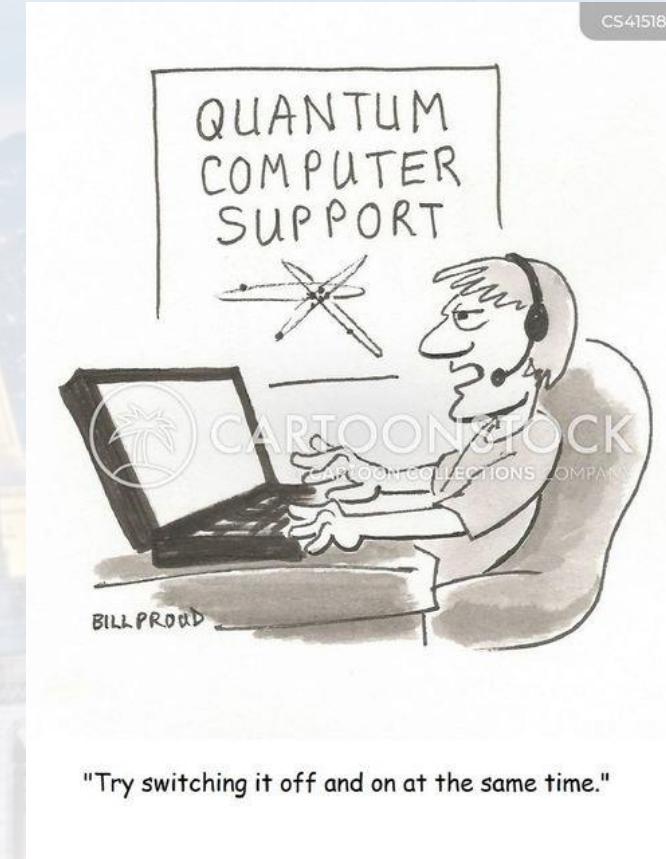




M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics





<u>syllabus:</u>	- Introduction to Unix & Python	(week 1 - 2)
	- Functions, Loops, Lists and Arrays	(week 3 - 4)
	- Visualization	(week 5)
	- Parsing, Data Processing and File I/O	(week 6)
	- Statistics and Probability, Interpreting Measurements	(week 7 - 8)
	- Random Numbers, Simulation	(week 9)
	- Numerical Integration and Differentiation	(week 10)
	- Root Finding, Interpolation	(week 11)
	- Systems of Linear Equations	(week 12)
	- Ordinary Differential Equations	(week 13)
	- Fourier Transformation and Signal Processing	(week 14)
	- Capstone Project Presentations	(week 15)



Week	Dates	Topics	Reading	Lecture Slides	Workshop	Homework
1 - 2	Aug 28th / Sep 4th	Introduction to Unix and Python	K&N Ch. 1			
3 - 4	Sep 11th / Sep 18th	Functions, Loops, Lists, Arrays	K&N Ch.2-3			
5	Sep 25th	Visualization	K&N Ch. 4, K&N Ch. 6.3-6.4			
6	Oct 2nd	Parsing, Data Processing, and File I/O	K&N Ch. 4			
7 - 8	Oct 9th / Oct 16th	Statistics and Probability, Interpreting Measurements	Hughes			



Week	Dates	Topics	Reading	Lecture Slides	Workshop	Homework
9 - 10	Oct 23rd / Oct 30th	Random Numbers, Simulation	K&N Ch. 6, Newman Ch. 10			
11 - 12	Nov 6th / Nov 13th	Numerical Integration and Differentiation	K&N Ch. 6 Newman Ch. 5			
13	Nov 20th	Root finding, Interpolation	K&N Ch. 6.5 Newman Ch. 6			
14	Nov 27th	Systems of Linear Equations	Newman Ch. 6 K&N Ch. 6.6			
15	Dec 4th	Ordinary Differential Equations	K&N Ch. 6.8-6.9 Newman Ch. 8			
16	Dec 11th	Fourier Transforms, Signal Processing	Newman Ch. 7			
17	Dec 18th	Capstone Project Presentations				



Lecture: Wednesdays, 2 - 4pm, Physics Building 251

Workshops: Fridays, 2 - 4pm (section 1) Social Science, 170 and
Fridays, 4 - 6pm (section 2) Social Science, 140

Office Hours: Wednesdays, 4 – 6pm, 397 Physics North (17)

Instruction begins: August 28, 2024

Instruction ends: Friday, December 13, 2024

my contact: markus.hohle@berkeley.edu

TA: Krish Desai krish.desai@berkeley.edu

check out: [Google Colab](#)

[bcourse](#)

[datahub.berkeley](#)

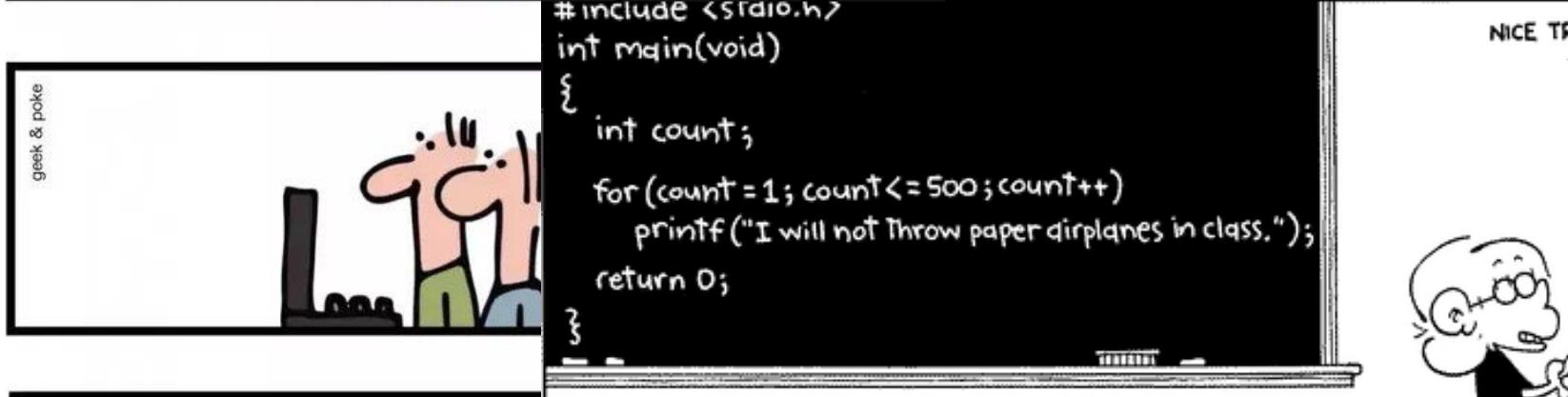


Mixed Audience

beginner



medium



advanced





motivation: The good old days:



