CSCI 77800: Ethics and Computer Science

Run of Session for Session #1 (Week #1) – 8/29/2024

Edgar E. Troudt, Ph.D. <et3076@hunter.cuny.edu>

P0: Administrative

1. Course documents will be on Blackboard: https://bbhosted.cuny.edu/.
2. Course Github repository: https://github.com/edgartroudt/778-2024fa/.
3. Bookmark the Zoom link for our synchronous sessions: https://us02web.zoom.us/j/81654051546?pwd=jwaUTPlE73MAGsQvsMgfWpj8agWlkJ.1.
4. Textbook for Python materials:
    a. Downey, Allen. Think Python, 3rd edition. https://greenteapress.com/wp/think-python-3rd-edition/
    b. More direct links to the Jupyter version of the textbook hosted in Google Colab: https://allendowney.github.io/ThinkPython/.
    c. Repository for the ThinkPython textbook code: https://github.com/AllenDowney/ThinkPython. Please note that you can download the actual Jupyter notebooks that comprise the text and run these locally on your own Jupyter installation. (Potential for customizing lessons?)
5. This course tackles ethics in computer science as well as the Python programming language. We aspire to use Python to help us understand ethical dilemmas better.

P1: Quick introductions

1. Who are you? How much Python do you know? If knowledgeable, what IDE do you use for Python?

P2: EthiCS

1. *Pairwise breakout then discussion:* When you think of algorithmic or computer science ethics, of what do you think? What are the major contemporary issues that you've been hearing of in the news?

2. *Structural review:* How is this course set up?
    a. Review syllabus.
    b. Review EthiCS video ASYNC assignment.

3. **Ground rules for discussions:** *We will grow/learn the most if we…*
    a. respect one another's positions.
    b. separate person (and their worth) from their opinions/stance.
    c. listen for understanding.
    d. step into others' shoes.
    e. sometimes play "Devil's advocate" to clarify unfamiliar/distasteful/problematic positions.

P3: Issues are Complex – A Stakeholder Analysis

4. *Stakeholders and Stakeholder analysis:* Should we be creating AI-generated recreations of people (a.k.a. "deepfakes") ethical? Who are the stakeholders and what might they think?
    a. Examine the articles and quotes listed in 5-9. Then, answer the questions in 10.

5. Historical Portrait (2023). "A Message from George Washington to the nation (Recreated with AI)" https://www.youtube.com/watch?v=AeTTVNqysYA.
    a. Historical Portrait: "Step back in time and experience a profound moment in American history as we bring you a message from the Father of Our Country, George Washington."
6. Michel, Arthur H. (2024). "The ACLU Fights for Your Constitutional Right to Make Deepfakes". Wired. https://www.wired.com/story/aclu-artificial-intelligence-deepfakes-free-speech/.
    a. Michel: "States across the US are seeking to criminalize certain uses of AI-generated content. Civil rights groups are pushing back, arguing that some of these new laws conflict with the First Amendment."
7. Cho, Winston (2024). "George Carlin Estate Sues Creators of AI-Generated Comedy Special in Key Lawsuit Over Stars' Likenesses". The Hollywood Reporter.
    a. Cho:"…an AI-generated George Carlin… tackles modern topics… Most subjects came to mainstream prominence after he died in 2008." "…it's explained that the AI program that created the special ingested five decades of Carlin's original stand-up routines, which are owned by the comedian's estate, as training materials…"
8. Bond, Shannon (2024). "A political consultant faces charges and fines for Biden deepfake robocalls". NPR. https://www.npr.org/2024/05/23/nx-s1-4977582/fcc-ai-deepfake-robocall-biden-new-hampshire-political-operative.
    a. Bond: "The Federal Communications Commission proposed a $6 million fine…" "On the other end was a AI-generated voice that sounded like Biden and discouraged Democrats from voting."
9. Macdonald, Cheyenne (2023). "Tom Hanks calls out dental ad for using AI likeness of him". Engadget. https://www.engadget.com/tom-hanks-calls-out-dental-ad-for-using-ai-likeness-of-him-161548459.html
    a. Macdonald: "Tom Hanks posted a warning on Instagram, stating he had "nothing to do" with the video."

