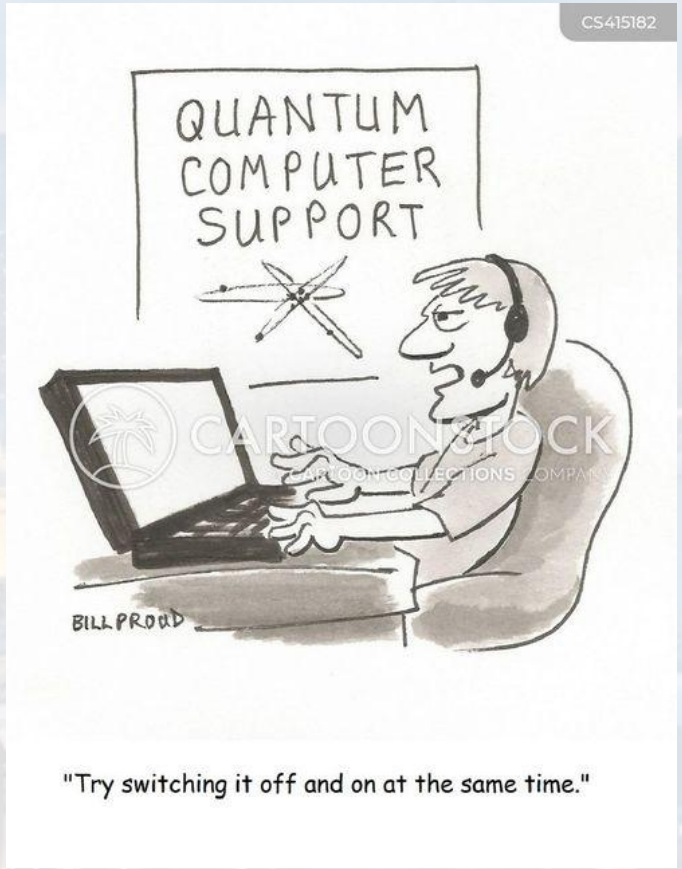
Berkeley
UNIVERSITY OF CALIFORNIA

M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics



Introduction to Computational Techniques in Physics: syllabus

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

bcourse

datahub.berkeley

my contact: markus.hohle@berkeley.edu

GSI (Sam): Sam D'Ambrosia shda@berkeley.edu

Introduction to Computational Techniques in Physics: syllabus

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 **asynchron**
which you can complete at any time before the du
workshops, office hours) which you will attend at a (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 codin

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

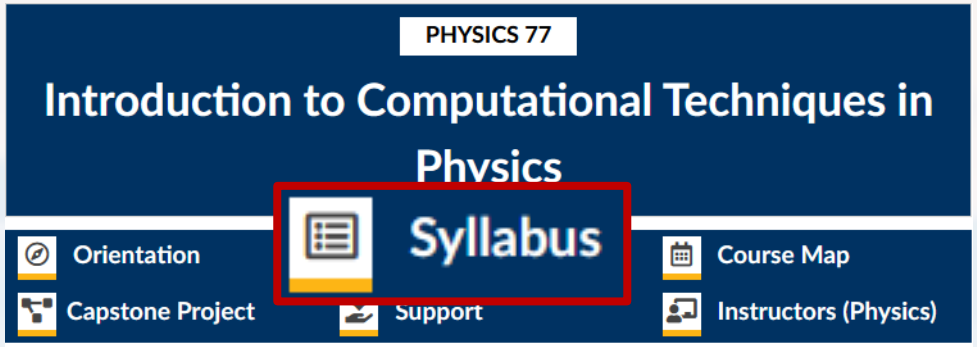
Office Hours Fridays, 3:00 - 4:30pm, here

Discussion/Workshops (Sam):

Monday, Wednesday 10:00 - 11:30am, here

Thursday 6:00 - 7:30pm, here

Introduction to Computational Techniques in Physics: syllabus



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 syl 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 li

If you prefer, you can set your own time zone to display throughc

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

Introduction to Computational Techniques in Physics: syllabus

Navigation Tips

- Select a section from the drop-down menu below to see the syl
  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 li
  If you prefer, you can set your own time zone to display throughc

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

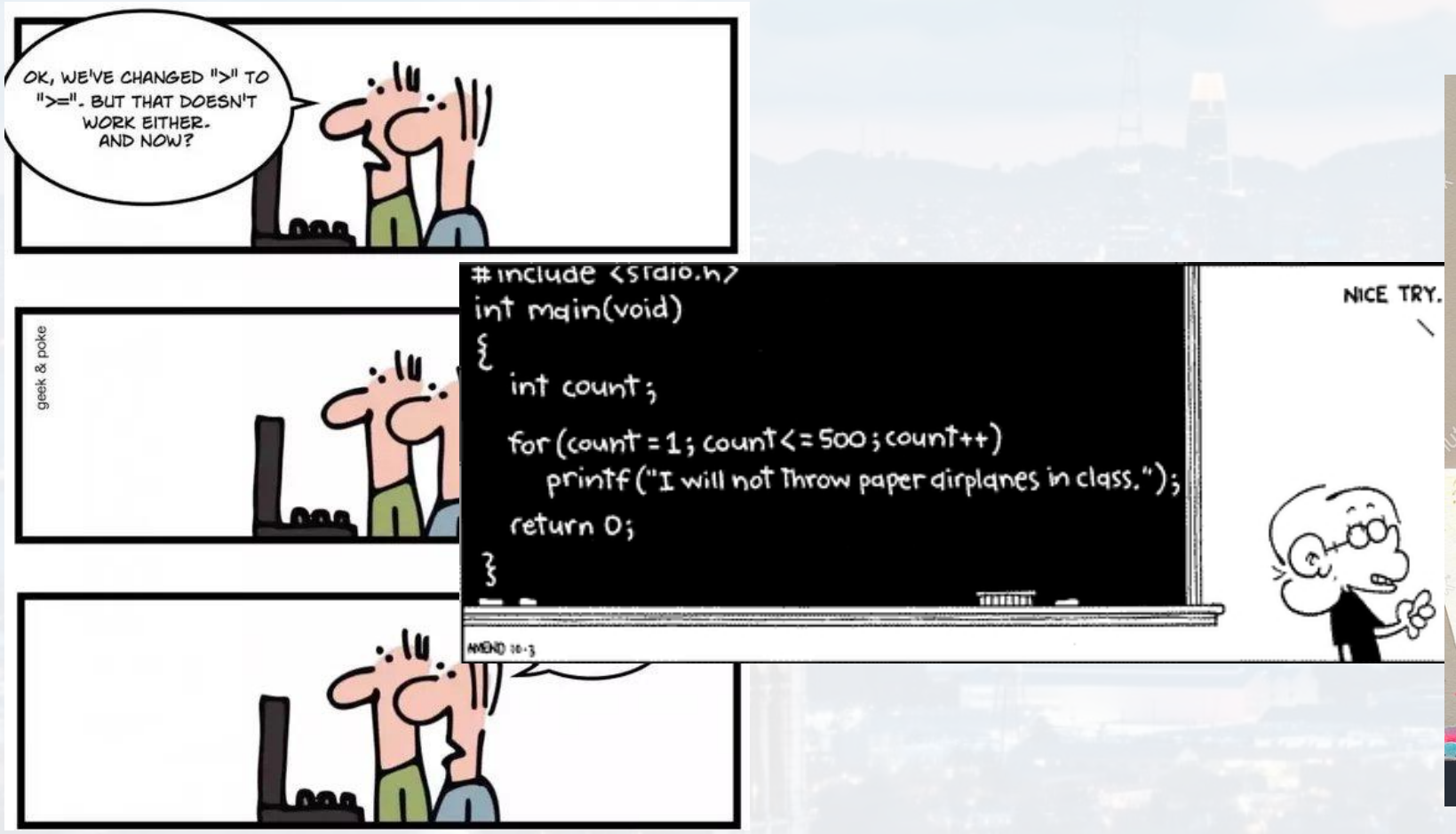
Introduction to Computational Techniques in Physics: syllabus

Week	Date	Topic
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	*June 19th*	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

Introduction to Computational Techniques in Physics: syllabus

Mixed Audience

beginner ★ **medium** ★ **advanced** ★

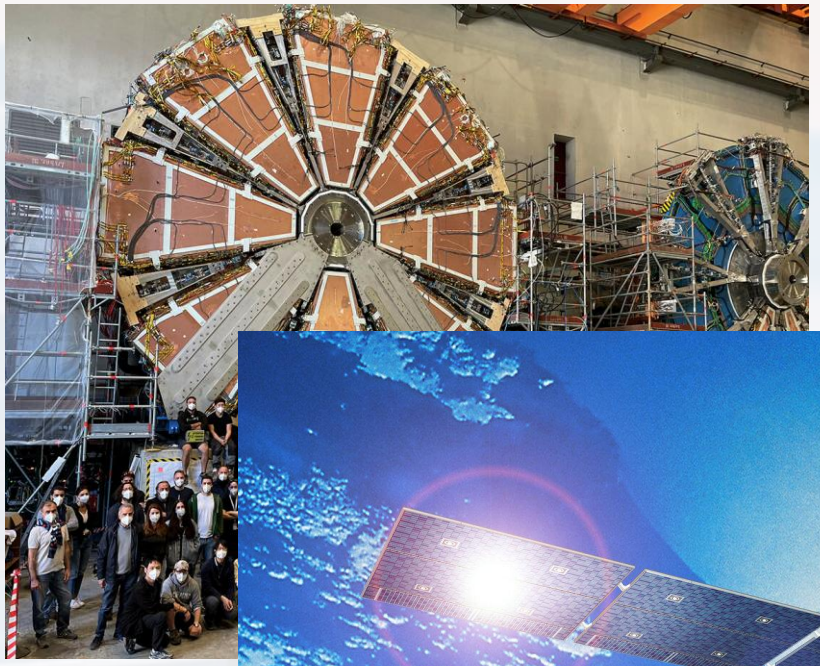


motivation: The good old days:

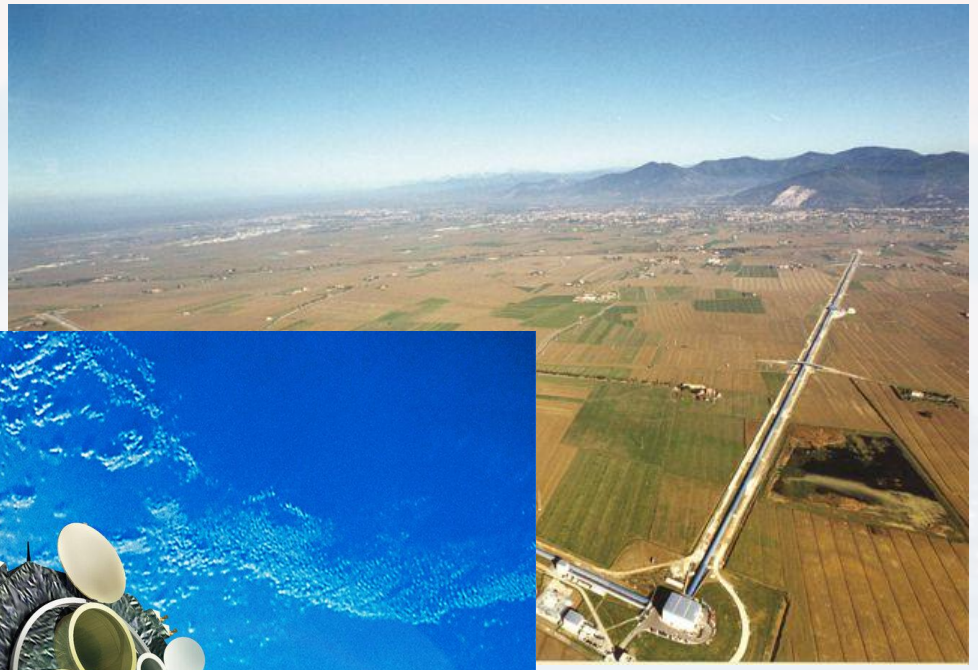
Salar de Uyuni, Bolivia, Mar 2003

Introduction to Computational Techniques in Physics: syllabus

motivation:



