CSE 140 Wrap-Up

CSE 140

Winter 2014

University of Washington

Presentations on Monday

- 2:30-4:20pm, Monday 3/17
- No more than 5 slides (including title slide)
- Time will be capped at TWO minutes
- Both partners should speak
- Slides are due BY noon on 3/17 to catalyst

Progress in 10 weeks

10 weeks ago: you knew no programming

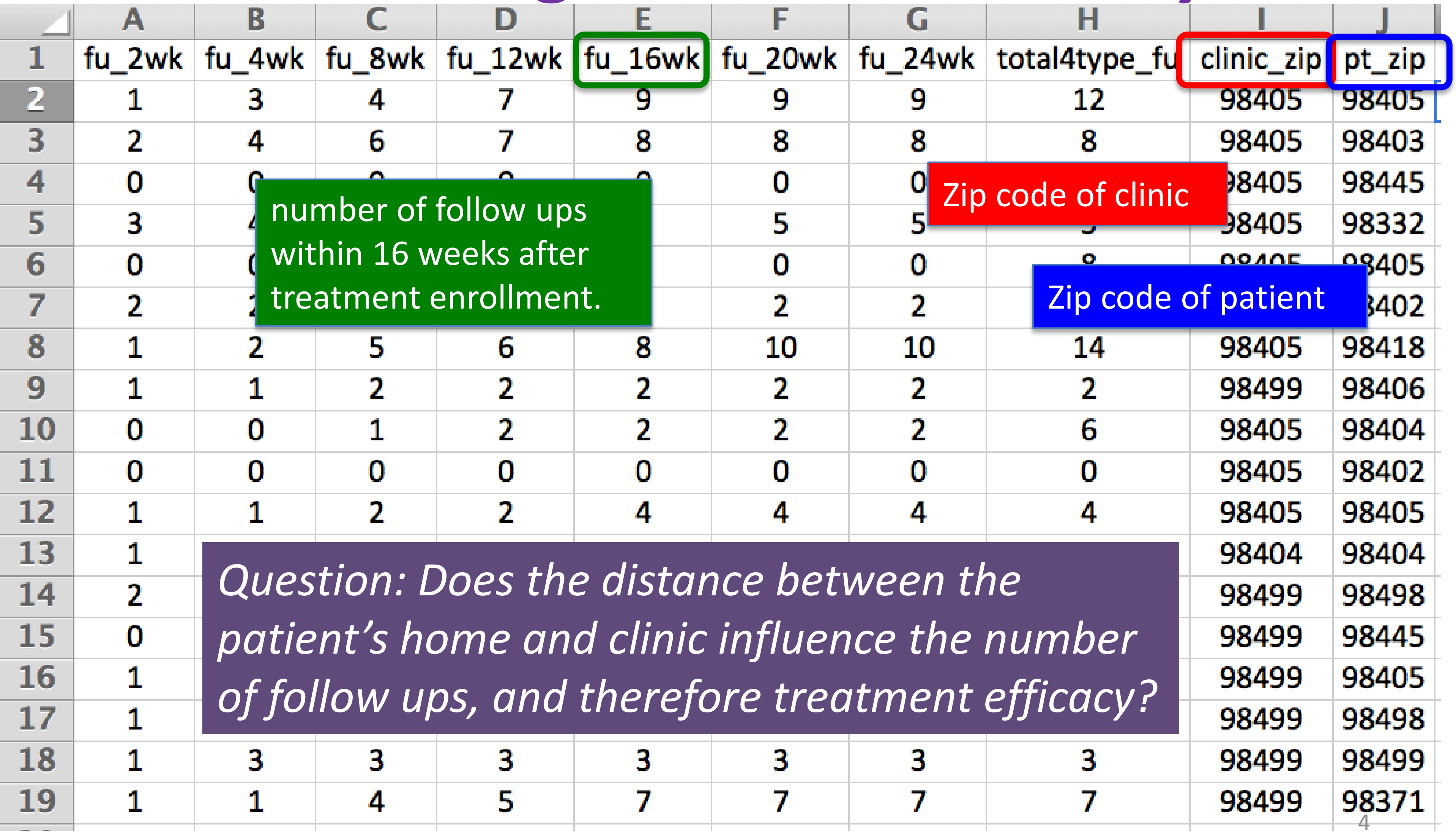
Goals:

