



PHYS 6260 – Computational Physics (with Python)

Prof. John Wise

Logistics

- **Instructor:** Prof. John Wise, Office: Boggs 1-90D,
 - Email: jwise@gatech.edu
 - **Office Hours:** Thurs 1-2pm
- **Teaching Assistant:** Matteo Reynoso
 - **Office hours:** TBA
 - Email: mreynoso@gatech.edu
- Bring your laptop to every class
- Today we will use it to install Python and the necessary software stack on your laptop

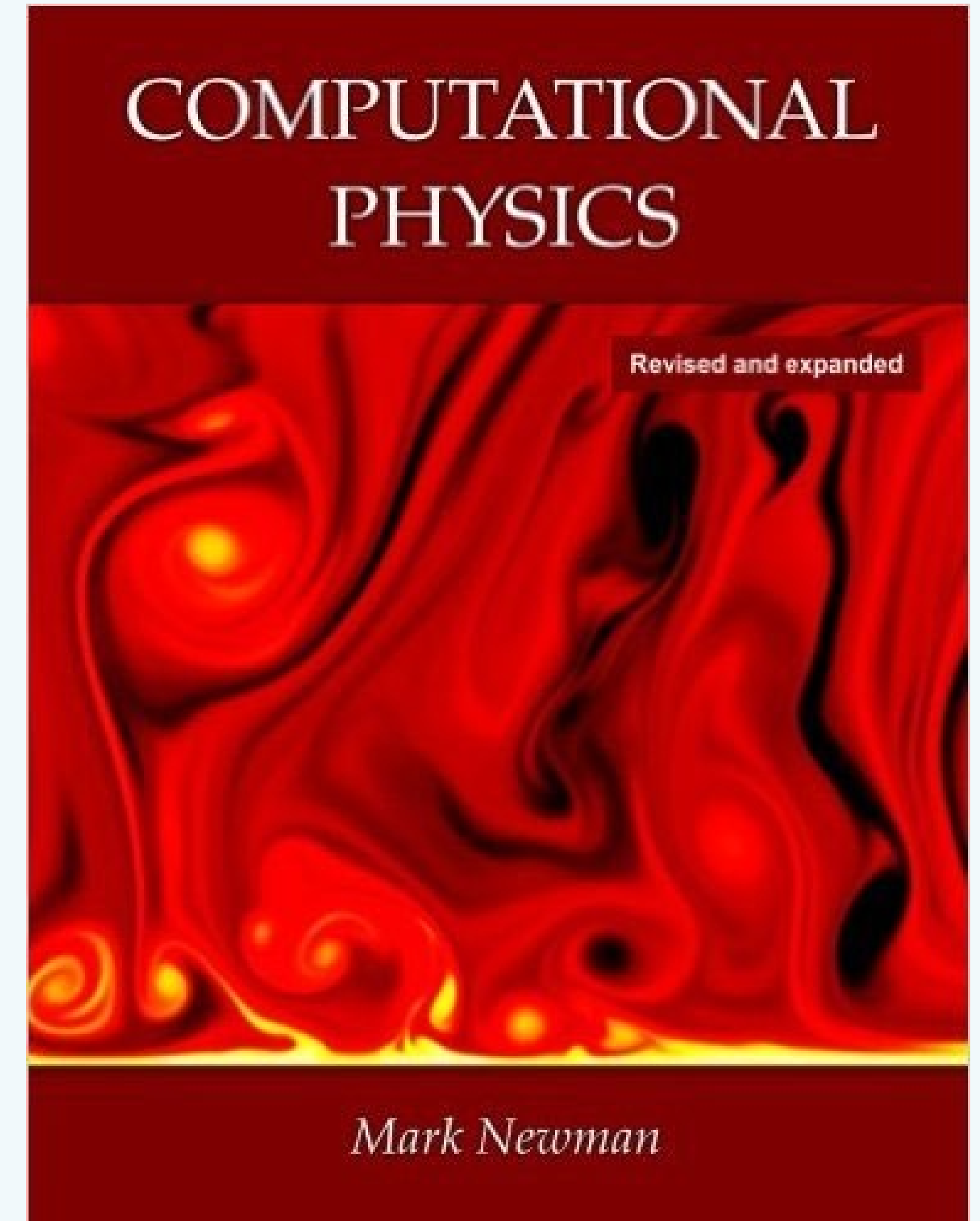
How to reach me

- In-person office hours: Thurs 1-2
- MS Teams chat
 - Faster: Mateo and I will answer your questions as soon as we see them
 - Efficient: Many students have the same concerns and questions
 - DM me if you want privacy
- E-mail: jwise@gatech.edu



