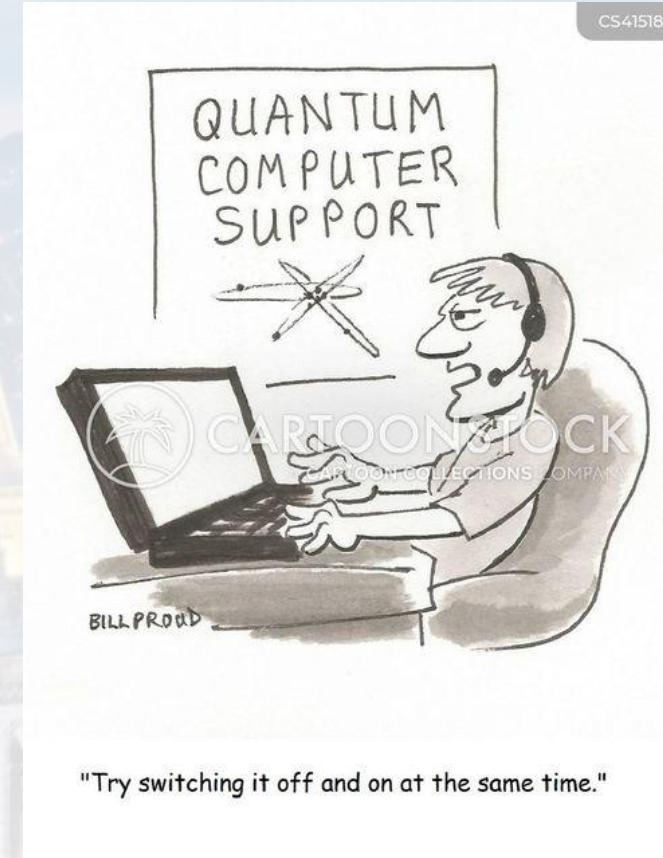




M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics





Lecture: asynchronous

Workshops: WS1 Mo 10:00 – 11:30am [here](#)

Wed 10:00 – 11:30am [here](#)

WS2 Tue 06:00 – 07:30pm [here](#)

Thu 06:00 – 07:30pm [here](#)

Office Hours Lecturer: Fr 03:00 – 04:30pm [here](#)

Instruction begins: Week of June 12th, 2025

Instruction ends: Week of August 14th, 2025

check out: [Google Colab](#) my contact: markus.hohle@berkeley.edu

[bcourse](#) GSI (Sam): Sam D'Ambrosia shda@berkeley.edu

[datahub.berkeley](#)



PHYSICS 77

Introduction to Computational Techniques in Physics

Orientation Syllabus Course Map
 Capstone Project Support Instructors (Physics)

Welcome!

Welcome to Physics 77, Introduction to Computational Techniques in Physics!

This course will introduce you to scientific programming in Python with examples from physics. The course includes lectures, interactive workshops, homework assignments and a Final Capstone Project. A key philosophy in the course is that the best way to learn Python and programming is *by doing*. So let's get started with learning and applying these computational techniques in physics.

Structure of the Course

Some of your learning activities will be **asynchronous**, which you can complete at any time before the due date. These include the **workshops**, office hours) which you will attend at a specific time and place.

(see Orientation)

The **lectures** are prerecorded and the videos can be watched if you click on the corresponding module. We strongly recommend to watch the lecture before you are attempting to solve the homework assignments.

The **workshops** are synchronous events and are held by the GSI and/or the lecturer. The workshops prepare you for the HW assignments and you answer small quizzes. Attendance is mandatory.

During **Office Hours** you can deepen your knowledge about the topics presented during the lecture, give feedback, ask for help or advice. Attendance is optional but highly recommended.

Enjoy the Course!

Helpful Links

1. Introduction to [Canvas platform](#) used for bCourse
2. Don't get lost: [Campus Map](#) Berkeley
3. Access [Zoom](#) via your CalID
4. Seek help when you are depressed and stressed [here](#)
5. You'd like to use [GitHub](#), but don't want to run it via command line
6. Link to download [ANACONDA Navigator](#) or [miniconda](#), the coding

Important Dates and Times (all times are PST)

Lecture

Start: Week of June 9th

End: Week of August 11th

Workshop & Office Hour Instructor (Markus):

Workshop Tuesdays, 6:00 - 7:30pm, [here](#)

Office Hours Fridays, 3:00 - 4:30pm, [here](#)

Discussion/Workshops (Sam):

Monday, Wednesday 10:00 - 11:30am, [here](#)

Thursday 6:00 - 7:30pm, [here](#)



PHYSICS 77

Introduction to Computational Techniques in Physics

Orientation Capstone Project Syllabus Course Map Support Instructors (Physics)

Welcome!

Welcome to Physics 77, Introduction to Computational Techniques in Physics!

This course will introduce you to scientific programming in Python with examples from physics. The course includes lectures, interactive workshops, homework assignments and a Final Capstone Project. A key philosophy in the course is that the best way to learn Python and programming is *by doing*. So let's get started with learning and applying these computational techniques in physics.

Structure of the Course

Some of your learning activities will be **asynchronous** (i.e., lectures, readings, homework assignments), which you can complete at any time before the due date. Other activities will be **synchronous** (i.e., workshops, office hours) which you will attend at a specific time via Zoom ([see Orientation](#)).

The **lectures** are prerecorded and the videos can be watched if you click on the corresponding module. We strongly recommend to watch the lecture before you are attempting to solve the homework assignments.

The **workshops** are synchronous events and are held by the GSI and/or the lecturer. The workshops prepare you for the HW assignments and you answer small quizzes. Attendance is mandatory.

During **Office Hours** you can deepen your knowledge about the topics presented during the lecture, give feedback, ask for help or advice. Attendance is optional but highly recommended.

Enjoy the Course!

Navigation Tips

- Select a section from the drop-down menu below to see the syllabus (use the arrows to the side) to navigate through this syllabus.
- See the [Course Map](#) for a weekly outline.
- See the [Course Summary](#) below for a list of due dates (use the links to the right). If you prefer, you can [set your own time zone](#) to display throughout the course.

Expand All Collapse All

▶ Course Description

▶ Learning Goals

▶ Course Map

▶ Instructor Information and Communication

▶ Course Materials and Technical Requirements

▶ Learning Activities and Assignments

▶ Grading

▶ Strategies for Successful Learning

▶ Course Policies



Navigation Tips

- Select a section from the drop-down menu below to see the syllabus for that section (use the arrows to the side) to navigate through this syllabus.
- See the [Course Map](#) for a weekly outline.
- See the [Course Summary](#) below for a list of due dates (use the links to the right) and other course details.
If you prefer, you can set your own time zone  to display throughout the course.

 Expand All  Collapse All

Course Description

Learning Goals

Course Map

Instructor Information and Communication

Course Materials and Technical Requirements

Learning Activities and Assignments

Grading

Strategies for Successful Learning

Course Policies

Course Map

Exact weeks and dates might differ slightly, depending on progress during class:

Course Map				
Week	Dates	Topics	Reading	Material
1	June 12th	Programming Environment & UIs for Python. Programming Fundamentals	K&N Ch. 1.	Module 1
2	June 19th	Basic Types in Python.	K&N Ch. 1.	Module 2
3	June 26th	Parsing, Data Processing and File I/O, Visualization	K&N Ch.2-3	Module 3
4	July 3rd	Functions, Map & Lambda	K&N Ch. 4, K&N Ch. 6.3-6.4	Module 4
5	July 10th	Random Numbers & Probability Distributions, Interpreting Measurements	DS Sivia Data Analysis	Module 5
6	July 17th	Numerical Integration and Differentiation	Hughes	Module 6
7	July 24th	Root finding, Interpolation	K&N Ch. 6, Newman Ch. 10	Module 7
8	July 31st	Systems of Linear Equations, Ordinary Differential Equations (ODEs)	K&N Ch. 6 Newman Ch. 5	Module 8
9	Aug 7th	Stability of ODEs, Examples	K&N Ch. 6.5 Newman Ch. 6	Module 9
10	Aug 14th	Final Project Presentations		Module 10



<u>Week</u>	<u>Date</u>	<u>Topic</u>
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	<i>June 19th</i>	Basic Types in Python
3	June 26th	Parsing, Data Processing and File I/O, Visualization
4	July 3rd	Functions, Map & Lambda
5	July 10th	Random Numbers & Probability Distributions, Interpreting Measurements
6	July 17th	Numerical Integration and Differentiation
7	July 24th	Root finding, Interpolation
8	July 31st	Systems of Linear Equations, Ordinary Differential Equations (ODEs)
9	Aug 7th	Stability of ODEs, Examples
10	Aug 14th	Final Project Presentations



Mixed Audience

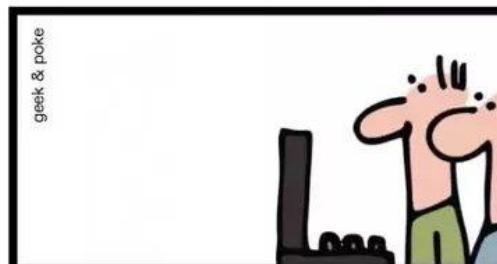
beginner



medium



advanced



```
#include <stdio.h>
int main(void)
{
    int count;
    for(count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
```