- **Computational problem-solving**
- **Python** programming language
- Experience with **real datasets**
- **Fun** of extracting understanding and insight from data, and of mastery over the computer
- Ability to go on to more advanced **computing** classes

Today: you can write a useful program to solve a real problem

- You can even pose the problem yourself

Example from lecture 0: Assessing treatment efficacy



	A	B	C	D	E	F	G	H	I	J
1	fu_2wk	fu_4wk	fu_8wk	fu_12wk	fu_16wk	fu_20wk	fu_24wk	total4type_fu	clinic_zip	pt_zip
2	1	3	4	7	9	9	9	12	98405	98405
3	2	4	6	7	8	8	8	8	98405	98403
4	0	0	0	0	0	0	0	[illegible]	98405	98445
5	3	4	[illegible]	[illegible]	[illegible]	5	5	5	98405	98332
6	0	0	[illegible]	[illegible]	[illegible]	0	0	8	98405	98405
7	2	2	[illegible]	[illegible]	[illegible]	2	2	[illegible]	[illegible]	98402
8	1	2	5	6	8	10	10	14	98405	98418
9	1	1	2	2	2	2	2	2	98499	98406
10	0	0	1	2	2	2	2	6	98405	98404
11	0	0	0	0	0	0	0	0	98405	98402
12	1	1	2	2	4	4	4	4	98405	98405
13	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98404	98404
14	2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98498
15	0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98445
16	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98405
17	1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	98499	98498
18	1	3	3	3	3	3	3	3	98499	98499
19	1	1	4	5	7	7	7	7	98499	98371

number of follow ups within 16 weeks after treatment enrollment.

Zip code of clinic

Zip code of patient

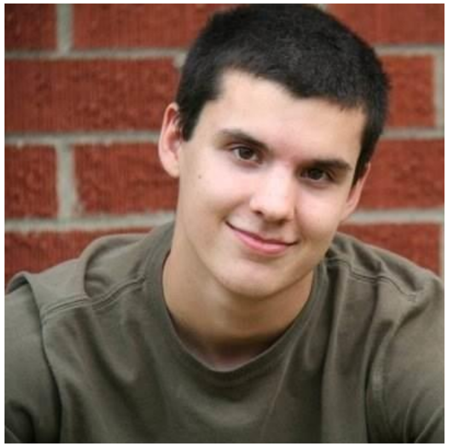
Question: Does the distance between the patient's home and clinic influence the number of follow ups, and therefore treatment efficacy?

Python program to assess treatment efficacy

```python
# This program reads an Excel spreadsheet whose penultimate
# and antepenultimate columns are zip codes.
# It adds a new last column for the distance between those zip
# codes, and outputs in CSV (comma-separated values) format.
# Call the program with two numeric values:  the first and last
# row to include.
# The output contains the column headers and those rows.

# Libraries to use
import random
import sys
import xlrd        # library for working with Excel spreadsheets
import time
from gdapi import GoogleDirections

# No key needed if few queries
gd = GoogleDirections('dummy-Google-key')

wb = xlrd.open_workbook('mhip_zip_eScience_121611a.xls')
sheet = wb.sheet_by_index(0)

# User input:  first row to process, first row not to process
first_row = max(int(sys.argv[1]), 2)
row_limit = min(int(sys.argv[2]+1), sheet.nrows)

def comma_separated(lst):
  return ",".join([str(s) for s in lst])

headers = sheet.row_values(0) + ["distance"]
print comma_separated(headers)

for rownum in range(first_row,row_limit):
  row = sheet.row_values(rownum)
  (zip1, zip2) = row[-3:-1]
  if zip1 and zip2:
    # Clean the data
    zip1 = str(int(zip1))
    zip2 = str(int(zip2))
    row[-3:-1] = [zip1, zip2]
    # Compute the distance via Google Maps
    try:
      distance = gd.query(zip1,zip2).distance
    except:
      print >> sys.stderr, "Error computing distance:", zip1, zip2
      distance = ""
   # Print the row with the distance
   print comma_separated(row + [distance])
   # Avoid too many Google queries in rapid succession
   time.sleep(random.random()+0.5)
```