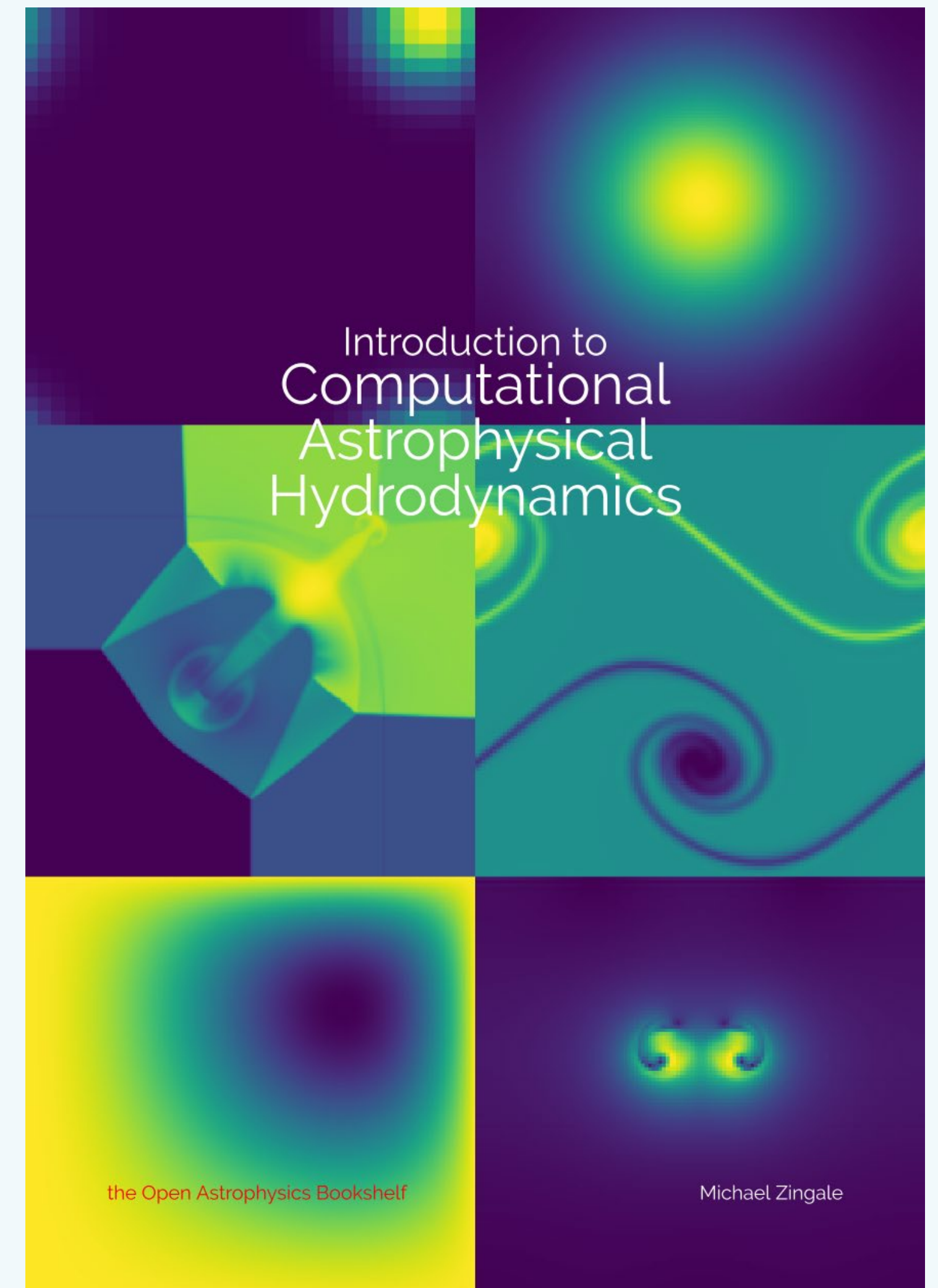
Textbooks

- ***First 4 weeks:** Computational Physics* (2013, “revised and expanded”) by M. Newman
- We will cover the highlights throughout the book.
- All lecture notes (sometimes in the form of notebooks) and recording will be posted on Canvas after class.
- Slides will be posted before class, so you can write notes on them.



Textbooks

- : *Computational Hydrodynamics for Astrophysics* by M. Zingale ([github](#); [PDF](#))
- All other material will come my own lecture notes.
- Optional to get up to speed with Python: [CodeAcademy](#)