Salar de Uyuni,
Bolivia, Mar 2003



motivation:



source: CERN



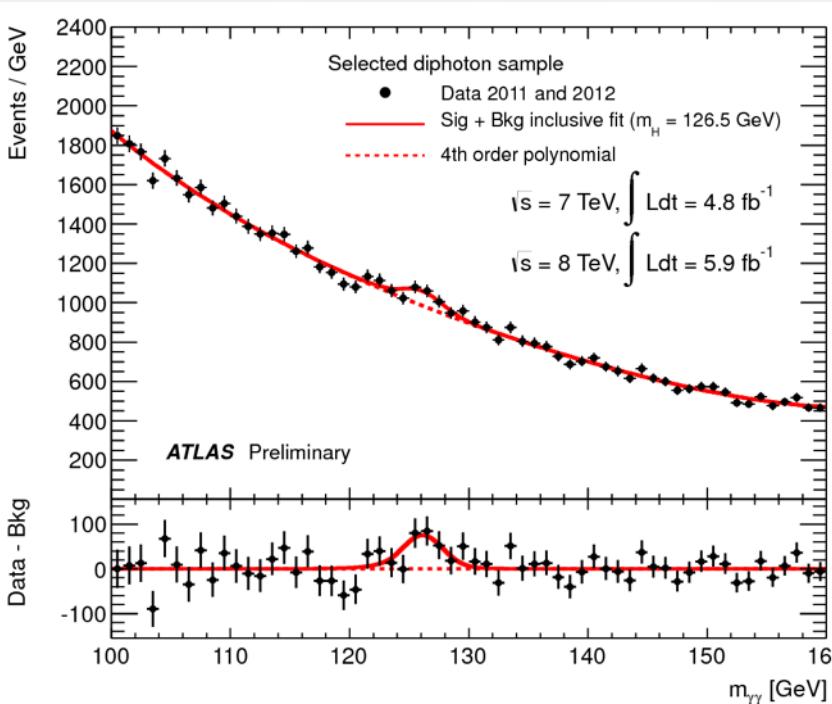
source: LIGO collaboration



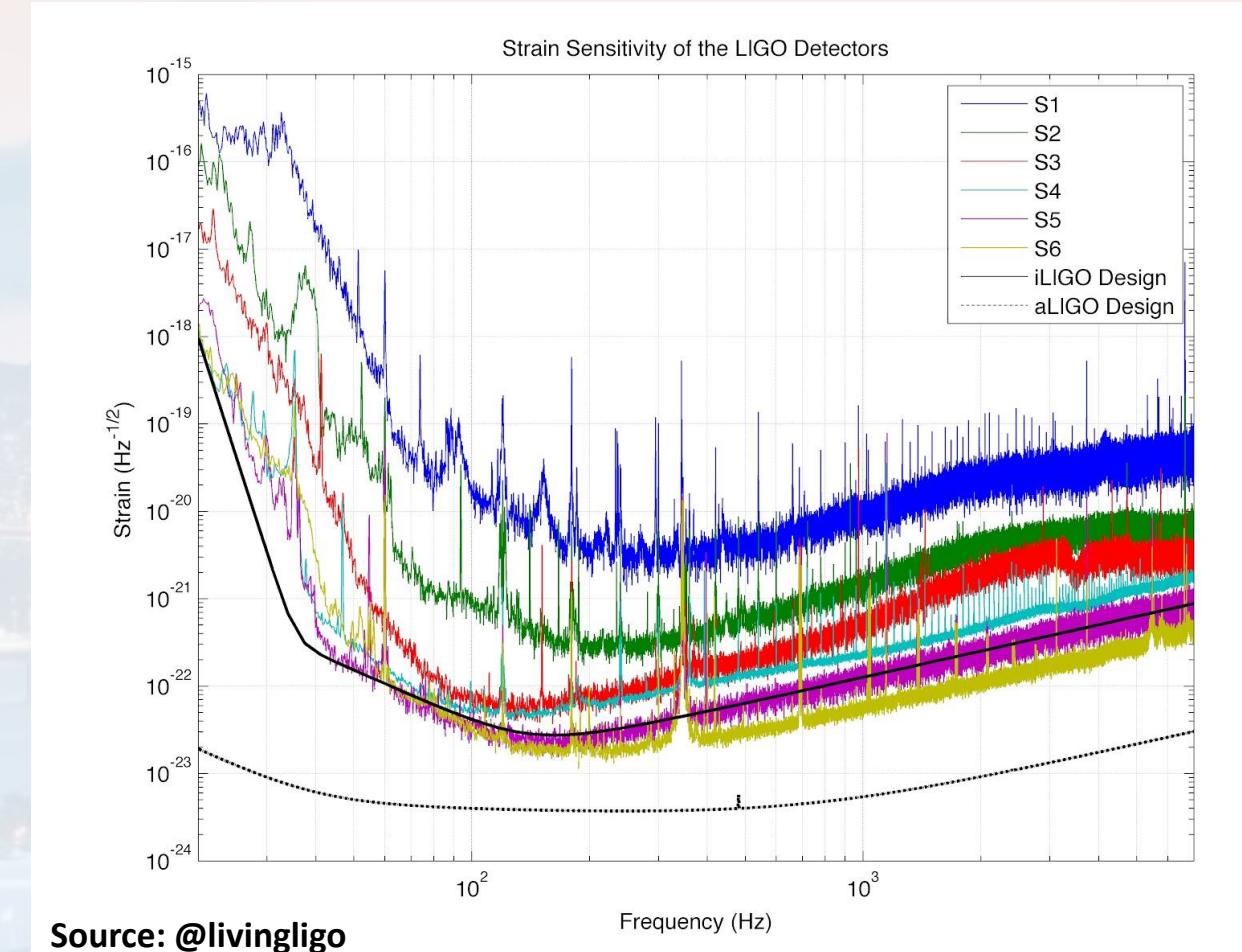
source: ESA



motivation:



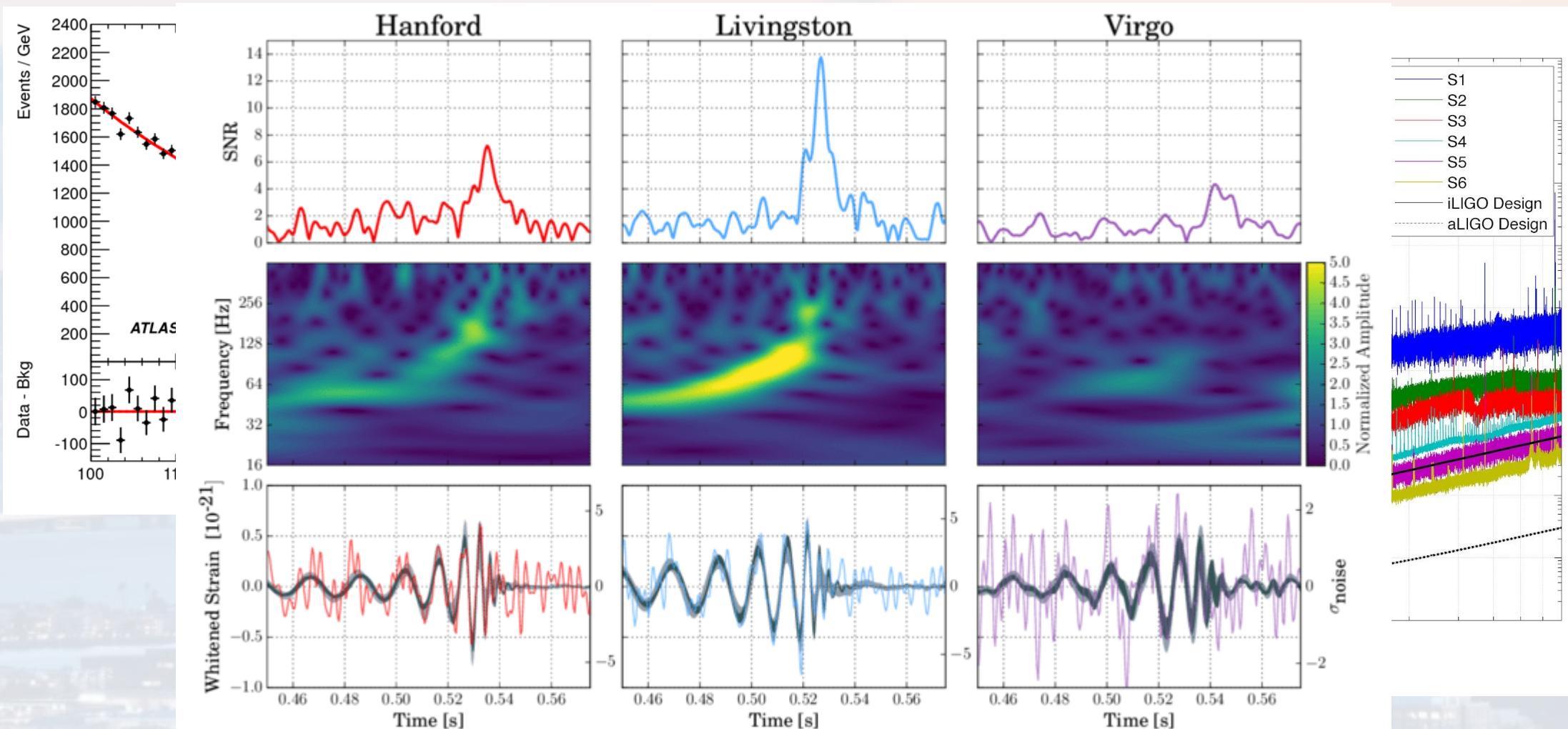
Source: CERN/ATLAS



Source: @livingligo



motivation:





goal: being able to write code like this...

