Python tutorial

Tips and tricks you may already know…

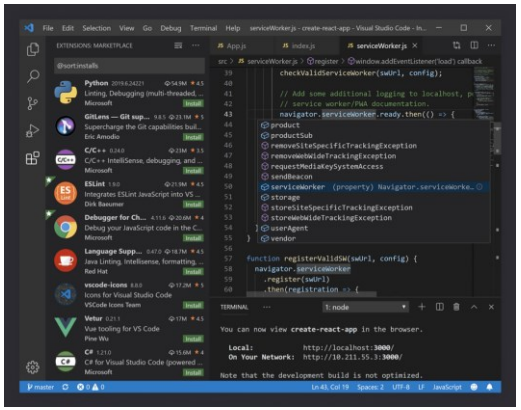
Prof. Maurício Girardi-Schappo (UFSC, Brazil)
girardi.s@gmail.com

Summary

- Coding:
  What is it really about?

- In the beginning…
  There was an IDE

- Python:
  must know



$$A \to B$$
$$C \to D$$
$$E \to F$$

Disclaimer

- This lecture **is not about**:
  - Coding from the scratch
  - Advanced coding/python
- This lecture **is about**:
  - Giving a hands-on intro…

Coding: what is it about?

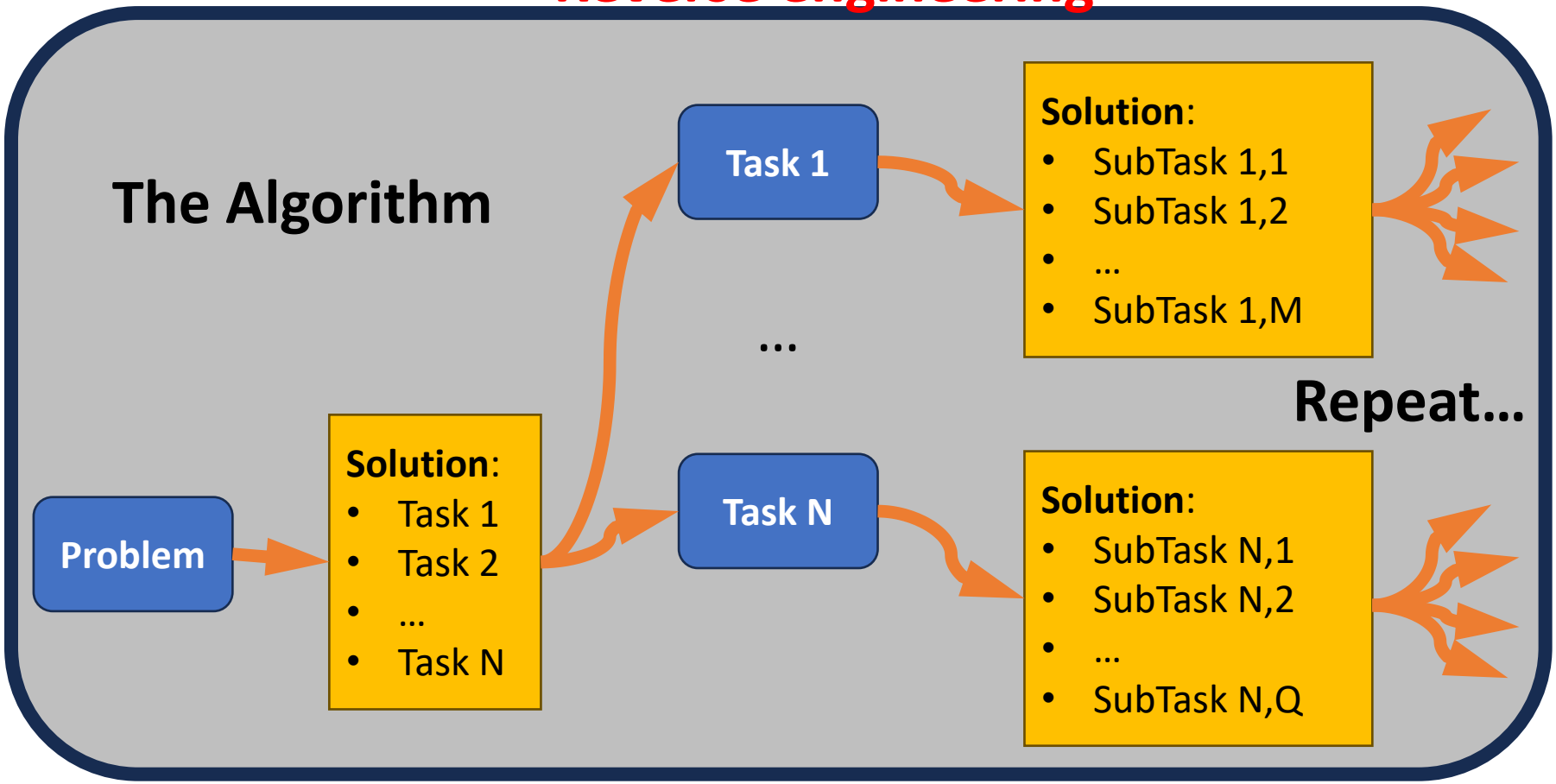
- **Solving problems** you didn't know you had!