10. **Use this worksheet:** NOAA. "Working with People: Stakeholder Analysis Exercise". https://coast.noaa.gov/data/digitalcoast/pdf/stakeholder-analysis-worksheet.pdf.
    a. Select an issue previously mentioned. For example:
        i. Breakout 1: stakeholder groups 1 and 6,
        ii. Breakout 2: stakeholder group 2 and 5,
        iii. Breakout 3: stakeholder group 3 and 4.
    b. Once you've identified at least one stakeholder group, attempt to answer the questions on page #2.

P4: The Python Language

1. Difference between a compiled (C++), interpreted (Python) and bytecode-compiled language (Java).
    a. **Code example:** HelloWorld.cpp
        i. C++ is a compiled language.
        ii. `g++ HelloWorld.cpp ; cat a.out`
    b. **Code example:** HelloWorld.java
        i. Java is a bytecode language.
        ii. `javac HelloWorld.java ; cat HelloWorld.class`
    c. **Code example:** HelloWorld.py
        i. Python is an interpreted language.
        ii. python3 HelloWorld.py
        iii. python3 then type code: `print( "Hello world")`
    d. Will see "transpiled" in 707 if a group discusses Kotlin.
    e. **Optional reading:** https://medium.com/@shishirmohire/compiled-vs-interpreted-vs-transpiled-vs-byte-code-interpreted-languages-92bfc3b3d319
    f. Want to read about the input function? Visit the Python documentation: https://docs.python.org/3/library/functions.html#input

2. How do libraries work?
    a. pandas – great data science library that we will use this semester.
    b. pip – Python package manager (officially: **p**ackage **i**nstaller for **p**ython). https://pypi.org/project/pip/.
    c. **Powershell:** `pip install pandas`
        i. Python package index: https://pypi.org/
    d. **Ubuntu:** `apt list | grep python | grep pandas`
        i. Can also use apt to list available packages.
    e. Libraries are loaded using an include statement. It gets more complex (future lesson).
        i. **Optional reading on include statements and aliasing:** https://www.w3schools.com/python/pandas/pandas_getting_started.asp

P5: Environments and Jupyter IDE

1. I strongly recommend making use of the Jupyter IDE this semester.
2. Open Google Colab to follow along:
    a. https://colab.research.google.com
    b. In the Open Notebook window (or File → Open Notebook) choose "Upload".
    c. Upload my **DynamicTyping.ipynb** notebook to get started.
    d. Please note that CoLab is enabled in our HunterSoE workspace.
3. Try Jupyter is a backup in case Colab does not work for you:
    a. https://jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb
    b. File → New → Notebook will let you follow along, though you may have to copy/paste the **DynamicTyping.py** program into a cell.
    c. Later, feel free to explore https://try.jupyter.org/.

4. Demonstrate Jupyter IDE.
    a. Kernels.ipynb.
    b. A kernel is the center or the brain of software (e.g., the Linux kernel).
    c. Team edited notebooks.
    d. "!" causes the kernel to execute a command on the underlying OS.
    e. Is this safe?
        i. Virtual machines – e.g., VMWare – isolated systems running on top of existing.
        ii. Containerized sandboxes – e.g., Docker – separate environments with limited access to shared system resources.
            1. Read more: Mell, Emily, Pariseau, Beth (2023). "What is container management and why is it important?". TechTarget. https://www.techtarget.com/searchitoperations/definition/Docker.
        iii. Python virtual environments – e.g. VirtualEnv (https://pypi.org/project/virtualenv/) – allows for local libraries limited to current project. Not isolated from underlying system.

5. **Code Example:** DynamicTyping.py.
    a. **Optional reading:** https://stackoverflow.com/questions/2329460/which-languages-are-dynamically-typed-and-compiled-and-which-are-statically-typ
        i. Lists languages that are dynamically typed and compiled and statically typed and interpreted.

P6: Python Practice (Overtime)

1. I prefer that you complete these exercises using the Jupyter notebook from try.jupyter.org. But, should you desire a simpler alternative for today, try this online interpreter: https://www.programiz.com/python-programming/online-compiler/.