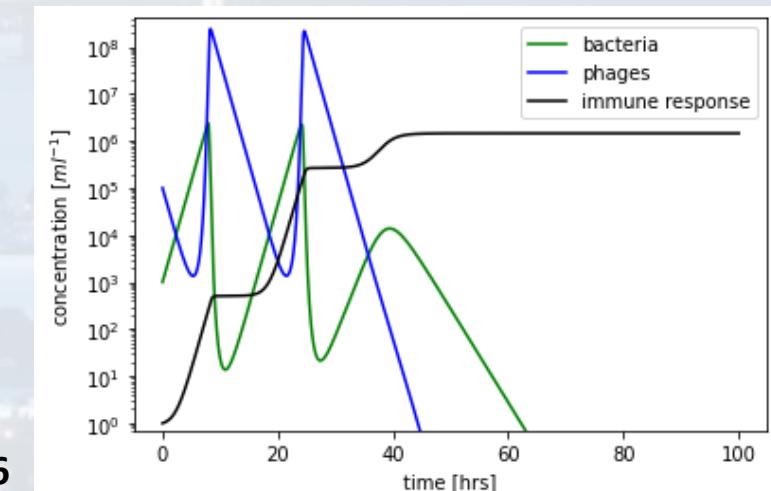
```
@my_timer
class SignalDetect():

    def __init__(self, T, Range_phi = [0, 2*np.pi], dphi = 0.01,\n                 MaxM = 20, **Opts):
```

```
L = listdir(getcwd())
[remove(i) for i in L if '.npy' in i]
```

...

...and simulating systems like that





<u>syllabus:</u>	- Introduction to Unix & Python	(week 1 - 2)
	- Functions, Loops, Lists and Arrays	(week 3 - 4)
	- Visualization	(week 5)
	- Parsing, Data Processing and File I/O	(week 6)
	- Statistics and Probability, Interpreting Measurements	(week 7 - 8)
	- Random Numbers, Simulation	(week 9)
	- Numerical Integration and Differentiation	(week 10)
	- Root Finding, Interpolation	(week 11)
	- Systems of Linear Equations	(week 12)
	- Ordinary Differential Equations	(week 13)
	- Fourier Transformation and Signal Processing	(week 14)
	- Capstone Project Presentations	(week 15)



syllabus:

- Introduction to Unix & Python	(week 1 - 2)
- Functions, Loops, Lists and Arrays	(week 3 - 4)
- Visualization	(week 5)
- Parsing, Data Processing and File I/O	(week 6)
- Statistics and Probability, Interpreting Measurements	(week 7 - 8)
- Random Numbers, Simulation	(week 9)
- Numerical Integration and Differentiation	(week 10)
- Root Finding, Interpolation	(week 11)
- Systems of Linear Equations	(week 12)
- Ordinary Differential Equations	(week 13)
- Fourier Transformation and Signal Processing	(week 14)
- Capstone Project Presentations	(week 15)



Operating System: interface between human and computer

→ **G**raphical **U**ser **I**nterface: Windows, OS X, iOS, Android

→ text based: MS-DOS

→ mixed: Unix

Unix: - Bell Labs, early 70s
 - Written in C and Assembly

Linux: - as an alternative to Unix
 - Linus Torvalds (1991)
 - Ubuntu, SUSE, ... **Scientific Linux**



efficient, fast, robust
optimized, but...

... not suitable for
the *standard* user



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

the “style” of programming

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, ...

how a programming language “talks” to you CPU or GPU

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

source: wikipedia

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



Bits & Bytes: We need only **two** states: **on** and **off**

How many different states can I create with 8 switches?



$$2 \times 2 \times 2 \dots = 2^n$$

smallest memory cell: 8bits = 1byte
 bit stands for **binary digit**

For some reason humans use **ten** states $a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$N_{dec} = \sum_{i=0} a_{ji} 10^i$$

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
...	



Bits & Bytes: We need only **two** states: **on** and **off**

For some reason humans use **ten** states $a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$$N_{dec} = \sum_{i=0} a_{ji} 10^i$$

vs **two** states $a = \{0, 1\}$

$$N_{bin} = \sum_{i=0} a_{ji} 2^i$$

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
...	

Let's write the number 15 in: *decimal*, *binary* and to base *three*:

**Bits & Bytes:**

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



So, we are fine with natural numbers...

... what is with negative numbers...

...or fractions...

...or π and e

→ float and double numbers need three *fields*:

- sign
- exponent
- fraction

more [here](#)

8 bits	= 1 byte (B)
1 kB	= 1024 B
1 MB	= 1024 kB etc

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
...	...



Bits & Bytes:

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

	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	$\approx 10^{-16}$	$\approx 10^{-308} \dots 10^{308}$

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



build your environment

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the 'EXTENSIONS: MARKETPLACE' tab open, displaying several extensions: Python (version 2019.6.24221), GitLens — Git supercharged (version 9.8.5), C/C++ (version 0.24.0), ESLint (version 1.9.0), Debugger for Chrome (version 4.11.6), Language Support for Java (version 0.47.0), vscode-icons (version 8.8.0), and Vetur (version 0.21.1). The main editor area shows a file named 'blog-post.js' with code related to a Gatsby blog post component. The code includes imports for React and gatsby-image, and a template literal for rendering a blog post. A tooltip is visible over the word 'data'. The bottom status bar shows the terminal output: 'info : [wdm]: Compiling...', 'DONE Compiled successfully in 26ms', and the date '3:57:58 PM'. The bottom right corner of the status bar also shows a bell icon.

```
src > components > JS blog-post.js > <function> > blogPost
1 import React from "react"
2 import Image from "gatsby-image"
3
4 export default ({ data }) => {
5   const blogPost = data.cms.blogPost
6   return (
7     <div>
8       {blogPost.debug}
9       {blogPost.debugger}
10      {blogPost.decodeURI}
11      {<Image decodeURIComponent=> default}
12      <h1>{blogPost.defaultStatus}
13      <div>Post {<div> delete
14      <div> dang {<div> departFocus
15      </div> devicePixelRatio
16      <div> dispatchEvent
17    }
18  }
19}
20
21 export const query = graphql`
```

PROBLEMS TERMINAL ... 2: Task - develop

info : [wdm]: Compiling...
DONE Compiled successfully in 26ms 3:57:58 PM
info : [wdm]:
info : [wdm]: Compiled successfully.

master 0 1 0 0 1 Gatsby Develop (gatsby-graphql-app) Ln 6, Col 21 Spaces: 2 UTF-8 LF JavaScript

<https://themanifest.com/software-development/blog/programming-engineer>



1) if your OS is Windows

→ install Windows Subsystem for Linux (WSL)

→ follow the instructions [here](#)



2) if your OS is Unix/Linux

→ we can start right away!

