分頁 1

Talk Resources

- **Safety, biosafety, and regulatory anchors:**
  - NIH Guidelines PDF (Apr 2024) https://osp.od.nih.gov/wp-content/uploads/NIH_Guidelines.pdf
  - NIH OSP Biosafety & Biosecurity Policy: https://osp.od.nih.gov/policies/biosafety-and-biosecurity-policy/
  - CDC BMBL hub: https://www.cdc.gov/labs/bmbl/index.html
- **DNA design + assembly planning:**
  - NEBuilder Assembly Tool: https://nebuilder.neb.com/
  - NCBI (PubMed/NCBI portal): https://static.pubmed.gov/portal/portal.fcgi/
  - NCBI BLAST: https://blast.ncbi.nlm.nih.gov/Blast.cgi
  - IDT OligoAnalyzer: https://www.idtdna.com/pages/tools/oligoanalyzer
  - IDT Codon Opt: https://www.idtdna.com/pages/tools/codon-optimization-tool
  - Twist Codon Optimization: https://www.twistbioscience.com/faq/using-your-twist-account/what-does-twist-codon-optimization-tool-do
- **Advanced:**
  - Plasmid screening: https://blog.addgene.org/plasmids-101-plasmid-screening-strategies
  - Molecular Biology Reference: https://www.addgene.org/mol-bio-reference/
- **Protocol development, reproducibility, and sharing**
  - protocols.io: https://www.protocols.io/
  - Benchling: https://www.benchling.com/
  - Opentrons Protocol Library: https://library.opentrons.com/

- **Useful resources borrowed from DNA Read Write Edit**
  - DNA Sequencing at 40: Past, Present, and Future (2017) Shendure, J., Balasubramanian, S., Church, G. *et al.* https://doi.org/10.1038/nature24286 DNA
  - Synthesis Technologies to Close the Gene Writing Gap (2023), Hoose, A., Vellacott, R., Storch, M. *et al.* https://doi.org/10.1038/s41570-022-00456-9
  - Recombineering and MAGE (2021), Wannier T, et al. Nat Rev Methods Primers, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9083505/
  - CRISPR Technology: A Decade of Genome Editing is Only the Beginning, Wang, Doudna, *et al.,* https://www.science.org/doi/10.1126/science.add8643

Talk Example Questions

Q1. In a real lab-automation workflow, what *most* distinguishes an "AI agent" from a normal scripted pipeline?

a. It always produces deterministic, identical outputs given the same input
b. It can **plan, execute, detect failures, and adapt** using tool feedback (closed loop)
c. It only works if all steps are manual and human-approved
d. It replaces the need for calibration and QC

Q2. Your agent generates an OT-2 protocol, but runs fail intermittently due to bead carryover and inconsistent pellet behavior. What's the best next step for the agent to do *before* changing biology-level assumptions?

a. Increase PCR cycles
b. Randomly adjust incubation times until yields improve
c. Add **instrument-aware constraints and liquid-handling parameters** (mixing strategy, aspiration height, delays, magnet timing) and run a small DOE
d. Switch to a different organism

Q3. A team built: (i) an LLM that drafts protocols, (ii) an OT-2 run that executes them, and (iii) a dashboard that visualizes results. Their system works once, but performance is inconsistent and they can't explain failures. What change most credibly turns this into an *advanced* automation+AI final project?

a. Add retrieval over papers and reagent manuals so the LLM cites sources
b. Add a simulator/digital twin so the OT-2 protocol can be "tested" before runs
c. Add **closed-loop evaluation + iteration**: explicit metrics, automated QC gates, and a repeatable plan that updates parameters based on measured outcomes
d. Add a human-in-the-loop approval step for every generated protocol

Q4. An agent is set up to generate and run enzymatic assembly protocols. Which design choice most increases the chance the system is *advanced and defensible*?

a. Let the agent choose any reagent brands so it can generalize
b. Evaluate success by "did a band appear on a gel?"
c. Use **typed protocol specs + unit checking**, pre-run validation (labware, volumes, mixing/aspiration constraints), and post-run QC metrics that feed parameter updates
d. Maximize autonomy by removing all guardrails once the first run succeeds

Python Basics

Resource:

Programming (in general)
📝**Ethan Itovitch (TA) - HTGAA Pre Bootcamp Python Course**

📝Harvard - CS50: Introduction to Computer Science
📝freecodecamp

Python for Biologists
📝Python for Biologists
📝Python Documentation - The Python Tutorial
📝Automate the Boring Stuff with Python
📝Coursera - Biology Meets Programming: Bioinformatics for Beginners
📝 Rosalind - learning bioinformatics through problem solving

Special Topics in Python
📝Luciano Ramalho - Fluent Python: Clear, Concise, and Effective Programming (££), Example Code
📝Ned Batchelder - Blog: #python

Git and Github
📝Hubspot - An Intro to Git and GitHub for Beginners (Tutorial)

Mathematics and Statistics
📹3Blue1Brown
📹StatQuest with Josh Starmer

Example Questions

Q1. What happens when you run this code?

```
class A:
    x = 1

a1 = A()
a2 = A()
a1.x = 5
print(a1.x, a2.x, A.x)
```

A. 5 5 5
B. 5 1 1
C. 5 1 5
D. 1 1 1

ANS: B

Q2. What does PEP 20 (The Zen of Python) say? Select all that apply.
(Hint: Open a Python shell and run `import this`)

A. Readability counts.
B. Errors should pass easily.
C. Nested is better.
D. Namespaces are one honking great idea -- let's do more of those!

ANS: A, D

Q3. Open a Python shell in your terminal (use Ethan's github tutorial).
Find the **smallest** integer n that satisfies **ALL** of these:

1. 2000 < n < 2500
2. n leaves remainder 6 when divided by 10 (`n % 10 == 6`)
3. n leaves remainder 3 when divided by 7 (`n % 7 == 3`)

ANS: 2026