CSE 190p wrapup

Michael Ernst

CSE 190p
University of Washington

Progress in 9 weeks

9 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

A huge thanks to the rest of the staff

Bill Howe

Dun-Yu Hsiao

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!

Python concepts

- Assignments
- Variables
- Types
- Programs & algorithms
- Control flow: loops (for), conditionals (if)
- 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

- Mutable and immutable data structures
  - Immutable: easier to reason about, less efficient
- Distinction between *identity* and *value*

$f(x) = x^2$

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

- 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
    - 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

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 number of times you iterate over data

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

Nested looping matters a lot

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 if, via a stub

There is more to learn

- Data analysis, data science, and data visualization
- Scaling up:
  - larger 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 190p

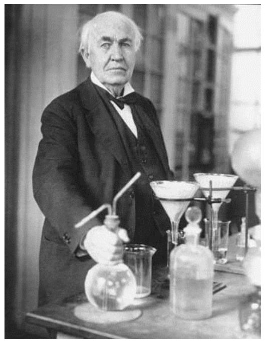
Compare your skills today to 9 weeks ago

- 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

What comes next?

Classes
- Java: CSE 142 (you may skip), CSE 143
- MATLAB, other programming languages

Data analysis: classes, research, jobs
- In software engineering & programming systems
- 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

Go forth and conquer

- System building is fun!
    - It’s even more fun when you build it successfully
- Pay attention to what matters
    - Use the techniques and tools of CSE 190p effectively