Space Exploration: A high-schooler's introduction to scientific coding

Product of Lasting Value (PLV) for the CEI Fellowship 2022

Tharindu W. Fernando

Department of Physics

Overview

- This PLV presents a proposed framework and one of many modules for a coding workshop that I will lead and run through the high-school outreach organization STEM Pals (http://stempals.club) that is sponsored by UW MEM-C.

- This workshop is currently under development, and this PLV is but a first steppingstone that is the result of several discussions.

Objective

- To introduce programming to **novice high school students with no prior coding experience** using the programming language Python.
- Python has several applications **invaluable, marketable skill**
  - Scientific and numeric computing (healthcare, STEM research, …)
  - Scripts for automating tasks (financial trading, stock portfolio rebalancing, shipping routes, …)
  - Supporting web development
  - Data analysis and visualization (demand forecasting, STEM, financial sector, etc)
  - Machine learning (robotics, IOT, …)

Desired Outcomes

- Introduce concepts fundamental to **several** programming languages:
  - Printing text
  - Variables
  - Functions and function calls
  - Packages
  - Lists
  - Plotting
  - Arrays
  - Loops
  - Scientific problem modeling

Strategies

- Continued exposure and practice:
  - 2-3 hours/session
  - 2-3 sessions/week
  - 3-4 weeks

- Use consistent **Space Exploration** examples/themes/activities in entire workshop
- Hands-on activities and group discussions
- Solo exercises
  - No required homework (only optional)
- In-person troubleshooting with facilitators
- Fun, personalized capstone project tying all concepts from workshop into an application in **Space Exploration**
- Successful participants receive certificate of completion

- Computer access: mostly in target school labs, but also possibly via a UW computer lab (pending logistics)
- Software access: pre-designed Jupyter notebooks that are accessible online using a web browser, with no extra software installations. Ex: https://www.programiz.com/python-programming/online-compiler/

- Schools recruited via *UW GEAR UP Achievers* (https://depts.washington.edu/gearupac/)
  - Gaining Early Awareness and Readiness for Undergraduate Programs
  - Federal grant (Department of Education)
  - Support middle/high school students from low-income families to access and succeed in college

STEM Pals

- Over 45 (grads, undergrads and postdoc) willing to participate when possible
  - 5 project organizers
- Advising: Danica Hendrickson, 4 faculty (Drs. Xiaosong Li, Ting Cao, Nikolai Tolich and Armita Nourmohammad)
- Coding workshop: I will lead, with a team of 3 others for content development and logistics
  - Others will volunteer once organized

Timeline

- May – August 2022: Module planning
- July 2022: Recruit cohorts for Fall 2022 pilot run
- November 2022: Pilot run (scheduling might be tricky)
- March 2023: Second cohort + Recruit 3-4 cohorts for Summer 2023
- Summer 2023: Run

Workshop:

- Week 1: introduction, pre-workshop evaluation, Jupyter Notebooks, start modules
- Week 2: modules, begin final project (facilitators will approve / help with topic ideas)
- Week 3-4: modules, final project presentations + certificate award, post-workshop evaluation

Modules

1) Flowcharts, representing logic/process flow, pseudocode
2) Jupyter Notebook
3) Printing text
4) Variables and printing variables
5) Functions and function calls
6) Lists
7) Plotting
8) Arrays
9) Loops
10) Scientific problem modeling
11) Final project workshopping

Module Structure

- Concept and motivation
  - Real-life examples of use
  - Analogies, figures, animations
  - Real-life tangible demos if sensible (paper puzzles/games to drive home key broad concepts)
- Python-specific implementation of concept
  - Underlying math (if applicable)
  - Group activity/game
- At the end of the day, what can you do with it? Short-term and long-term? How can it fit in final project?

- Solo exercises involving scientific modeling (live troubleshooting with facilitators).

- Introduce optional at-home exercises.
- Perspective on how modules build off each other so far.

Printing Module

At the most basic level, what do you think ‘printing’ means in the context of coding?