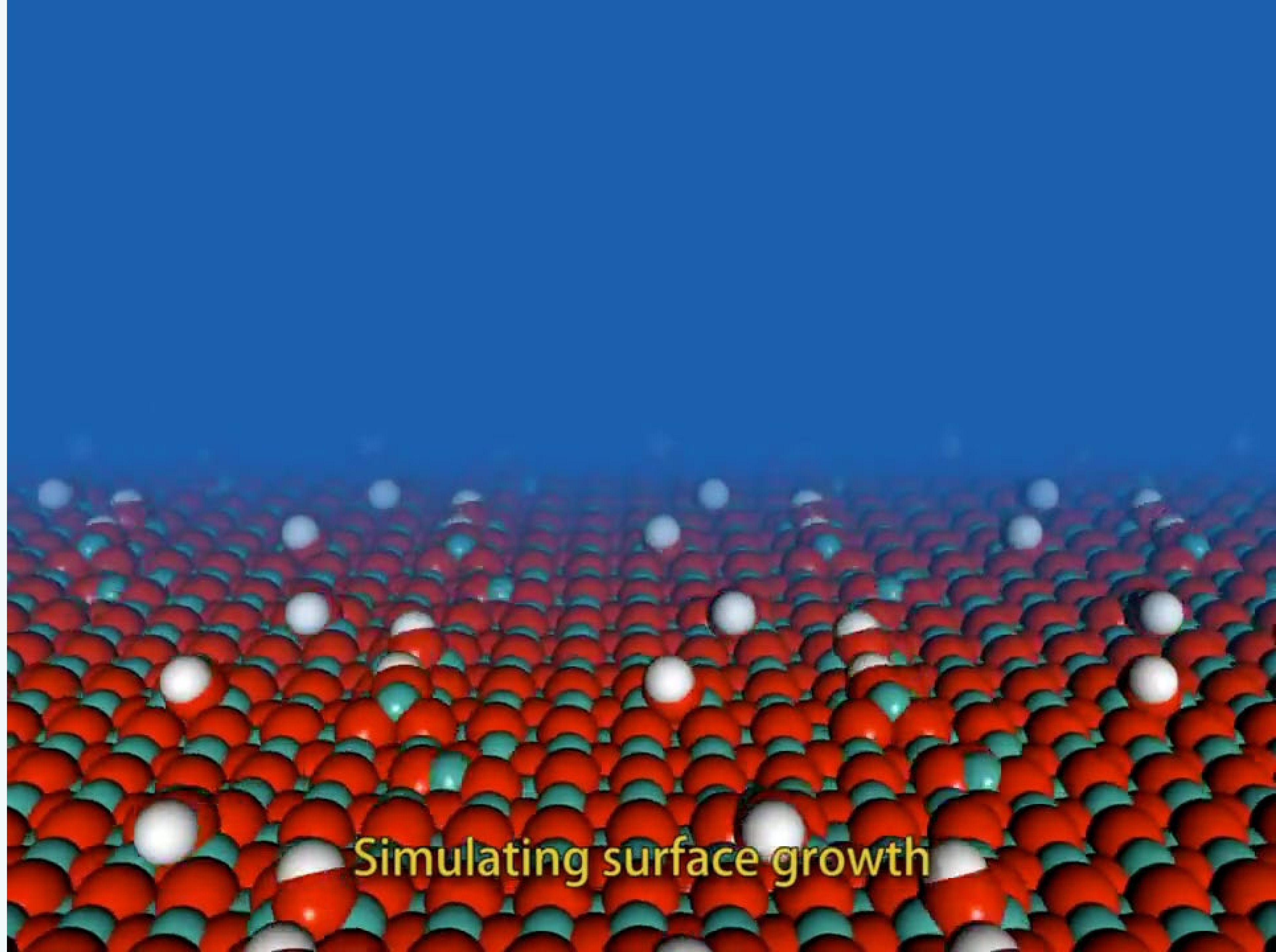


Course Goals

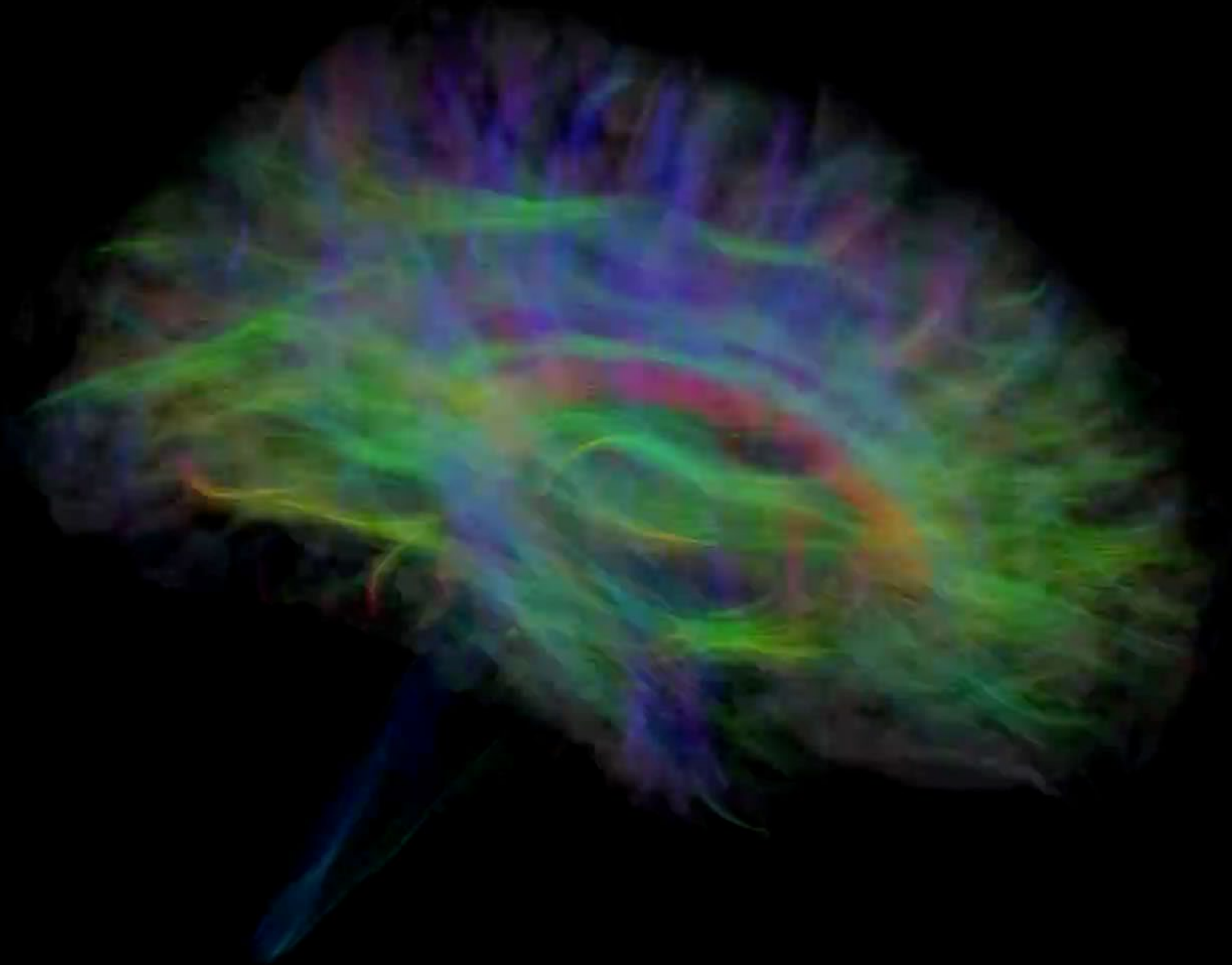
- The purpose of this course is to introduce various numerical methods that are used in solving physics problems
- Class time will be dedicated to covering theoretical aspects of a numerical method and discussing its potential applications
- The homework sets will apply these methods to physics problems
- While the class is not a programming class, you will be exposed to modern programming techniques and expected to reproduce them in exercises and projects
- Example applications include topics such as quantum chromodynamics, classical mechanics, hydrodynamics, astrophysics, biophysics, and material science.