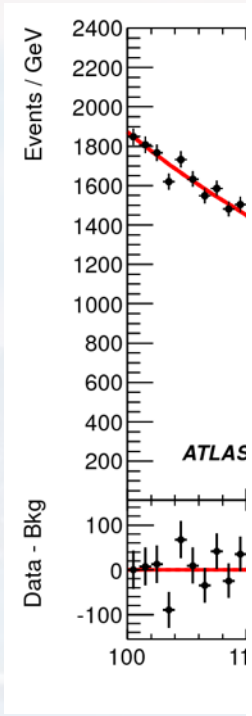
source: CERN



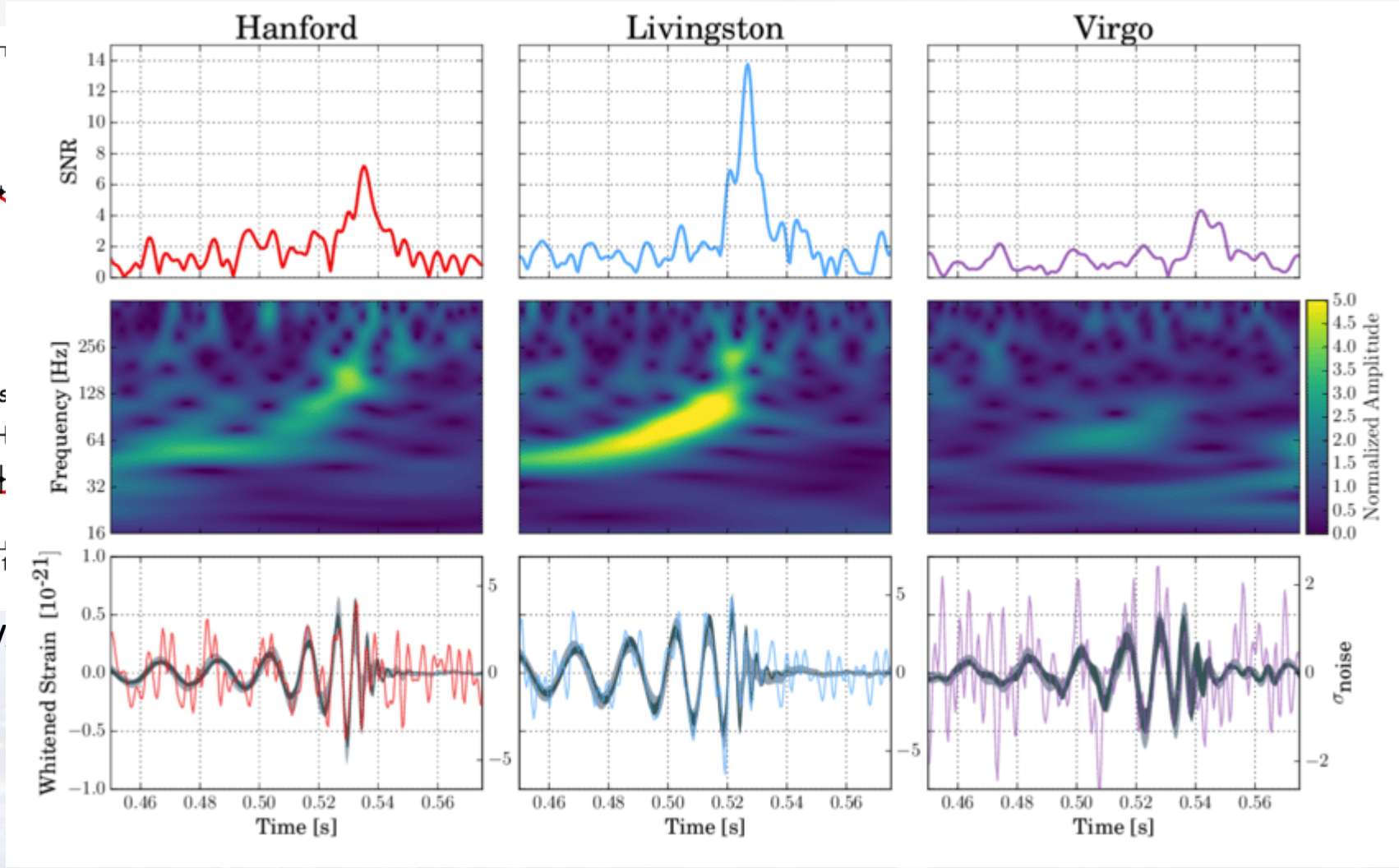
source: LIGO collaboration

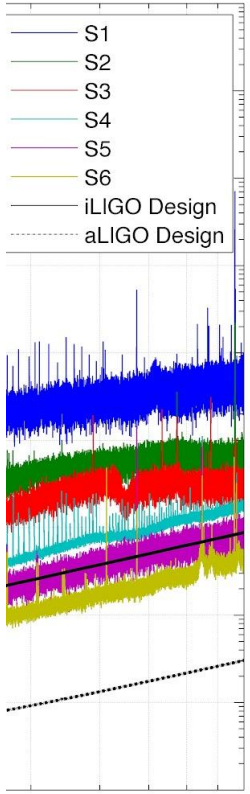
source: ESA

motivation:



Source: CERN/





Source: Sergio Gaudio Embry-Riddle Aeronautical University, Prescott, United States · Physics