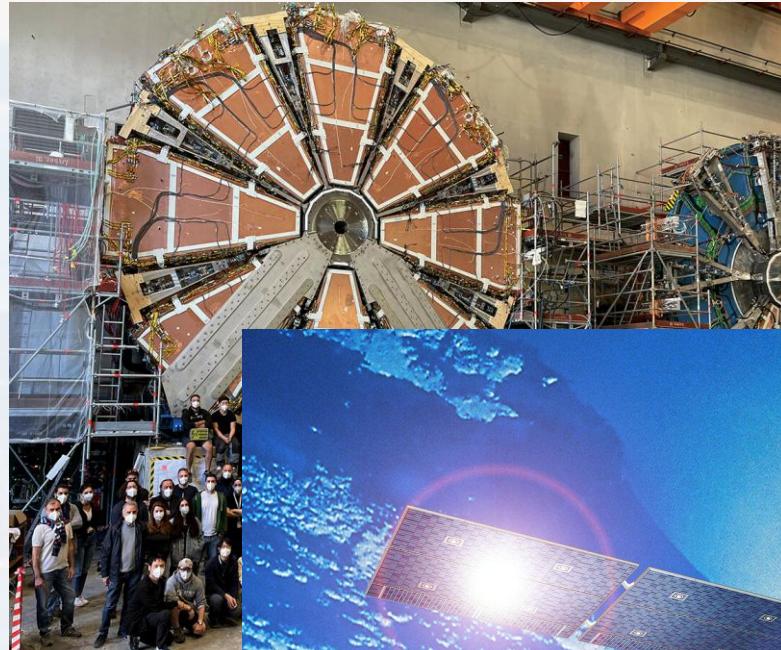
motivation: The good old days:



Salar de Uyuni,
Bolivia, Mar 2003



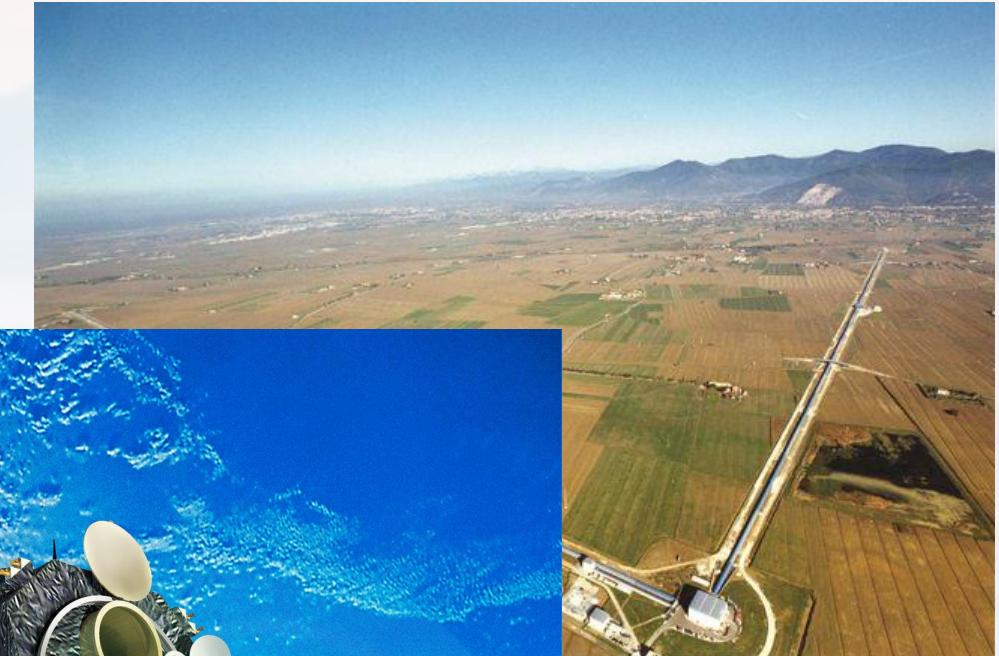
motivation:



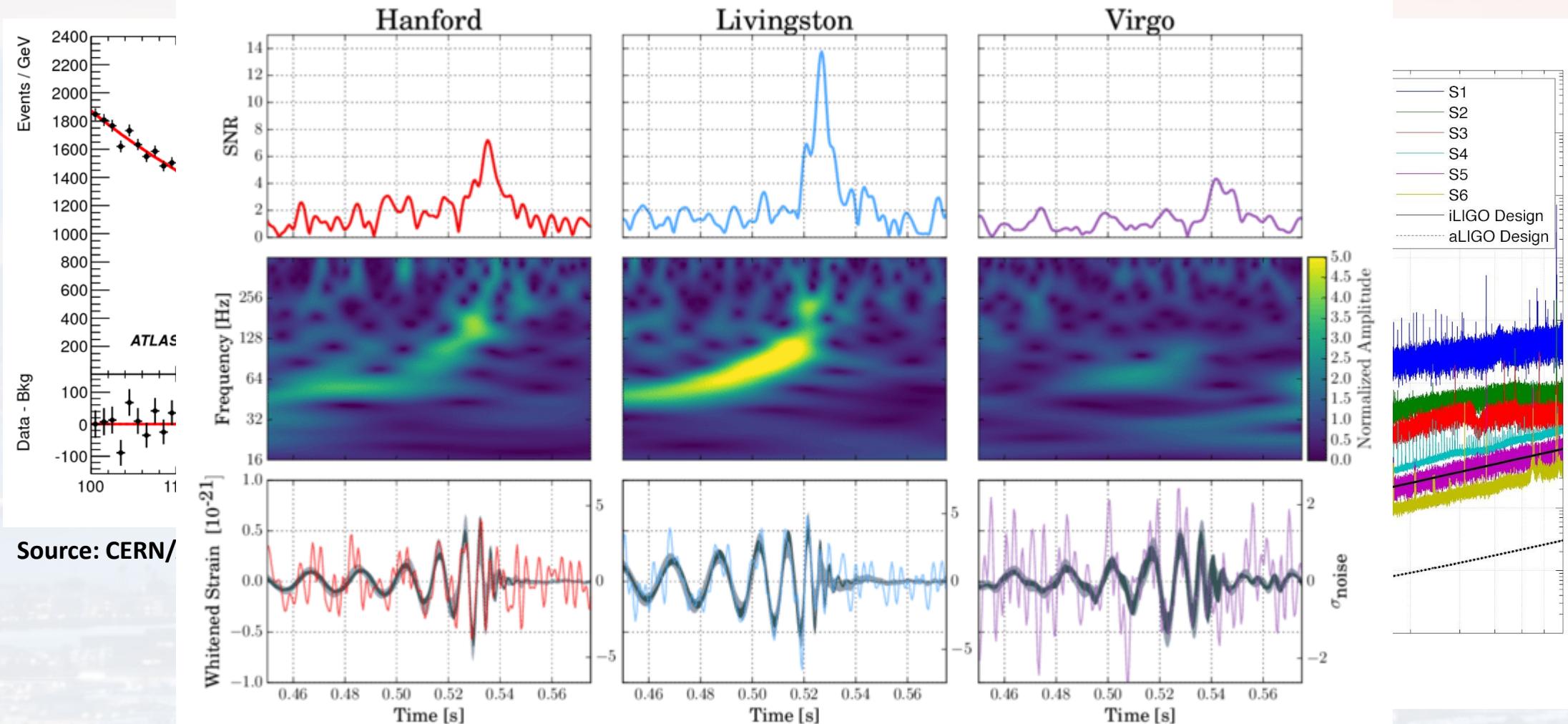
source: CERN



source: ESA



source: LIGO collaboration

**motivation:**



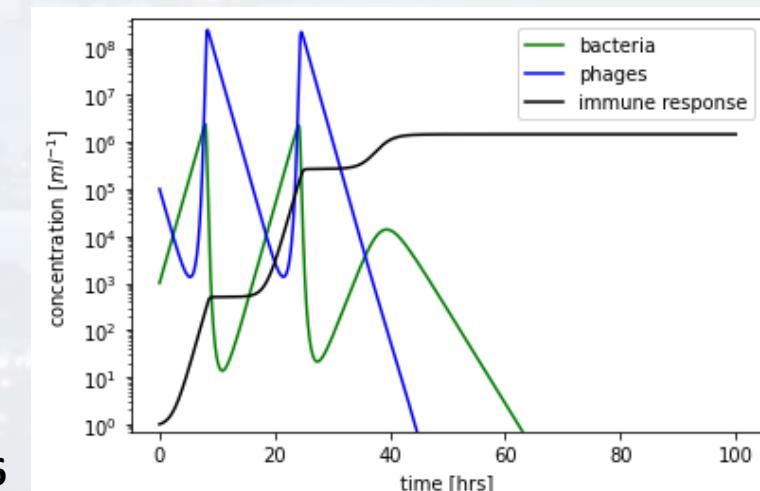
being able to write code like this...

```
@my_timer
class SignalDetect():

    def __init__(self, T, Range_phi = [0, 2*np.pi], dphi = 0.01,
                  MaxM = 20, **Opts):
        L = listdir(getcwd())
        [remove(i) for i in L if '.npy' in i]

    ...
    ...
```

...and simulating systems like that





<u>Week</u>	<u>Date</u>	<u>Topic</u>
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	<i>June 19th</i>	Basic Types in Python
3	June 26th	Parsing, Data Processing and File I/O, Visualization
4	July 3rd	Functions, Map & Lambda
5	July 10th	Random Numbers & Probability Distributions, Interpreting Measurements
6	July 17th	Numerical Integration and Differentiation
7	July 24th	Root finding, Interpolation
8	July 31st	Systems of Linear Equations, Ordinary Differential Equations (ODEs)
9	Aug 7th	Stability of ODEs, Examples
10	Aug 14th	Final Project Presentations



