# Welcome to CS 2100: Program Design and Implementation

Prof. Rasika Bhalerao

# Rasika Bhalerao, Ph.D.

- (Assistant) Teaching Professor
- Likes:
  - Rock climbing
  - Meditation
- Research areas:
  - Ethical tech
  - Practical/applied AI
  - CS Pedagogy

# In-class participation

Helps students to:

- connect with each other
- learn the material more deeply
- let the instructor know about their understanding.

It is also worth a portion of the grade.

Try it out: https://pollev.com/rasika

Log in with your Northeastern account
(May need to log out of other account)

## Poll: What motivates you? (Choose your top 3)

- Grades
- Learning
- Preparing for job / co-op interviews
- Doing well in the job / co-op
- Doing well in later courses
- Making friends
- Letter of recommendation from instructor
- Enjoying college life
- Addressing societal issues
- Health and well-being

In-class exercises are graded on completion, not correctness.
(And there is no "correct answer" for this poll.)

# Course Structure

- 3 lectures and 1 lab per week
- ~ weekly homework assignments
- 3 codewalks
- 4 quizzes and a final exam
- Office hours and the discussion board

# The AI Policy

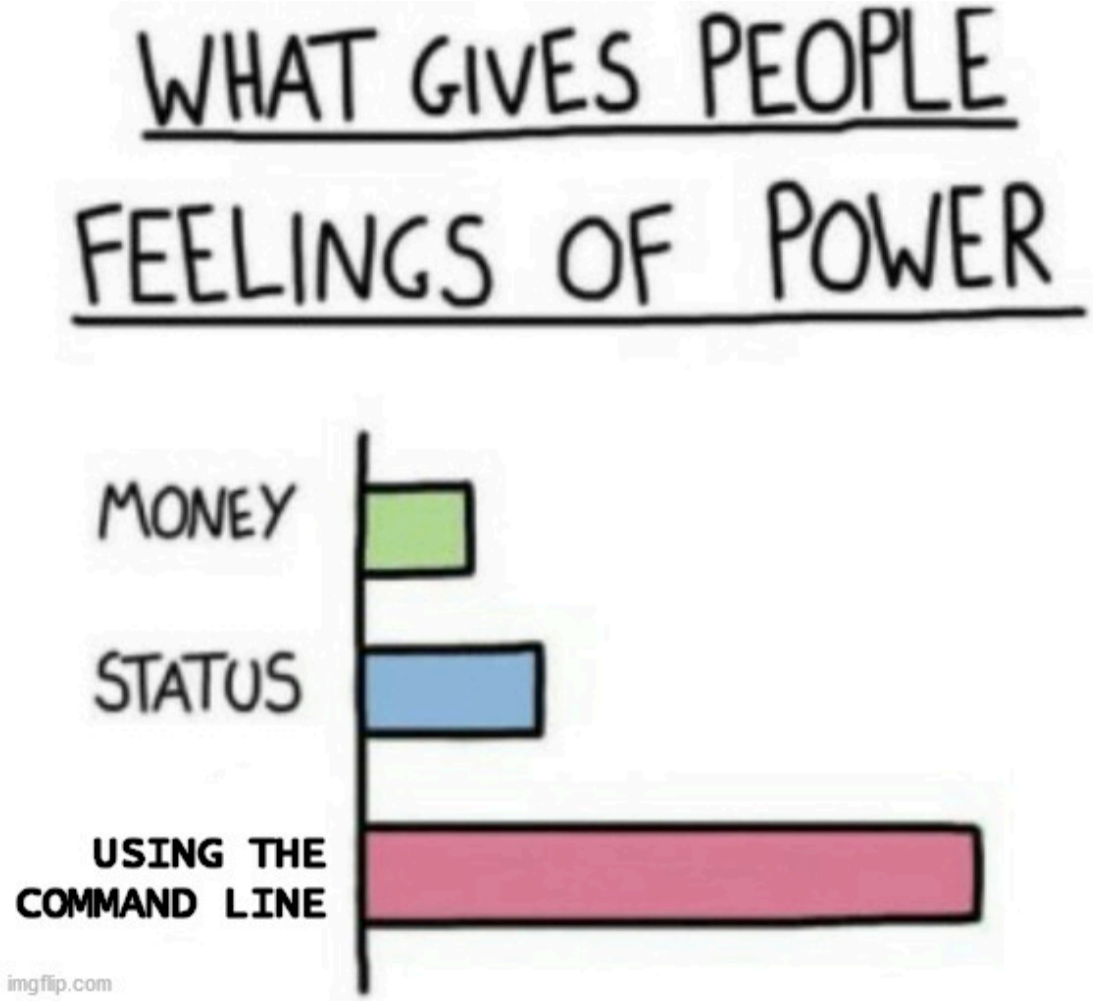
- No AI coding assistants like Cursor, Windsurf, or Copilot
- No chat models like ChatGPT, Claude, or Gemini
- Exception: the AI overview that appears when using a search engine like Google. You may use it to look up documentation, errors, concepts, etc.
- Instructor may ask to meet with the student and have them explain their code
- There are three "codewalks" this semester