being able to write code like this…

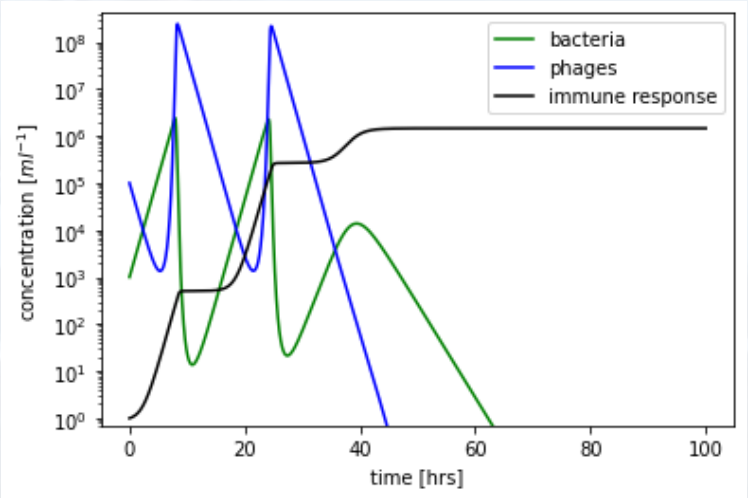
```
@my_timer
class SignalDetect():

    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



Leung & Weitz 2016

Introduction to Computational Techniques in Physics: syllabus

Week	Date	Topic
1	June 12th	Programming Environment & UIs for Python, Programming Fundamentals
2	*June 19th*	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

Introduction to Computational Techniques in Physics: syllabus

Week	Date	Topic
1	**June 12th**	**Programming Environment & UIs for Python, Programming Fundamentals**
2	*June 19th*	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

Introduction to Computational Techniques in Physics: fundamentals

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

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**

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

- for Windows:

Windows **S**ubsystem for **L**inux (**WSL**), see later

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

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

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**!

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

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

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

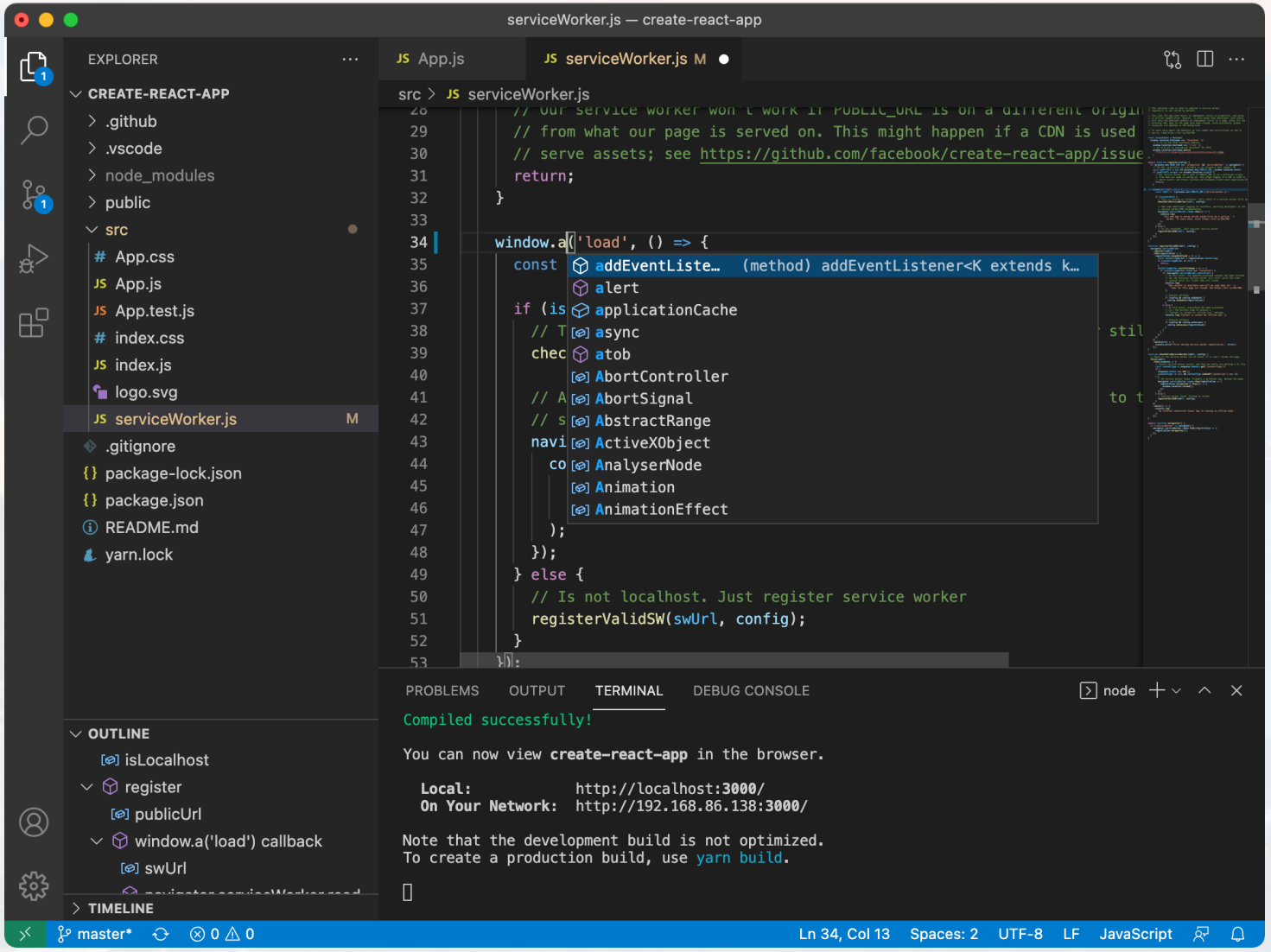
- for actual coding
- in principle: any editor is fine

Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

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

VS Code
(professionals)

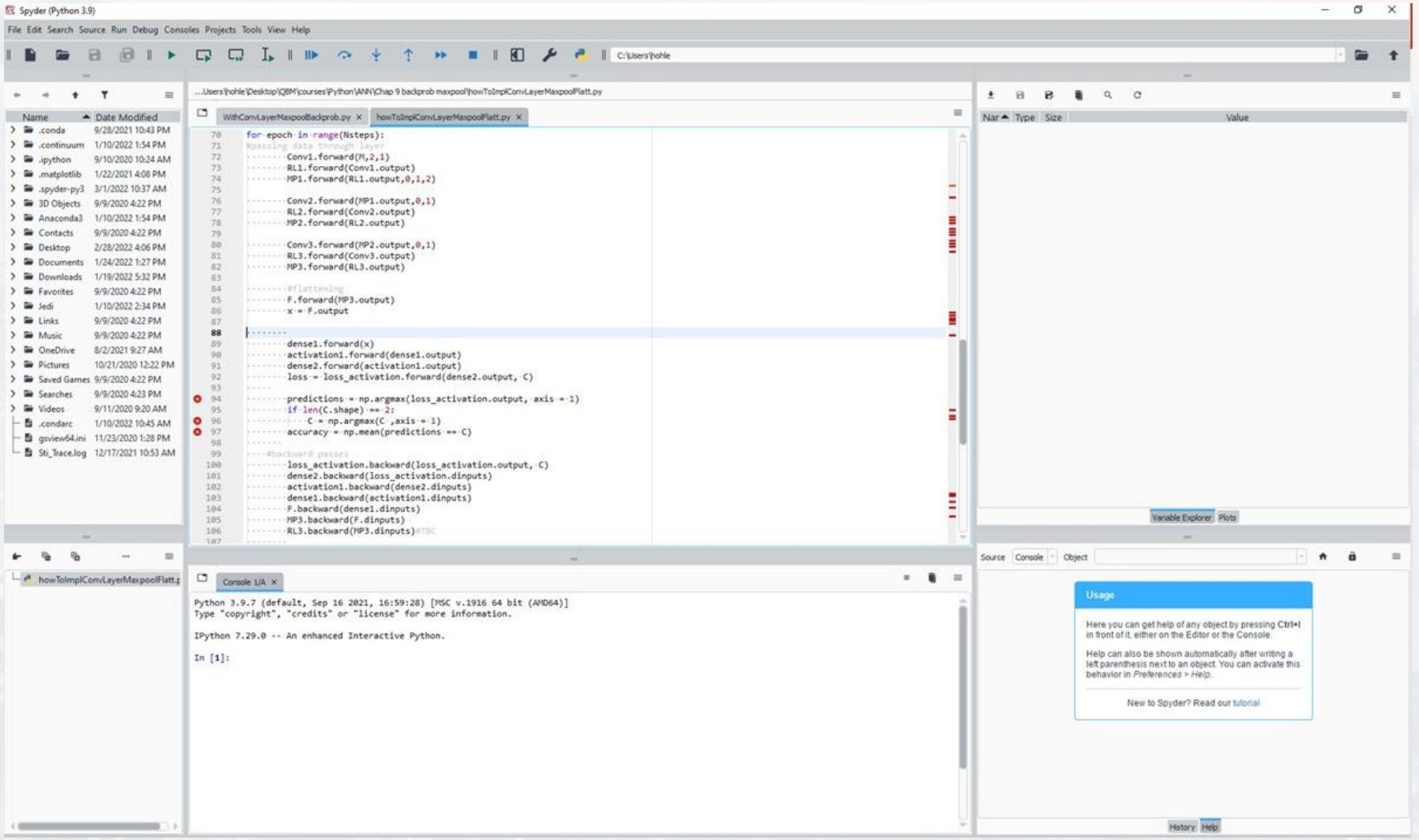


Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

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

Spyder (see later)



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

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

Jupyter (see later)

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

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

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

created 1990, Guido van Rossum

named after **"Monty Python"**, not the serpent

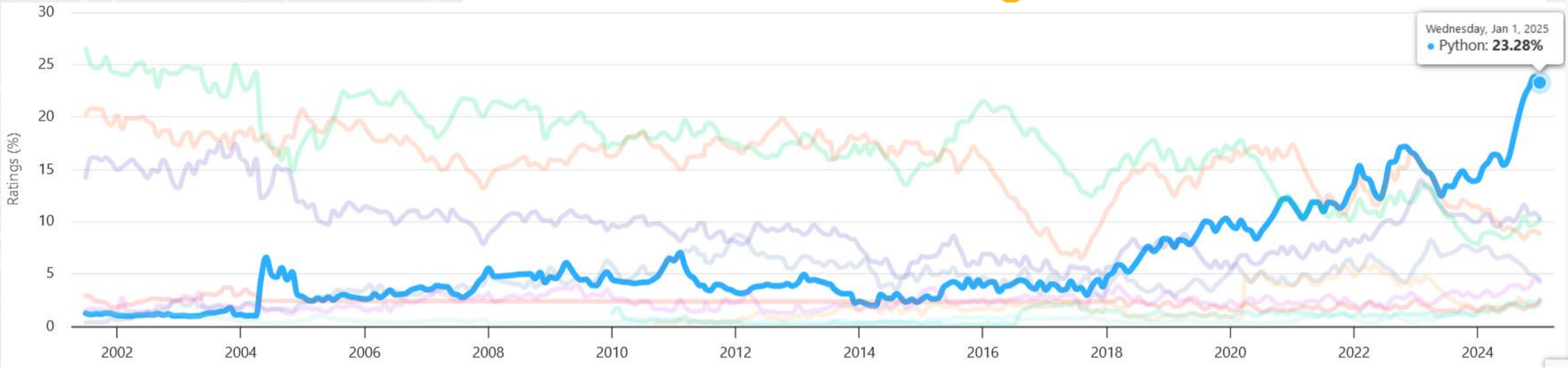
idea: flexible, simple and compact syntax, extendable

Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

TIOBE index Jan 2025

Jan 2025	Jan 2024	Change	Programming Language	Ratings	Change
1	1		Python		23.28%
2	3				
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)

object oriented (OOP): creating objects/types of different properties (see later)
C++, Fortran 2003, Java, MATLAB, **Python**, Ruby, …

the *"style"* of programming

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

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

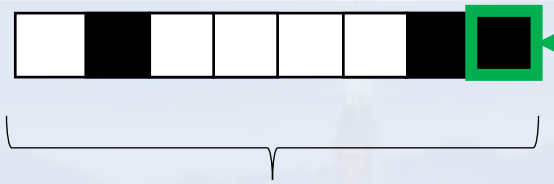