23 lines of executable code!

Thanks!



Why do you care about processing data?

- The world is awash in data
- Processing and analyzing it is the difference between success and failure
  - for a team or for an individual
- Manipulating and understanding data is essential to:
  - Astronomers
  - Biologists
  - Chemists
  - Economists
  - Engineers
  - Entrepreneurs
  - Linguists
  - Political scientists
  - Zoologists
  - … and many more!

Programming Concepts

- Variables
- Assignments
- Types
- Programs & algorithms
- Control flow: loops (for), conditionals (if)
- Functions
- File I/O
- Python execution model
  – How Python evaluates expressions, statements, and programs

Data structures: managing data

- List
- Set
- Dictionary
- Tuple
- Graph

- List slicing (sublist)
- List comprehension: shorthand for a loop

$f(x) = x^2$

Functions

- Procedural abstraction
  - avoid duplicated code
  - the implementation does not matter to the client
- Using functions
- Defining functions
- A function is an ordinary value
  - assign to variables
  - in a call, use an expression as the function: myfns[i](arg)
- Method syntax: put first argument before a period (.)
  - arg1.methodname(arg2, arg3)
  - used for “objects”
  - (period also means “look up variable in a namespace”)

Data abstraction

Dual to procedural abstraction (functions)

A module is: operations

An object is: data + operations

- Operations: create, query, modify
- Clients use the operations, never directly access data
- The representation of the data does not matter
- Programmer defines a class.
  Each instance of a class is an object.

Testing and debugging

Write enough tests:
- Cover every branch of each boolean expression
  - especially when used in a conditional expression (if statement)
- Cover special cases:
  - numbers: zero, positive, negative, int vs. float
  - data structures: empty, size 1, larger

Assertions are useful beyond tests

Debugging: after you observe a failure
- Divide and conquer
  - In time, in data, in program text, in development history
  - this is also a key program design concept
- The scientific method
  - state a hypothesis; design an experiment; understand results

Think first
- Be systematic: record everything; have a reason for each action

Data analysis

Statistics

- Run many simulations
- How uncommon is what you actually saw?

Graphing/plotting results

Program design

How to write a function:

1. Name, arguments, and documentation string
2. Tests
3. Body/implementation

How to write a program:

1. Decompose into parts (functions, modules)
   - Each part should be a logical unit, not too large or small
2. Write each part
   - Define the problem
   - Choose an algorithm
   - In English first; test it via manual simulation
   - Translate into code

When necessary, use *wishful thinking*

- Assume a function exists, then write it later
- Can test even before you write it, via a stub

Recursion

- Base case: does all the work for a small problem
- Inductive case:
  - passes the buck for *most of* a large problem
  - does a small amount of work (or none) to the subanswer
  - returns whole result

Speed of algorithms

Affected primarily by the number of times you iterate over data

“Constant factors” don’t matter (looping 2 times or 3 times)

Nested looping matters a lot

Data!

- DNA
- Images
- Social Networks
- Election Results/Polls
- Detecting Fraudulent Data
- Your Choice!

