CSE 160 Wrap-Up

Ruth Anderson

UW CSE 160

Winter 2020

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

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

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 program
- 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

Bonus Material - Recursion

- Base case: does all the work for a small problem
- Inductive case:
  - Divide the problem, creating one or more smaller problems
  - Ask someone else to solve the smaller problems
    - Recursive call to do most of the work
  - (Maybe) Do a small amount of postprocessing on the result(s) of the recursive call(s)

Speed of algorithms

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

Data!

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

There is more to learn!

- Data analysis, data science, and data visualization
- Scaling up:
  - Larger and more complex programs
  - Algorithm selection
  - “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
  - Graphical User Interfaces (GUIs), user interaction
  - Data structures and algorithms
  - Working in a team

What you have learned in CSE 160

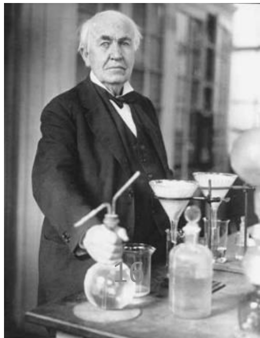
Compare your skills today to 10 weeks ago

- 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



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

What comes next?

Courses:

- Python: CSE 163 Intermediate Data Programming
- Java: CSE 142 (you might skip – your will know a lot of the ideas), CSE 143, CSE 143X
- HDCE 310: Python for interactive systems
- 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

More UW Computer Science Courses!!

You could take any of these now!

- CSE 163 Intermediate Data Programming
- CSE 142, 143, 143x Programming in Java (143x only in fall)
- CSE 154 Web Programming
- CSE 416 Intro to Machine Learning (requires Stat 311/390)
- INFO/STAT/CSE 180 Intro to Data Science (some Math pre-req) (all year)

Require CSE 143:

- CSE 373 Data Structures & Algorithms (all year)
- CSE 414 Databases
- CSE 374 Intermediate Programming Concepts & Tools

Require CSE 373:

- 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 160 effectively