how a programming language *"talks"* to your CPU or GPU

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	

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



$n = 8$

one bit (**bi**nary dig**it**)

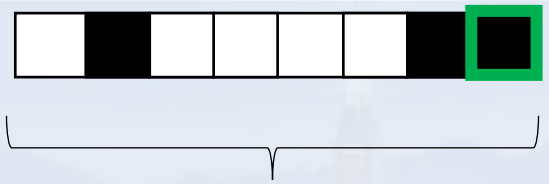
yes/no
on/off
up/down

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

How many different states can I create with 8 switches? 2 x 2 x 2 … = 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

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



$n = 8$

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

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
…	

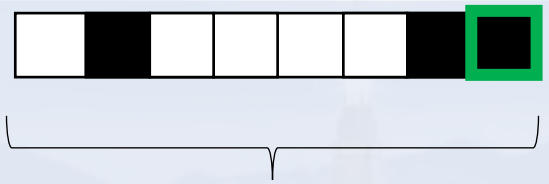
We need only **two** states: **on** and **off**

$$0 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$
$$1 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$
$$2 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$$

$$7 = 0 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$$

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	
…		

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



$n = 8$

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

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

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

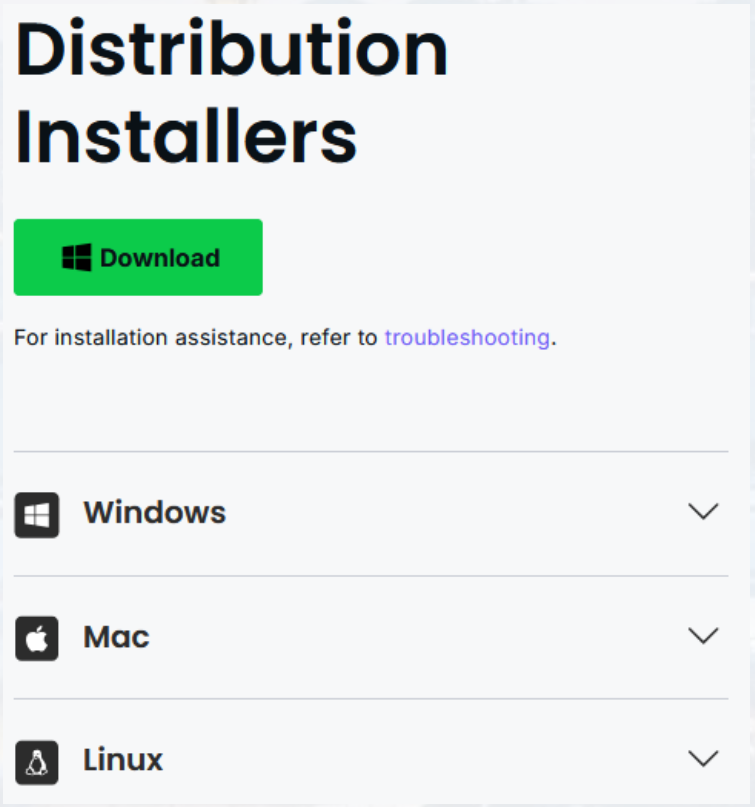
- requires a few more steps to install
- free
- you need to install the specific tools
- your choice if you prefer flexibility over convenience

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

ANACONDA Navigator

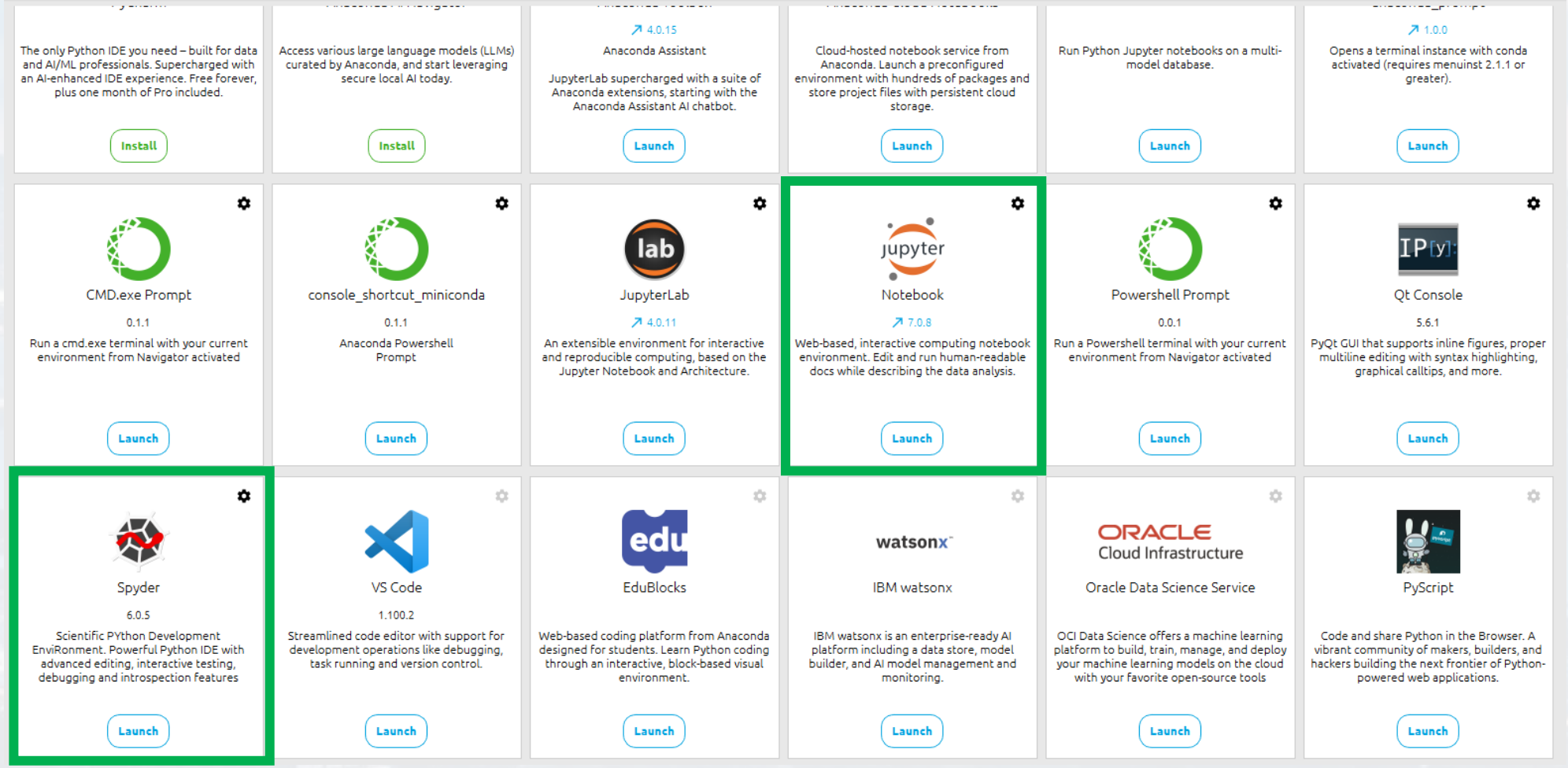
- straight forward installation
- free for teaching/instructions
- more tools than we actually need
- your choice if you have never programmed

download **here**



ANACONDA Navigator

- straight forward installation
- free for teaching/instructions
- more tools than we actually need



Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

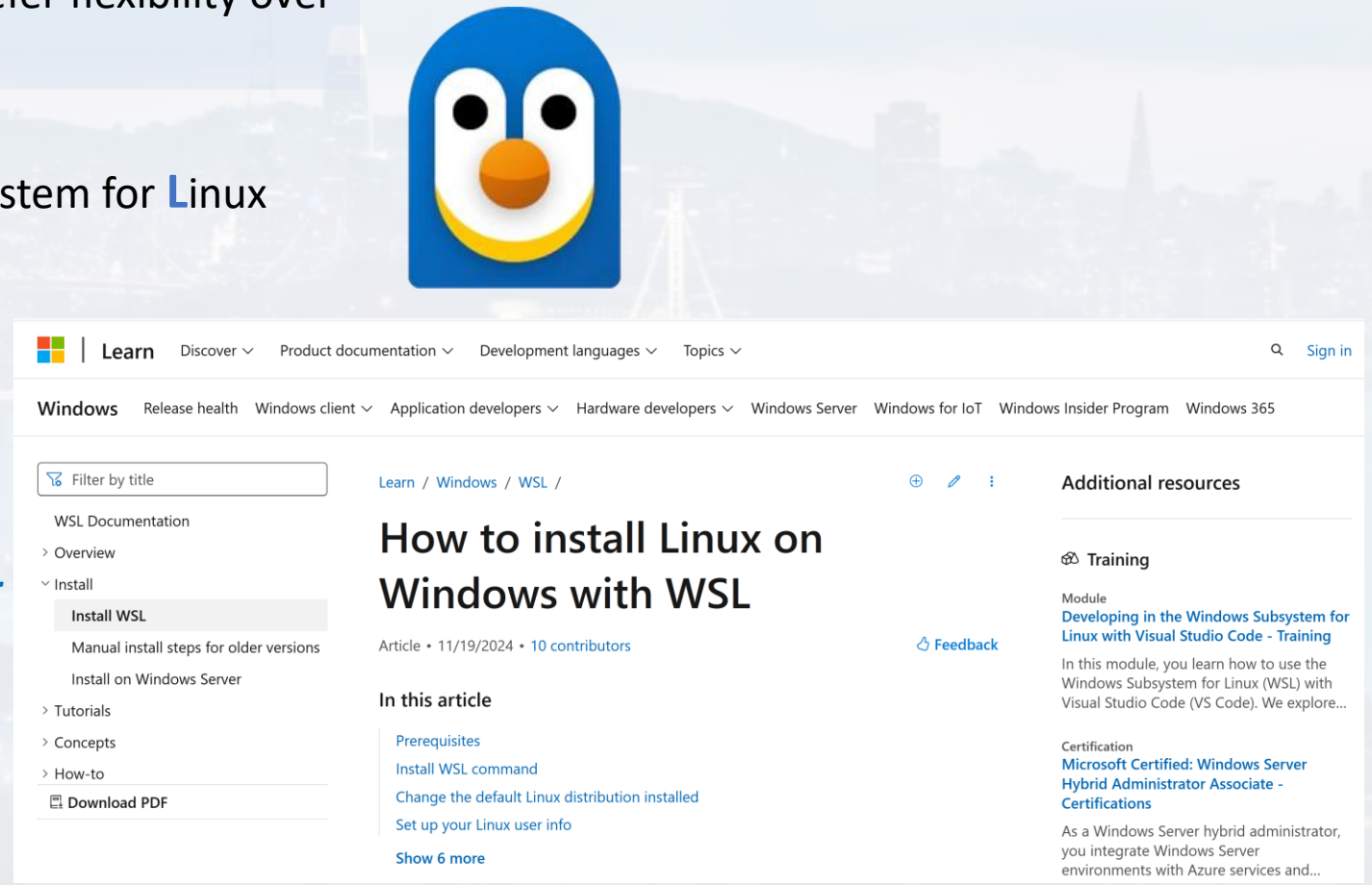
miniconda

- 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 **W**indows **S**ubsystem for **L**inux

follow the instructions here

watch video ***01a_Installing_WSL***



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***

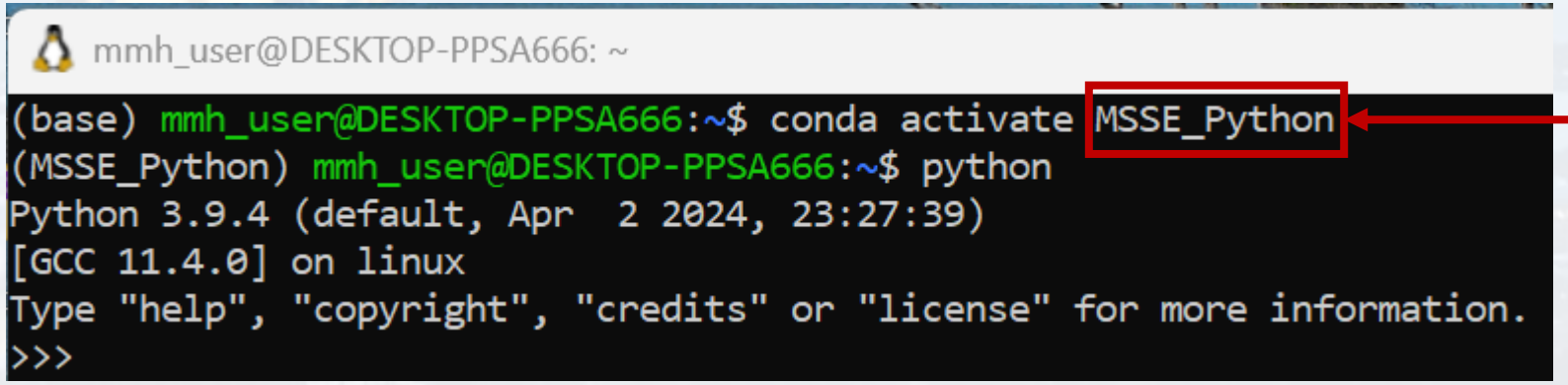
miniconda **on PC:**

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

6) check environment: `conda env list`

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

8) check python:



<My_Environment>

miniconda **on PC:**

9) install Jupyter: `conda install jupyter`

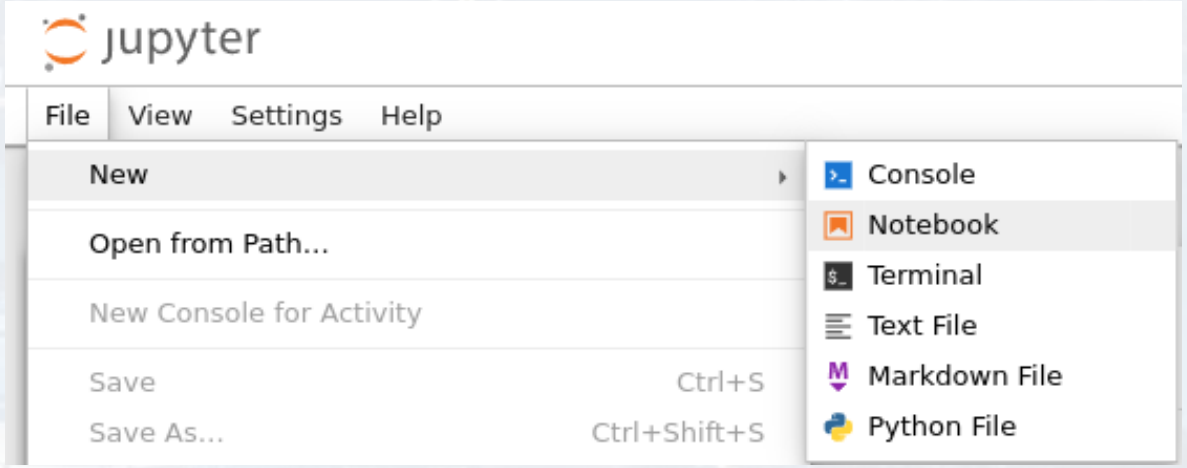
```
(MSSE_Python) mmh_user@DESKTOP-PPSA666:~$ conda install jupyter
```

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***

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
```