2. **Pairwise Exercise 1a:** Input and output operations. Write a program that asks for a name at runtime and then greets that person. (1)
    a. HINT: In other languages we print then read from the keyboard. In Python you can do this in one line with a prompt directly in the input() call.
3. **Pairwise Exercise 1b:** Revise the program above to give a Brady Bunch style greeting (e.g., "Marsha, Marsha, Marsha").
    a. HINT: Python is built for simple data and text manipulation. You can actually multiply strings. Use the "*" operator to multiply the name.

4. **Pairwise Exercise 2:** Numeric/mathematical operations. Write a program that asks for two numbers at runtime and prints the sum and product.
        i. HINT: will need to use type-casting.

5. **Pairwise Exercise 3:** Write a program that asks for a number during the runtime and prints the number preceding and following it.
    a. **PAIRWISE REVISION:** Write a program that pulls a number from the command line and does exactly what the previous example did.
        i. HINT: You will need to use the SYS library and the ARGV list. (Python has its own terms for its constructs. Lists can be accessed like Java arrays using [].)

PROBLEM: You will only be able to run this from the Jupyter console, not from the web-based editor linked above.

6. **Pairwise Exercise 4:** Manually convert a string containing a 4-digit hexadecimal number (e.g., hex_num = “0xABCD”) into decimal.
    a. HINT: You will need to use [] to grab individual characters, += to add values, and ** to perform exponents.)

7. **Pairwise Exercise 5:** Write a program that determines if a number is even or odd.
    a. HINT: Use an if statement, which blocks with “:”, not “{ }”. Indentation matters in Python.

8. **Pairwise Exercise 6:** Write a program that prints the classic left aligned, right triangle from small to large. i.e.:

```
*
**
***
```

    1. Hint: You will need to use the for function, controlled by a range function. Don’t forget the string duplication that we previously learned.

[End SYNC Notes.]

Week 1 HW

P0: Administrative

1. As usual, the expected out of class time for ASYNC + HW is 7.5 hours.
2. Please see Blackboard for ASYNC-WK1 – you are to create EthiCS video #1.

P1: Optional Reading

1. A number of "optional reading" links were embedded in the SYNC notes. As this is a graduate class, it is incumbent upon you to review the links and determine which may strengthen areas for your future professional work. At the very least, take note of the resources available, in case you later decide you want to engage in the reading.
2. If you want to learn more about using Jupyter, Binders and Github repositories, you can view this demo on your own: https://the-turing-way.netlify.app/communication/binder/zero-to-binder.html. (You may wish to read about binders *after completing* this week's HW.)
3. Although interesting, we will not be using Processing.py from https://py.processing.org/ which brings the Processing visual literacy library to Python. You can briefly read about this library.
4. Although interesting, we also will not be using the Thonny IDE (https://thonny.org/); though you can unofficially use it if so desired. You can briefly read about this IDE.
5. And, a reminder that there are web-based code visualizers, should you need these: https://pythontutor.com/. You can briefly familiarize yourself with this again, should it help your understanding later.
6. Finally, you should know that Jupyter Notebooks can get really interactive. See: grasshopperx03 (2023). "Interactive Controls in Jupyter Notebooks". Geeks for Geeks. https://www.geeksforgeeks.org/interactive-controls-in-jupyter-notebooks/.

P2: Required Reading

1. ThinkPython/Downey, Chapter 1: Programming as a way of thinking. https://colab.research.google.com/github/AllenDowney/ThinkPython/blob/v3/chapters/chap01.ipynb
    a. Read the linked subsection on "If you are not familiar with Jupyter notebooks…"
    b. As you read, it's useful to run all of the Jupyter code cells. Remember that the Jupyter notebook keeps track of its entire state. Some later cells may require the variables in cells that preceded it. If you encounter an error in the code, this may be why.
2. ThinkPython/Downey, Chapter 2: Variables and Statements. https://colab.research.google.com/github/AllenDowney/ThinkPython/blob/v3/chapters/chap02.ipynb
3. ThinkPython/Downey, Chapter 3: Functions. https://colab.research.google.com/github/AllenDowney/ThinkPython/blob/v3/chapters/chap03.ipynb