Quantum Chromodynamics

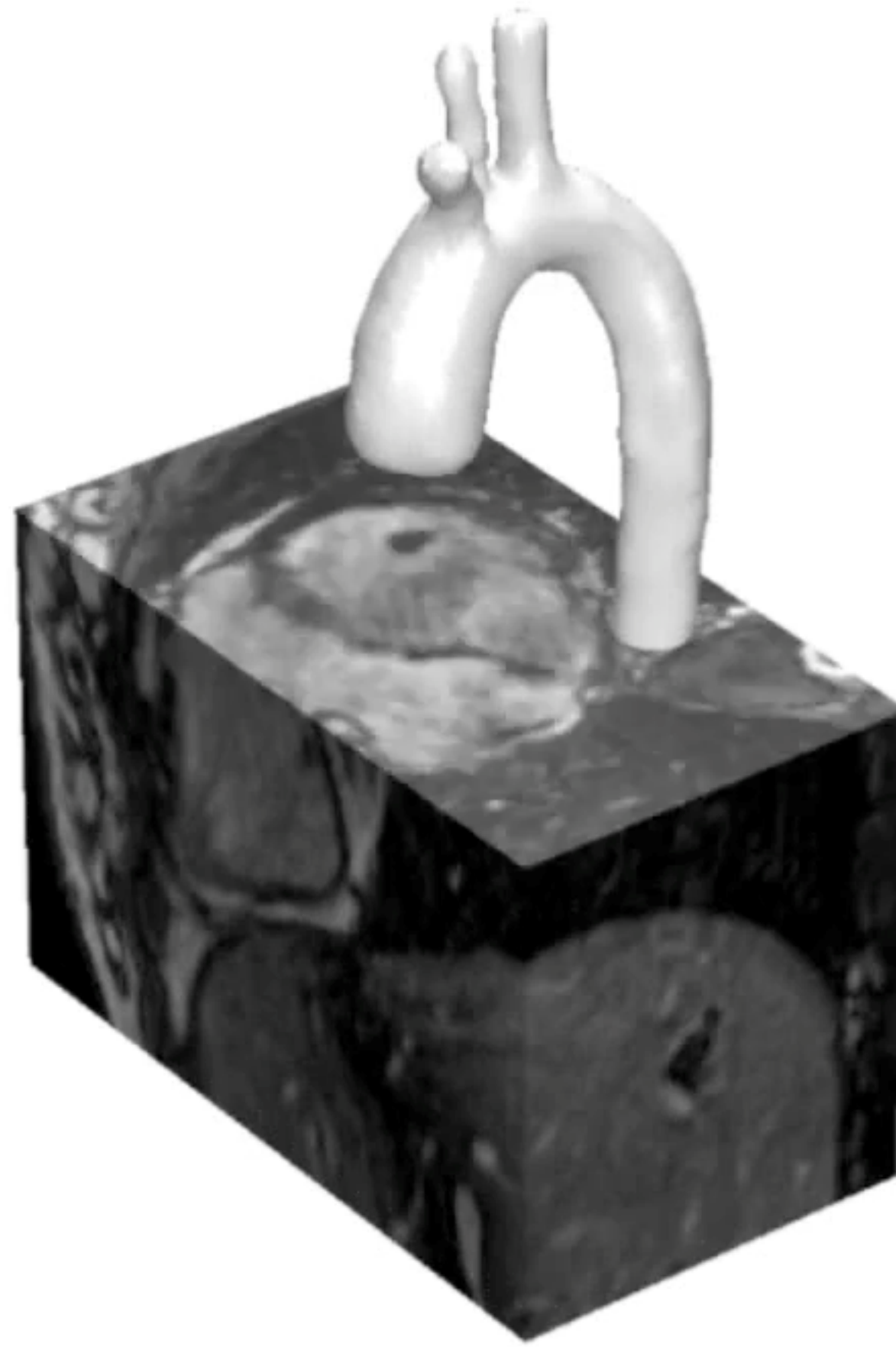
QCD



Simulating surface growth



Patient-specific
aortic segmentation
from MRI data



BIG DATA



Class Modules

1. Introduction to Python, Jupyter notebooks, and GitHub
2. Introduction to Numerical Methods
 - a. Differentiation and integration
 - b. ODEs
 - c. PDEs
 - d. Monte Carlo methods
3. Parallel programming: MPI, Accelerators, HPC
4. Machine Learning: Supervised learning, Deep Learning
5. Applications
 - a. Elliptic solvers
 - b. Transport phenomena (radiation, neutrinos, etc.)
 - c. Molecular dynamics
 - d. Fluid dynamics