<u>Week</u>	<u>Date</u>	<u>Topic</u>
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	<i>June 19th</i>	Basic Types in Python
3	June 26th	Parsing, Data Processing and File I/O, Visualization
4	July 3rd	Functions, Map & Lambda
5	July 10th	Random Numbers & Probability Distributions, Interpreting Measurements
6	July 17th	Numerical Integration and Differentiation
7	July 24th	Root finding, Interpolation
8	July 31st	Systems of Linear Equations, Ordinary Differential Equations (ODEs)
9	Aug 7th	Stability of ODEs, Examples
10	Aug 14th	Final Project Presentations



Berkeley

UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:

fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter





the environment

programming fundamentals

downloading UI

Spyder & Jupyter

before we can start coding...we need an **environment**

your computer

Operational System (OS)

interface between the computer and all programs,
usually, Windows or Linux



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

your computer

Operational System (OS)

interface between the computer and all programs,
usually Windows or **Linux**

- in Computer Science, Software Engineering, Physics etc:
some version of Linux/Unix
- for Windows:

Windows Subsystem for Linux (WSL), see later





the environment

programming fundamentals

downloading UI

Spyder & Jupyter

your computer

Operational System (OS)

Linux/WSL

Text Editor

- for actual coding
- in principle: any editor is fine

- but: code needs to be **compiled** and **ran**!

before we can start coding...we need an **environment**



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

your computer

Operational System (OS)

Linux/WSL

Coding platform

- ANACONDA (editor + compiler: **Jupyter**, **Spyder** + many other tools)
- miniconda (like ANACONDA but only basics, editor + compiler: **Jupyter**)
- VS Code (editer + compiler)

Text Editor

- for actual coding
- in principle: any editor is fine



the environment

programming fundamentals
downloading UI
Spyder & Jupyter

VS Code (professionals)

before we can start coding...we need an **environment**

```
// our service worker won't work if PUBLIC_URL is on a different origin
// from what our page is served on. This might happen if a CDN is used
// serve assets; see https://github.com/facebook/create-react-app/issue
return;
}

window.addEventListener('load', () => {
  const addEventListener = (method) => addEventListener<K extends keyof WindowEventMap>(method, handler);
  if (isNode) {
    if (applicationCache) {
      // This is async
      check(atob);
      if (AbortController) {
        // AbortSignal
        // AbstractRange
        navigator[ActiveXObject];
        const AnalyserNode;
        const Animation;
        const AnimationEffect;
      }
    }
  } else {
    // Is not localhost. Just register service worker
    registerValidSW(swUrl, config);
  }
});
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Compiled successfully!

You can now view `create-react-app` in the browser.

Local: <http://localhost:3000/>
On Your Network: <http://192.168.86.138:3000/>

Note that the development build is not optimized.
To create a production build, use `yarn build`.



the environment

programming fundamentals
downloading UI
Spyder & Jupyter

Spyder (see later)

before we can start coding...we need an **environment**

The screenshot shows the Spyder Python IDE interface. The code editor window displays a Python script named `howToImplConvLayerMaxpoolFlatt.py`. The script contains code for implementing a neural network layer, specifically a convolutional layer followed by a max pooling layer and a flattening step. The code uses various libraries like `numpy`, `keras`, and `tensorflow`. The console window at the bottom shows the Python and IPython versions being used. The variable explorer window on the right shows the current state of variables in the session.

```
for epoch in range(nsteps):
    #passing data through layer
    Conv1.forward(M,2,1)
    RL1.forward(Conv1.output)
    MP1.forward(RL1.output,0,1,2)
    Conv2.forward(MP1.output,0,1)
    RL2.forward(Conv2.output)
    MP2.forward(RL2.output)
    Conv3.forward(MP2.output,0,1)
    RL3.forward(Conv3.output)
    MP3.forward(RL3.output)
    flattening
    F.forward(MP3.output)
    x = F.output
    dense1.forward(x)
    activation1.forward(dense1.output)
    dense2.forward(activation1.output)
    loss = loss_activation.forward(dense2.output, C)
    predictions = np.argmax(loss_activation.output, axis = -1)
    if len(C.shape) == 2:
        C = np.argmax(C, axis = 1)
    accuracy = np.mean(predictions == C)

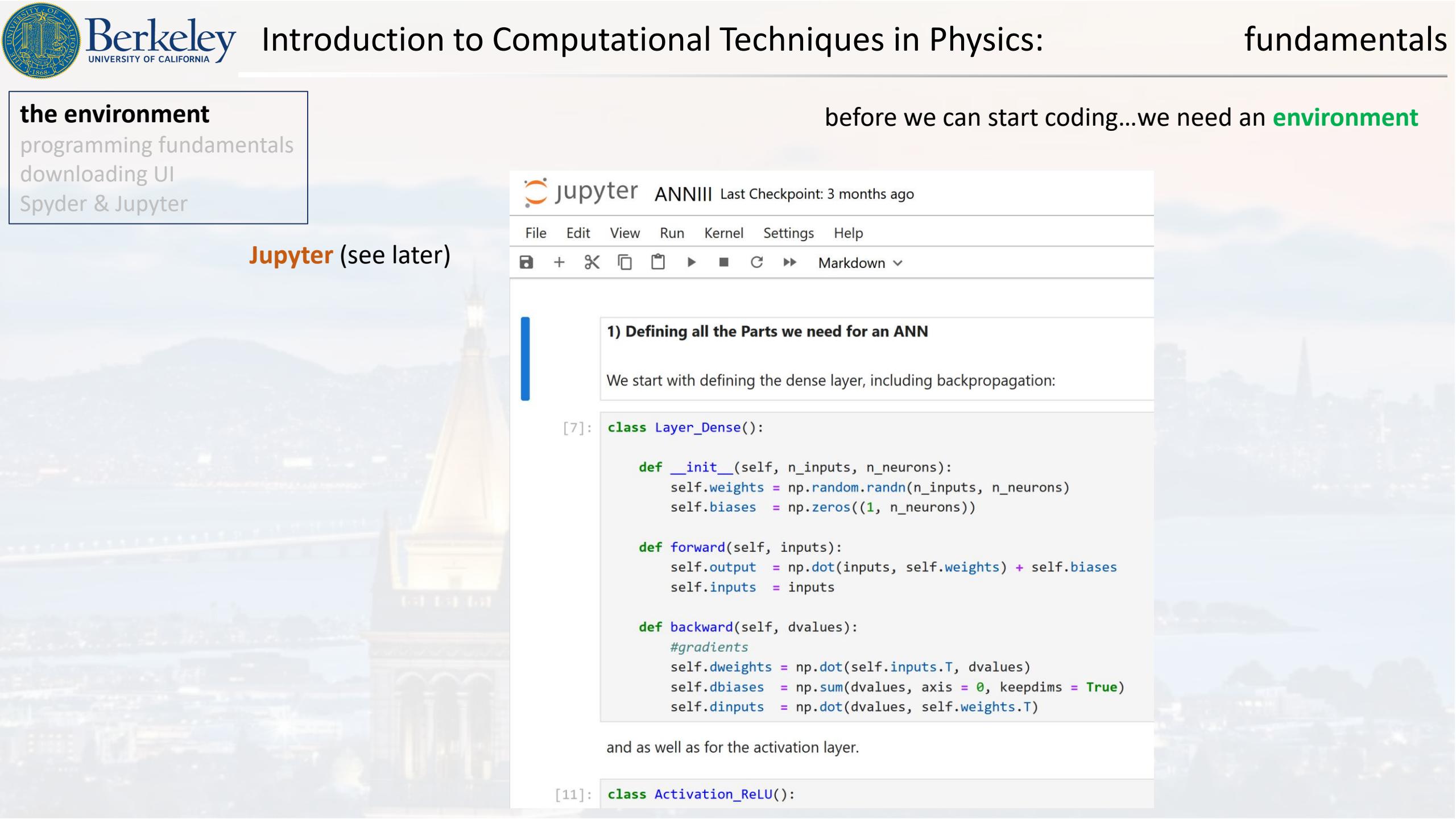
    #backward passes
    loss_activation.backward(loss_activation.output, C)
    dense2.backward(loss_activation.dinputs)
    activation1.backward(dense2.dinputs)
    dense1.backward(activation1.dinputs)
    F.backward(dense1.dinputs)
    MP3.backward(F.dinputs)
    RL3.backward(MP3.dinputs)@TBC
```

**the environment**

programming fundamentals
downloading UI
Spyder & Jupyter

Jupyter (see later)

before we can start coding...we need an **environment**



Jupyter ANNIII Last Checkpoint: 3 months ago

File Edit View Run Kernel Settings Help

Markdown

1) Defining all the Parts we need for an ANN

We start with defining the dense layer, including backpropagation:

```
[7]: class Layer_Dense():

    def __init__(self, n_inputs, n_neurons):
        self.weights = np.random.randn(n_inputs, n_neurons)
        self.biases = np.zeros((1, n_neurons))

    def forward(self, inputs):
        self.output = np.dot(inputs, self.weights) + self.biases
        self.inputs = inputs

    def backward(self, dvalues):
        #gradients
        self.dweights = np.dot(self.inputs.T, dvalues)
        self.dbiases = np.sum(dvalues, axis = 0, keepdims = True)
        self.dinputs = np.dot(dvalues, self.weights.T)
```

and as well as for the activation layer.