Coding: what is it about?

- If you **can make a sandwich**, you **can code**!

a.k.a.

"Reverse engineering"



Coding: what is it about?

- If you **can make a sandwich**, you **can code**!

```
from kitchen import SandwichMaker,fridge

sandwich = SandwichMaker()

if sandwich.is_vegan():
    stuffing = fridge.get_available_vegs()
else:
    stuffing = fridge.get_everything()

sandwich.open() # we need to open the bread!
for item in stuffing:
    sandwich.add(item) # add stuff
sandwich.close() # and then close it!

grill(sandwich,temperature=150*Celsius)
eat(sandwich)
```

Coding: elements…

- **Statements**:

Tells the “interpreter” what to do

E.g.:

- Create variables
- Check conditions
- Iterate over array
- Import stuff
- …

Each line here!

```
from kitchen import SandwichMaker,fridge

sandwich = SandwichMaker()

if sandwich.is_vegan():
    stuffing = fridge.get_available_vegs()
else:
    stuffing = fridge.get_everything()

sandwich.open() # we need to open the bread!
for item in stuffing:
    sandwich.add(item) # add stuff
sandwich.close() # and then close it!

grill(sandwich,temperature=150*Celsius)
eat(sandwich)
```

Coding: elements…

- **Functions**:

Transform some values into other values

```
from kitchen import SandwichMaker,fridge

sandwich = SandwichMaker()

if sandwich.is_vegan():
    stuffing = fridge.get_available_vegs()
else:
    stuffing = fridge.get_everything()

sandwich.open() # we need to open the bread!
for item in stuffing:
    sandwich.add(item) # add stuff
sandwich.close() # and then close it!

grill(sandwich,temperature=150*Celsius)
eat(sandwich)
```

Coding: elements…

- **Statements**:

Tells the “interpreter” what to do

E.g.:

- Create variables
- Check conditions
- Iterate over array
- Import stuff
- …

Each line here!

```
from kitchen import SandwichMaker,fridge

sandwich = SandwichMaker()

if sandwich.is_vegan():
    stuffing = fridge.get_available_vegs()
else:
    stuffing = fridge.get_everything()

sandwich.open() # we need to open the bread!
for item in stuffing:
    sandwich.add(item) # add stuff
sandwich.close() # and then close it!

grill(sandwich,temperature=150*Celsius)
eat(sandwich)
```

Coding: elements…

- **Variables**:

names that hold values… (i.e., “boxes”)

They are assigned with the “=“ operator

- int
- float
- str
- numpy.ndarray
- SandwichMaker

- **Types**:

Define what a variable can hold

- **Objects**:

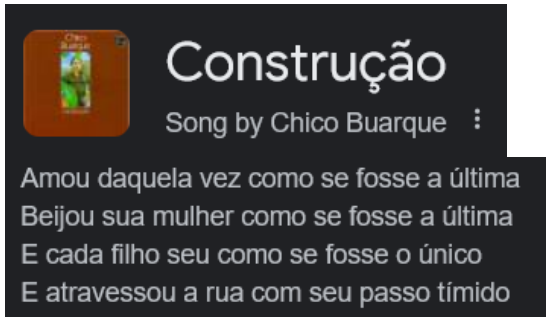
Special variables that hold *abstract* **Types**