Course Grades

- The course grade will be entirely determined from your scores on the homework sets, team project, and class participation.
- There will be no tests or final exam as this class is project-based.
- Rounding to the nearest tenth:
A = >89.5%, B = 79.5-89.4%,
C = 69.5-79.4%, D = 59.5-69.4%,
F = <59.4%
- Course grade division:
 - Homework: 50%
 - Term Project: 45%
 - Participation: 5%

Homework (50%)

- There will be seven homework sets during the semester.
- Homework problems will require writing computer programs in Jupyter notebooks or python scripts based on the numerical algorithms discussed in class.
- In addition, there will be one free response question on concepts discussed during the lectures.
- Programs must be written completely from scratch, with the essential steps fully commented in the notebook markup language.
- However, the structure of the program can be based, if necessary, on programs written or discussed by the instructor.

Homework (50%)

- Students are encouraged to work and discuss problems together, but the programming and written work must be your own.
- The instructor reserves the right to request the student to reproduce the results submitted in homework assignments.
- Delays in the submission of the assignment will be penalized 10% per day up to a maximum of 3 days late.
- All homework submissions are encouraged to be in the form of a Jupyter notebook.
- However, for topics like parallel programming and accelerators, python scripts and associated files are acceptable.

Term Project (45%)

- You will formulate, solve, and present a class project. Either an individual or a group project.
- This is an open-ended project in which the teams will pose a research question, devise a plan, and write a program to explore the system.
- Students are free to choose a particular field of physics -- astrophysics, physics of living systems, condensed matter, non-linear dynamics, etc.
- Students must consult with the instructor on the viability and scope of the question before finalizing it.

Term Project (35%)

- Proposal due: **Friday, February 28** (10% of grade)
 - Students must finalize their research question with the instructor before proposal due date
- Progress report due: **Friday, March 28** (15% of grade)
- Poster presentations: **Monday/Wednesday, April 14/16** (25% of grade)
- Final report due: **Tuesday, April 29** (50% of grade)

Class Participation (5%)

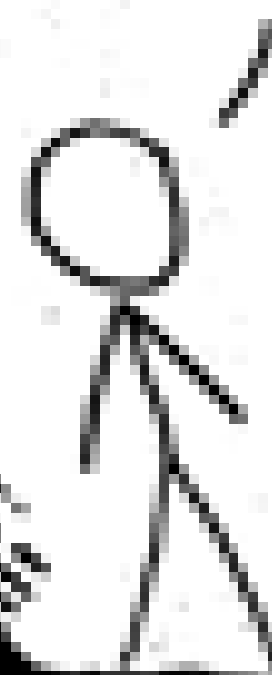
- The lecture periods will involve in-lecture problems and associated discussions. *Please have your Jupyter environment ready to go.*
- Every class period, one or two students will present the results of their in-class assignment.
- Your class participation will be computed based on your completion of an in-class presentation and your engagement during such assignments.

Installing Python (and associated development environments)



PYTHON!

YOU'RE FLYING!
HOW?



Installation of a Linux-like environment

Windows

- It's possible to install a Linux distribution natively in Windows with the Windows Subsystem for Linux (WSL). Be sure to use WSL2.
- Follow these [instructions](#)
- My distribution preference: Ubuntu ([24.04 LTS](#))
- My terminal preference: [Windows Terminal](#)
- X Windows Server (to open Linux GUIs): [VcXsrv](#)

Installation of a Linux-like environment

macOS

- macOS has its root in BSD Unix and has a native terminal (spotlight: Terminal)
- You will have to install a Linux-like package manager and X Windows Server
- Package manager: [Homebrew](#)
- X Windows Server: [Xquartz](#)

Other options: Dual booting and Virtual Machines (personally I don't like VMs for research)

Installing software dependencies

- When initializing a new system, I usually just install packages as-needed to reduce bloat
- You can use either “apt search ...” in Ubuntu (or Debian-based systems) or “brew search” in macOS
- Python: You will need to install a Python environment that’s suitable for scientific computation for this class
- There are a few package managers (e.g. Anaconda, pip, miniconda) for Python that can create conflicting installations

Installing software dependencies

- My advice is to pick one and keep with it.
 - In the past couple of years, Anaconda has gotten better about keeping track of packages installed with pip and conda.
- If you already use one, just install packages with your chosen manager.
- Personally, I use pip. To install in Ubuntu, “sudo apt install python-pip”
- Useful packages to begin: **numpy**, **scipy**, **matplotlib**, **Jupyter**, Jupyter-lab, unyt (for later modules: mpi4py, pycuda, scikit-learn)


Development Environments for Python

- Plain text editor and running scripts in the terminal
 - Please don't use Notepad, Wordpad, or TextEdit
 - Suggestions: Sublime Text, Atom, vim, emacs, **VScode** (my preference)
- Web-based editing and execution: [Jupyter notebook](#) or [JupyterLab](#) (my preference)
 - [Test Jupyter out now here!](#)
- Integrated Development Environments (IDE)
 - [Spyder](#) seems to be popular
 - <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>
 - <http://noeticforce.com/best-python-ide-for-programmers-windows-and-mac>

Anaconda | Individual Edition

anaconda.com/products/individual

www.anaconda.com/products/individual

ANACONDA

Products

Pricing


Solutions

Resources

Blog

Company

Get Started




Individual Edition


Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.


