Course Description

This course introduces business students to **computational thinking** and **programming** through Python. You'll learn how to **analyze business problems**, **design solutions**, and **implement them programmatically**.

Along the way, you will:

- Gain **hands-on experience** with **Python** as your primary programming language.
- Learn **Git** and **GitHub** for version control and collaborative development.
- Develop **problem-solving strategies** and apply computational thinking to real-world business problems.
- Complete a **capstone project**: an **equity portfolio management system** that integrates multiple programming concepts.

By the end of the semester, you will be able to:

- Write Python programs to solve real-world business problems.
- Use Git and GitHub effectively for **version control**.
- Apply computational thinking to **design efficient, scalable solutions**.
- Build a working **portfolio management system**.

Course Components

This course is hands-on and cumulative. Each component builds toward your final project.

1. Labs

- **4 labs**, one due every **3 weeks**.
- Labs increase in complexity and culminate in the **Portfolio Management System**.

2. Quizzes

- **6 quizzes**, roughly **one every two weeks**.
- The **lowest quiz grade** will be **dropped**.

3. Ungraded Weekly Assignments

- Designed to **prepare you for labs** and **reinforce concepts**.
- Reviewed in class to address common mistakes.

4. Classroom Participation

- Active engagement is essential.
- **Come prepared** with **at least one relevant question** for every class.

Key Notes & Expectations

1. Staying on Track

- This course is **cumulative** — each topic builds on previous ones.
- Falling behind makes it **very difficult to catch up**.
- **Recommendation:** Dedicate 6 - 10 **hours of independent practice** per week.

---

2. Use of AI Tools

AI tools like **ChatGPT, Copilot, or Gemini** can be helpful, but there are **clear guidelines**:

☑ **Acceptable Uses** *(Learning Aid)*:

- Asking AI to **explain concepts** or **debug error messages**.
- Using AI to **summarize documentation** or **provide examples**, **then writing your own code**.

✕ **Unacceptable Uses** *(Submission Violation)*:

- Submitting AI-generated code or solutions **without understanding them**.
- Copy-pasting entire answers directly into labs or quizzes.
- Using AI during **quizzes or exams** unless explicitly allowed.

Example – NOT ACCEPTABLE:

“Write a program that sums 1 through n.”
AI provides complete code → You submit it directly. ✕

Example – ACCEPTABLE:

“How do I convert user input to an integer in Python?”
AI explains the int() function → *You implement it yourself.* ☑

You must always be able to **explain your work**. If you can’t, it will be treated as a potential **academic integrity violation**.

---

3. Academic Honesty

- Babson College expects **integrity in all academic work**.
- **Cheating, plagiarism, and unauthorized collaboration** are prohibited.
- If evidence of cheating is found, I will **refer the case to Community Standards** and **recommend a failing grade** for the course.

---

4. Deadlines & Late Work

- **Quizzes and labs must be submitted on time.**
- **No late submissions** will be accepted unless you have an **exceptional circumstance**.
- If you anticipate a conflict, **contact me as early as possible**.

---

5. Attendance & Participation

- Attendance is **not mandatory**, but **highly encouraged**.
- Class activities are structured around **your questions** and **collaborative exercises**.
- To get the most from the course, **participate actively**.

Classroom Format

Most classes are divided into three segments:

1. **Reviewing ungraded assignments** – *30 minutes*
2. **Answering student questions & discussion** – *30 minutes*
3. **Building the Portfolio Management System lab** – *30 minutes*

4. **Always bring name card to class**
5. **Always sit in same seat**
6. **Print your full name in the green notebook**
7. **Quizzes and final will be done in the lockdown browser. They are open green notebook only. All entries in the notebook must be hand written.**
8. **Green notebooks need to be turned in with the final**

Setting up coding environment:

Go to github.com and create a free user account with user:

FirstName-babson2025

This guide walks you through setting up GitHub and Codespaces - your cloud-based coding workspace. By the end, you'll have your own repository, a working Python IDE, and the tools needed to write and submit assignments.