- **Classes**:

Statements that define *abstract* **Types**

```
from kitchen import SandwichMaker,fridge

sandwich = SandwichMaker()

if sandwich.is_vegan():
    stuffing = fridge.get_available_vegs()
else:
    stuffing = fridge.get_everything()

sandwich.open() # we need to open the bread!
for item in stuffing:
    sandwich.add(item) # add stuff
sandwich.close() # and then close it!

grill(sandwich,temperature=150*Celsius)
eat(sandwich)
```

Coding: elements…



- Remember:

A **programming language** is a

Formal Language not Poetry

i.e. no ambiguity; only “one way” of writing it; writing rules are strict (syntax is “absolute”)

```
1  "111" + 10
```

```
TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 "111" + 10

TypeError: can only concatenate str (not "int") to str
```

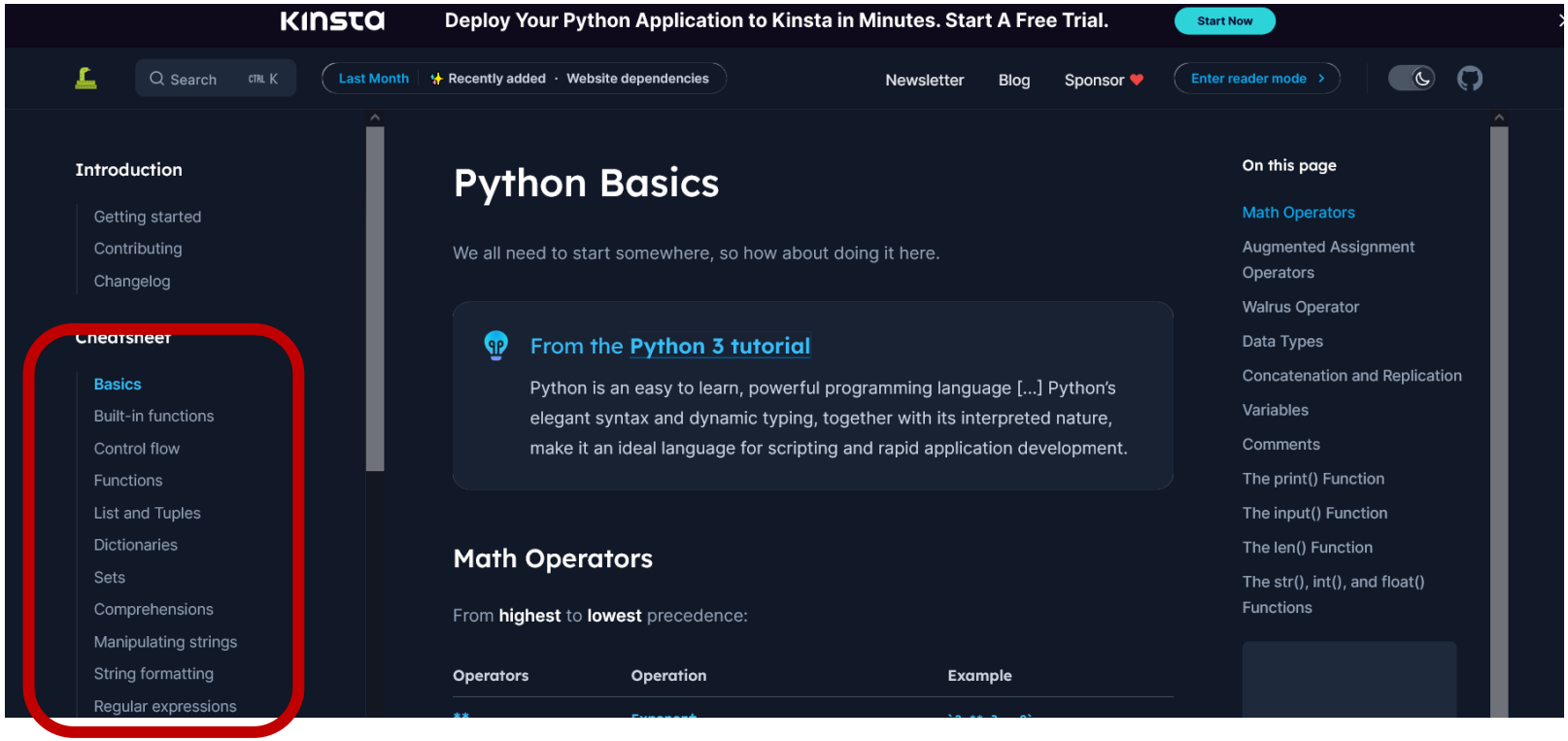
- **Errors will happen**

Don’t avoid them, **read** and **understand**!