Download



Open Source



Conda Packages



Manage Environments

Keyboard shortcuts are your friends!

- Your fingers never have to move to the trackpad, mouse, or screen.
- Once memorized, your productivity can skyrocket as you fly across the screen.

The background shows a Jupyter Notebook interface with a file browser at the top, a menu bar (File, Edit, View, Insert), and a toolbar. The notebook content includes a title 'The Lorenz attractor', a plot, and several code cells. The first code cell imports matplotlib and ipywidgets. The second code cell imports the Lorenz system from a module. The third code cell defines the Lorenz system. The fourth code cell defines the initial conditions. The fifth code cell defines the time step. The sixth code cell defines the time range. The seventh code cell defines the number of time steps. The eighth code cell defines the time vector. The ninth code cell defines the initial conditions. The tenth code cell defines the time step. The eleventh code cell defines the time range. The twelfth code cell defines the number of time steps. The thirteenth code cell defines the time vector. The fourteenth code cell defines the initial conditions. The fifteenth code cell defines the time step. The sixteenth code cell defines the time range. The seventeenth code cell defines the number of time steps. The eighteenth code cell defines the time vector. The nineteenth code cell defines the initial conditions. The twentieth code cell defines the time step. The twenty-first code cell defines the time range. The twenty-second code cell defines the number of time steps. The twenty-third code cell defines the time vector. The twenty-fourth code cell defines the initial conditions. The twenty-fifth code cell defines the time step. The twenty-sixth code cell defines the time range. The twenty-seventh code cell defines the number of time steps. The twenty-eighth code cell defines the time vector. The twenty-ninth code cell defines the initial conditions. The thirtieth code cell defines the time step. The thirty-first code cell defines the time range. The thirty-second code cell defines the number of time steps. The thirty-third code cell defines the time vector. The thirty-fourth code cell defines the initial conditions. The thirty-fifth code cell defines the time step. The thirty-sixth code cell defines the time range. The thirty-seventh code cell defines the number of time steps. The thirty-eighth code cell defines the time vector. The thirty-ninth code cell defines the initial conditions. The fortieth code cell defines the time step. The forty-first code cell defines the time range. The forty-second code cell defines the number of time steps. The forty-third code cell defines the time vector. The forty-fourth code cell defines the initial conditions. The forty-fifth code cell defines the time step. The forty-sixth code cell defines the time range. The forty-seventh code cell defines the number of time steps. The forty-eighth code cell defines the time vector. The forty-ninth code cell defines the initial conditions. The fiftieth code cell defines the time step. The fifty-first code cell defines the time range. The fifty-second code cell defines the number of time steps. The fifty-third code cell defines the time vector. The fifty-fourth code cell defines the initial conditions. The fifty-fifth code cell defines the time step. The fifty-sixth code cell defines the time range. The fifty-seventh code cell defines the number of time steps. The fifty-eighth code cell defines the time vector. The fifty-ninth code cell defines the initial conditions. The sixtieth code cell defines the time step. The sixty-first code cell defines the time range. The sixty-second code cell defines the number of time steps. The sixty-third code cell defines the time vector. The sixty-fourth code cell defines the initial conditions. The sixty-fifth code cell defines the time step. The sixty-sixth code cell defines the time range. The sixty-seventh code cell defines the number of time steps. The sixty-eighth code cell defines the time vector. The sixty-ninth code cell defines the initial conditions. The seventieth code cell defines the time step. The seventy-first code cell defines the time range. The seventy-second code cell defines the number of time steps. The seventy-third code cell defines the time vector. The seventy-fourth code cell defines the initial conditions. The seventy-fifth code cell defines the time step. The seventy-sixth code cell defines the time range. The seventy-seventh code cell defines the number of time steps. The seventy-eighth code cell defines the time vector. The seventy-ninth code cell defines the initial conditions. The eightieth code cell defines the time step. The eighty-first code cell defines the time range. The eighty-second code cell defines the number of time steps. The eighty-third code cell defines the time vector. The eighty-fourth code cell defines the initial conditions. The eighty-fifth code cell defines the time step. The eighty-sixth code cell defines the time range. The eighty-seventh code cell defines the number of time steps. The eighty-eighth code cell defines the time vector. The eighty-ninth code cell defines the initial conditions. The ninetieth code cell defines the time step. The ninety-first code cell defines the time range. The ninety-second code cell defines the number of time steps. The ninety-third code cell defines the time vector. The ninety-fourth code cell defines the initial conditions. The ninety-fifth code cell defines the time step. The ninety-sixth code cell defines the time range. The ninety-seventh code cell defines the number of time steps. The ninety-eighth code cell defines the time vector. The ninety-ninth code cell defines the initial conditions. The hundredth code cell defines the time step.

Keyboard shortcuts

The Jupyter Notebook has two different keyboard input modes. Edit mode allows you to type code or text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level commands and is indicated by a grey cell border with a blue left margin.