P3: Do/Code

1. This is individual work. Feel free to discuss it with peers after you've completed it on your own.
2. Submit evidence of each of the below exercises working fully correctly. You have the following options:
    a. Submit one document with screenshots that show all of the correct answers and the output.
    b. Or, create a PDF of your final work (File→Print).
    c. You may also want to save a copy of the notebook (in the cloud: File→Save copy in Drive) or (locally: File→ Download → Download ipynb).
3. Chapter 1 exercises –
    a. You can either use the Jupyter box at the top of the exercises or your own Jupyter notebook.
    b. Rounding (indicate your thought, perhaps as comment);
    c. Purposeful mistakes;
    d. Typing (guess then try);
    e. Five simple arithmetic expressions.
4. Chapter 2 exercises –
    a. What goes wrong?
    b. Python as calculator.
5. Chapter 3 exercises –
    a. Try "Ask a virtual assistant". If you have signed up for Github education, you have access to CoPilot both on the Github website and embedded as a plug-in to VS Code. You can also use https://bing.com/copilot. Please be aware of your personal privacy when sending information to AI tools.
    b. Print_right exercise.
    c. Triangle exercise.
    d. Rectangle exercise.
    e. Bottle verse exercise.

P4: Getting Accustomed to Jupyter

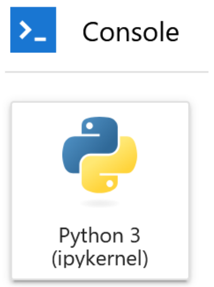
1. The next several sections have you familiarize yourself with common tools and set up your IDE for this semester.
2. For P6, P7, and P8 – submit a single document that contains screenshots that demonstrate that you have completed the final portion of that section. (P5 is just a setup and does not require a screenshot.)
3. Student feedback noted a desire to learn about web-based IDEs since the closing of Replit. This section provides some options.
4. This course's language is Python. It is important to select your preferred IDE for the course.

P5: Github, Environmental Setup, and IDEs

1. In previous semesters, we've noted the benefit of Github for Educators (https://github.com/education/teachers). (There is also a program for learners: https://github.com/education/students). If you haven't already had the opportunity to participate, in this homework we will make use of this. (If you are uncomfortable with signing up, please contact your instructor to waive this portion of the assignment. This will likely make you unable to complete P7 of the homework.)
2. In previous semesters we've made use of the Visual Studio Code IDE (integrated development environment) to write code in Java. This environment will also work for Python. The purpose of this section, though, is to bring you familiarity with other, web-based IDEs. Jupyter Notebooks, in particular, is popular option for Python development.
3. Set up a public Github repository. Write the very simple, but important traditional code: helloworld.py. Commit it to the repository. Please note that since this and the next step is "public", do not put any personal or private information in the program. (Hello world should not contain anything private anyhow.)
4. Although I strongly recommend you practice with your own repo, if you feel more comfortable using my repo, you can use this URL: https://github.com/edgartroudt/binder-test/.

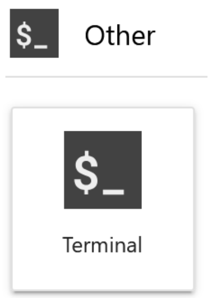
P6: Jupyter and My Binder

1. MyBinder is a project associated with Jupyter. It allows you to build a web-based Jupyter IDE environment that allows manipulation of code within an existing public Github repository.
2. Create a binder of the repo you created in section P0 (or my public repo). Visit https://mybinder.org/.
3. The building process will be slow. It should launch the Jupyter server once it builds the sandboxed container for it.
4. Demonstrate that your ability to run code directly in the interpreter:
    a. Click helloworld.py to open the code. You can use the editor to open and edit this code.
    b. In the Launcher tab, under Console, run Python3 (ipykernel). (If you've closed your Launcher tab, you can reopen it here: File → New Launcher.)



    c. You will now be at a Python command line. You can type Python code directly for the interpreter to run.
    d. Type **`run helloworld.py`** and hit **`SHIFT+ENTER`** to execute.
    e. Type **`print ("hello world")`** and hit **`SHIFT+ENTER`** to execute. **[SCREENSHOT]**