## Open ended poll: How do you feel about the AI policy?

# Resources

- Pawtograder (https://app.pawtograder.com)
  - Homeworks and labs
  - Discussion board
  - Grades
- Canvas: Announcements / resources specific to our section
- Git / GitHub: Submitting assignments to Pawtograder
- Lecture notes (https://neu-pdi.github.io/cs2100-public-resources)
- Textbook: "Python 3 Object Oriented Programming: Harness the Power of Python 3 Objects" by Dusty Phillips
  - we will not follow it exactly

# The command line




The command line is a powerful way to navigate a computer.

In this course, we will use the command line to navigate assignment files.

- Mac or Linux: Terminal app
- Windows: Git Bash

# Most popular commands:

| Command | Description | Example |
| --- | --- | --- |
| `ls` | short for “list”; prints the contents of the current working directory | `ls` |
| `cd` | short for “change directory”; moves you to the given directory | `cd ~/Desktop` |
| `pwd` | short for "print working directory" | `pwd` |
| `mkdir` | short for “make directory”; creates a new subdirectory at the current location with the given name | `mkdir nuresources` |
| `touch` | creates an empty file in the current directory with the given filename | `touch lecture8.py` |

## Changing the directory using `cd`

- Using the absolute directory, the complete path from the root to a given directory or file: `cd ~/Desktop/Lectures/2100`
- Using the relative directory, the path from our current working directory to a given directory or file: `cd Lectures/2100` (from the Desktop)

## Poll: The command `cd sp26-rasikabh-hw3/src/data` results in an error. What is NOT likely to be the cause of this error?

1. The current location is not the directory where I store my homework assignments
2. There is no directory called `src` in the directory `sp26-rasikabh-hw3`
3. `data` is a single file, not a directory
4. The assignment submission for Homework 3 is closed