3) optional

→ once you have Linux, there are lots of [useful tools](#) you can install and link together

- Google Chrome
- Visual Studio Code (aka VS Code)
- NodeJS
- Docker



Berkeley
UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics:

Unix & Python



- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment





list your files and folders

ls

```
(base) mmh_user@DESKTOP-PPSA666:~$ ls
Anaconda3-2023.09-0-Linux-x86_64.sh  AppAcademy
Anaconda3-2024.06-1-Linux-x86_64.sh  Downloads
                                         Untitled.ipynb  snap
                                         anaconda3
```

files

folder

shell scripts
→like executables



list your files and folders

ls

-lrt

- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment

called *flags*

- **l** stands for *long*: shows a file with all permissions and properties, each per line
- **r** stands for *reverse* (oldest first)
- **t** stands for the *time* flag (as criterium for r)

```
(base) mmh_user@DESKTOP-PPSA666:~$ ls -lrt
total 2158464
drwx-----  2 mmh_user mmh_user      4096 Apr  8 20:03 Downloads
drwx-----  3 mmh_user mmh_user      4096 Apr 10 19:16 snap
drwxr-xr-x  5 mmh_user mmh_user      4096 Apr 24 18:29 AppAcademy
-rw xr-xr-x  1 mmh_user mmh_user 1153404010 Jul 24 12:36 Anaconda3-2023.09-0-Linux-x86_64.sh
-rw xr-xr-x  1 mmh_user mmh_user 1056829859 Aug  5 18:53 Anaconda3-2024.06-1-Linux-x86_64.sh
drwxr-xr-x 31 mmh_user mmh_user      4096 Aug  5 19:22 anaconda3
-rw-r--r--  1 mmh_user mmh_user       616 Aug  5 19:30 Untitled.ipynb
```



list your files and folders

ls -lrt

but these are only those files you *see*.
try:

ls -la

```
(base) mmh_user@DESKTOP-PPSA666:~$ ls -la
total 2158640
drwxr-x--- 23 mmh_user mmh_user          4096 Aug  5 19:41 .
drwxr-xr-x  3 root     root            4096 Apr  2 21:35 ..
drwxr-xr-x  4 mmh_user mmh_user          4096 Apr  3 04:49 .aa-setup-checker
drwxr-xr-x  3 mmh_user mmh_user          4096 Aug  4 13:14 .anaconda
lrwxrwxrwx  1 mmh_user mmh_user          21 Apr  3 04:31 .aws -> /mnt/c/Users/MMH/.aws
lrwxrwxrwx  1 mmh_user mmh_user          23 Apr  3 04:31 .azure -> /mnt/c/Users/MMH/.azure
-rw-----  1 mmh_user mmh_user        26898 Aug  5 23:07 .bash_history
-rw-r--r--  1 mmh_user mmh_user         220 Apr  2 21:35 .bash_logout
-rw-r--r--  1 mmh_user mmh_user        4582 Jul 24 12:40 .bashrc
drwx----- 11 mmh_user mmh_user          4096 Aug  5 19:42 .cache
drwxr-xr-x  2 mmh_user mmh_user          4096 Jul 24 12:42 .conda
-rw-r--r--  1 mmh_user mmh_user          25 Aug   5 19:03 .condarc
drwx-----  6 mmh_user mmh_user          4096 Aug  5 19:45 .config
drwxr-xr-x  5 mmh_user mmh_user          4096 Apr  3 04:31 .docker
```

will be
important
(see later)

pointer/
links/aliases



list your files and folders

```
ls -lrt
```

```
ls -la
```

You can search for files/folders with particular substrings using a “wildcard”

```
ls *.py
ls *py*
```

```
(base) mmh_user@DESKTOP-PPSA666:~$ ls -lrt *A*
-rw-r-xr-x 1 mmh_user mmh_user 1153404010 Jul 24 12:36 Anaconda3-2023.09-0-Linux-x86_64.sh
-rw-r-xr-x 1 mmh_user mmh_user 1056829859 Aug  5 18:53 Anaconda3-2024.06-1-Linux-x86_64.sh
```

```
AppAcademy:
total 12
drwxr-xr-x 13 mmh_user mmh_user 4096 Apr 17 18:34 HTMLExercises
drwxr-xr-x  4 mmh_user mmh_user 4096 Apr 23 21:13 GitExercises
drwxr-xr-x  7 mmh_user mmh_user 4096 May  3 23:36 JavaExercises
```



changing your directory

cd

always leads back to the home directory

cd ../

one level up

cd ../../my_dir

one level up, down to my_dir

cd another/dir

one level down to dir

mkdir test

creating the new directory *test*

rm -r test

removing the directory using the flag *r* (here: recursively)

rm any_file

when removing a file, no flag is needed

- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment

Danger: Be careful with rm! There is no undelete command in Unix!



copying files and folders

- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment

```
cp my_file ../somewhere/else
```

```
cp my_file_original my_file_copy
```

```
cp my_file ../somewhere/else/my_file_copy
```

```
cp -r /entireDirectory somewhere/else/to/new_destination
```

note: the flag *r* is needed to copy the directory with all the sub directories

note: there are way more commands and flags we will be learning soon :)



You can install ANACONDA Navigator for Windows
[here](#)

- 1) useful commands in Linux
- 2) **installing ANACONDA**
- 3) setting up your environment

The screenshot shows the Anaconda Navigator application window. On the left, there's a sidebar with options like 'Download Now' (highlighted in green), 'Anaconda Navigator', 'Windows', 'Python 3.12', and links to 'Documentation' and 'Anaconda Blog'. The main area is titled 'ANACONDA.NAVIGATOR' and displays a grid of icons for different tools. Each tool has a name, version, and a brief description. Buttons for 'Install', 'Launch', or 'Connect' are provided for each. The tools listed include:

- PyCharm Professional
- Anaconda AI Navigator
- Anaconda Toolbox
- Anaconda Cloud Notebooks
- anaconda_prompt
- CMD.exe Prompt
- console_shortcut_miniconda
- JupyterLab
- Notebook
- Powershell Prompt
- IPyConsole
- Spyder
- watsonx™
- ORACLE Cloud Infrastructure
- Glueviz
- Orange 3
- powershell_shortcut_miniconda
- RStudio

The interface is clean with a light blue background and white cards for each tool.



You can install ANACONDA Navigator for Linux

- 1) we move into the home directory (cd) and run the curl (Client URL) command in order to download the ANACONDA installer from [here](#)

```
curl -O https://repo.anaconda.com/archive/Anaconda3-2024.06-1-Linux-x86_64.sh
```

- 2) running the installer

```
bash ~/Downloads/Anaconda3-2024.06-1-Linux-x86_64.sh
```

- 3) it might give us an error message “permission denied”
→ turning the installer shell script into an “executable” using chmod

```
chmod +x Anaconda3-2024.06-1-Linux-x86_64.sh
```

- 4) run the installer manually

```
./Anaconda3-2024.06-1-Linux-x86_64.sh
```

- 1) useful commands in Linux
- 2) **installing ANACONDA**
- 3) setting up your environment



3) it might give us an error message “permission denied”

```
chmod +x Anaconda3-2024.06-1-Linux-x86_64.sh
```

4) run the installer manually

```
./Anaconda3-2024.06-1-Linux-x86_64.sh
```

5) confirm license terms by typing *yes* and press *enter*

6) you might need to refresh your terminal by running

```
source ~/.bashrc
```

Now Anaconda should be working. Type

```
conda install python
```

finally, run

```
jupyter-notebook &
```

1) useful commands in Linux

2) **installing ANACONDA**

3) setting up your environment

Name	Last Modified	File Size
anaconda3	5 hours ago	
AppAcademy	3 months ago	
Downloads	3 months ago	
snap	3 months ago	
Untitled.ipynb	5 hours ago	616 B
Anaconda3-2023.09-0-Linux-x86_64.sh	12 days ago	1.1 GB
Anaconda3-2024.06-1-Linux-x86_64.sh	6 hours ago	1007.9 MB