```
[11]: class Activation_ReLU():
```



the environment

programming fundamentals
downloading UI
Spyder & Jupyter

your computer

Operational System (OS)

Linux/WSL

Coding platform

finetuning settings (ANACONDA or miniconda *environment*)

- ANACONDA (editor + compiler: **Jupyter**, **Spyder** + many other tools)
- miniconda (like ANACONDA but only basics, editor + compiler: **Jupyter**)

- VS Code (editer + compiler)

Text Editor

- for actual coding
- in principle: any editor is fine



Berkeley

UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:

fundamentals

the environment

programming fundamentals

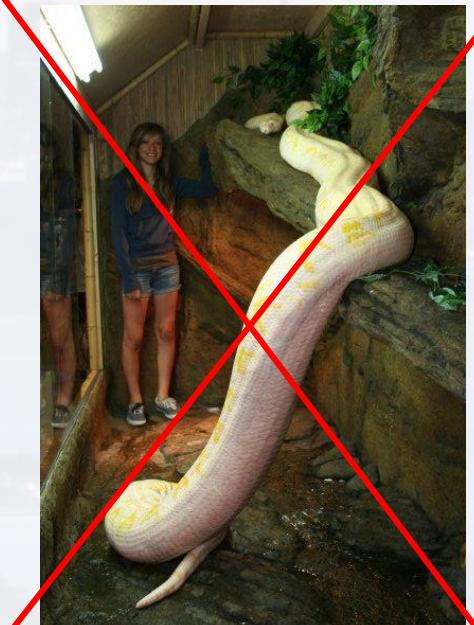
downloading UI

Spyder & Jupyter



created 1990, Guido van Rossum

named after “**Monty Python**”, not the serpent



idea: flexible, simple and compact syntax, extendable

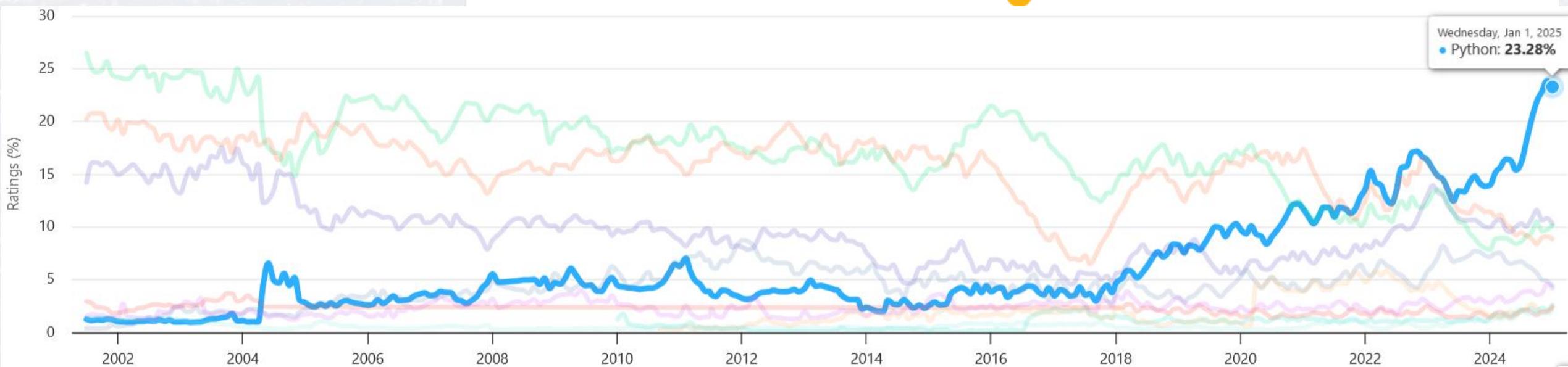


the environment

programming fundamentalsdownloading UI
Spyder & Jupyter

TIOBE index Jan 2025

Jan 2025	Jan 2024	Change	Programming Language	Ratings	Change
1	1		Python		
2	3				23.28%
3	4	▲	Java	10.15%	+2.28%
4	2	▼	C	8.86%	-2.59%
5	5		C#	4.45%	-2.71%
6	6		JavaScript	4.20%	+1.43%
7	11	▲	Go	2.61%	+1.24%
8	9	▲	SQL	2.41%	+0.95%





Programming Languages: translate human instructions to a form understandable by a computer

procedural:

functions/ routines that call each other
(Fortran, ALGOL, COBOL, BASIC, Pascal, C)

**the “style”
of programming**

object oriented (OOP):

creating objects/types of different properties (see later)
C++, Fortran 2003, Java, MATLAB, **Python**, Ruby, ...

compiled language:

close to the resulting machine code, **fast**
Fortran, C, C++, Java, Cobol, Pascal

**how a programming
language “talks” to your
CPU or GPU**

interpreted language:

an interpreter translates between source code and
machine code. **Slower, but simpler syntax**
Perl, Raku, **Python**, MATLAB



Python libraries/modules

```
from my_module import my_method as my_alias
```

1) reading files (.xlsx, .xls, .csv, .txt, ...)

pandas (standard), dask, polars

2) plotting

matplotlib, seaborn

3) numerical methods

math, numpy, scipy

4) machine learning

scikitlearn

5) ANN/AI/DeepLearning



TensorFlow



Keras



PyTorch



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

We need only **two states: on and off** $n = 8$ smallest memory cell: **8bits = 1byte**one bit (**binary digit**)yes/no
on/off
up/downHow many different states can I create with 8 switches? $2 \times 2 \times 2 \dots = 2^n$

8 bit :	$2^8 =$	256
16 bit:	$2^{16} =$	65.536
32 bit:	$2^{32} =$	4.294.967.296
64 bit:	$2^{64} =$	18.446.744.073.709.551.616

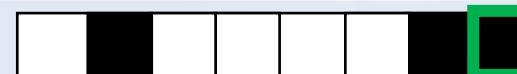


the environment

programming fundamentals

downloading UI

Spyder & Jupyter

We need only **two states: on and off** $n = 8$ smallest memory cell: **8bits = 1byte**

$$8 \text{ bit} : 2^8 = 256$$

$$16 \text{ bit}: 2^{16} = 65.536$$

$$32 \text{ bit}: 2^{32} = 4.294.967.296$$

$$64 \text{ bit}: 2^{64} = 18.446.744.073.709.551.616$$

0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
10	00001010
11	00001011
12	00001100
13	00001101
14	00001110
15	00001111
16	00010000

...



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

We need only **two** states: **on** and **off**

$0 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$	0	00000000
$1 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$	1	00000001
$2 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$	2	00000010
	3	00000011
	4	00000100
	5	00000101
	6	00000110
$7 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$	7	00000111
	8	00001000
	9	00001001
	10	00001010
	11	00001011
	12	00001100
	13	00001101
	14	00001110
	15	00001111
	16	00010000
	...	

$$8 \text{ bit} : 2^8 = 256$$

$$16 \text{ bit}: 2^{16} = 65.536$$

$$32 \text{ bit}: 2^{32} = 4.294.967.296$$

$$64 \text{ bit}: 2^{64} = 18.446.744.073.709.551.616$$

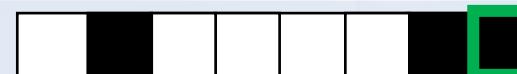


the environment

programming fundamentals

downloading UI

Spyder & Jupyter



)
 $n = 8$

smallest memory cell: **8bits = 1byte**

8 bit :	$2^8 =$	256
16 bit:	$2^{16} =$	65.536
32 bit:	$2^{32} =$	4.294.967.296
64 bit:	$2^{64} =$	18.446.744.073.709.551.616

8 bits = 1 byte (B)

1 kB = 1024 B
1 MB = 1024 kB etc

accuracy of numerical operations:

```
import sys  
sys.float_info.epsilon
```

2.220446049250313e-16

machine epsilon



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

	rel. approx. error (ϵ)	range
16 bit int		-32768 ... 32767
32 bit int		$\approx -10^9 \dots 10^9$
32 bit float	$\approx 10^{-8}$	$\approx 10^{-38} \dots 10^{38}$
64 bit double	$\approx 10^{-16}$	$\approx 10^{-308} \dots 10^{308}$

0	00000000
1	00000001
2	00000010
3	00000011
4	00000100
5	00000101
6	00000110
7	00000111
8	00001000
9	00001001
10	00001010
11	00001011
12	00001100
13	00001101
14	00001110
15	00001111
16	00010000

...

Luckily, Python will tell us when an operation doesn't make sense based on precision, but still: **be cautious!**



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

beginner



ANACONDA Navigator

- straight forward installation
- free for teaching/instructions
- more tools than we actually need
- your choice if you have never programmed

medium



miniconda

- requires a few more steps to install
- free
- you need to install the specific tools
- your choice if you prefer flexibility over convenience



the environment
programming fundamentals
downloading UI
Spyder & Jupyter



ANACONDA Navigator

- straight forward installation
- free for teaching/instructions
- more tools than we actually need
- your choice if you have never programmed

download [here](#)

Distribution Installers

Download

For installation assistance, refer to [troubleshooting](#).

Windows



Mac



Linux





the environment
programming fundamentals
downloading UI
Spyder & Jupyter



ANACONDA Navigator

- straight forward installation
- free for teaching/instructions

more tools than we actually need

<p>The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.</p> Install	<p>Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.</p> Install	<p>4.0.15 Anaconda Assistant JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.</p> Launch	<p>Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.</p> Launch	<p>Run Python Jupyter notebooks on a multi-model database.</p> Launch	<p>1.0.0 Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).</p> Launch
 CMD.exe Prompt 0.1.1 Run a cmd.exe terminal with your current environment from Navigator activated Launch	 console_shortcut_miniconda 0.1.1 Anaconda Powershell Prompt Launch	 JupyterLab 4.0.11 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Launch	 Notebook 7.0.8 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch	 Powershell Prompt 0.0.1 Run a Powershell terminal with your current environment from Navigator activated Launch	 Qt Console 5.6.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch
 Spyder 6.0.5 Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features Launch	 VS Code 1.100.2 Streamlined code editor with support for development operations like debugging, task running and version control. Launch	 EduBlocks IBM watsonx™ IBM watsonx IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring. Launch	 ORACLE Cloud Infrastructure Oracle Data Science Service OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools Launch	 PyScript Code and share Python in the Browser. A vibrant community of makers, builders, and hackers building the next frontier of Python-powered web applications. Launch	



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

miniconda

- requires a few more steps to install
- free
- you need to install the specific tools
- your choice if you prefer flexibility over convenience

on PC: install [Windows Subsystem for Linux](#)

follow the instructions [here](#)

watch video [01a_Installing_WSL](#)



Filter by title

Learn Discover Product documentation Development languages Topics

Windows Release health Windows client Application developers Hardware developers Windows Server Windows for IoT Windows Insider Program Windows 365

Learn / Windows / WSL /

How to install Linux on Windows with WSL

Article • 11/19/2024 • 10 contributors

In this article

- Prerequisites
- Install WSL command
- Change the default Linux distribution installed
- Set up your Linux user info
- Show 6 more

Additional resources

Training

Module [Developing in the Windows Subsystem for Linux with Visual Studio Code - Training](#)

In this module, you learn how to use the Windows Subsystem for Linux (WSL) with Visual Studio Code (VS Code). We explore...

Certification [Microsoft Certified: Windows Server Hybrid Administrator Associate - Certifications](#)

As a Windows Server hybrid administrator, you integrate Windows Server environments with Azure services and...





the environment
programming fundamentals
downloading UI
Spyder & Jupyter

miniconda on PC:

open your **WSL** shell and run the following commands:

1) download installer:

```
mmh_user@DESKTOP-PPSA666: ~  
(base) mmh_user@DESKTOP-PPSA666:~$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

2) run installer:

```
bash Miniconda3-latest-Linux-x86_64.sh
```

confirm - license agreement,
- folder location and
- settings

3) close and reopen WSL**4) activate conda:**

```
conda activate
```

watch video [**01b_Installing_Miniconda**](#)





the environment
programming fundamentals
downloading UI
Spyder & Jupyter

miniconda on PC:



5) create environment: `conda create --name <My_Environment>`

6) check environment: `conda env list`

7) activate environment: `conda activate <My_Environment>`

<My_Environment>

8) check python:

```
mmh_user@DESKTOP-PPSA666: ~
(base) mmh_user@DESKTOP-PPSA666:~$ conda activate MSSE_Python
(MSSE_Python) mmh_user@DESKTOP-PPSA666:~$ python
Python 3.9.4 (default, Apr  2 2024, 23:27:39)
[GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

miniconda on PC:



9) install Jupyter:

conda install jupyter

```
(MSSE_Python) m mh_user@DESKTOP-PPSA666:~$ conda install jupyter
```

confirm settings

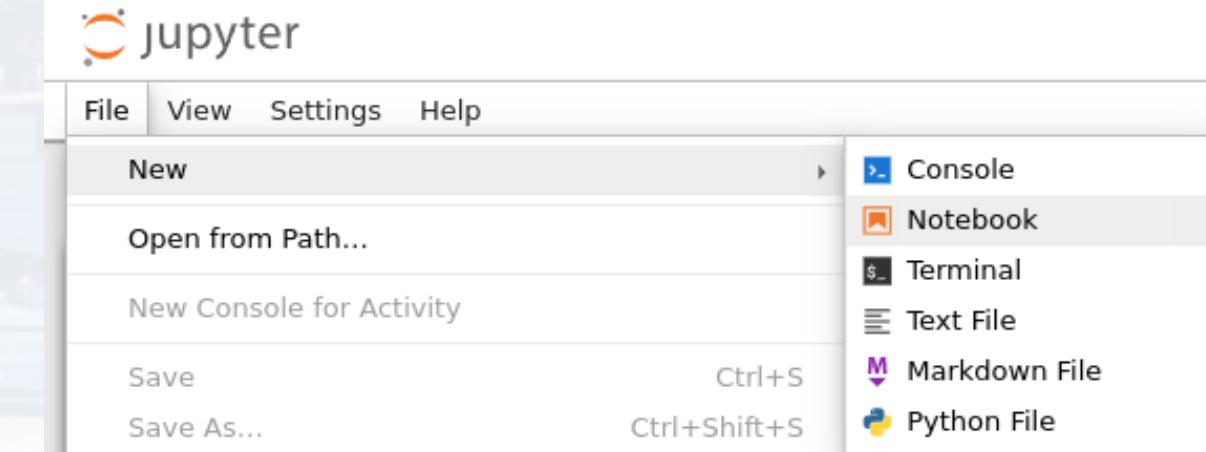
10) open Jupyter:

jupyter notebook &

a browser will open

go to → File
→ New
→ Notebook
→ confirm kernel

the notebook will open



watch video [*01c_Installing_Jupyter*](#)



the environment

programming fundamentals

downloading UI

Spyder & Jupyter

miniconda on PC:



11) creating spyder environment

```
conda create -c conda-forge -n spyder-env spyder numpy scipy pandas matplotlib sympy cython
```

12) activating and configure environment

```
conda activate spyder-env
```

```
conda config --env --add channels conda-forge
```

```
conda config --env --set channel_priority strict
```

13) open spyder

```
spyder &
```

watch video [01e_Installing_Spyder](#)



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

miniconda on MAC:

follow instructions [here](#)



Installing Miniconda and compilers

Click the appropriate tab for your operating system to see set-up instructions.

Mac OS

Linux

Windows

Miniconda Installation

You can download and run the installer at this [link](#).

Compilers

MacOS users should [install XCode](#). An easy way to install XCode is through the [Mac App Store](#).



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

watch video [*01f_Quick_Guide_Spyder*](#)

<p>The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.</p> <p>Install</p>	<p>Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.</p> <p>Install</p>	<p> 4.0.15 Anaconda Assistant JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.</p> <p>Launch</p>	<p>Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.</p> <p>Launch</p>	<p>Run Python Jupyter notebooks on a multi-model database.</p> <p>Launch</p>	<p> 1.0.0 Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).</p> <p>Launch</p>
<p> CMD.exe Prompt 0.1.1 Run a cmd.exe terminal with your current environment from Navigator activated</p> <p>Launch</p>	<p> console_shortcut_miniconda 0.1.1 Anaconda Powershell Prompt</p> <p>Launch</p>	<p> JupyterLab  4.0.11 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.</p> <p>Launch</p>	<p> Notebook  7.0.8 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.</p> <p>Launch</p>	<p> Powershell Prompt 0.0.1 Run a Powershell terminal with your current environment from Navigator activated</p> <p>Launch</p>	<p> Qt Console 5.6.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.</p> <p>Launch</p>
<p> Spyder 6.0.5 Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features</p> <p>Launch</p>	<p> VS Code 1.100.2 Streamlined code editor with support for development operations like debugging, task running and version control.</p> <p>Launch</p>	<p> EduBlocks IBM watsonx</p> <p>Web-based coding platform from Anaconda designed for students. Learn Python coding through an interactive, block-based visual environment.</p> <p>Launch</p>	<p> watsonx™ IBM watsonx</p> <p>IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring.</p> <p>Launch</p>	<p> ORACLE Cloud Infrastructure Oracle Data Science Service</p> <p>OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools</p> <p>Launch</p>	<p> PyScript A vibrant community of makers, builders, and hackers building the next frontier of Python-powered web applications.</p> <p>Launch</p>



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

**content of
current
folder**

The screenshot shows the Spyder Python IDE interface. On the left, a 'content of current folder' panel displays a file tree with various files and folders. In the center, a 'py script: for coding' panel shows a Python script with code for a neural network's forward pass and backward passes. To the right, a 'workspace' panel displays current plots or variables, with two tabs circled in green: 'Variable Explorer' and 'Plots'. A 'console' panel at the bottom shows the Python and IPython environments. A 'help' panel on the right provides usage information.

py script: for coding

```
for epoch in range(Nsteps):
    #passing data through layers
    Conv1.forward(M,2,1)
    RL1.forward(Conv1.output)
    MP1.forward(RL1.output,0,1,2)

    Conv2.forward(MP1.output,0,1)
    RL2.forward(Conv2.output)
    MP2.forward(RL2.output)

    Conv3.forward(MP2.output,0,1)
    RL3.forward(Conv3.output)
    MP3.forward(RL3.output)

    #flattening
    F.forward(MP3.output)
    x = F.output

    dense1.forward(x)
    activation1.forward(dense1.output)
    dense2.forward(activation1.output)
    loss = loss_activation.forward(dense2.output, C)

    predictions = np.argmax(loss_activation.output, axis=-1)
    if len(C.shape) == 2:
        C = np.argmax(C, axis=1)
    accuracy = np.mean(predictions == C)