# git

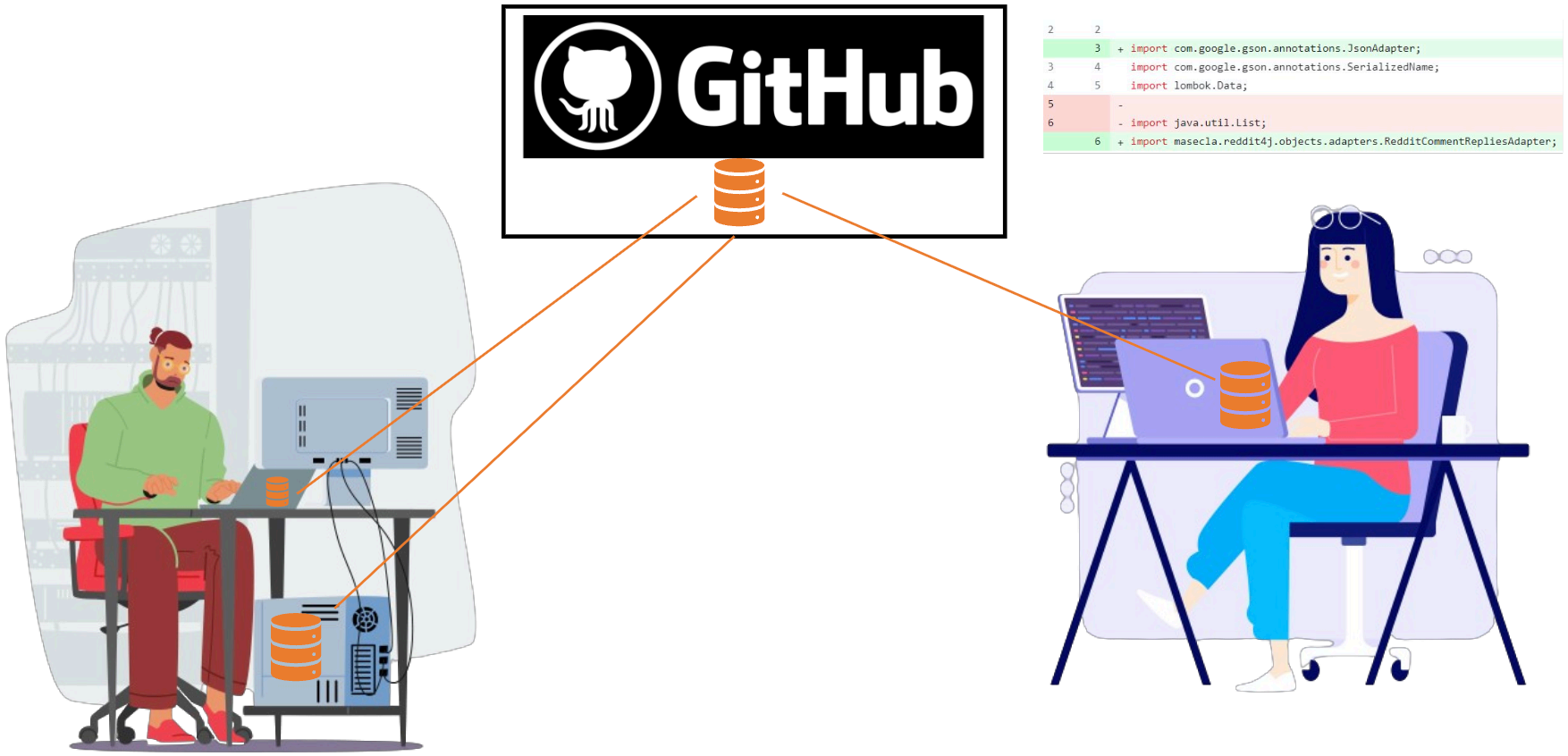
Have you ever wished you could...

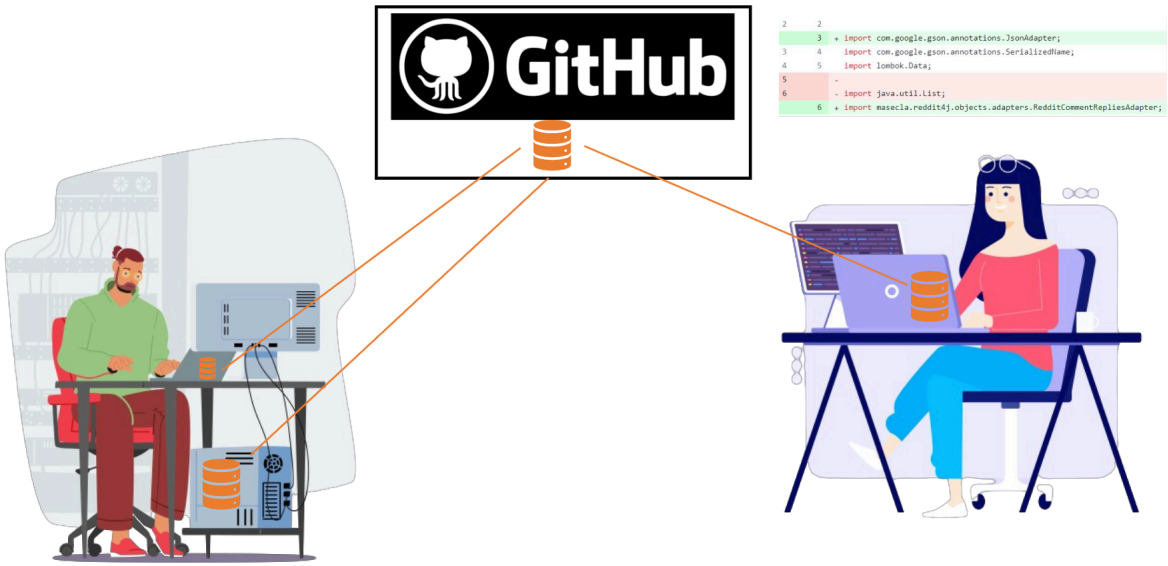
- go back to an earlier working version of a project?
- see what changed between versions of your code?
- tell who made a certain change, when and why?
- switch between using your laptop and desktop to work on a project?
- work with a friend without worrying about overwriting each other's changes?
- have someone review your changes before adding them to a project?
- make some experimental changes with the option of undoing them?
- create your own version of an existing project, change it, and incorporate updates to the original project?