- **ChatGPT/Google/Stackoverflow** can help. **Use with caution**. Use clever queries with proper “keywords”

Coding: elements…

- Python cheat sheet: https://www.pythoncheatsheet.org/cheatsheet/basics



Coding: elements…

- Python cheat sheet: https://www.pythoncheatsheet.org/cheatsheet/basics

Math Operators

From **highest** to **lowest** precedence:

Operators	Operation	Example
`**`	Exponent	`2 ** 3 = 8`
`%`	Modulus/Remainder	`22 % 8 = 6`
`//`	Integer division	`22 // 8 = 2`
`/`	Division	`22 / 8 = 2.75`
`*`	Multiplication	`3 * 3 = 9`
`-`	Subtraction	`5 - 2 = 3`
`+`	Addition	`2 + 2 = 4`

Coding: elements…

- Python cheat sheet: https://www.pythoncheatsheet.org/cheatsheet/built-in-functions

etc

Coding: elements…

- Python cheat sheet: https://www.pythoncheatsheet.org/cheatsheet/control-flow

Comparison Operators

Operator	Meaning
`==`	Equal to
`!=`	Not equal to
`<`	Less than
`>`	Greater Than
`<=`	Less than or Equal to
`>=`	Greater than or Equal to

Boolean Operators

There are three Boolean operators: `and`, `or`, and `not`. In the order of precedence, highest to lowest they are `not`, `and` and `or`.

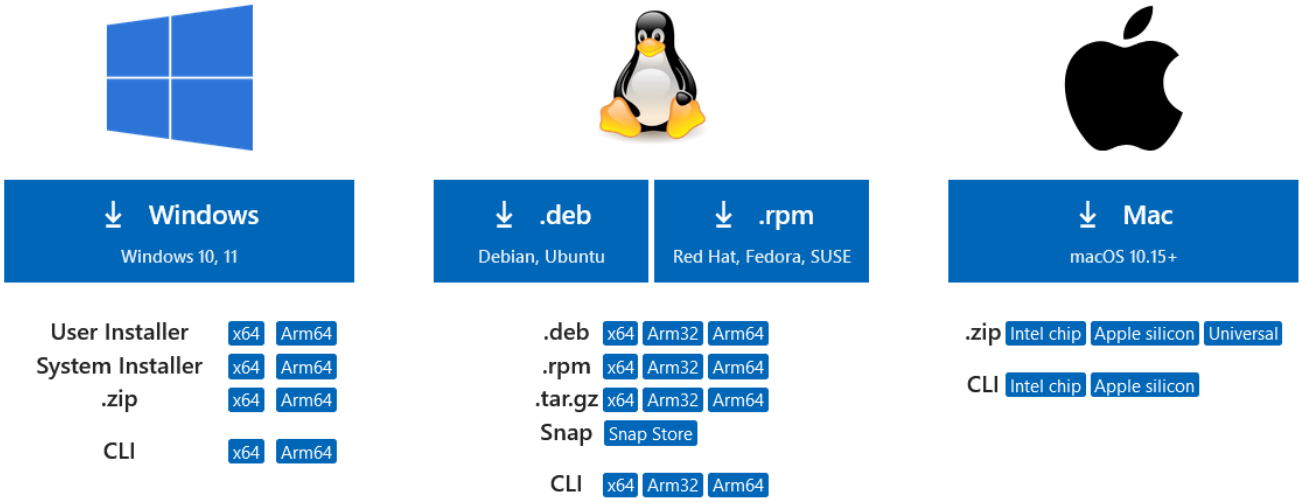
etc

Coding: elements…

- Python cheat sheet:
  - https://www.pythoncheatsheet.org/cheatsheet/functions
  - https://www.pythoncheatsheet.org/cheatsheet/lists-and-tuples
  - https://www.pythoncheatsheet.org/cheatsheet/dictionaries
  - https://www.pythoncheatsheet.org/cheatsheet/sets
  - https://www.pythoncheatsheet.org/cheatsheet/comprehensions
  - https://www.pythoncheatsheet.org/cheatsheet/string-formatting