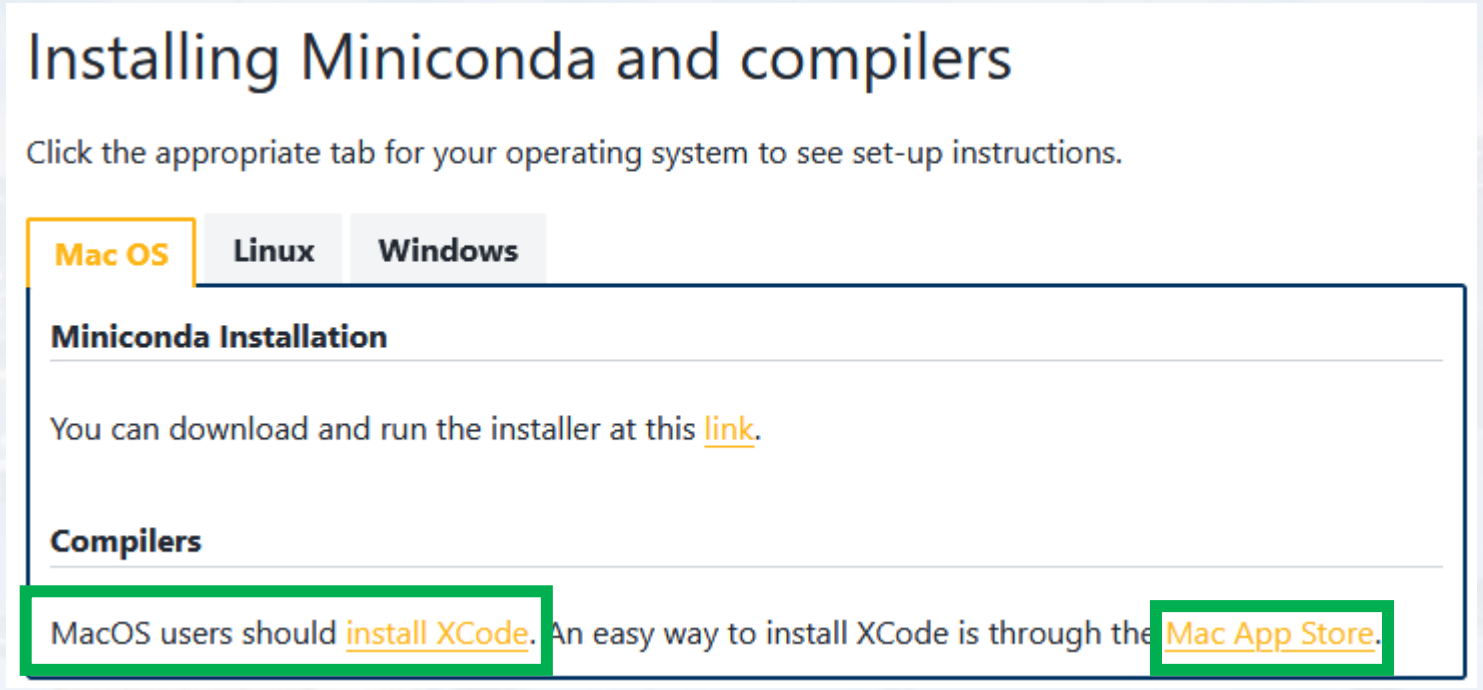
13) open spyder

```
spyder &
```

watch video ***01e_Installing_Spyder***

miniconda **on MAC:**

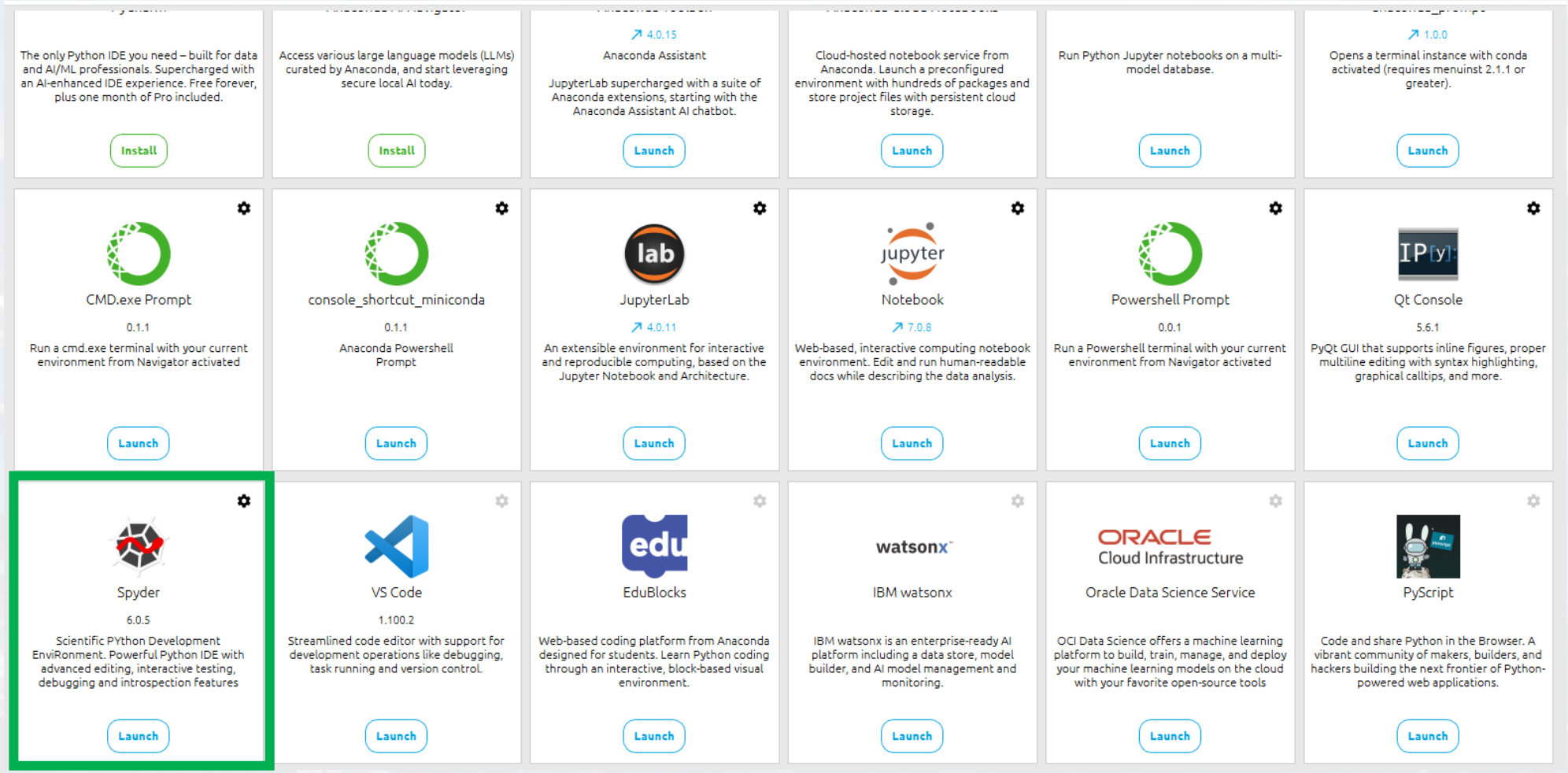
follow instructions **here**



Introduction to Computational Techniques in Physics: fundamentals

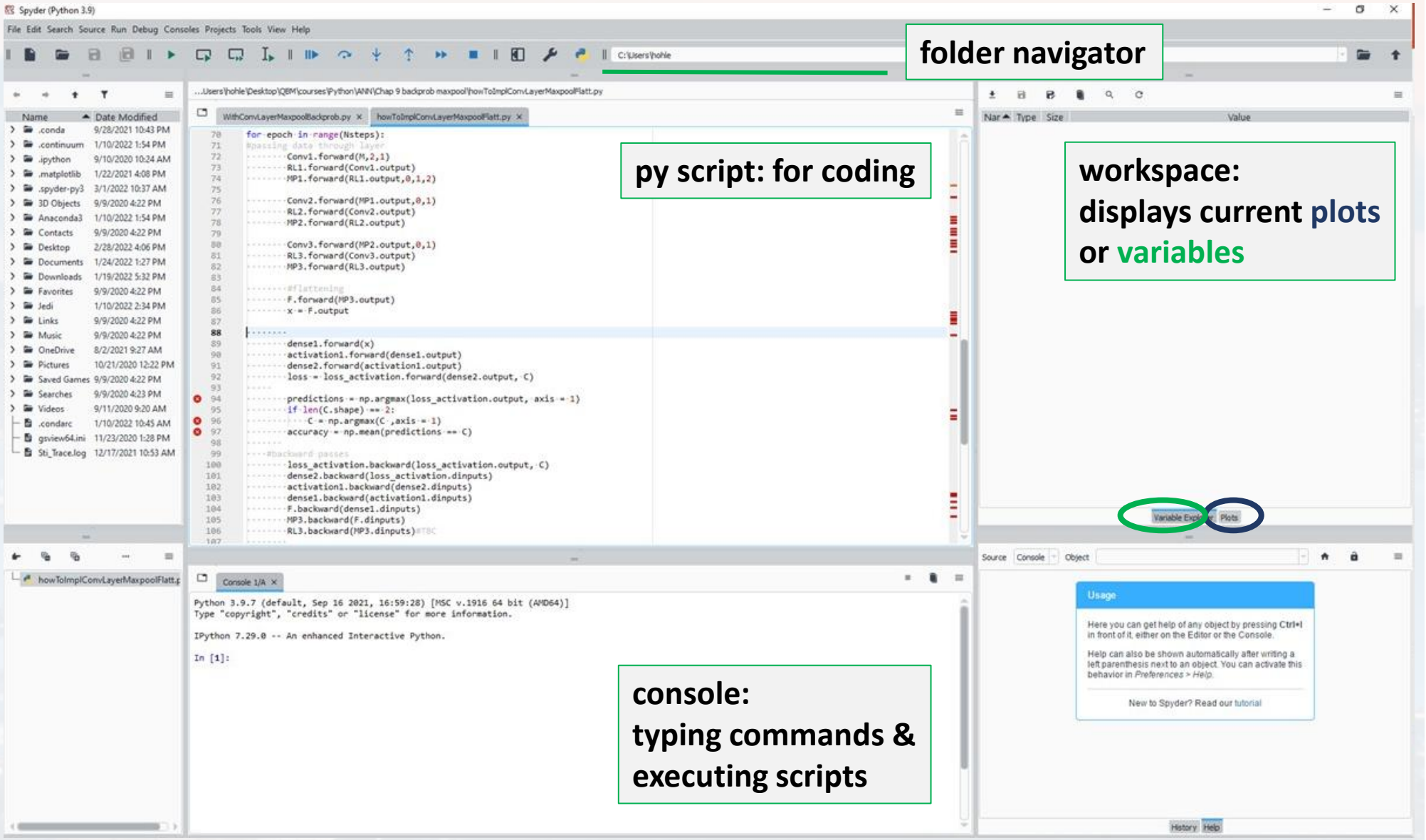
the environment
programming fundamentals
downloading UI
Spyder & Jupyter

watch video ***01f_Quick_Guide_Spyder***



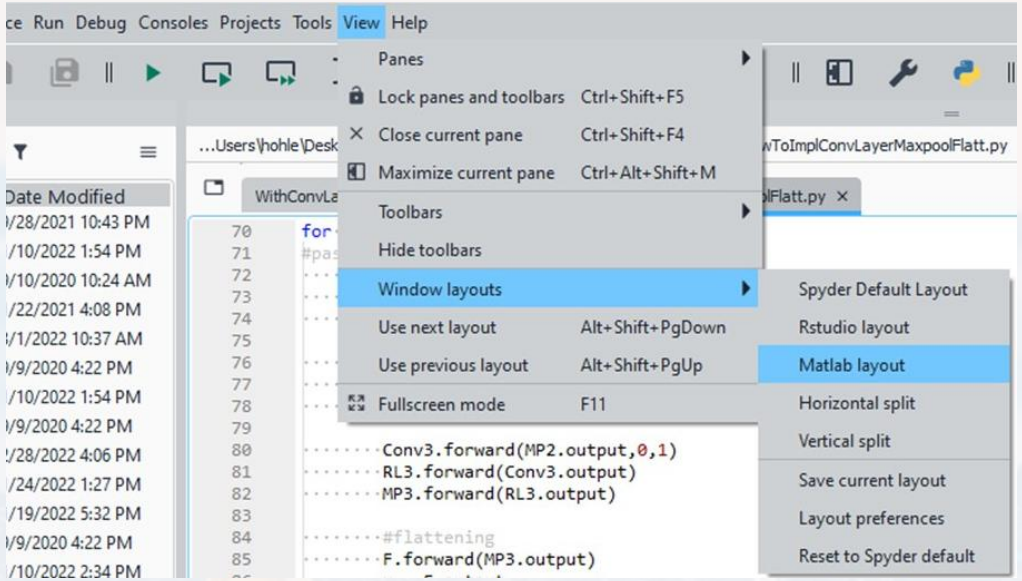
Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter



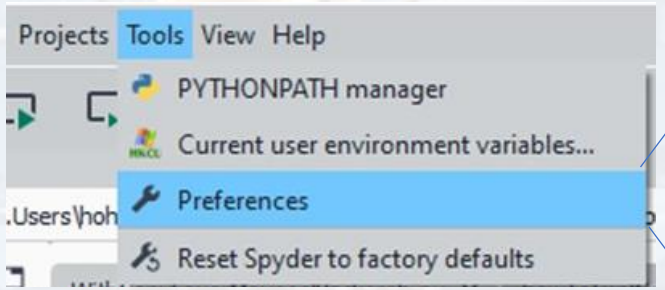
Introduction to Computational Techniques in Physics: fundamentals

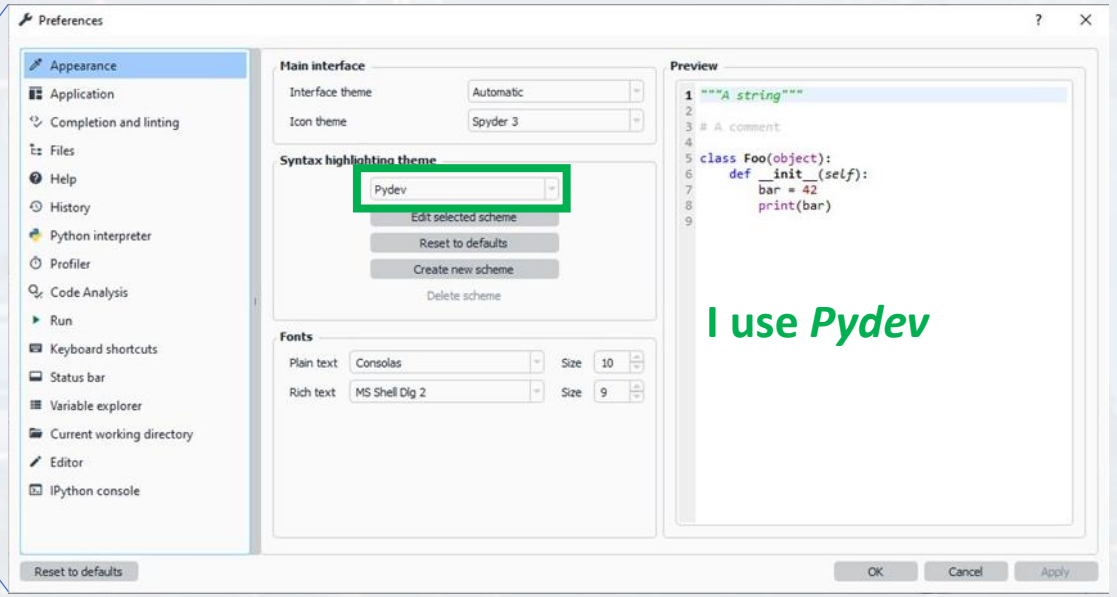
the environment
programming fundamentals
downloading UI
Spyder & Jupyter



settings:
toolbar: *View* → *Window layouts*

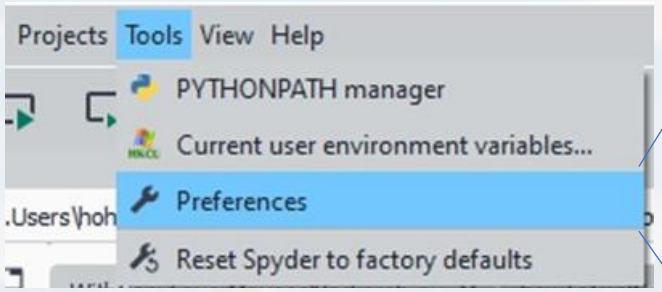
e. g. Matlab

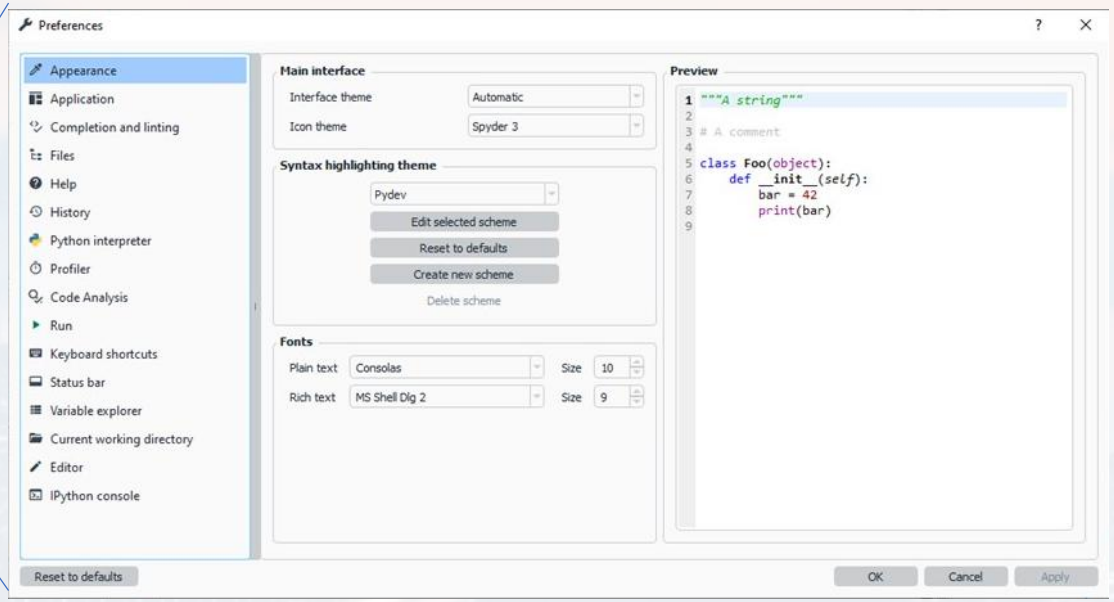


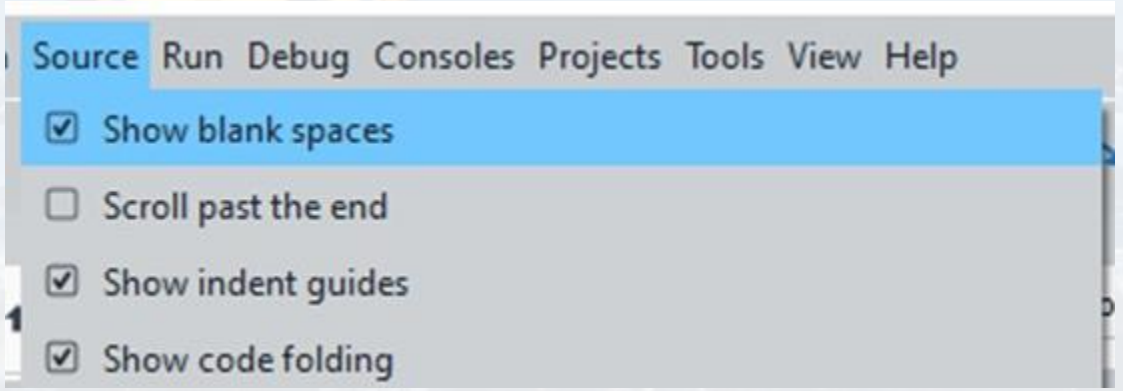


Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter



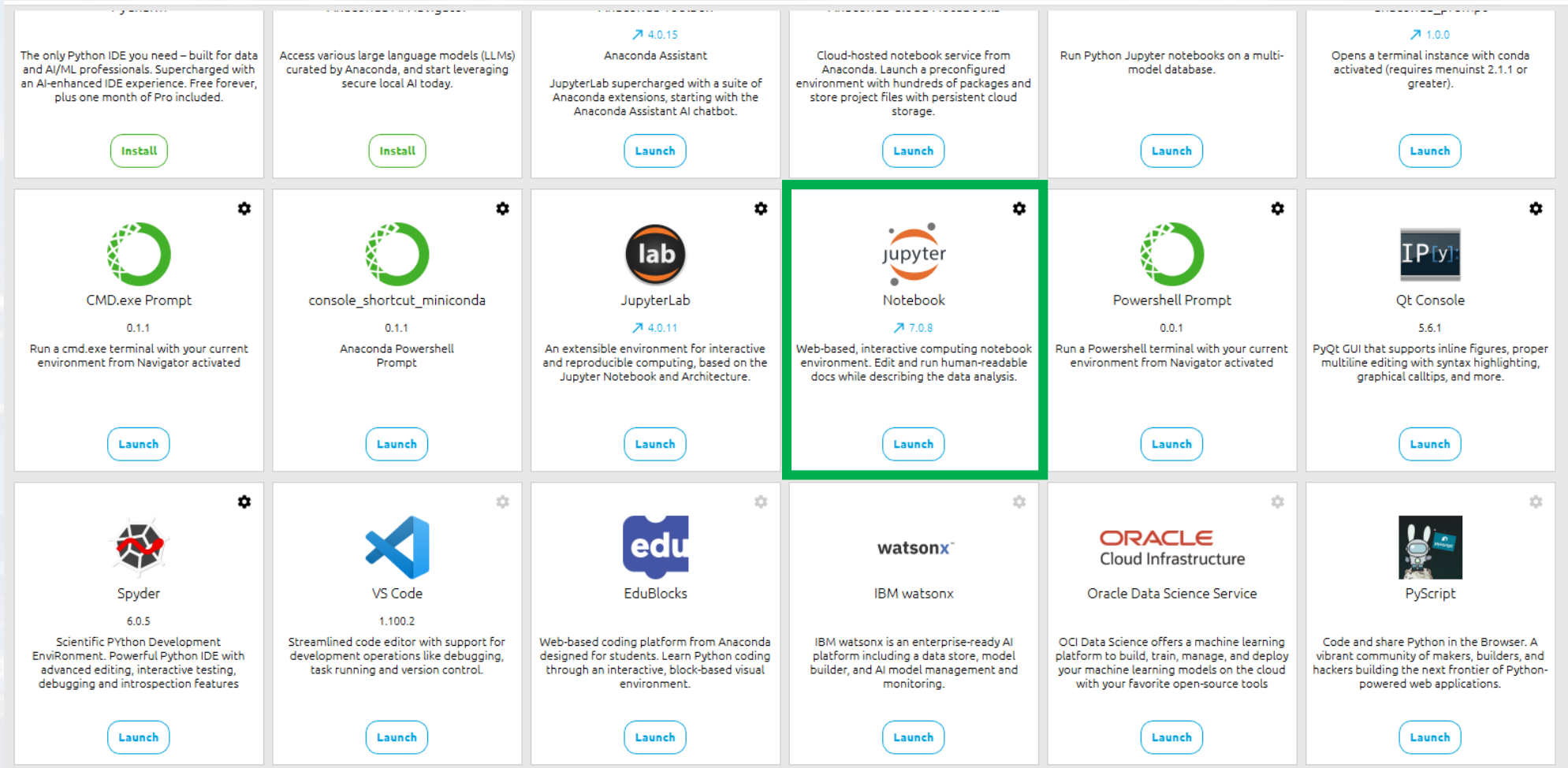




spaces are relevant for syntax!

