# CSE 160 Wrap-Up

Rob Thompson

UW CSE 160

Winter 2021

## 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 (from nothing!) to solve a real problem

# Join me in thanking your TAs!

| | | | | |
|---|---|---|---|---|
| | | | | |
| Amanda Ong | Austin Jenchi | Kushagra Kumar | Wilson Tang | Joely Nelson |
| She/Her | He/Him | He/Him | | She/Her |
| AF | AC | AB | AC | AA, AE |
| | | | | |
| Zoe Kaputa | Jack Venberg | Niamh Froelich | Brian Zhu | David Chang |
| She/Her | He/Him | She/Her | He/Him | He/Him |
| AA | AB | AE | AD | AD |

## 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
- Anonymous functions
  - Lambda

## 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 to the client
  - Programmer defines a class.
    Each instance of a class is an object.

## Testing and debugging

- Use small data sets to test your <u>program</u>
- 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 ("lost in the woods" analogy)
  - 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. Choose name, arguments, and documentation string
2. Write tests
3. Write 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

## Speed of algorithms

- Affected primarily by the number of times you iterate over data
- Nested looping matters a lot

## Data!

- DNA
- Images
- Social Networks
- Election Results
- 2D points and Handwriting Samples

## What you have learned in CSE 160

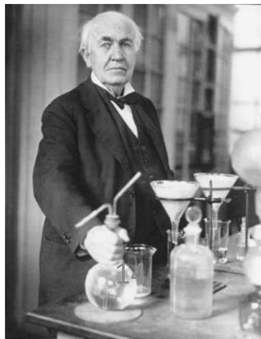
Compare your skills today to 10 weeks ago

- 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



# Python program to assess treatment efficacy

```
# 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!

## 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 160 effectively

## Final Exam

- Topics: Everything in the course up to and including List Comprehensions is fair game.
  - Part 1 - similar to the midterm: writing small functions, although now dictionaries, sets, and tuples could be involved.
  - Part 2 - short answer questions and code writing that involves writing less than an entire function.
- Released by end of day Mon 3/15, due by 11pm Thurs 3/18
- Similar Policies to Midterm – Groups of up to 4