In the beginning… There was an IDE

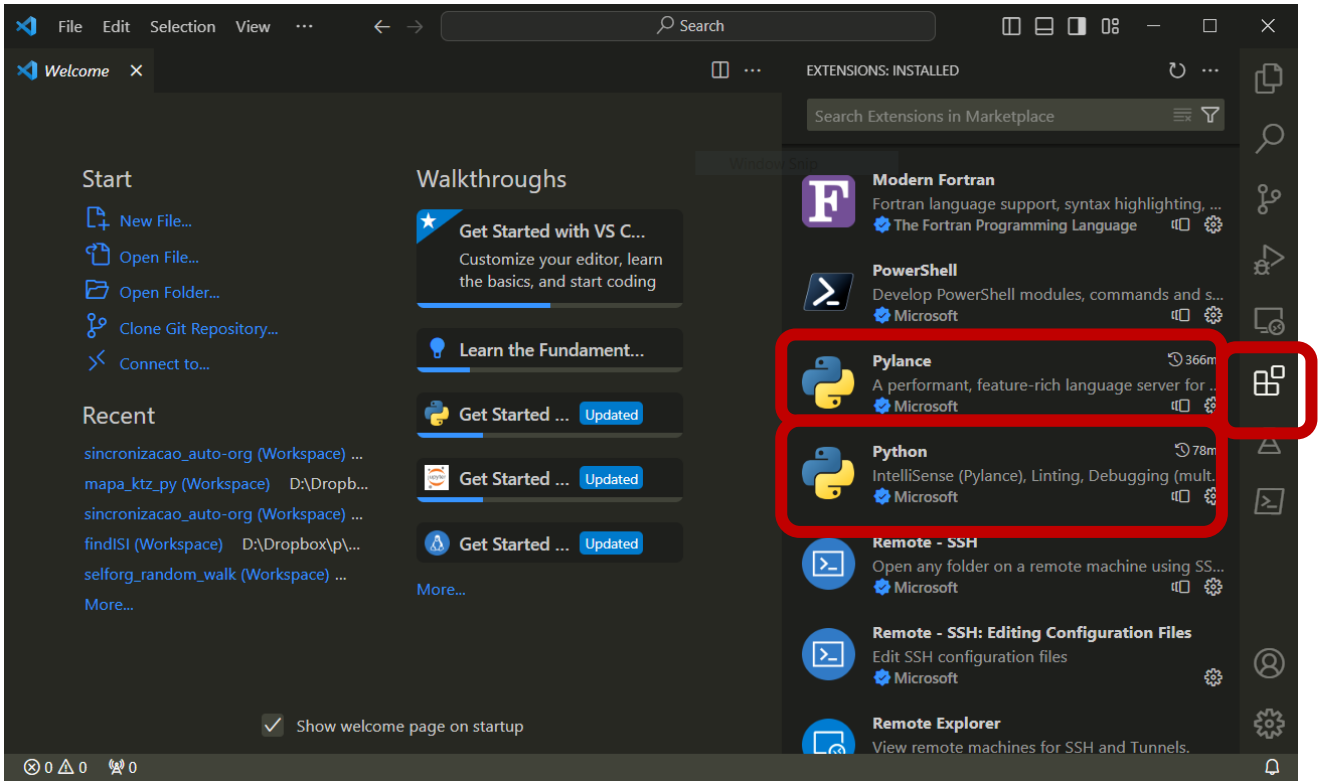
- VS Code
  https://code.visualstudio.com/#alt-downloads



Lightweight, Intellisense (suggestions/corrections of code snippets), syntax highlighting, Jupyter integrated support (view variable workspace on the fly)