- ❑ Using a printer to put computer code on physical paper
- ❑ Displaying text in the output (on the screen, etc)
- ❑ Using code to ask a program to start a printer
- ❑ Converting images to text using code

printer

At the most basic level, what do you think ‘printing’ means in the context of coding?

- ❑ Using a printer to put computer code on physical paper
- ❑ **Displaying text in the output (on the screen, etc)**
- ❑ Using code to ask a program to start a printer
- ❑ Converting images to text using code

printer

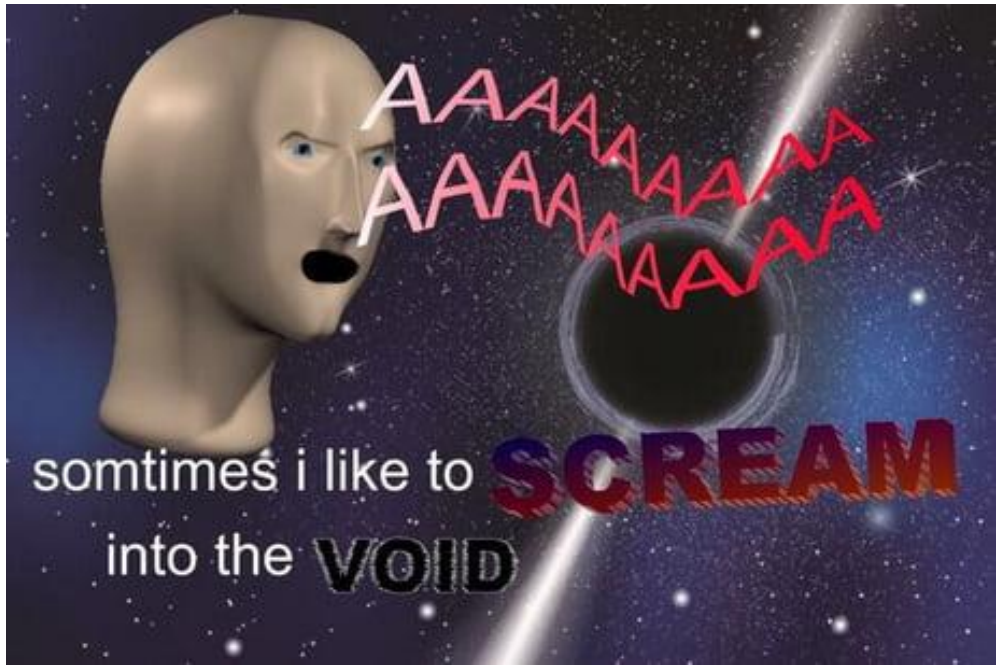
Day 1 at mission control (͡° ͜ʖ ͡°)

- You are asked to introduce yourself to fellow scientists using your name and ID number (ex: Sunil 666) but you won't be screaming it out into the void **all the time** unless you:
  - want to respond to someone who asks you for your name
  - want to independently practice speaking your introduction

- Screaming ↔ Printing

- Ask the program to display something on a screen/file as:
  - **a response to something** (ex: you press a big red button on a machine to start refueling spaceship engines → in response, the screen shows "Commencing refueling protocol.")
  - **standalone text** (ex: the machine simply shows "Please wait until refueling is complete.")

- In general – depending on the coding language – there is no reason for text to appear on its own unless asked to.
- A simple way for code to communicate to humans. Human-code interface.