(List source: Ellen Spertus)

**Git** is the leading distributed version control software, created by Linus Torvalds in 2005

**GitHub** is a website owned by Microsoft that hosts git repositories and has a web interface (plus other tools like automated testing and issue tracking)






When you "start" an assignment using Pawtograder, it creates a code repository on GitHub containing the "starter code" for that assignment.

You then use `git clone` to copy that code repository to your laptop, and open / edit the code using VSCode on your laptop.

# Key git concepts

- Repository (repo): a set of code and its history
  - local: on your computer
  - remote: on another computer (like GitHub)
- Commit
  - the codebase at a given point in time (noun)
  - to add a set of changes to the repository (verb)
  - Push: to move code from a local to remote repository

**Tip:** git repositories have a directory in them called `.git` which is invisible by default. To get it to show up when you use the `ls` command, you must do this: `ls -al`

## Locations of versions of code

| Location | Definition | git command to put code there | Postal analogy |
|---|---|---|---|
| working area | code that you are currently writing / saving in VSCode | | Writing on a paper |
| staging area | code that is ready to be commited | `git add .` | Add a stamp and put it in your backpack |
| local repository | code that has been committed | `git commit -m "description"` | Put all stamped cards in the mailbox |
| remote repository | code on GitHub | `git push` | Workers move cards to destinations |

## Most assignments will go like this:

- Accept the assignment on Pawtograder
- Pawtograder creates your GitHub repo.
- Using your command line, navigate to this course's directory and do `git clone <GitHub repo URL (ssh version, not html)>`
- You open the resulting files using VSCode and work on the assignment, saving as you go.
- After each significant chunk of progress on the assignment:
    - `git add .` to stage changes in all files in this directory
    - `git commit -m "descriptive message"` to commit the changes
    - `git push` to push the changes to the online repo in GitHub
        - Pawtograder will automatically take that as your submission (if the submission is still open)

## Other commands

- `git pull` takes changes any that others pushed to the repo on GitHub, and copies them to your local repo
- `git status` reports which files have been changed and staged
- `git diff` shows every changed line
- `git diff --staged` shows the difference between staged and committed changes
- `history` shows the history of commands you typed into the command line

## Poll: Why is this not an ideal commit message? `"Complete Homework 3"`

1. It doesn't describe the code changes
2. It implies that all changes to the entire assignment were submitted in a single commit
3. We can't change Homework 3 again, since we said we completed it
4. All of the above
5. (1) and (2) only

# Code-level Design Practices (style guidlines)

## Why enforce a Style Guide?

- Any code that we write will be read by at least one other person (probably more)
    - Sometimes that "other person" is actually ourselves, years in the future, attempting to use our old code.
- Many employers require code to be verified by at least two other people before it is accepted into the code base
- Helps other people to easily contribute to our codebase
- TA needs to read it to grade it

We use the "official" Python style guide, PEP8: https://peps.python.org/pep-0008

## Style: Variable naming conventions

Names of variables, functions, and modules use `snake_case`: words in lowercase, separated by underscores (_).

## Pylint

Pylint (https://marketplace.visualstudio.com/items?itemName=ms-python.pylint) checks that our code follows the style guidelines.

Please set up the Pylint VSCode extension using the steps in the Setup Guide.

