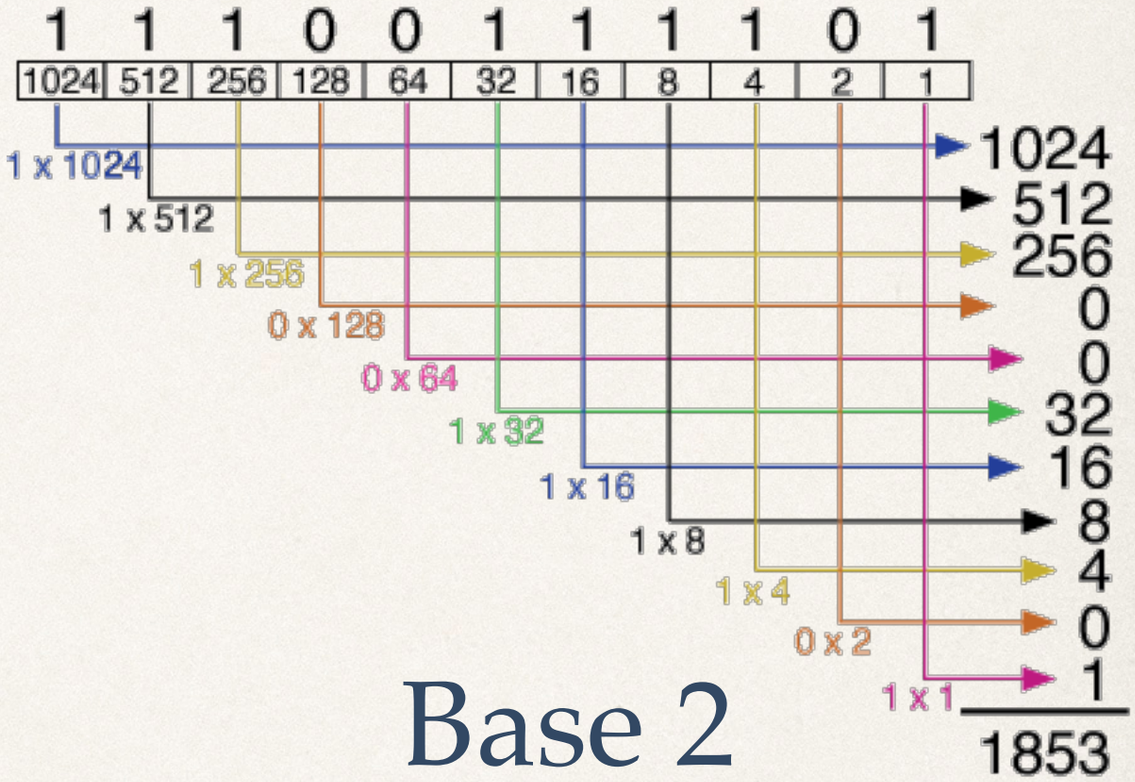
Key questions

- How to represent a boolean?
- How large might a number be?
- How many characters in a string?
- How many choices in a factor (category)?

Binary representation - 1853



Base 10



Base 2
(binary)

- via https://www.unm.edu/~tbeach/terms/binary.html

Why does it matter?

- Often, it won't
- If you need **speed**, data representation matters
- If you have **volume**, data representation matters
- Often, efficiency doesn't matter **until it does**
- What does "big data" mean to you?

Understanding your data

- What are the variables, what is an observation?
- What are the domain and range, max and min?
- What do missing values mean?
- How fast is this growing?
- How big can this get?
- How can I model and manage it efficiently?

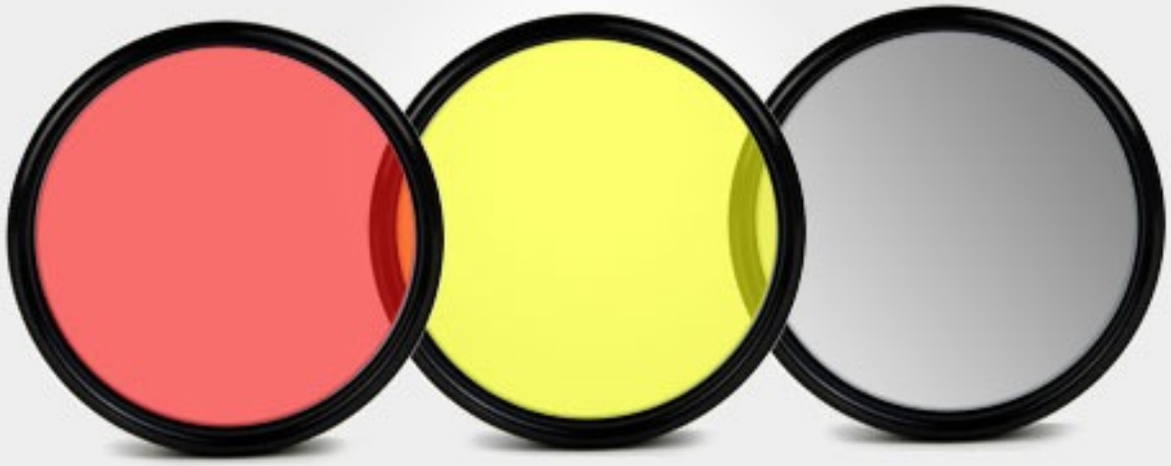
Filters


FILTERS

RED ENHANCER

YELLOW FILTER

UV FILTER



- sort -n
- grep -oE '\w{2,}'
- uniq
- cat
- tr '[:upper:]' '[:lower:]'
- uniq -c
- head -25

Line orientation

- Easy to understand
- Works well with text
- Works well with tidy data
- Only read one line at a time
- Friendly to pipes

Easy to write

simplefilter.py

```
#!/usr/bin/env python

"""
A filter that ___.
"""

import fileinput

def process(line):
    """For each line of input, ___."""
    print line.strip()


for line in fileinput.input():
    process(line)

```

Parallel

Why process in parallel?

- We often need to scale up in **speed** or **volume**
- Processors have become inexpensive
- Many algorithms can be parallelized
- Easier than ever before

Examples of parallelism

- Your operating system (win/osx/linux/ios/android)
- Shell pipeline
- Your computer hardware - CPU and graphics (GPU)
- Hadoop and map/reduce paradigm

Three kinds of parallelism

- Data parallel - split data into chunks (map/reduce)
- Task or logic parallel - different tasks, different handlers (CPU and GPU)
- Pipeline parallel - maximize flow through series of process (shell pipeline)
- Combinations of the above

GNU Parallel

- Simplifies data-parallel tasks in the shell
- Simple case: from 1 book to 100 books
- Read more: https://www.gnu.org/software/parallel/

Project 01