OIM3640 - Problem Solving and Software Design

2021 Fall

Session 26 (12/02)

Today's Agenda

1. Welcome/News/Announcements
2. Social Time
3. Course Review
4. ~~Project Demo~~
5. Session 26
    i. Data Analytics with `pandas`, `numpy`, `matplotlib`
    ii. Demo

Announcements

1. **Assignment 4**: Build a Better Python Community
    i. Due: **Saturday** 12/04
    ii. In case you need help to get more upvotes...
2. Project:
    i. Due: **Next Friday**
    ii. I will be available to meet you anytime next week. **Please sign up office hour timeslot on Canvas.**
3. Grading Scheme
    - Participation and In-Class Exercises/Quizzes:40%
    - Assignments: **30%**
    - Term Project: 30%

~~Social~~ Reflection Time (8 mins)

The last day of class can also be an important beginning.

Please answer the following 3 questions in Google form.

1. Can you identify at least **TWO** important concepts/theories/techniques/ideas/fun things that you learned from this class?
2. How would you **apply** what you have learned from this class to some aspect of your **life/work**?
3. What **question(s)/mystery(-ies)** has the class answered/clarified for you? What are you **still wondering** about?

Progress in 13 Weeks

- **13 weeks ago**:
  - you knew **nothing** or **little** about programming
  - you never used **VS Code/GitHub**
  - you never processed data from **API**
  - you never created a **web application**
- **Today**:
  - well, let us take a look...

Programming Concepts

- Variables, Expressions, Statements
- Types: int, float, string, boolean, None, string, list, dictionary, tuple, set, ...
- Functions
- Control flow:
  - Conditional Statements: `if...elif...else`
  - Iterations: `for`, `while`
- I/O
- Web Framework

Data Structures

- List
- Dictionary
- Set
- Tuple
- Basic operations: slicing, ...
- List comprehension: shorthand for a loop

Functions

- Procedural **abstraction**
  - avoid duplicated code
  - the implementation does not matter to the client
- Using(calling) functions
- Defining functions

Object Oriented Programming

- Class and object
- Attributes and methods
- Inheritance

Testing

- Write **enough** tests:
  - Cover **every branch** of each **boolean** expression
  - Cover **special cases**:
    - numbers: zero, positive, negative, `int` vs. `float`
    - data structures: empty, length of 1, larger, ...
- **Assertions** are useful beyond tests

Debugging

- When you observe a failure/**error**
  - **Divide and conquer**
  - `print()`
- Use **debugging tools**:
  - breakpoints
  - other debuggers
- A more scientific method:
  - state a **hypothesis**
  - design an **experiment**
  - understand **results**
- **Think first**

Program Design

- How to write a **function**
  - Name, parameter(s)
  - Docstring
  - Tests
  - Body/implementation

Program Design (cont.)

- How to write a **program**
  - i. **Decompose** into parts (functions, modules)
    - Each part should be a logical unit, not too large or small
  - ii. Write each part
    - Define the problem
    - Choose an algorithm
    - In English (**pseudo-code**) first
    - Translate into code
- When necessary, use **wishful thinking**
  - Assume a function/module exists, then write it later - *Fake it till make it!*

OK. What you have learned in this class

- Compare your skills **today** to **13 weeks ago**:
  - **Theory**:
    - abstraction, specification, design...
  - **Practice**:
    - implementation, testing, collaboration...
- **Bottom line**: previous assignments look much easier for you today

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

What to Learn Next

- **Data** related:
  - Data analytics, data science, data visualization, machine learning, big data...
- **Scaling up**:
  - larger and more complex programs
  - i.e. *Flask* → *Django*
- **Ensuring correctness**
  - Principled, systematic design, testing, and programming
  - Coding style
- **Managing complexity**
  - More programming tools: testing, version control, debugging, deployment
  - Data structures and **algorithms**

How to Learn Next

- Python learning resources
- GitHub Explore
- Many MOOCs
  - coursera/udacity/edx, e.g. *MIT 6.00.1x*
  - freecodecamp
  - udemy, e.g. *automate the boring stuff*
- Participate in community - SO/Reddit/GitHub
- Working in a **team/open source project**

What *Other* Technologies to Learn Next

- Excel
- HTML/CSS/JavaScript
- Other programming languages
  - Java/Swift/TypeScript/Go/Rust
  - 2021 Stack Overflow Developer Survey
- R
- UI/UX Design
- Being a product manager

Being a Product Manager (PM)

- Understand demand
- Collect demand
- Convert to tasks
- Project management
- ...repeat after launching the product

Being a PM: Front-end PM/Feature PM/UI PM

- Technical skill sets:
  - UI, i.e. Sketch/Figma
  - A/B testing
  - Front-end experience
  - Data Analytics
- Intracting with other roles:
  - UIUX Designer
  - Engineers
  - Data Analyst/Data Scientists
  - User Researchers
  - Users

Being a PM: Back-end PM/PM-Technical

- Technical skill sets:
  - Depending on technologies of the product
    - i.e. cloud infrastructure, database products, APIs, ML algorithm
  - Data
  - Basic Coding/Software Development
- Interacting with other roles
  - UIUX Designer
  - Engineers
  - Data Analyst/Data Scientists
  - User Researchers
  - Users

Data x Python

- Useful libraries:
  - Data analytics:
    - numpy, pandas, scikit-learn, matplotlib, ipython...
    - anaconda
  - Deep learning:
    - `pytorch`, `tensorflow`, ...
- Python for Data Analysis
  - book
  - code

Thank you