Introduction to Computational Techniques in Physics: fundamentals

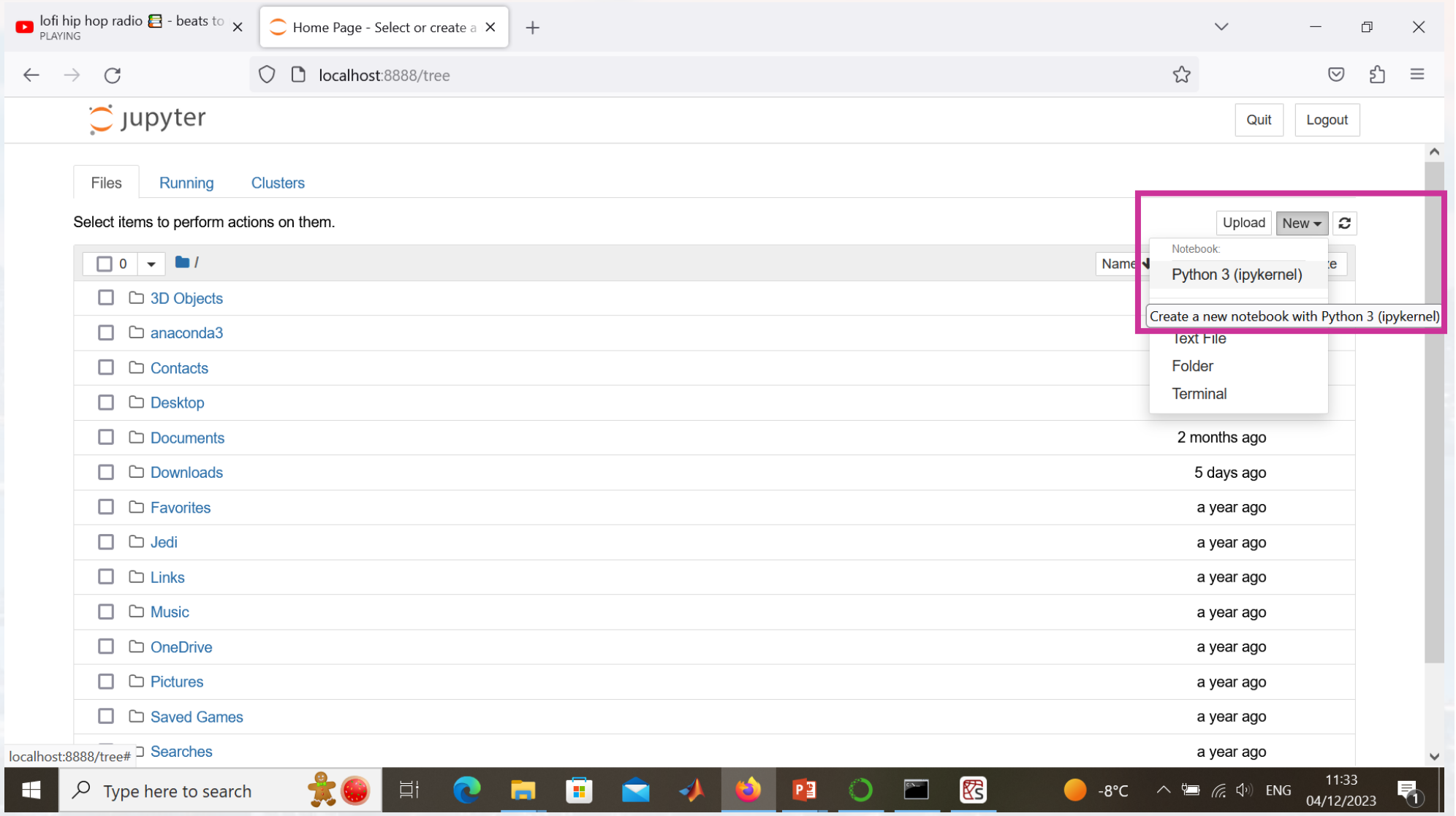
the environment
programming fundamentals
downloading UI
Spyder & Jupyter



Introduction to Computational Techniques in Physics: fundamentals

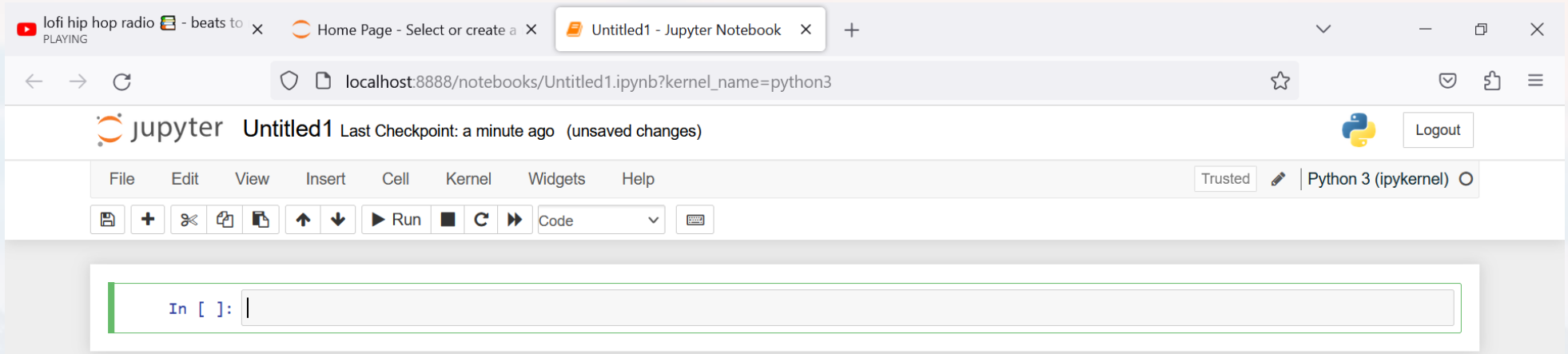
the environment
programming fundamentals
downloading UI
Spyder & Jupyter

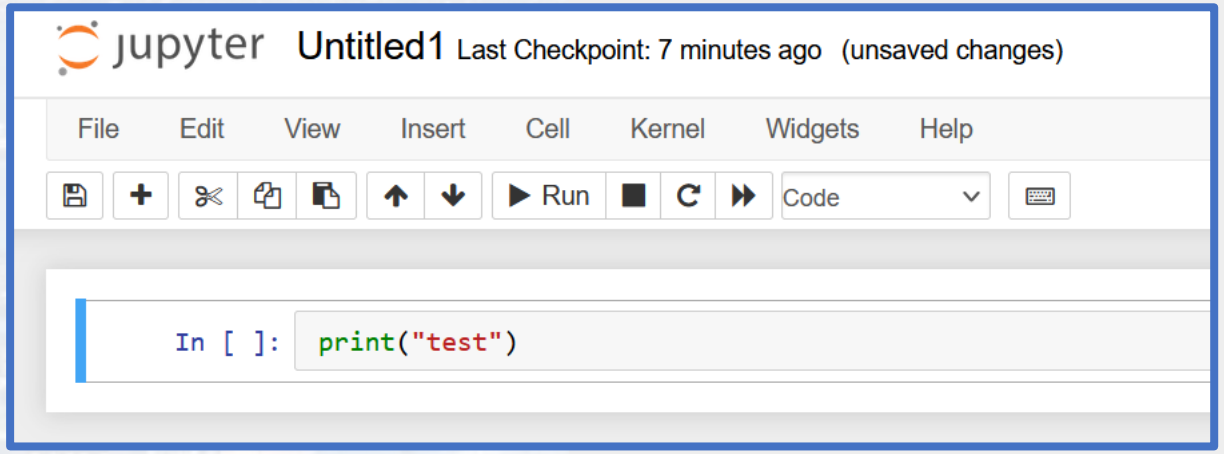
watch video ***01d_Quick_Guide_Jupyter***

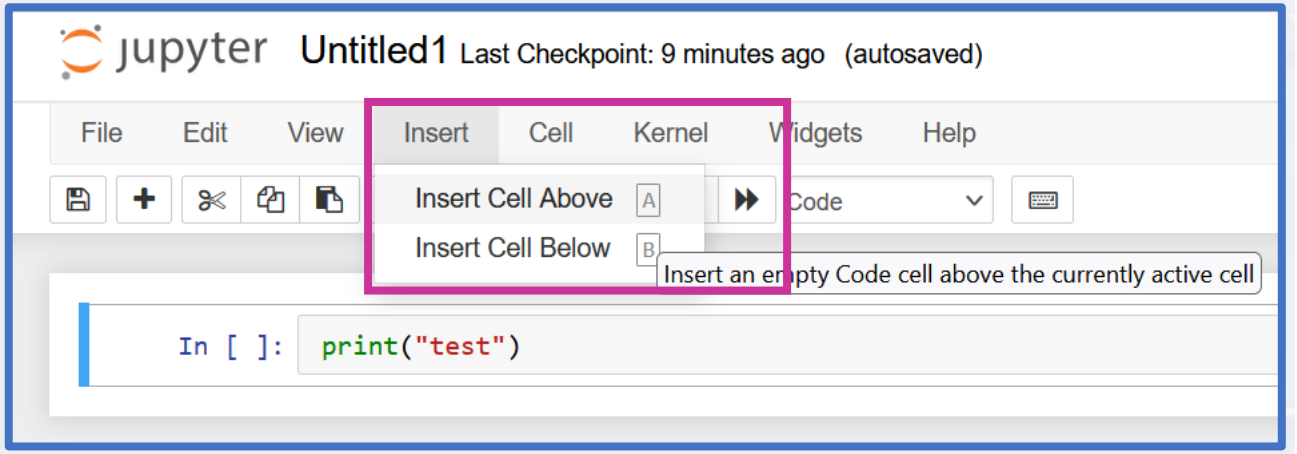


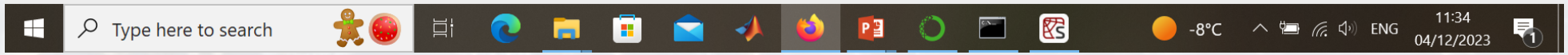
Introduction to Computational Techniques in Physics: fundamentals