Note: conda on unix doesn't work well with Spyder

- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment**

```
(base) mmh_user@DESKTOP-PPSA666:~$ spyder &
[1] 52032
(base) mmh_user@DESKTOP-PPSA666:~$ Could not load the Qt platform plugin "xcb" in "" even though it was found.
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: eglfs, minimal, minimalegl, offscreen, vnc, webgl, xcb.

[1]+ Aborted                  spyder
(base) mmh_user@DESKTOP-PPSA666:~$
```

creating a conda environment for the Spyder IDE:

```
conda create -n spyder-env -c conda-forge python=3.11 spyder
```

activating the environment:

```
conda activate spyder-env
```



Note: conda on unix doesn't work well with Spyder

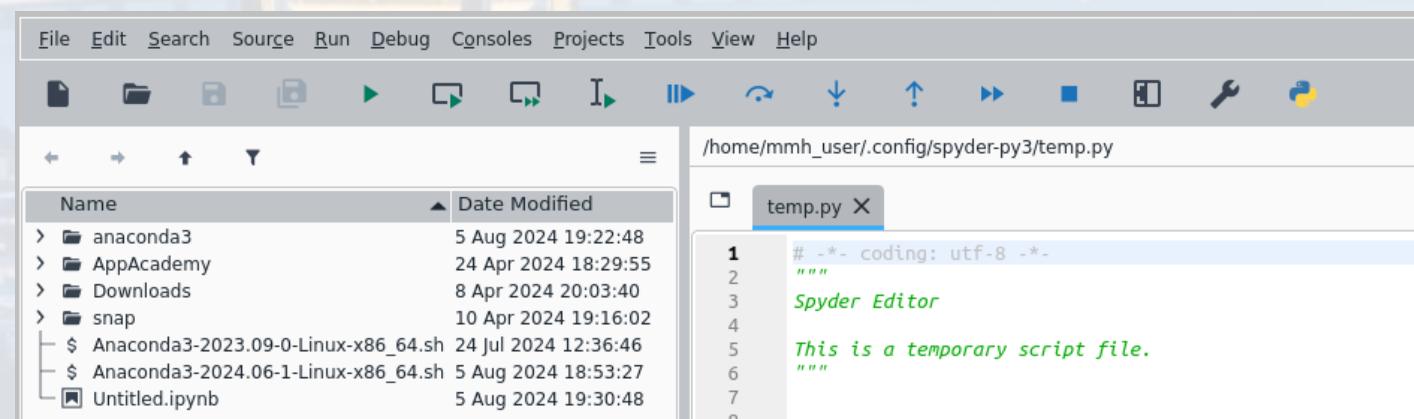
creating a conda environment for the Spyder IDE:

```
conda create -n spyder-env -c conda-forge python=3.11 spyder
```

activating the environment:

```
conda activate spyder-env
```

```
(base) mmh_user@DESKTOP-PPSA666:~$ conda activate spyder-env
(spyder-env) mmh_user@DESKTOP-PPSA666:~$ spyder &
[1] 53245
(spyder-env) mmh_user@DESKTOP-PPSA666:~$ fromIccProfile: failed minimal tag size sanity
```





depending on your project, you can create many different
conda environments

- 1) useful commands in Linux
- 2) installing ANACONDA
- 3) setting up your environment**

```
conda create -n <MyEnv> python=3.10  scipy=0.17.3
```

showing all libraries:

```
conda list
```

showing all environments

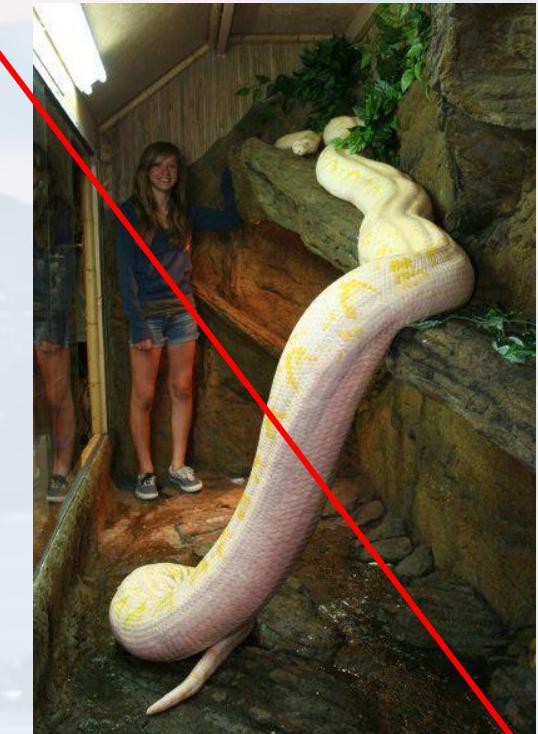
```
conda info --envs
```

```
(base) mmh_user@DESKTOP-PPSA666:~$ conda info --envs
# conda environments:
#
base                  * /home/mmh_user/anaconda3
spyder-env            /home/mmh_user/anaconda3/envs/spyder-env
```



Now it is time for python!

- 1990, von Guido van Rossum
- named after "**Monty Python**",
not the serpent



- idea: flexible, simple and compact syntax, extendable

**TIOBE index July 2024**

Jul 2024	Jul 2023	Change	Programming Language	Ratings	Change
1	1		Python	16.12%	+2.70%
2	3	▲	C++	10.34%	-0.46%
3	2	▼	C	9.48%	-2.08%
4	4		Java	8.59%	-1.91%
5	5		C#	6.72%	-0.15%
6	6		JavaScript	3.79%	+0.68%
7	13	▲	Go	2.19%	+1.12%
8	7	▼	Visual Basic	2.08%	-0.82%
9	11	▲	Fortran	2.05%	+0.80%
10	8	▼	SQL	2.04%	+0.57%
11	15	▲	Delphi/Object Pascal	1.89%	+0.91%
12	10	▼	MATLAB	1.34%	+0.08%
13	17	▲	Rust	1.18%	+0.29%
14	16	▲	Ruby	1.16%	+0.25%
15	12	▼	Scratch	1.15%	+0.08%
16	9	▼	PHP	1.15%	-0.27%



<https://www.anaconda.com/>

The screenshot shows the Anaconda Navigator application window. On the left is a sidebar with icons for Help, Home, Environments, Learning, and Community. The main area displays a grid of application tiles. The tiles include:

- CMD.exe Prompt (0.1.1) - Run a cmd.exe terminal.
- JupyterLab (1.2.6) - An extensible environment for interactive and reproducible computing.
- Notebook (6.0.3) - A web-based, interactive computing notebook environment.
- Powershell Prompt (0.0.1) - Run a Powershell terminal.
- IPyConsole (4.6.0) - A PyQt GUI for inline figures and multiline editing.
- Spyder (4.0.1) - A scientific Python development environment.
- VS Code (1.45.1) - A streamlined code editor.
- Glueviz (0.15.2) - Multidimensional data visualization.
- Orange 3 (3.23.1) - A component-based data mining framework.
- RStudio (1.1.456) - Integrated tools for R development.

Two specific applications are highlighted with red boxes: Jupyter Notebook and Spyder. Both have "Launch" buttons below them.



Spyder

The screenshot shows the Spyder Python IDE interface. The top navigation bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The title bar indicates "Spyder (Python 3.9)".

content of current folder: A callout box points to the left sidebar's file browser, which lists files and folders in the current directory.

py script: yet untitled, we are going to need later: A callout box points to the main code editor window containing a Python script with code for a neural network forward pass and backward pass.

folder navigator: A callout box points to the top right corner of the interface.