In the beginning… There was an IDE

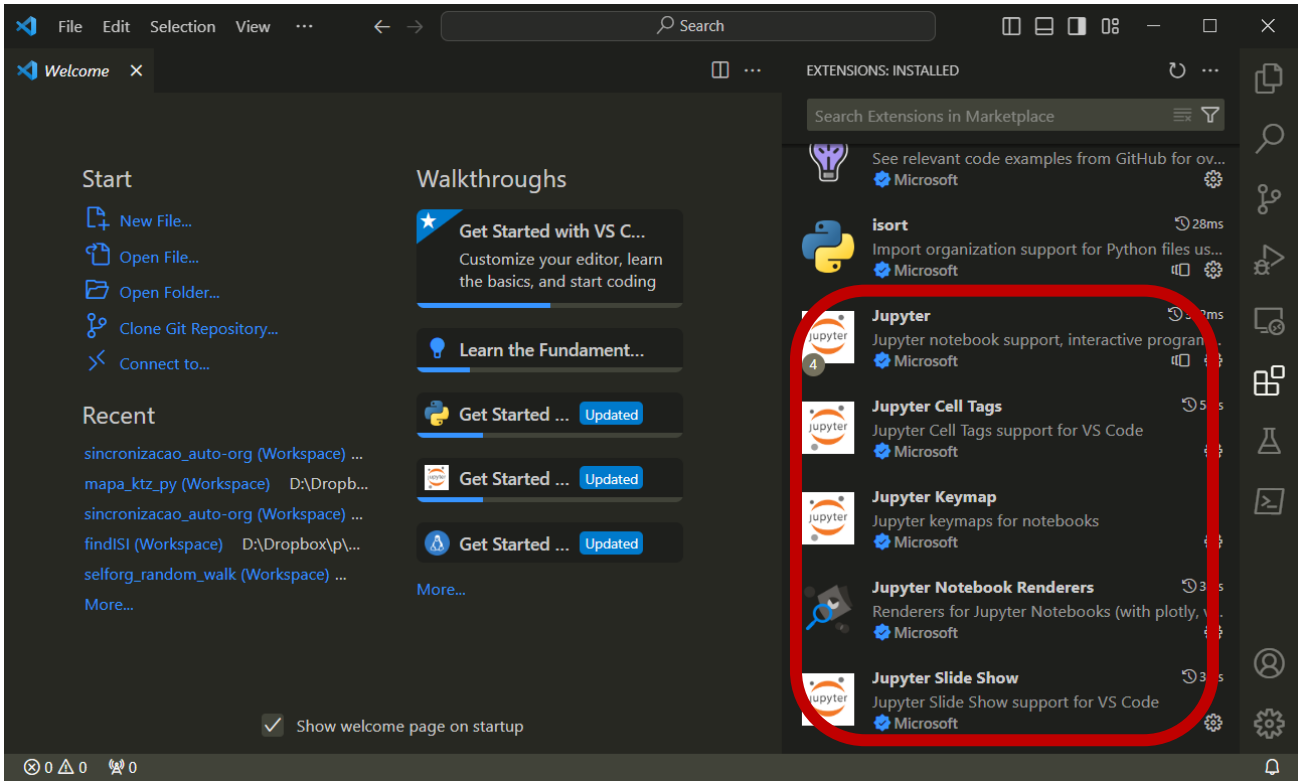
- Python in VS Code: https://code.visualstudio.com/docs/languages/python



- Python
- Pylance

In the beginning… There was an IDE

- Python in VS Code: https://code.visualstudio.com/docs/languages/python

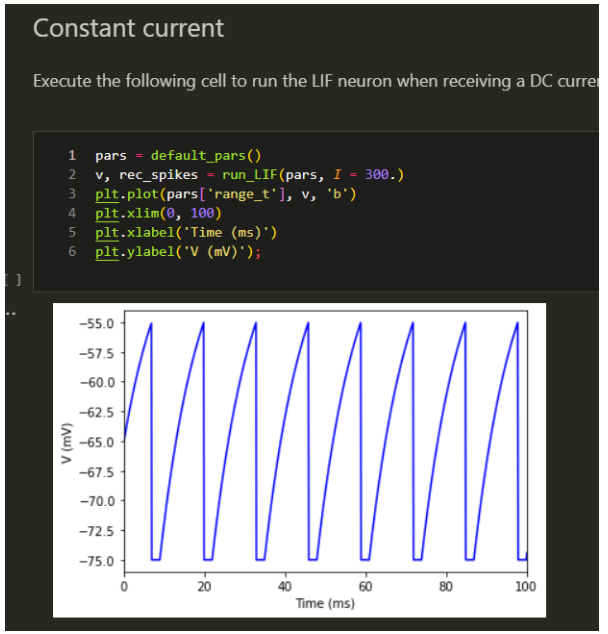


- Python
- Pylance
- Jupyter
- Other Jupyter related

Python: must know

- Python is not a “go-to” language for performance!
- Python is optimized for user-friendliness