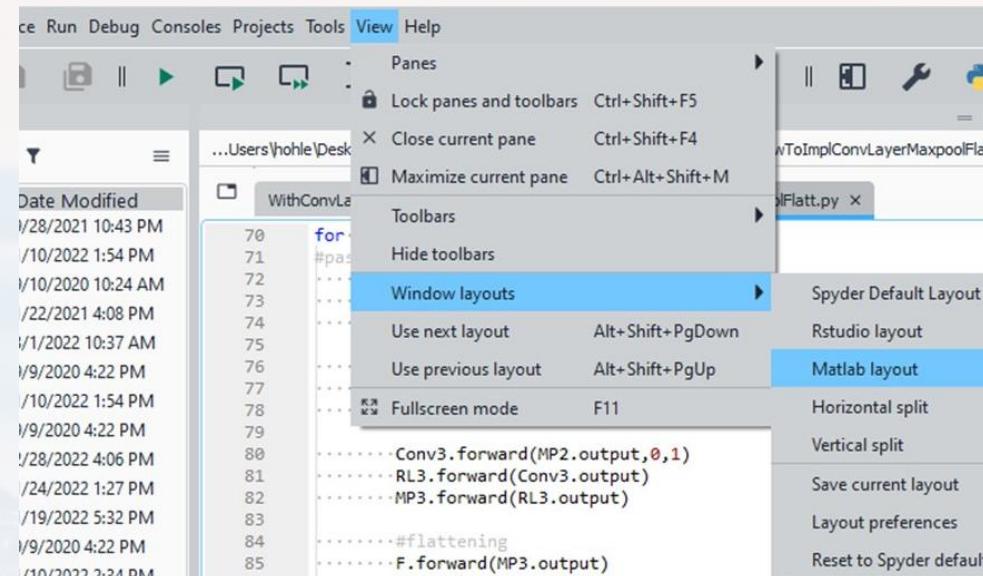
    #backward passes
    loss_activation.backward(loss_activation.output, C)
    dense2.backward(loss_activation.dinputs)
    activation1.backward(dense2.dinputs)
    dense1.backward(activation1.dinputs)
    F.backward(dense1.dinputs)
    MP3.backward(F.dinputs)
    RL3.backward(MP3.dinputs) #BC
```

workspace:
displays current plots
or variables

console:
typing commands &
executing scripts



the environment
programming fundamentals
downloading UI
Spyder & Jupyter



settings:
toolbar: *View → Window layouts*

e. g. Matlab

I use Pydev

Projects Tools View Help

PYTHONPATH manager

Current user environment variables...

Preferences

Reset Spyder to factory defaults

Preferences

Appearance

Main interface

Syntax highlighting theme

Pydev

Plain text

Rich text

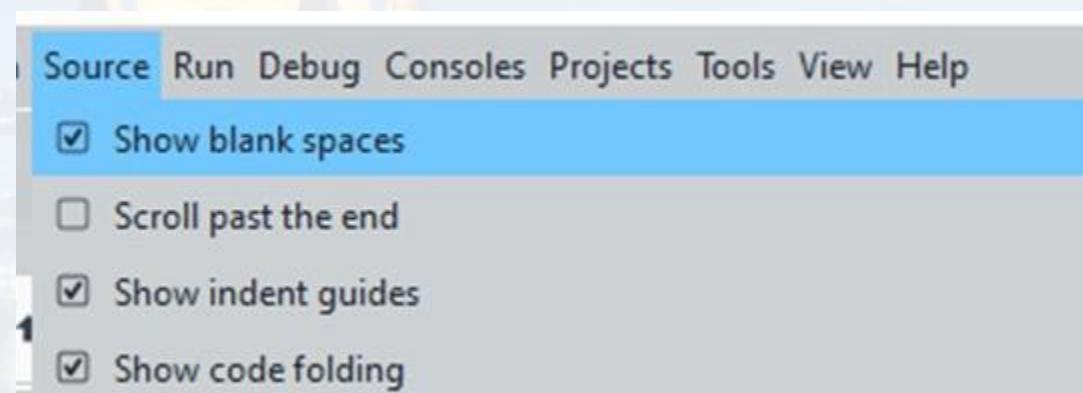
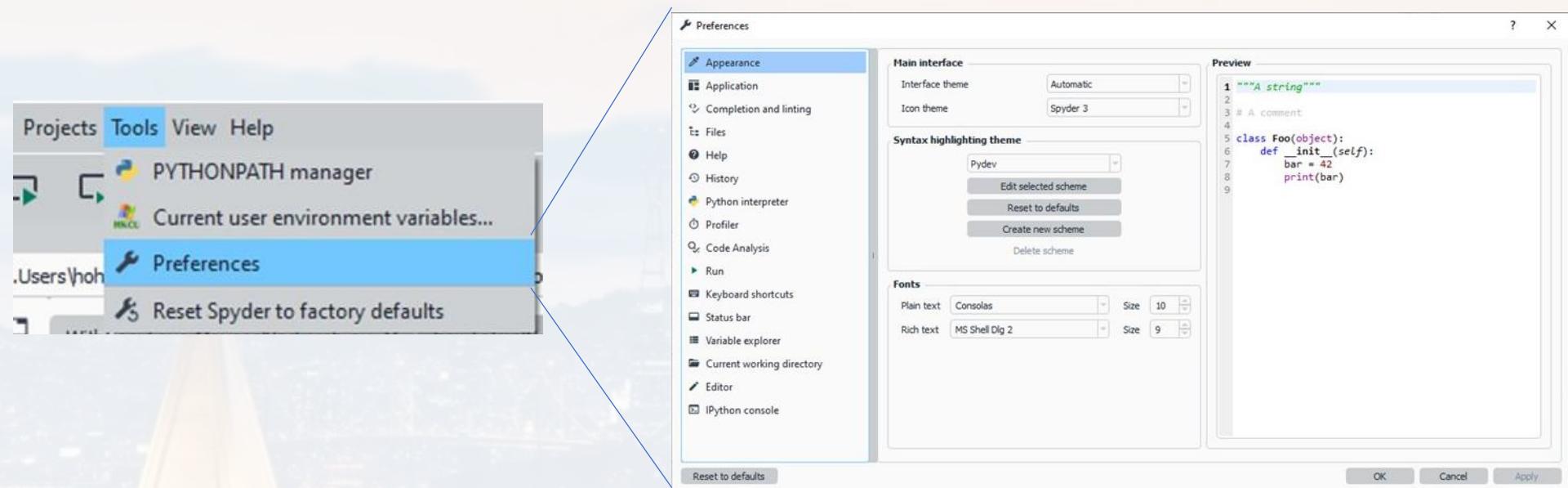
Preview