Coming on Monday…

- Financial Crisis
- Music
- Allen Brain Atlas
- Neighborhood Safety
- Olympic Results
- Sports Statistics
- Flight Delays
- Housing Prices
- DNA – Alzheimer’s
- Temperature Change
- School Performance
- Health Outcomes
- Galaxy Identification
- Energy use
- Agricultural & Economic Reports
- Financial Aid
- Student Admissions

There is more to learn

- Data analysis, data science, and data visualization
- Scaling up:
  - Larger and more complex programs
  - "Big data": out-of-memory data, parallel programming, …
- Ensuring correctness
  - Principled, systematic design, testing, and programming
  - Coding style
- Managing complexity
  - Programming tools: testing, version control, debugging, deployment
  - GUIs, user interaction
  - Data structures and algorithms
  - Working in a team

What you have learned in CSE 140

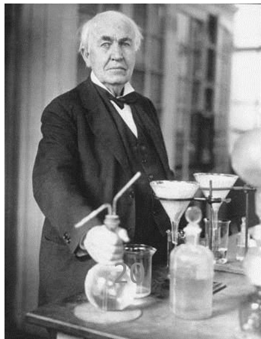
Compare your skills today to 10 weeks ago

- Theory: abstraction, specification, design
- Practice: implementation, testing
- Bottom line: The assignments would be easy for you today
  - This is a measure of how much you have learned

There is no such thing as a "born" programmer!

Your next project can be more ambitious

Genius is 1% inspiration and 99% perspiration.
Thomas A. Edison



What you will learn later

- Your next project can be much more ambitious
- Know your limits
  - Be humble (reality helps you with this)
- You will continue to learn
  - Building interesting systems is never easy
    - Like any worthwhile endeavor
  - Practice is a good teacher
    - Requires thoughtful introspection
    - Don’t learn *only* by trial and error!
    - Get lots of practice *and* feedback

Why the Python language?

	Python	Excel	MATLAB	R	C/C++	Java
Readable syntax	☺	☹	☹	☹	☹	☺
Easy to get started	☺	☺	😐	☹	☹	☹
Powerful libraries	☺	😐	☺	☺	😐	☺

Comparison of Python with Java

- Python is better for learning programming
- Python is better for small programs
- Java is better for large programs

Main difference: dynamic vs. static typing

- Dynamic typing: put anything in any variable
- Static typing:
  - Source code states the type of the variable
  - Cannot run code if any assignment might violate the type

What comes next?

Classes

– Java: CSE 142 (you might skip), CSE 143, CSE 143X
– HDCE 310: Python for Internet mashups
– MATLAB, other programming languages
– Self-study: books & websites

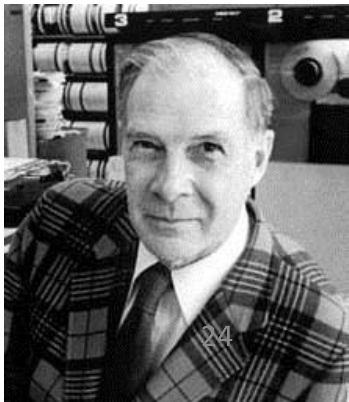
Data analysis: classes, research, jobs

– In programming and software engineering
– In any topic that involves software

Having an impact on the world

– Jobs (and job interviews)
– Larger programming projects

The purpose of computing is insight, not numbers.
Richard W. Hamming
Numerical Methods for Scientists and Engineers



More Computer Science Courses!!

- CSE 142, 143 Programming in Java
- CSE 154 Web Programming
- CSE 373 Data Structures & Algorithms
- CSE 374 Intermediate Programming Concepts & Tools
- CSE 410 Computer Systems
  (Operating Systems & Architecture)
- CSE 413 Programming Languages
  and their Implementation
- CSE 415 Artificial Intelligence
- CSE 417 Algorithms and Complexity

Note: These classes are all open to NON-majors.
You may also be interested in applying for the CSE major!

Go forth and conquer

- System building and scientific discovery are fun!
  - It’s even more fun when your system works
- Pay attention to what matters
  - Use the techniques and tools of CSE 140 effectively