- **Jupyter -> literally a “notebook”**
  - Code executed by parts
  - You can mix code with annotations
  - Direct visual I/O feedback of your code
  - Good for plotting
  - Good for probing unknown problems
  - Usually worse performance
  - Runs in the browser or in VS Code



- **Scripting -> more like an “old school program”**
  - Write all the code -> run all at once
  - Another script needed to analyze the data
  - Better when you need performance
  - Run from the terminal: python myscript.py
  - Useful tools for performance gain:
    numba, **pythran**, **cython**, **f2py**

```
import quantities
import modules.output as output
import modules.neurons as neu
import modules.plot as p
import modules.input as inp
import modules.misc as misc

parser = argparse.ArgumentParser(description='Run c
parser = inp.add_neuron_parameters(parser)
#parser = inp.add_neuronparamset_parameters(parser)
#parser = inp.add_stimulus_parameters(parser,tStim=
parser = inp.add_stimulus_parameters(parser,tStim=[
#parser = inp.add_stimulus_parameters(parser,tStim=
parser = inp.add_output_parameters(parser,out=['cur
parser = inp.add_simulation_parameters(parser,T=[50
parser.add_argument('-show', required=False, action
parser.add_argument('-savefig', required=False, act
parser.add_argument('-savevoltage', required=False,

args = parser.parse_args()
saveData = args.save
saveVoltage = args.savevoltage
saveFig = args.savefig
```