# Python code with types

Your classmate wants help finding a bug in their code:

```
def get_area_of_rectangle(width, height):
    return width * height

width = '3'
height = 4

result = get_area_of_rectangle(width, height)

print(f'Area of a {width} by {height} rectangle: {result}')
```

Its output is `Area of a 3 by 4 rectangle: 3333`, which is false.

## Unconvinced? How about this code:

```
num1 = input('Please enter a number: ')
num2 = input('Please enter another number: ')
print(num1 + num2)
```

It says that 3 + 5 is 35. We'll revisit that in a bit.

## Python:

- strongly typed language (variables have types)
- dynamically typed language (checks the types for consistency at run time, not compile time)

## We love Python, but...

this makes it hard for introductory learners, and hard to catch bugs in code generated by someone else.

# Python supports putting types in code

```
def get_area_of_rectangle(width: int, height: int) -> int:
    return width * height

width: int = '3'
height: int = 4

result: int = get_area_of_rectangle(width, height)

print(f'Area of a {width} by {height} rectangle: {result}')
```

Python does not enforce the types. (The above code runs exactly the same as before, even after adding the types.)

So we use MyPy to enforce the type checking.

# MyPy

Please set up the MyPy VSCode extension using the steps in the Setup Guide.

Missing or mismatched types will be reported in the "Problems" tab every time you save or open a file:

- Mac: Cmd + Shift + M
- Windows: Ctrl + Shift + M

# MyPy

If MyPy is set up properly, then this code:

```
def add(num1: int, num2) -> int:
    return num1 + num2

result: str = add(3, 'hi')

def func() -> int:
    pass
```

should result in three errors:

1. `num2` 's missing type
2. `add()` 's returning something other than the promised `int`
3. `result` 's value being an `int` when the variable type is `str`

If MyPy is set up properly, then this code:

```
def add(num1: int, num2) -> int:
    return num1 + num2

result: str = add(3, 'hi')

def func() -> int:
    pass
```

If there is an error about `func()` missing a return, then the arg `--disable-error-code=empty-body` was not specified correctly in the settings.

## Back to the example where 3 + 5 is 35:

```
num1: int = input('Please enter a number: ')
num2: int = input('Please enter another number: ')
print(num1 + num2)
```

Adding the types for `num1` and `num2` prompted MyPy to remind us that the `input()` function returns a `str`, not an `int`.

# Functions require documentation and tests

Write tests to ensure our code works, but also to convince *others* that our code works.

Each function's documentation must include:

- All parameters
- Any returns
- Any errors or exceptions that might be raised

Formatting the documentation properly makes it show up in official places like `str.__doc__` and `help(str)`.

```
def get_area_of_rectangle(width: int, height: int) -> int:
    """Returns the area of a rectangle.

    Parameters
    ----------
    width : int
        The width of the rectangle
    height : int
        The height of the rectangle

    Returns
    -------
    int
        The area of the rectangle

    Raises
    ------
    ValueError
        If width or height is negative
    """
    if (width < 0 or height < 0):
        raise ValueError("Rectangle dimensions cannot be negative")
    return width * height
```

(Same file, continued)

```
class TestArea(unittest.TestCase):
    """Tests for the function get_area_of_rectangle(width: int, height: int) -> int"""

    def test_3_by_4(self) -> None:
        """3 by 4 rectangle"""
        self.assertEqual(12, get_area_of_rectangle(3, 4))

    def test_negative_area(self) -> None:
        """Make sure it raises a ValueError for a negative width"""
        with self.assertRaises(ValueError):
            get_area_of_rectangle(-1, 4)


if __name__ == '__main__':
    unittest.main()
```

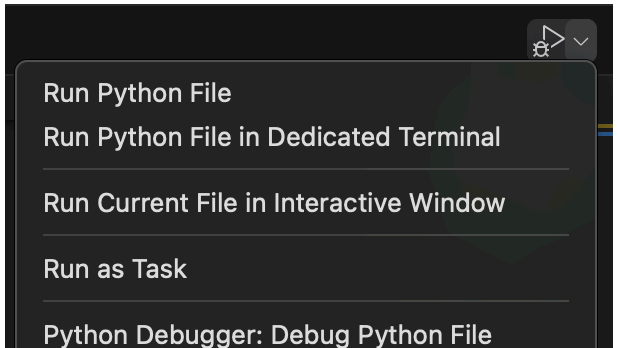
## Poll: My code keeps printing the wrong thing. What should I do?