the environment
programming fundamentals
downloading UI
Spyder & Jupyter

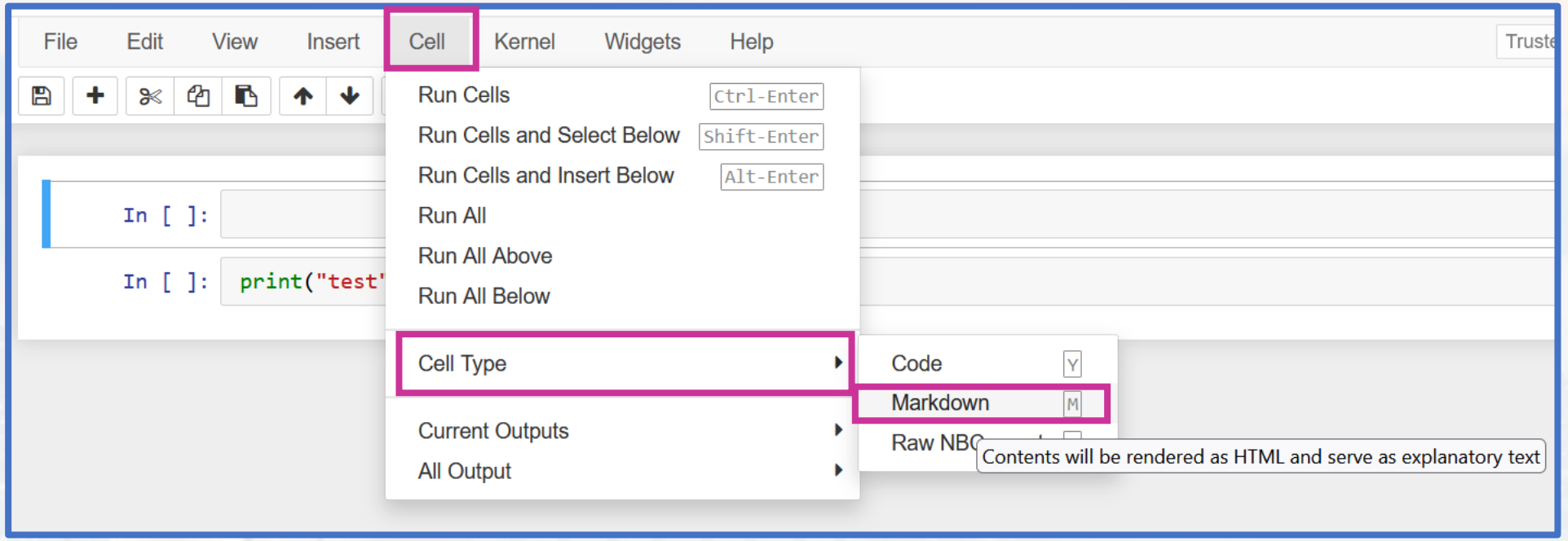




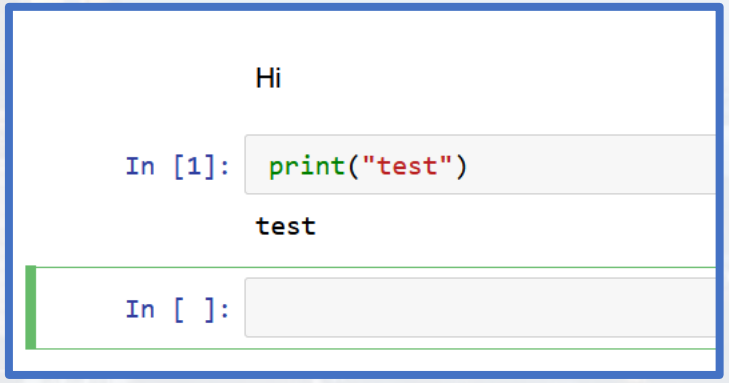




the environment
programming fundamentals
downloading UI
Spyder & Jupyter

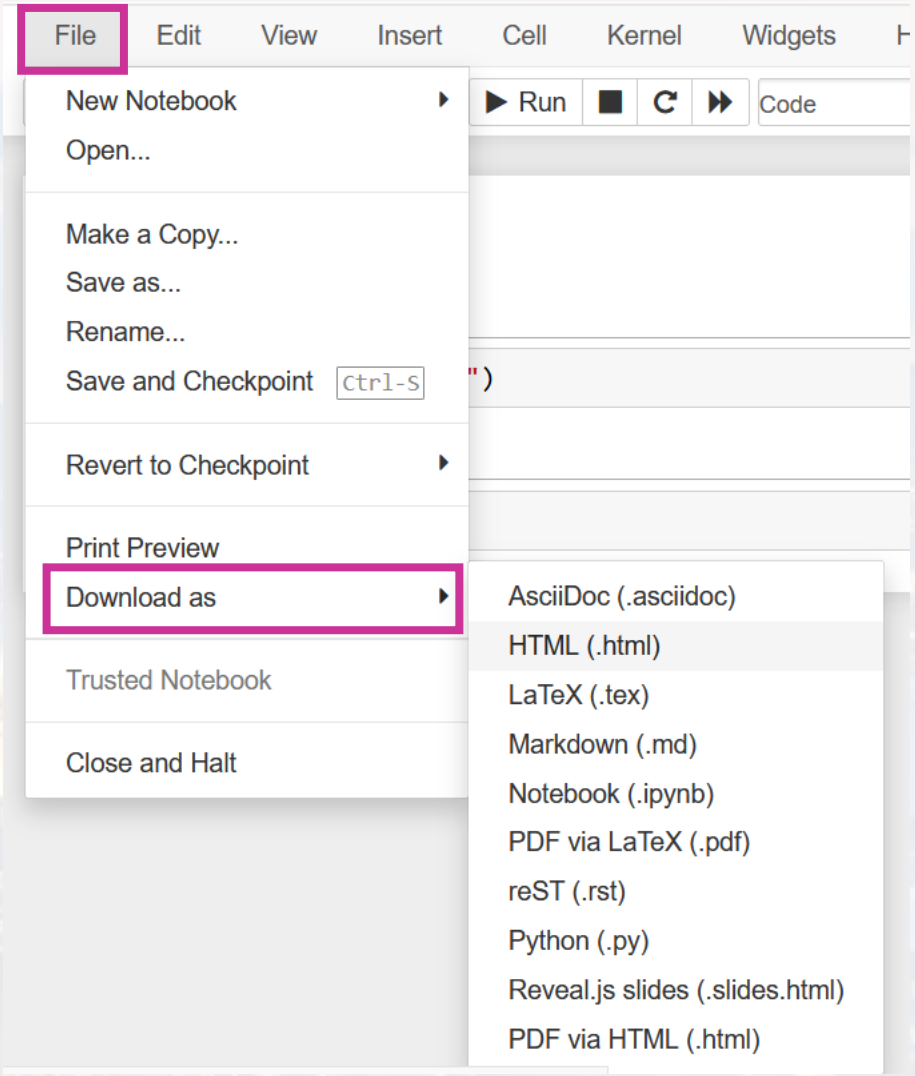


type something → click ***run***



web: Jupyter → markdown styles (including LaTeX)

the environment
programming fundamentals
downloading UI
Spyder & Jupyter



the environment
programming fundamentals
downloading UI
Spyder & Jupyter

You can run Jupyter on Datahub too!

Berkeley
UNIVERSITY OF CALIFORNIA

Operated by the College of Computing, Data Science, and Society

Log in to continue

Welcome to the University of California, Berkeley DataHub.

jupyter

RStudio

Berkeley
UNIVERSITY OF CALIFORNIA

CalNet Authentication Service

sign in

CalNet ID:

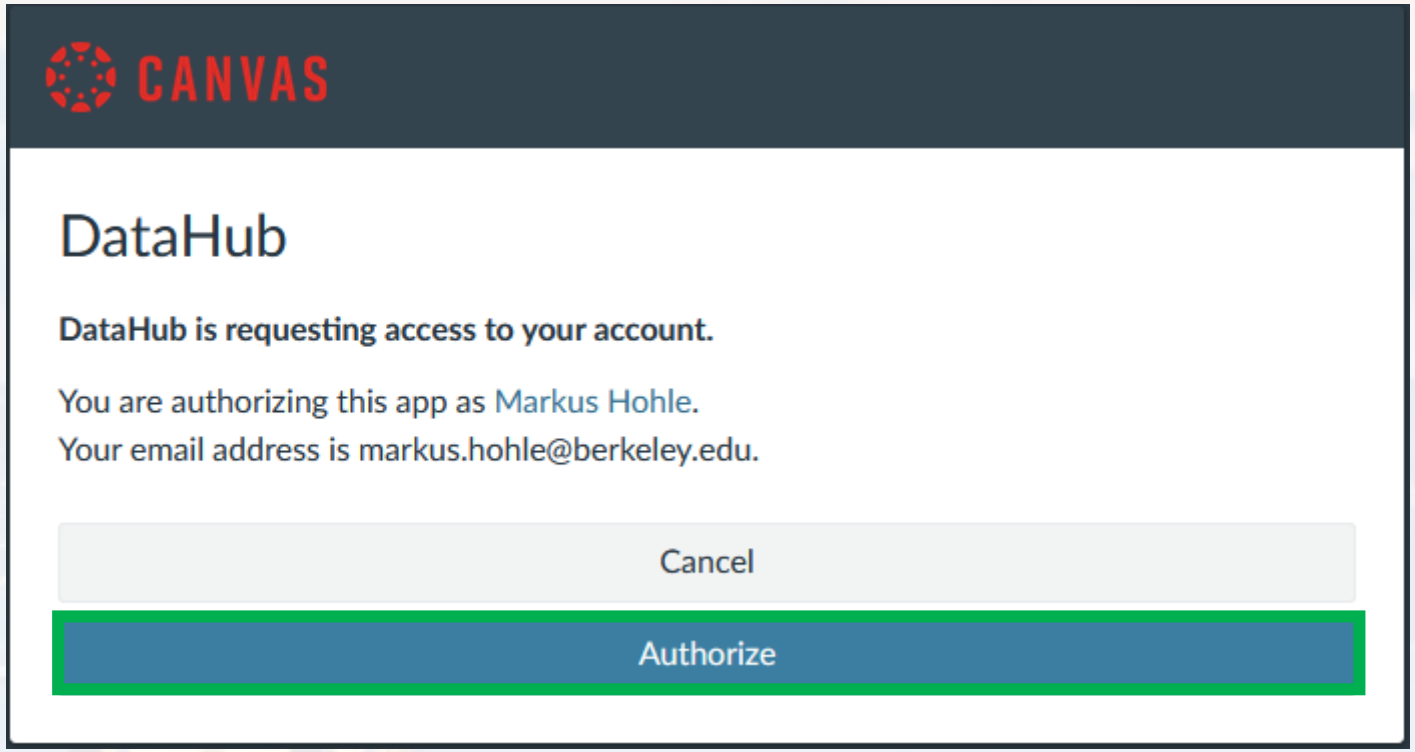
Passphrase (Case Sensitive):

SIGN IN HELP Sponsored Guest Sign In

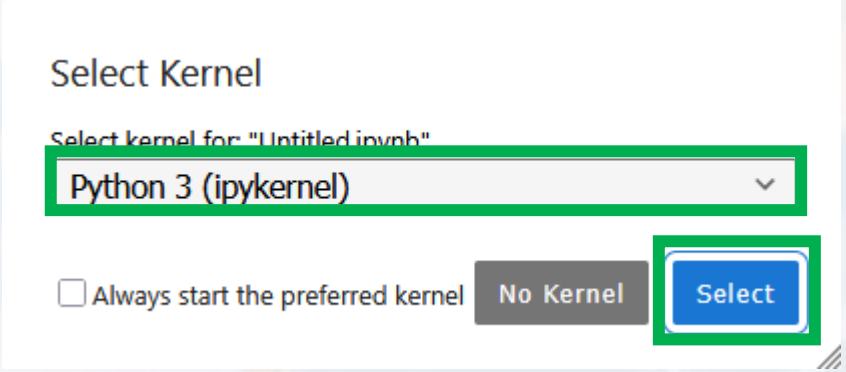
FORGOT CALNET ID OR PASSPHRASE?

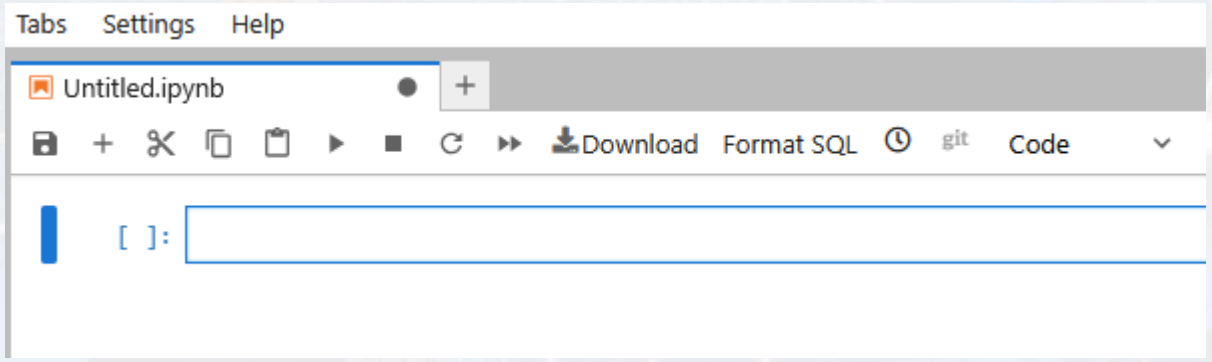
MANAGE MY CALNET ACCOUNT

You can run Jupyter on Datahub too!



You can run Jupyter on Datahub too!





Introduction to Computational Techniques in Physics:

Thank you for your attention!