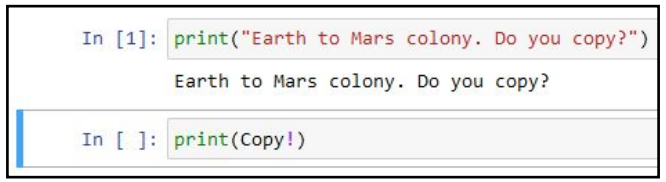


Printing in Python

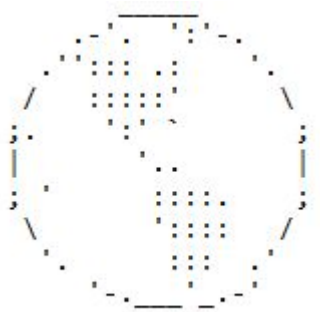
- Function: print(“Earth to Mars colony. Do you copy?”)
  - Must use quotes:
    - NOT: print(Copy!)
  - Can use either “ or ‘ – but be consistent!
    - NOT: print(“FATAL ERROR: Inconsistent use of apostrophes: Initiate spaceship crashing protocol.’)
    - YES: print(‘Okay, stop crashing.’), or print(“YIKES d00dette”)
    - YES: print(“The team scientist Margie asked us to ‘stop being kids’ when Javier complained about using company resources to play Minecraft.”)
  - Repeat text using multiplication sign *: print(5 * “NYOOOM”) and print(5 * “NYOOOM ”)
  - Spaces:
    - Regular space(s): print(“ “) or print(“ “)
    - Regular space with multiplication: print(3 * ‘ ‘)
    - Tab sign \t: print(“\tNYOOOM”) □ SPACES NOT NECESSARY AFTER SIGN. Amount of ‘space’ added by \t depends on font (usually ~4-5).
  - Blank lines:
    - Simply: print()
    - New line character \n for each blank line: print(“\n”) or print(“\n\n\n\n\n\n\n”)
    - Multiplication sign: print(“\n” * 5)
    - New line between text: print(‘Oh no!\nCan you fix this?’)
  - Is your text too long? Break for readability using \:
    - INSTEAD OF: print(“Yoooo, I heard an alien say ‘BOOP BOOP BEEP BEEP BLOPBLOP GOBBLE DOOK POOP BOG BOG BEEP BEEP BLAH GAH BOOP’”)
    - TRY:

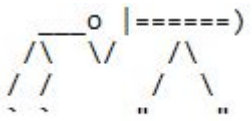
```
print(“Yoooo, I heard an alien say \
‘BOOP BOOP BEEP BEEP BLOPBLOP \
        GOBBLE DOOK POOP BOG BOG \
        BOG BEEP BEEP BLAH GAH BOOP’”)
```

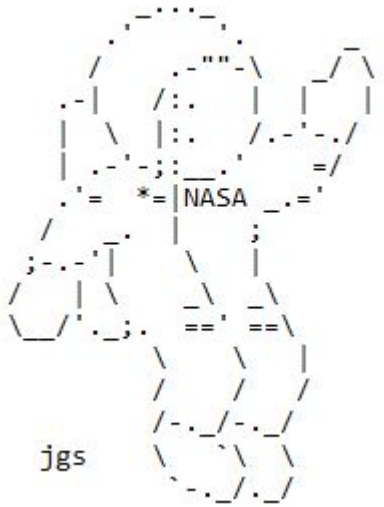
  - What is you want to display “\t” or “\n”?
    - Print raw string: print(r”Haha, I beat the code: \n”)
- **Solo exercise**: Open Jupyter notebook and try all highlighted code, using a new Jupyter cell for each line. Feel free to make modifications to sentences! What do you observe?

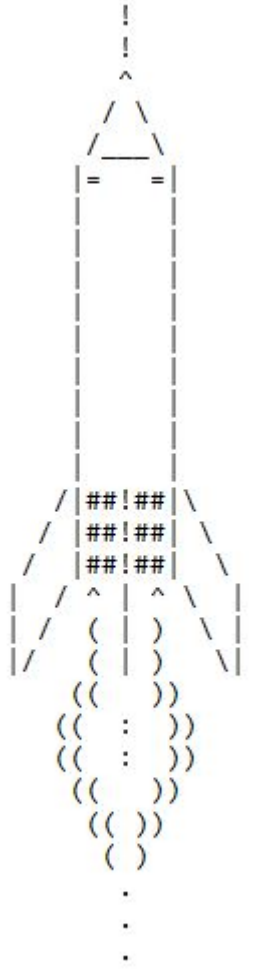


<u>Group Exercise</u>: Using only Jupyter Notebook, make some text art from scratch with your group! One computer per group.









Credit: Google

Next Steps

- We will use printing to help develop a program-human interface that will help with our **Space Exploration**!