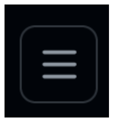
5. Demonstrate that you can run code from the containerized environment Linux prompt:
    a. Reopen the Launcher tab.
    b. Choose the regular, non-Python terminal to launch a Linux shell prompt.



    c. Perform an **`ls`** to show the available files, including helloworld.py.
    d. Then perform a **`python3 hello.py`** to demonstrate that you can launch this program from the prompt. **[SCREENSHOT]**
6. Consider MyBinder as an option for when you need to run web-based coding demonstrations. Although you aren't able to edit the repo, you can create an environment for others to manipulate.

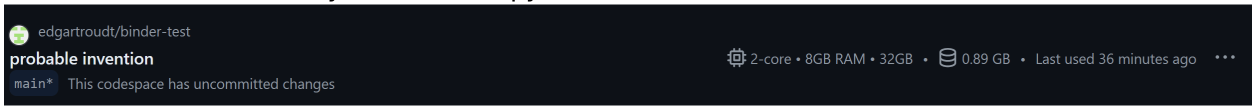
P7: Github's Web-based IDEs

1. We will now demonstrate our ability to use the web-based interfaces of Github.
2. Demonstrate that you can use the web-based editor from Github:
    a. On GIthub.com click the menu button in the upper left corner.



    b. Choose "Codespaces".
    c. Choose the green button "New Codespaces".
    d. Create a code space from this repository.
    e. Demonstrate that you can run your helloworld.py using the online code space editor (a web version of VS Code). **[SCREENSHOT]**

3. Demonstrate that you can use Jupyter via GIthub:



edgartroudt/binder-test
probable invention
main* This codespace has uncommitted changes
2-core • 8GB RAM • 32GB • 0.89 GB • Last used 36 minutes ago ···

    a. From the list of codespaces click the three dots the right of your codespace.
    b. Choose Open in JupyterLab.
    c. You will see that you can perform the same functions as on the Mybinder website. **[SCREENSHOT]**

P8: Installing Python and the Jupyter IDE locally

1. Although the above web-based options are useful for classroom environments, teaching and sharing code, you might benefit the most from installing Jupyter directly on your PC. (It likely is also the fastest and most portable option.)
2. For Windows users, I strongly recommend installing Ubuntu Linux WSL (Windows Services for Linux) from the Windows Store and running Jupyter in that environment. This gives you access to apt for easily installing additional libraries.
3. Windows users may also install Python directly in Windows, though I will only provide limited insight and support for these installations.
4. MacOS users should explore Homebrew and the Apple Developer Tools.

5. Install the Python language:
   a. On Windows, download from: https://www.python.org/downloads/.
   b. On MacOS, download from: https://www.python.org/downloads/macos/.
   c. On Ubuntu/WSL: **`sudo apt install python3`**
6. Install JupyterLab (https://jupyter.org/install)
   a. Instructions for installation via pip (Linux + Windows) or Homebrew (macOS) are available above.

   b. On Ubuntu (recommended):
      i. **`sudo apt install python3-jupyterlab-server`**
      ii. **`jupyter notebook`**

   c. On Windows/MacOS (after Python3 installation):
      i. **`pip install jupyterlab`** (https://pypi.org/project/jupyterlab/; most features).
      ii. **`pip install jupyter`** (https://pypi.org/project/jupyter/; smaller installation).
   d. If you're directed to update pip: You might be directed to update pip: **`python.exe -m pip install --upgrade pip`**.
      i. You will have to locate the executable to run it. If your Python3 installation has the system path set up, this will be easier. On my system, where the path was not correctly set up, I executed it using:
      ```
      C:\Users\edgar\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.12_qbz5n2kfra8p0\LocalCache\local-packages\Python312\Scripts\jupyter-lab
      ```

7. Python has its own package manager.
   a. Remember that APT is the package manager for Linux.
   b. PIP is the package manager for Python. Allows for the installation of libraries.

8. Submit a screenshot of HelloWorld.py running on your own Jupyter setup. **[SCREENSHOT]**