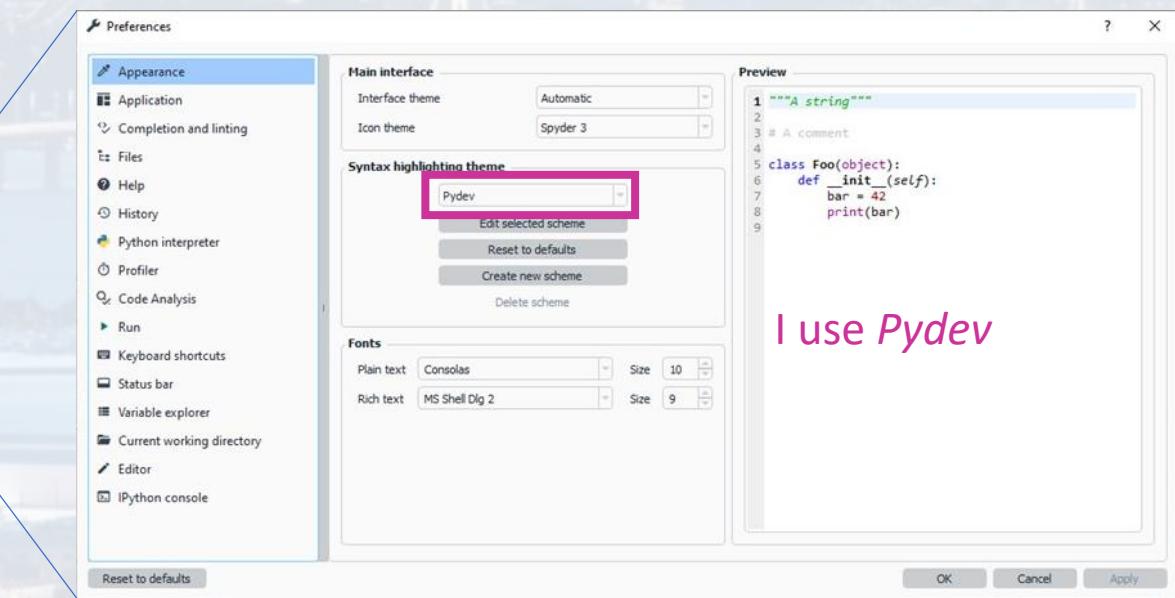
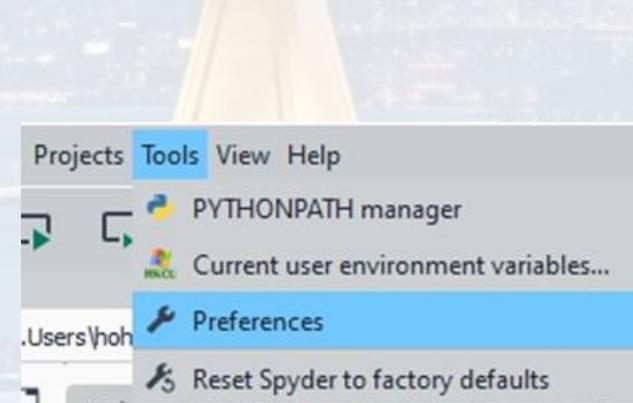
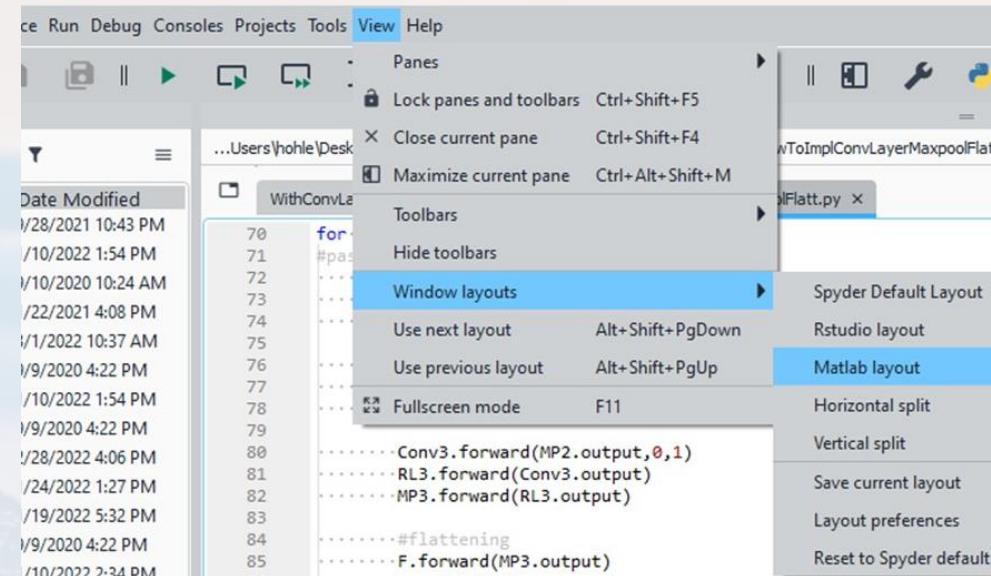
workspace: displays current plots or variables: A callout box points to the right sidebar's "Variable Explorer" tab, which is circled in orange. Below it, the "Plots" tab is also circled in blue.

console: typing commands & executing scripts: A callout box points to the bottom left "Console" tab, which shows the Python interpreter prompt and a help message.

Usage: A callout box points to the bottom right help section, which provides instructions for getting help on objects.



Spyder



settings:

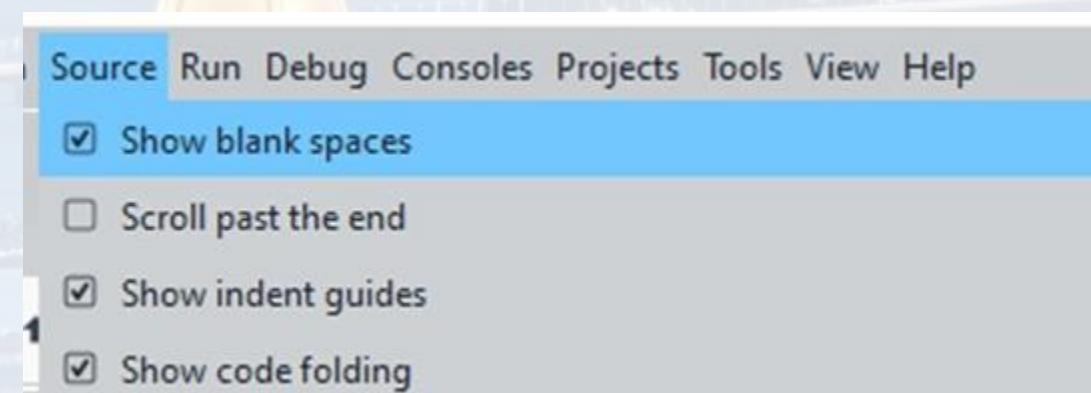
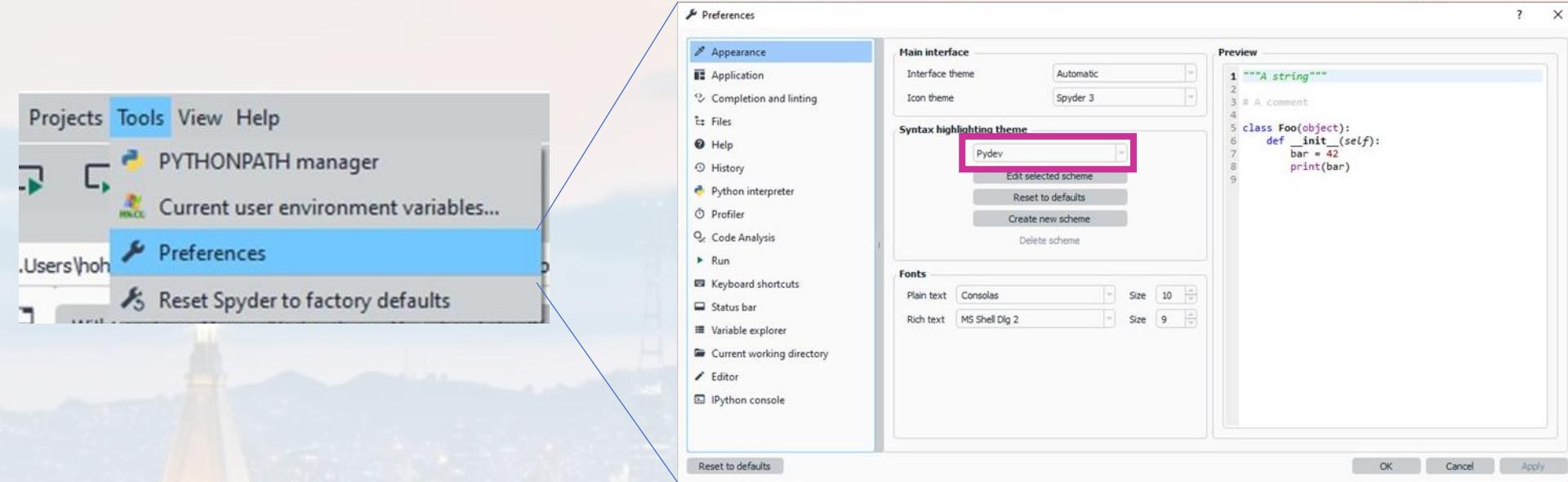
toolbar: View → Window layouts

e. g. Matlab

I use Pydev



Spyder



blanks are relevant for syntax!