```
1 """A string"""
2
3 # A comment
4
5 class Foo(object):
6     def __init__(self):
7         bar = 42
8         print(bar)
9
```

OK Cancel Apply



the environment
programming fundamentals
downloading UI
Spyder & Jupyter



spaces are relevant for syntax!



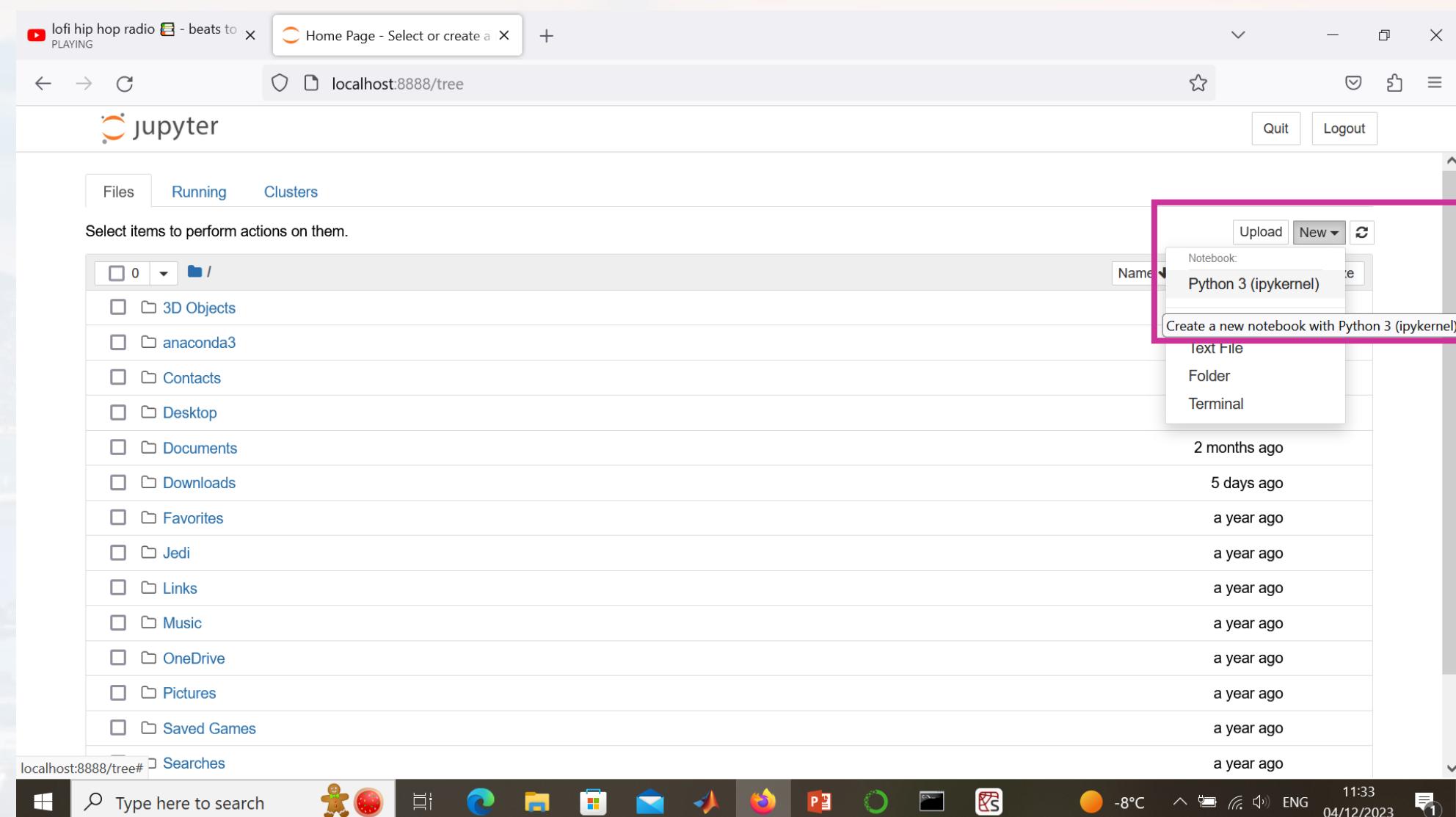
the environment
programming fundamentals
downloading UI
Spyder & Jupyter

<p>The only Python IDE you need – built for data and AI/ML professionals. Supercharged with an AI-enhanced IDE experience. Free forever, plus one month of Pro included.</p> Install	<p>Access various large language models (LLMs) curated by Anaconda, and start leveraging secure local AI today.</p> Install	<p>4.0.15 Anaconda Assistant JupyterLab supercharged with a suite of Anaconda extensions, starting with the Anaconda Assistant AI chatbot.</p> Launch	<p>Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.</p> Launch	<p>Run Python Jupyter notebooks on a multi-model database.</p> Launch	<p>1.0.0 Opens a terminal instance with conda activated (requires menuinst 2.1.1 or greater).</p> Launch
 CMD.exe Prompt 0.1.1 Run a cmd.exe terminal with your current environment from Navigator activated Launch	 console_shortcut_miniconda 0.1.1 Anaconda Powershell Prompt Launch	 JupyterLab 4.0.11 An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Launch	 Notebook 7.0.8 Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Launch	 Powershell Prompt 0.0.1 Run a Powershell terminal with your current environment from Navigator activated Launch	 Qt Console 5.6.1 PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Launch
 Spyder 6.0.5 Scientific Python Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features Launch	 VS Code 1.100.2 Streamlined code editor with support for development operations like debugging, task running and version control. Launch	 EduBlocks IBM watsonx IBM watsonx is an enterprise-ready AI platform including a data store, model builder, and AI model management and monitoring. Launch	 ORACLE Cloud Infrastructure Oracle Data Science Service OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools Launch	 PyScript Code and share Python in the Browser. A vibrant community of makers, builders, and hackers building the next frontier of Python-powered web applications. Launch	



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

watch video [01d_Quick_Guide_Jupyter](#)



lofi hip hop radio - beats to X Home Page - Select or create a X +

localhost:8888/tree

Quit Logout