Command Mode (press **Esc** to enable)

Edit Shortcuts

F : find and replace	Shift-J : extend selected cells below
Ctrl-Shift-F : open the command palette	Ctrl-A : select all cells
Ctrl-Shift-P : open the command palette	A : insert cell above
Enter : enter edit mode	B : insert cell below
P : open the command palette	X : cut selected cells
Shift-Enter : run cell, select below	C : copy selected cells
Ctrl-Enter : run selected cells	Shift-V : paste cells above
Alt-Enter : run cell and insert below	V : paste cells below
Y : change cell to code	Z : undo cell deletion
M : change cell to markdown	D , D : delete selected cells
R : change cell to raw	Shift-M : merge selected cells, or current cell with cell below if only one cell is selected
1 : change cell to heading 1	Ctrl-S : Save and Checkpoint
2 : change cell to heading 2	S : Save and Checkpoint
3 : change cell to heading 3	L : toggle line numbers
4 : change cell to heading 4	O : toggle output of selected cells
5 : change cell to heading 5	Shift-O : toggle output scrolling of selected cells
6 : change cell to heading 6	H : show keyboard shortcuts
K : select cell above	I , I : interrupt the kernel
Up : select cell above	0 , 0 : restart the kernel (with dialog)
Down : select cell below	Esc : close the pager
J : select cell below	Q : close the pager
Shift-K : extend selected cells above	Shift-L : toggles line numbers in all cells, and persist the setting
Shift-Up : extend selected cells above	Shift-Space : scroll notebook up
Shift-Down : extend selected cells below	

Close

Configuring Access to GT Compute Cluster

Access to PACE (GT Compute Cluster)

- Everyone should have access to the PACE Instructional Compute Environment ([PACE-ICE](#))
 - [User's guide](#) & Orientation [slides](#)
- Useful for problem sets with parallel computing and accelerators (GPUs)
- The primary benefit will come for your projects if you need more computing power and/or memory

ICE Compute Nodes

- 15GB of backed up storage
- Scratch storage (100 GB) is not backed up
- Access to over 100 computational nodes

Quantity	CPU	Memory	GPU	Local Scratch
30	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	192GB DDR4 2933 MHz		1.6TB NVMe SSD
22	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	192GB DDR4 2933 MHz		1.9TB NVMe SSD
1	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	384GB DDR4 2933 MHz		8TB SAS HDD RAID
5	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	768GB DDR4 2933 MHz		1.6TB NVMe SSD
2	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	768GB DDR4 2933 MHz		1.9TB NVMe SSD
4	Dual Xeon Gold 6248 (40 cores/node, 2.50 GHz)	192GB DDR4 2933 MHz	1x Tesla V100 PCIe 32GB	512GB SATA SSD
4	Dual Xeon Gold 6248 (40 cores/node, 2.50 GHz)	192GB DDR4 2933 MHz	4x Tesla V100 PCIe 32GB	512GB SATA SSD
6	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	192GB DDR4 2933 MHz	4x Quadro Pro RTX6000 24GB	1.6TB NVMe SSD
4	Dual Xeon Gold 6226 (24 cores/node, 2.70 GHz)	384GB DDR4 2933 MHz	4x Quadro Pro RTX6000 24GB	1.9TB NVMe SSD

PACE Cluster Documentation

- Home
- Getting Started ▾
- FAQs ▾
- Storage and File Transfer ▾
- Scheduler ▾
- Hive Cluster Documentation ▾
- Phoenix Cluster Documentation ▾
- Firebird Cluster Documentation ▾
- Buzzard (OSG) Cluster Documentation ▾
- ICE Cluster Documentation ▾
- Open OnDemand ^
 - Open OnDemand Guide
- Firebird Software ▾
- Phoenix, Hive, and ICE Software ▾
- Usage Metrics ▾
- Troubleshooting your Jobs ▾

Updated 2023-05-08

Open OnDemand Instances for PACE's Clusters

Cluster	Link
Phoenix	Phoenix OnDemand
Hive	Hive OnDemand
ICE	ICE OnDemand

★ Must be connected to GT VPN (even if on eduroam)

⚠️

Warning

OnDemand does not have IE11 support and does not work well with Safari. Use Chrome, Firefox, or Edge for best performance.

Introduction

If you require assistance with this system, please contact your course instructor or teaching assistant (TA).


[Home](#) / [My Interactive Sessions](#) / Jupyter

IDEs

 VSCode IDE/Editor

Interactive Apps

[Compute Node Jobs](#)

 Jupyter

 Matlab

 RStudio

 Interactive Shell

[Desktops](#)

 Interactive Desktop

Jupyter version: aba8b02

This app will launch a Jupyter Notebook server on one or more nodes.

Anaconda Module

Anaconda 3 - 2021.05

Quality of Service

Default (none)

Node Type

CPU (first avail)

Nodes

1

Cores Per Node

1

Number of cores (CPUs) per node

GPUs Per Node

0

Memory Per Core (GB)

1

Leave blank if unsure.

Number of hours

1

☐ I would like to receive an email when the session starts

Launch

* The Jupyter session data for this session can be accessed under the [data root directory](#).