Jupyter

The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the page is "localhost:8888/tree". The main area displays a file tree with various folders and files. A context menu is open over a folder named "Python 3 (ipykernel)", which is highlighted with a pink box. The menu options include "Upload", "New", "Folder", and "Terminal".

lofi hip hop radio - beats to play

Home Page - Select or create a

localhost:8888/tree

jupyter

Quit Logout

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

Name: Python 3 (ipykernel)

Upload New

Notebook: Python 3 (ipykernel)

Create a new notebook with Python 3 (ipykernel)

Text File

Folder

Terminal

2 months ago

5 days ago

a year ago

localhost:8888/tree# Searches

Type here to search

-8°C ENG 11:33 04/12/2023



Jupyter

A screenshot of a Jupyter Notebook interface. The title bar shows tabs for "lofi hip hop radio - beats to PLAYING", "Home Page - Select or create a", and "Untitled1 - Jupyter Notebook". The address bar indicates the notebook is running on "localhost:8888/notebooks/Untitled1.ipynb?kernel_name=python3". The main window title is "jupyter Untitled1 Last Checkpoint: a minute ago (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like save, new, and delete, and code execution buttons. A code cell labeled "In []:" contains the text "print('test')".

A screenshot of a Jupyter Notebook interface. The title bar shows "jupyter Untitled1 Last Checkpoint: 7 minutes ago (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like save, new, and delete, and code execution buttons. A code cell labeled "In []:" contains the text "print('test')".

A screenshot of a Jupyter Notebook interface. The title bar shows "jupyter Untitled1 Last Checkpoint: 9 minutes ago (autosaved)". The menu bar includes File, Edit, View, Insert (which is highlighted with a pink box), Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like save, new, and delete, and code execution buttons. A code cell labeled "In []:" contains the text "print('test')". A tooltip for the "Insert" menu option is visible, stating "Insert an empty Code cell above the currently active cell".



Jupyter

The screenshot shows the Jupyter Notebook interface. A blue box highlights the top navigation bar. The 'Cell' menu item is selected and highlighted with a pink box. A dropdown menu is open under 'Cell Type'. The 'Markdown' option is also highlighted with a pink box. A tooltip for 'Markdown' states: 'Contents will be rendered as HTML and serve as explanatory text'. The main workspace shows an input cell with 'In []:' and a code cell with 'In []: print("test")'.

type something → click **run**

The screenshot shows the Jupyter Notebook interface with a green box highlighting the output area. It displays the word 'Hi' followed by the output of the code cell 'print("test")', which is 'test'. Below the output is an empty input cell 'In []:'.

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



Jupyter

The screenshot shows the Jupyter Notebook interface with the 'File' menu open. The 'File' menu is highlighted with a pink box, and the 'Download as' option is also highlighted with a pink box. A dropdown menu lists various formats for saving the notebook, including AsciiDoc (.asciidoc), HTML (.html), LaTeX (.tex), Markdown (.md), Notebook (.ipynb), PDF via LaTeX (.pdf), reST (.rst), Python (.py), Reveal.js slides (.slides.html), and PDF via HTML (.html). The background of the slide features a blurred image of the San Francisco skyline.

- New Notebook
- Open...
- Make a Copy...
- Save as...
- Rename...
- Save and Checkpoint Ctrl-S
- Revert to Checkpoint
- Print Preview
- Download as
- Trusted Notebook
- Close and Halt

- AsciiDoc (.asciidoc)
- HTML (.html)
- LaTeX (.tex)
- Markdown (.md)
- Notebook (.ipynb)
- PDF via LaTeX (.pdf)
- reST (.rst)
- Python (.py)
- Reveal.js slides (.slides.html)
- PDF via HTML (.html)



basic types:

```
String  = 'Hello ' + "'World'"  
  
List    = [1, 2, 3, 5, "'World'" ]      #default  
  
Tuple   = (1, 2)  
  
Dict    = { 'A': 1, 'B': 2}  
  
Array   = np.array([1, 2, 3, 5])  
  
pd.DataFrame
```

other objects:

```
class  
  
def     #function/method
```



- labels and titles of plots
- paths and file names
- error messages

```
String    = 'Hello ' + 'World'  
List     = [1, 2, 3, 5, 'World' ]  
Tuple    = (1, 2)  
Dict     = { 'A': 1, 'B': 2}  
Array    = np.array([1, 2, 3, 5])  
pd.DataFrame
```

```
string1 = 'Hello Students'
```

```
string2 = ', how are you'
```

```
string12 = string1 + string2
```

```
Out[1]: 'Hello Students, how are you'
```

concatenating is incredibly easy!

```
S = 'abc'
```

```
3*S
```

```
Out[2]: 'abcabcabc'
```

```
string12[2:6]
```

slicing

```
[ | 1, | 5, | 0, | -3 ]
```

slices: 0 1 2 3 4



- labels and titles of plots
- paths and file names
- error messages

string12[2:6]

String = 'Hello ' + 'World'
List = [1, 2, 3, 5, 'World']
Tuple = (1, 2)
Dict = { 'A': 1, 'B': 2}
Array = np.array([1, 2, 3, 5])
pd.DataFrame

slicing

[| 1, | 5, | 0, | -3 |]
slices: 0 1 2 3 4

[1, 5, 0, -3]
index: 0 1 2 3

indexing

index: -4 -3 -2 -1

string12[-1]

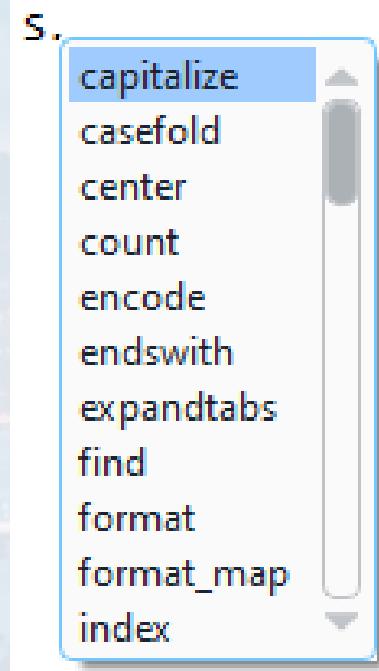
string12[1:]

string12[:-1]