jupyter

Files Running Clusters

Select items to perform actions on them.

0 / 3D Objects anaconda3 Contacts Desktop Documents Downloads Favorites Jedi Links Music OneDrive Pictures Saved Games Searches

Name: Python 3 (ipykernel)

Create a new notebook with Python 3 (ipykernel)

Upload New

Text File

Folder

Terminal

2 months ago

5 days ago

a year ago

localhost:8888/tree#

Type here to search

11:33 -8°C ENG 04/12/2023



Berkeley
UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:

fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

jupyter Untitled1 Last Checkpoint: a minute ago (unsaved changes)

In []: |

jupyter Untitled1 Last Checkpoint: 7 minutes ago (unsaved changes)

In []: print("test")

jupyter Untitled1 Last Checkpoint: 9 minutes ago (autosaved)

Insert Cell Above

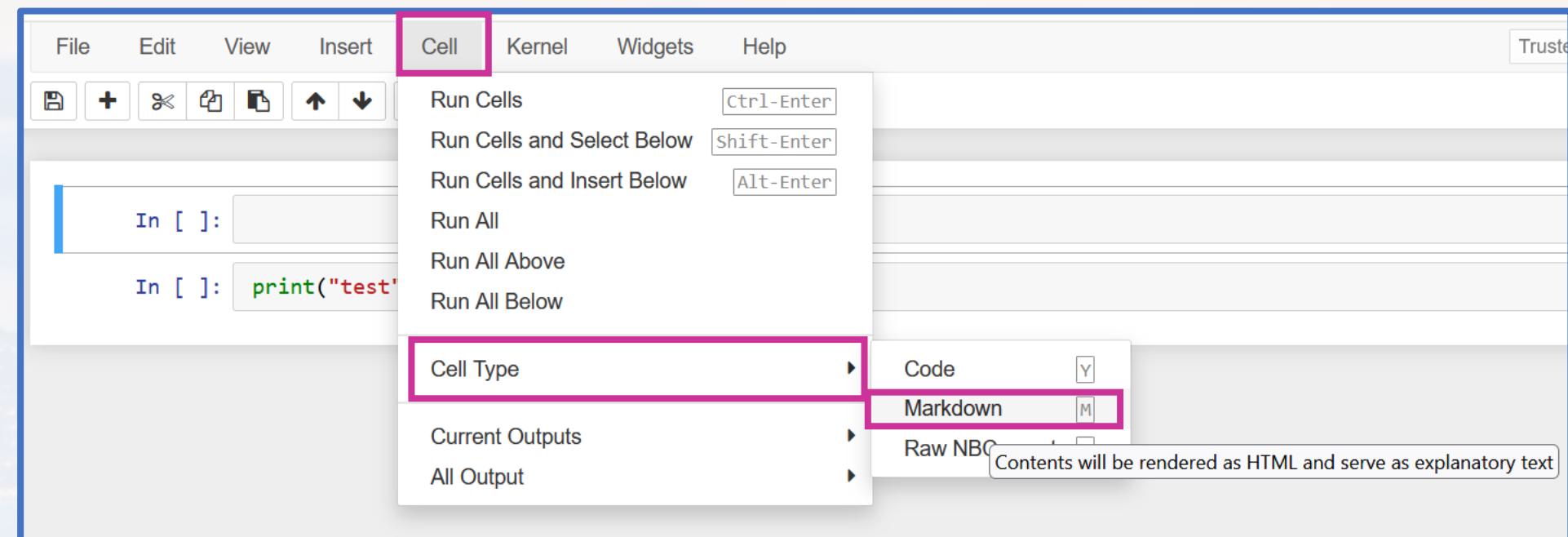
Insert Cell Below

Insert an empty Code cell above the currently active cell

In []: print("test")



the environment
programming fundamentals
downloading UI
Spyder & Jupyter



type something → click **run**

The screenshot shows the Jupyter Notebook interface with a single code cell. The cell is labeled 'In [1]:' and contains the command `print("test")`. The output of the cell is 'test'. Below the cell is an empty input field labeled 'In []:'.

**web: Jupyter → markdown styles
(including LaTeX)**

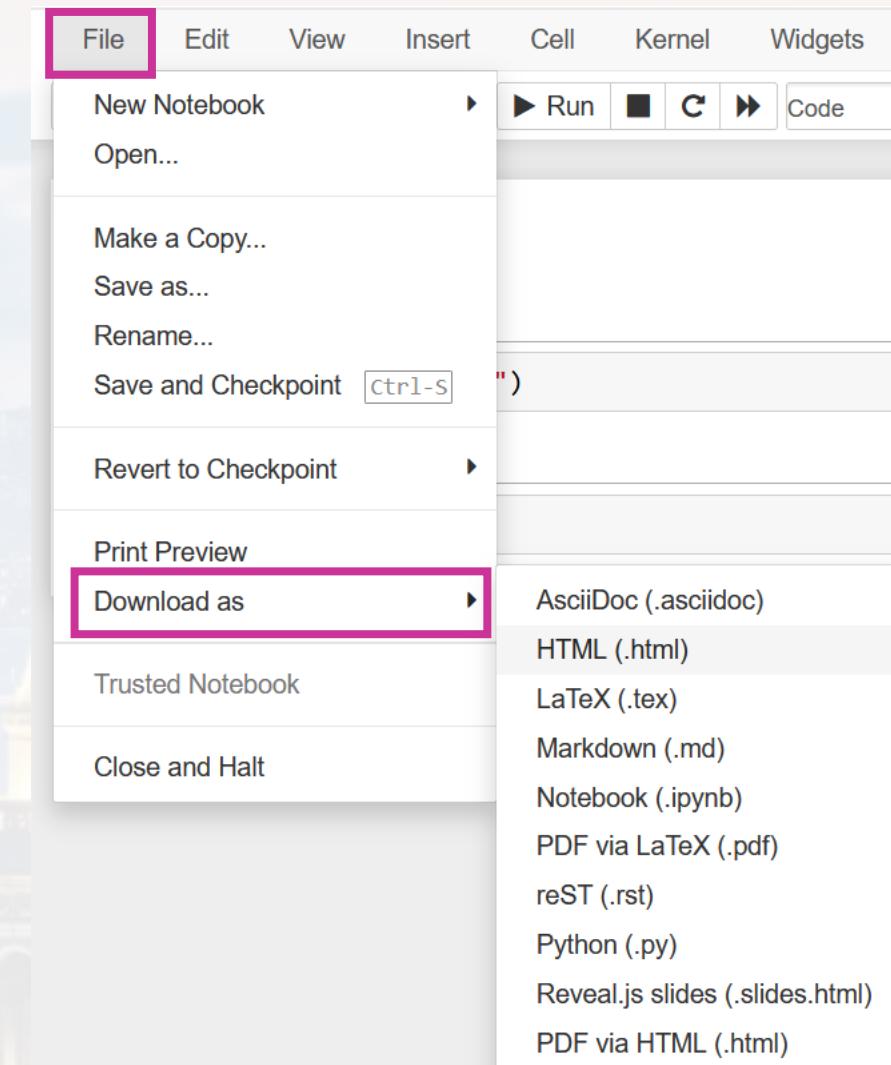


the environment

programming fundamentals

downloading UI

Spyder & Jupyter





Berkeley

UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:

fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

You can run Jupyter on [Datahub](#) too!



Welcome to the University of California, Berkeley
DataHub.



Berkeley
UNIVERSITY OF CALIFORNIA

CalNet Authentication Service

[sign in](#)

CalNet ID:

Passphrase (Case Sensitive):

[SIGN IN](#)

[HELP](#)

[Sponsored Guest Sign In](#)

[FORGOT CALNET ID OR PASSPHRASE?](#)

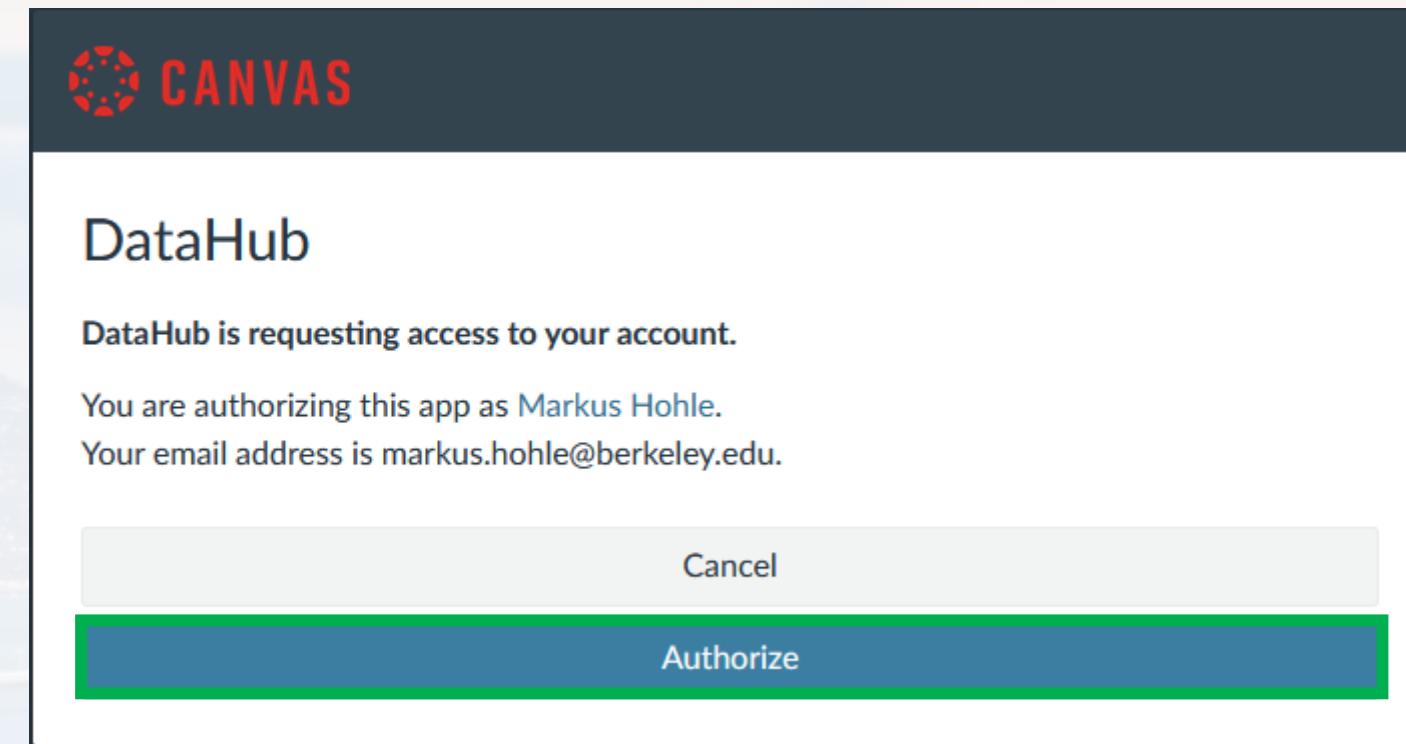
[MANAGE MY CALNET ACCOUNT](#)

Copyright © 2025 UC Regents. All rights reserved.



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

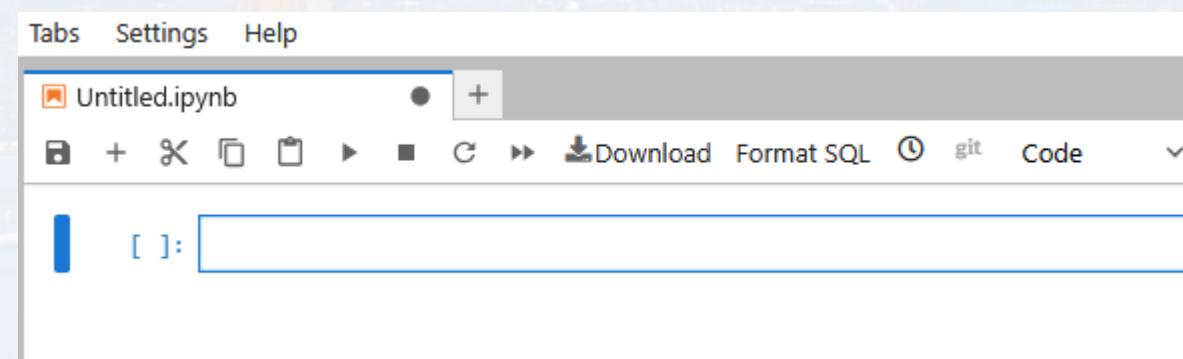
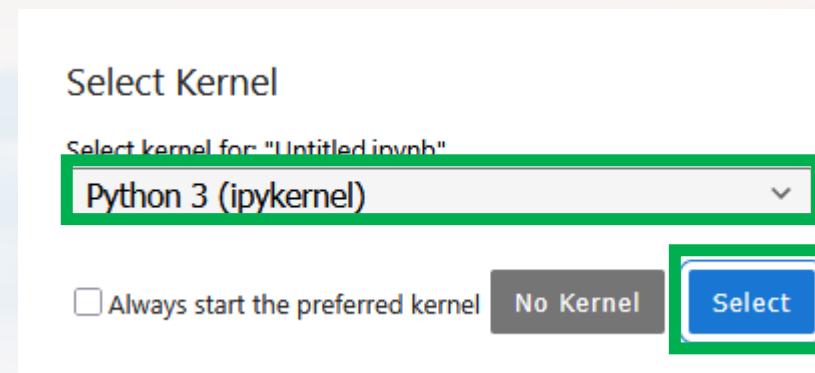
You can run Jupyter on [Datahub](#) too!





the environment
programming fundamentals
downloading UI
Spyder & Jupyter

You can run Jupyter on [Datahub](#) too!

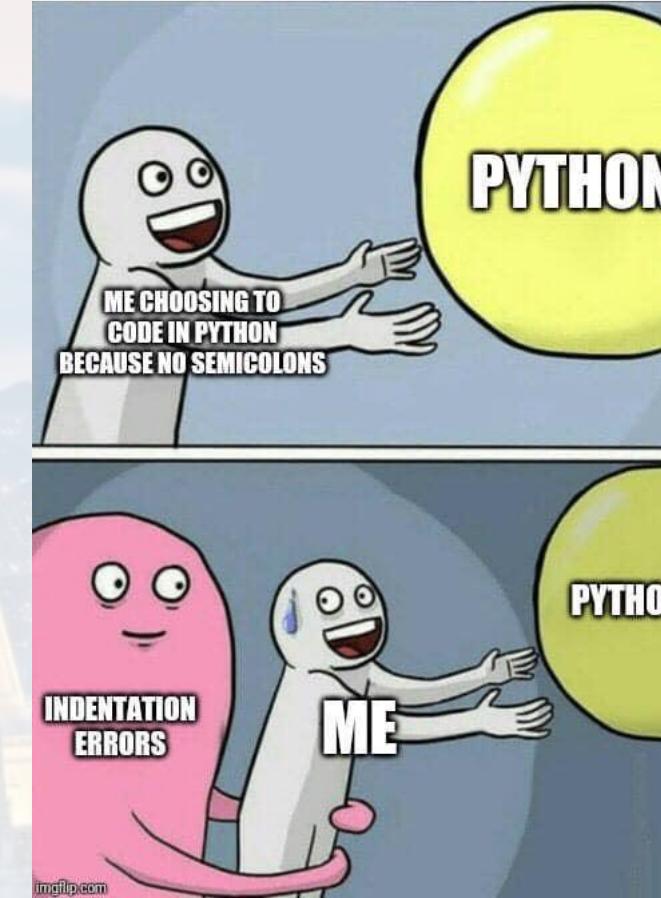




Berkeley

UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:



Thank you for your attention!