1. Run it again -- maybe it will magically work this time!
2. Stare at the code for a couple more hours trying random things
3. Use the debugger
4. Nothing -- this is unfixable

# To use the debugger:

1. set a breaking point on at least one line
    - To set a breaking point, click next to the line number. A red dot will appear. (Click the same spot again to remove the breaking point.)
    - When we start the debugger, it will execute all of the code up to (but not including) that line.


2. To start the debugger, rather than clicking the "run" button, use the menu to select "Python Debugger: Debug Python File"

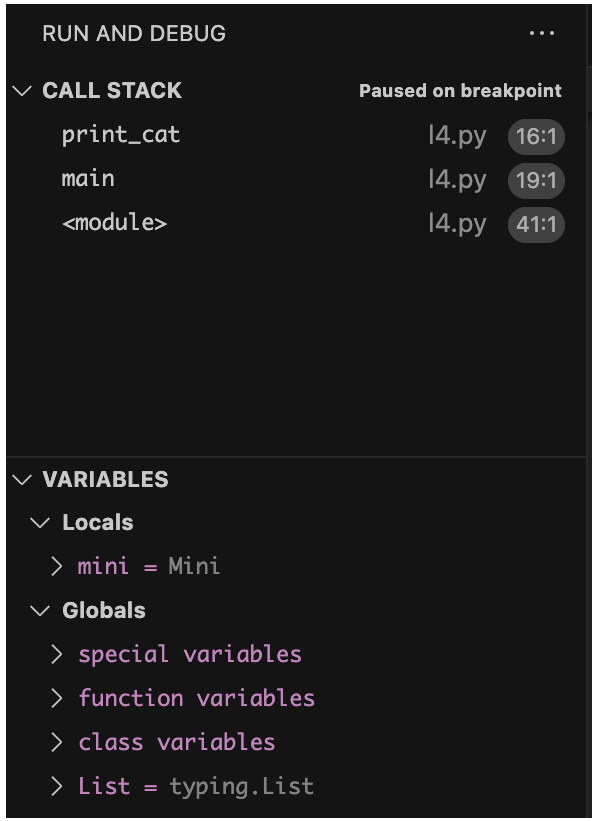


# To use the debugger:

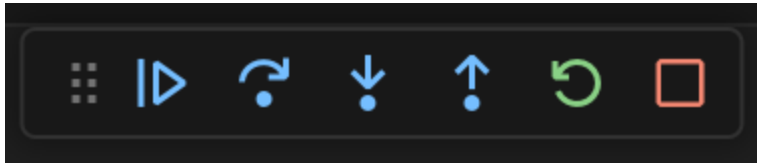
3. Two things will appear: the "Run and Debug" window, and the menu of controls

The "Run and Debug" window contains:

- "call stack": functions that are currently running
- current values stored inside all local and global variables




Called `main()`, and from there, called `print_cat()` (which is currently running, waiting for us to use the menu of controls)

# The menu of controls: 

1. Six dots: drag the menu elsewhere on your screen
2. Continue running the program until the next breaking point (or the end if none left)
3. Run the current (highlighted) line of code.
    - If that line contains a function call, execute the entire function.
4. Run the current (highlighted) line of code.
    - If that line contains a function call, enter that function and run only its first line.
5. Continue running the code in the current function, until we return control to the function that called it. Then stop at the next line in the outer function.
6. Restart the debugger from the beginning of the program.
7. Stop the debugger and close it without running the remaining code.