- labels and titles of plots
- paths and file names
- error messages

```
String      = 'Hello ' + 'World''  
List       = [1, 2, 3, 5, 'World' ]  
Tuple      = (1, 2)  
Dict       = { 'A': 1, 'B': 2}  
Array      = np.array([1, 2, 3, 5])  
pd.DataFrame
```



try some of the functions like

```
S.count()  
S.find()
```



default type in python

```
L = [1, 2, 3, -2]
```

```
2*L
```

```
Out[3]: [1, 2, 3, -2, 1, 2, 3, -2]
```

```
type(L)
```

```
Out[4]: list
```

```
String      = 'Hello ' + 'World''  
List        = [1, 2, 3, 5, 'World']  
Tuple       = (1, 2)  
Dict        = {'A': 1, 'B': 2}  
Array       = np.array([1, 2, 3, 5])  
pd.DataFrame
```



default type in python

be careful:

```
L1 = [1, 2]
```

```
L2 = L1
```

```
L1[0] = 5
```

```
print(L2[0])
```

```
L1 = [1, 2]
```

```
L2 = L1.copy()
```

```
L1[0] = 5
```

```
print(L2[0])
```

```
String   = 'Hello ' + 'World''  
List    = [1, 2, 3, 5, 'World']  
Tuple   = (1, 2)  
Dict    = {'A': 1, 'B': 2}  
Array   = np.array([1, 2, 3, 5])  
pd.DataFrame
```

What is the result? What happens if you do the same with an **int**?

lists, dictionaries, sets and bytearrays are **mutable**.



```
L1 = [1, 2, 4]
```

```
L2 = ['a', 'b', 'c']
```

```
T = (L1, L2)
```

```
len(T)
```

```
Out[5]: 2
```

```
T[1]
```

```
Out[6]: ['a', 'b', 'c']
```

```
type(T)
```

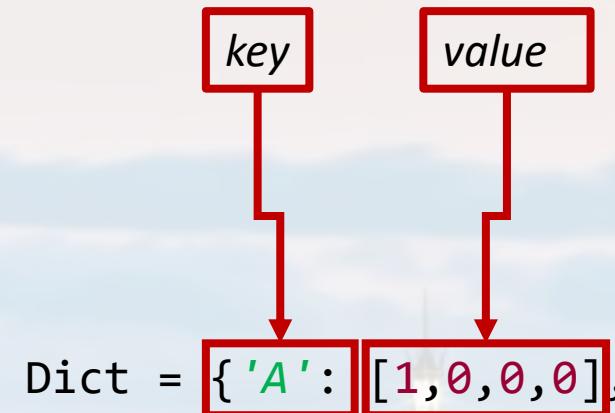
```
Out[7]: tuple
```

```
(out1, out2) = T
```

```
out1
```

```
Out[8]: [1, 2, 4]
```

```
String      = 'Hello ' + "'World'"  
List        = [1, 2, 3, 5, "'World'" ]  
Tuple       = (1, 2)  
Dict        = {'A': 1, 'B': 2}  
Array       = np.array([1, 2, 3, 5])  
pd.DataFrame
```



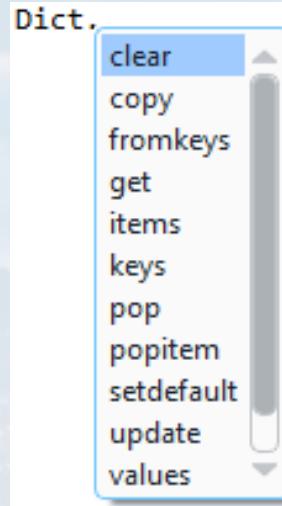
```
Dict['A']
Out[9]: [1, 0, 0, 0]
```

```
type(Dict)
Out[21]: dict
```

```
String    = 'Hello ' + 'World'
List     = [1, 2, 3, 5, 'World' ]
Tuple    = (1, 2)
Dict     = { 'A': 1, 'B': 2}
Array    = np.array([1, 2, 3, 5])
pd.DataFrame
```



check out



```
String      = 'Hello ' + ''World''  
List       = [1, 2, 3, 5, ''World'' ]  
Tuple      = (1, 2)  
Dict       = { 'A': 1, 'B': 2}  
Array      = np.array([1, 2, 3, 5])  
pd.DataFrame
```

try some of the functions like

```
Dict.values()  
Dict.pop('A')  
Dict.update({ 'U': [2, 0, 0, 0]})
```

syntax:

[] → **arrays** (lists, np.array, data frames → see next week)

() → functions or **tuple**

T = (a, v)

(a_new, v_new) = T

{ } → **Dictionaries**

Dict = { 'A': [1,0,0,0], 'C': [0,1,0,0], 'G': [0,0,1,0], 'T': [0,0,0,1] }

Dict['A']

Out[8]: [1, 0, 0, 0]

```
String  = 'Hello ' + 'World''  
List   = [1, 2, 3, 5, 'World' ]  
Table  = (1, 2)  
Dict   = { 'A': 1, 'B': 2}  
Array  = np.array([1, 2, 3, 5])  
pd.DataFrame
```



syntax:

Dict.

```
clear  
copy  
fromkeys  
get  
items  
keys  
pop  
popitem  
setdefault  
update  
values
```

type:

dir(dict)

```
'__class_getitem__',  
'__contains__',  
'__delattr__',  
'__delitem__',  
'__dir__',  
'__doc__',  
'__eq__',  
'__format__',  
'__ge__',  
'__getattribute__',  
'__getitem__',  
'__getstate__',  
'__gt__',  
'__hash__',  
'__init__',  
'__init_subclass__',  
'__ior__',  
'__iter__',  
'__le__',  
'__len__',  
'__lt__',  
'__ne__',  
'__new__',  
'__or__',  
'__reduce__',  
'__reduce_ex__',  
'__repr__',  
'__reversed__',  
'__ror__',  
'__setattr__',  
'__setitem__',  
'__sizeof__',  
'__str__',  
'__subclasshook__',  
'clear',  
'copy',  
'fromkeys',  
'get',  
'items',  
'keys',  
'pop',  
'popitem',  
'setdefault',  
'update',  
'values']
```

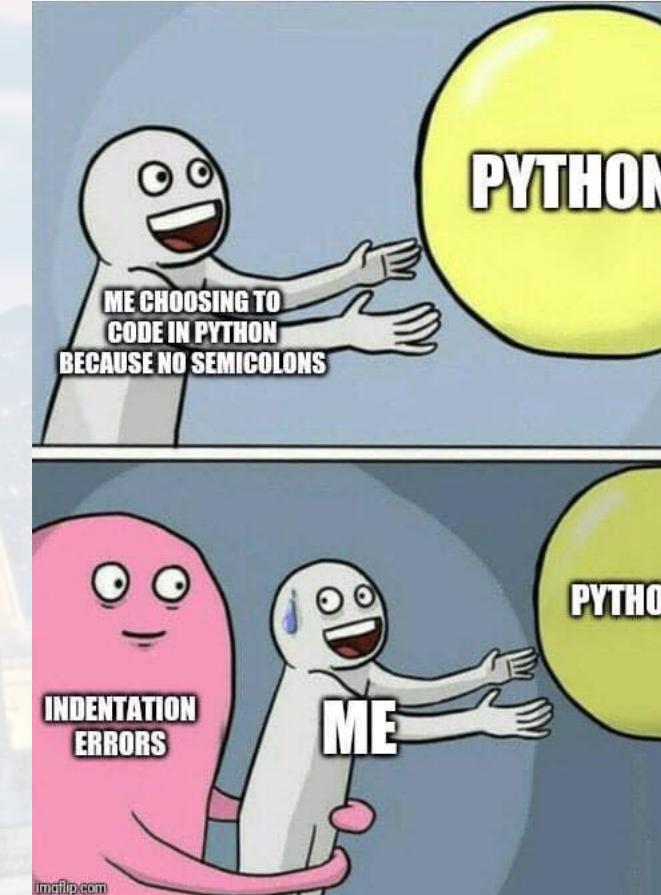
```
String = 'Hello ' + 'World'  
List = [1, 2, 3, 5, 'World']  
Table = (1, 2)  
Dict = {'A': 1, 'B': 2}  
Array = np.array([1, 2, 3, 5])  
pd.DataFrame
```

attributes

functions aka methods



Introduction to Unix & Python



Thank you for your attention!