Follow each step carefully. If something doesn't look right, ask for help!

Step 1: Log into your GitHub account. This is where your code will live.

Step 2: Open the Canvas assignment link and click 'Accept'.

This creates your own personal copy of the class repository - like getting your own digital notebook.

Step 3: Your new repo will be named:

python-YOUR_GITHUB_USER_NAME

This is your private workspace for the course.

Step 4: Go back to github.com, refresh, and click on your new repo.

Step 5: Click Code -> Codespaces -> Create codespace on main.

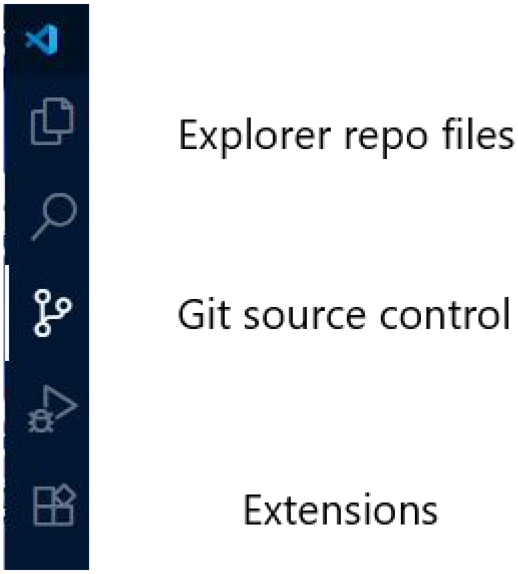
This launches a cloud-based coding environment with your files already loaded.

Step 6: Open Settings: hamburger menu -> File -> Preferences -> Settings.

Step 7: Search for 'git.untrack', change MIXED to HIDDEN, then close settings.

This hides unnecessary Git files so you can focus on your code.

Visual reference for Settings panel:



Step 8: Click Extensions in the left sidebar, search 'Live Server', and install the one by Ritwick Dey.

This lets you preview HTML files in your browser.

Step 9: In the terminal (just once for the course), enter:

```
git remote add upstream https://github.com/babson-org/classroom-week00-python_class.git
```

This connects your repo to the teacher's master copy so you can pull updates.

Git Source Control view in Codespaces shows a commit icon on the upper left. Clicking it

adds a suggested commit message:

What's Going On?

GitHub is your code's home base. When you accept the assignment, GitHub creates a private copy of the teacher's repository just for you.

Codespaces is your coding workspace - like a virtual computer in the cloud. It opens your repo in a full-featured Python editor.

Git tracks changes to your files. When you 'commit', you're saving a snapshot of your work. You'll write a short message each time to describe what you changed.

To keep your repo up to date with the teacher's version, you'll use:

```
git pull --no-edit upstream main
```

This command does three things:

* Connects to the teacher's repo (called 'upstream')

* Pulls in the latest changes from the main branch

* Merges them into your repo without asking for a commit message

** Before running this command, you must commit your changes and sync. If you haven't committed and synced, Git won't let you pull - and you risk losing unsaved work.

** You'll use this command often:

* After your instructor announces updates

Make it a habit:

Commit -> Sync -> **git pull --no-edit upstream main** -> Code

IMPORTANT: Always commit and sync your work BEFORE updating from the master repository.

Setup Checklist

# Accepted the GitHub Classroom assignment

# Created a Codespace on main

# Installed Live Server extension

# Set git.untrack to HIDDEN

# Added upstream remote

# Committed at least once

# NEVER DELETE YOUR REPO

# NEVER DELETE YOUR CODESPACE

Turing Machine

1. Left
2. Right
3. Scan
4. Print
5. Change state
6. Stop

Computational Thinking

1. Abstraction
2. Decomposition
3. Pattern recognition
4. Algorithms
5. Scale

Board Game

Reverse the red and blue circles

HomeWork:

1. What's the minimum number of moves?
2. Explain the strategy as efficiently as you can.
3. Watch Udemy course Intro to python
4. Take notes
5. Prepare questions for class