# Tip:

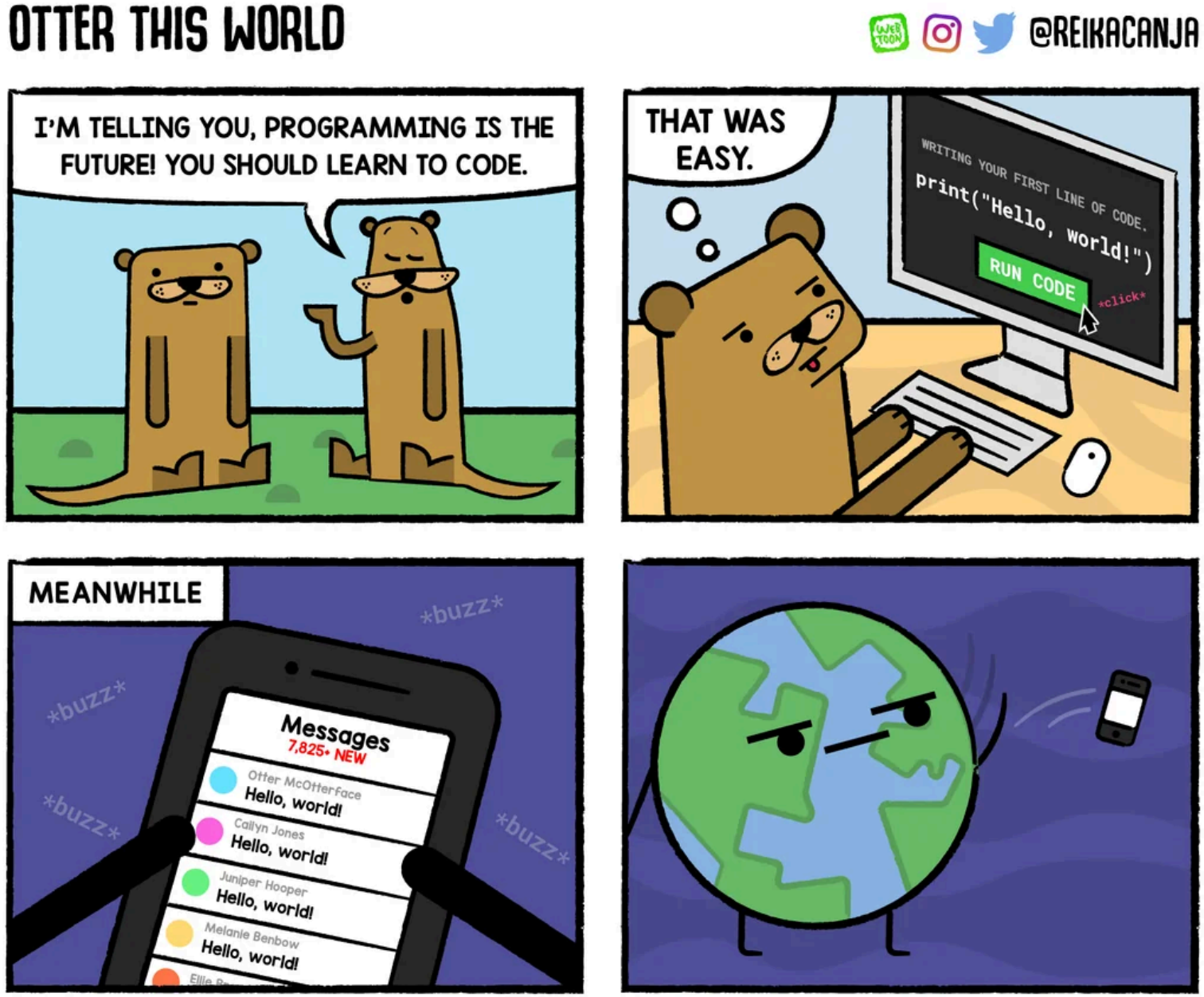
Use tests to debug buggy functions.

1. If there is a function with a bug, write a test that calls that function in a way that is likely to make the bug happen
2. The test will fail at first, which is good because if the test passes, then you know you have fixed the bug
3. Place a breaking point in the buggy function, and start the debugger on the test

# Why Python?

## It's popular

- Employers
- Interviews
- Hundreds of thousands of public Python packages
- Popular among data scientists, web developers, game developers, machine learning engineers, and many others
- Many online resources for learning Python




Source: https://www.reddit.com/r/ProgrammerHumor/comments/64s93u/hello_world_oc

## Poll:

1. What is your main takeaway from today?

2. What would you like to revisit next time?

Source: https://www.reddit.com/r/ProgrammerHumor/comments/64s93u/hello_world_oc