- Can launch an interactive environment for Jupyter notebooks through PACE ICE on-demand
- You may already have access to Phoenix or Hive through your research group
- For this class, you can simply use ICE for all PACE calculations (does not require any payment)

VS Code: Easy to use with PACE & Git

GITHUB

PULL REQUESTS

> Local Pull Request Branches

> Waiting For My Review

> Assigned To Me

- #137691: update to windows latest by @TylerLeonhardt
 - Description →

build/azure-pipelines

- win32
 - ! product-build-win32.yml M
 - ! product-build.yml M
 - ! sdl-scan.yml M
- src/vs/workbench/services/search/test/node
 - TS search.integrationTest.ts M

> Created By Me

> All Open

ISSUES

Pull Request #137691 X

update to windows latest #137691 ✎ 📄

Checkout Refresh

Open TylerLeonhardt wants to merge changes into microsoft:main from microsoft:TylerLeonhardt/upgrade-to-windows-latest Created 6 days ago

Reviewers +

- joaomoreno ✓
- raghavthind2005 ✓
- sbatten ✓
- deepak1556 ●

Assignees +

- TylerLeonhardt ×
- gregvanl ×

Labels +

None yet

Milestone +

No milestone

TylerLeonhardt commented 6 days ago

No description provided.

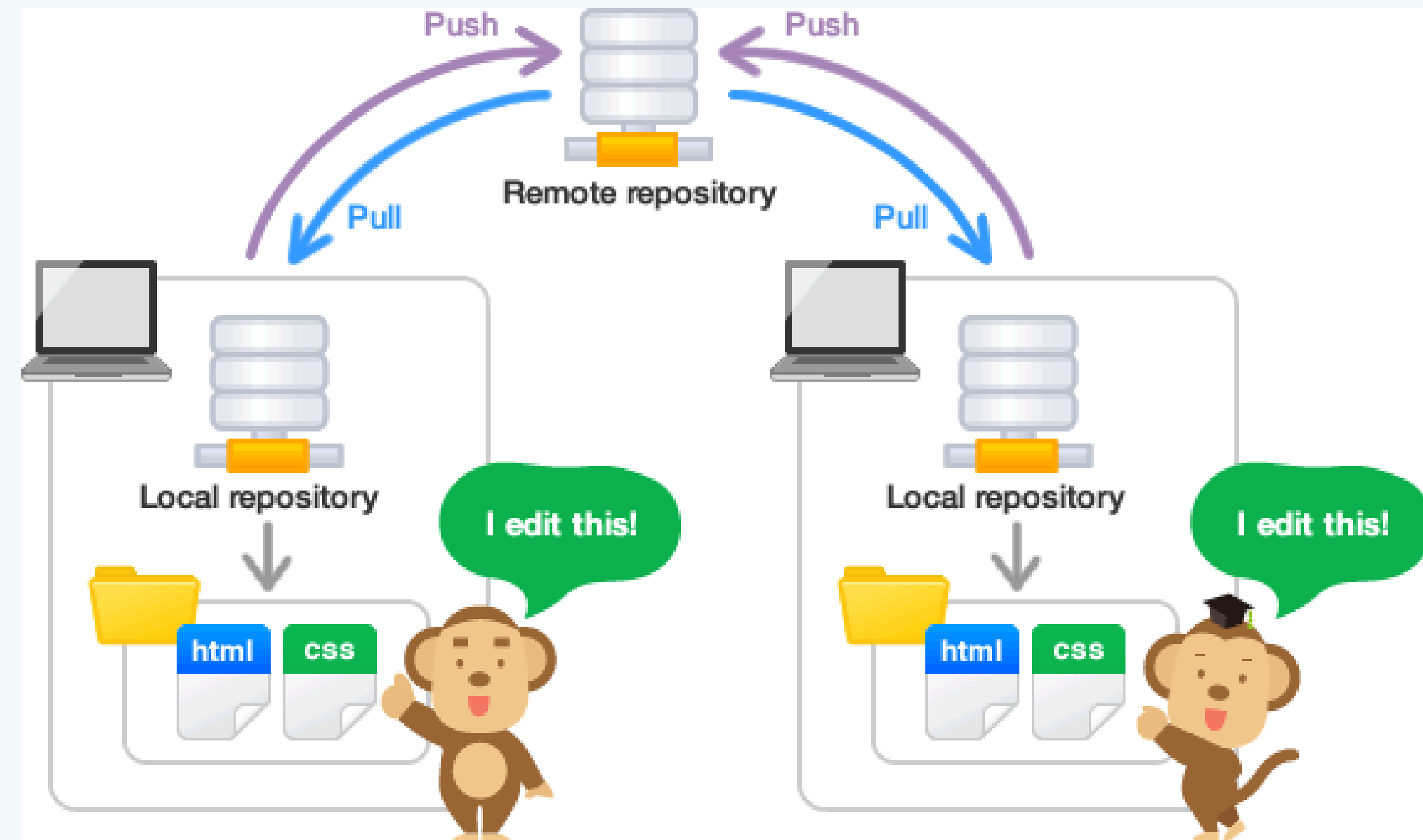
deepak1556 commented 6 days ago

Thanks for looking into this @TylerLeonhardt , adding some data from my attempt previously.

Required use of Github for this course

Distributed version control systems (git)

- You will be required to submit your homework through GitHub Classroom
- You will need to become familiar with git for your homework, and I highly encourage you use it for your project
- This is a distributed version control system
 - There exists a remote server (github.com)
 - These repositories can be cloned onto any machine



For next class ---

- Download the following:
 - VS Code
 - Python environment of your choice (e.g. Jupyter notebook, VS Jupyter extension, spider, etc.)
- We will have an interactive class where we will cover the basics of github and python