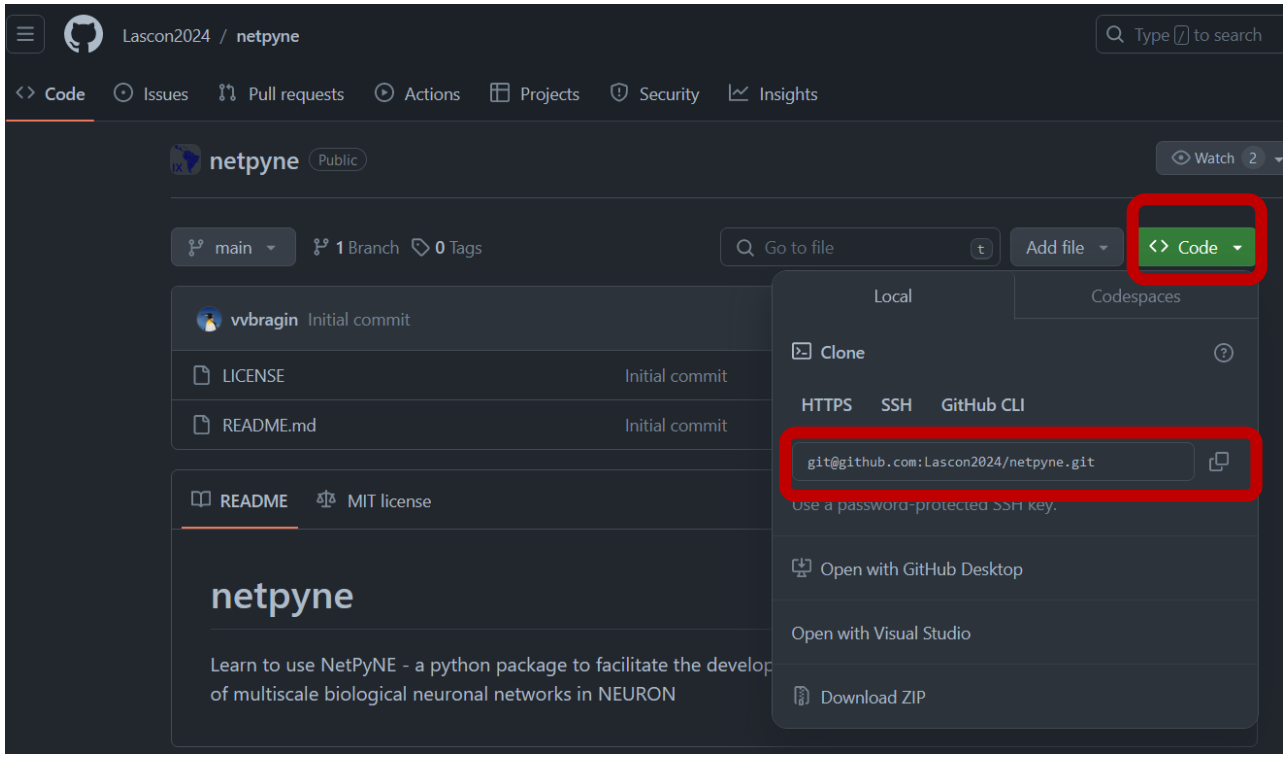
Python: must know

- git https://github.com/Lascon2024
  this.python_tutorial -> https://github.com/Lascon2024/python-tutorial

mkdir lascon2024-python-tutorial

cd lascon2024-python-tutorial

git clone git@github.com:Lascon2024/python-tutorial.git .



Python: must know

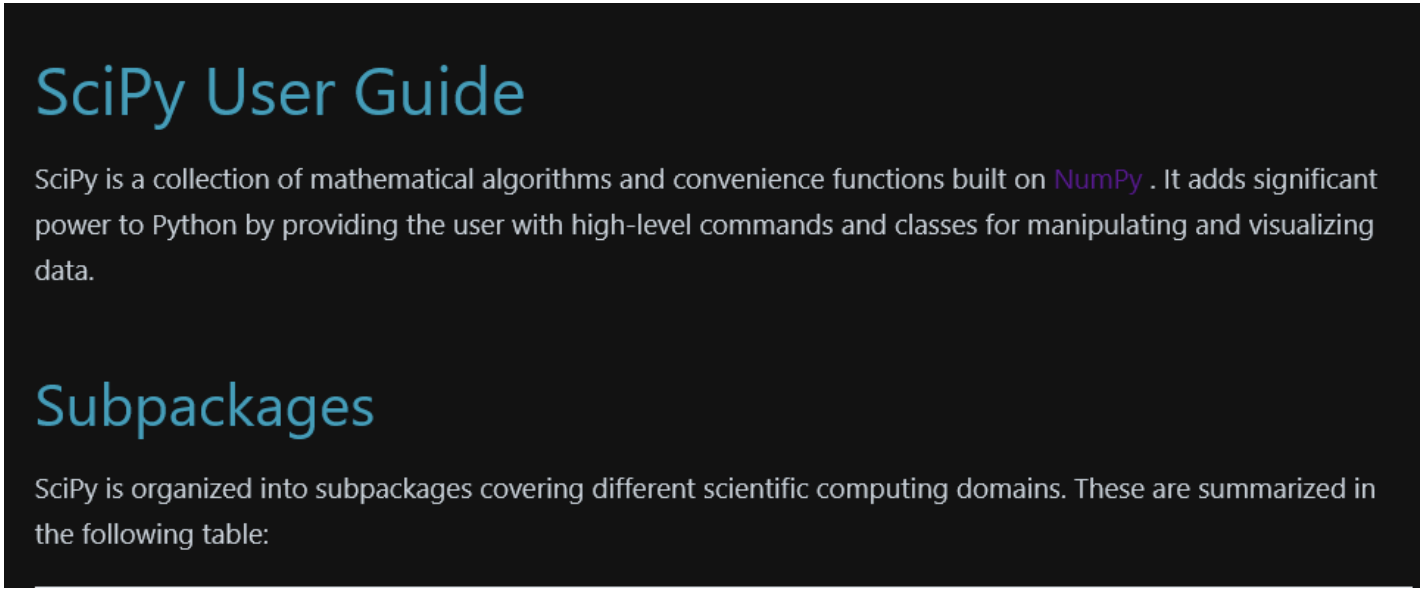
- **numpy**: MATLAB-like vector/matrix manipulation: **vectorized operations**
  https://numpy.org/doc/stable/user/quickstart.html

```
>>> a = np.array(1, 2, 3, 4)    # WRONG
Traceback (most recent call last):
  ...
TypeError: array() takes from 1 to 2 positional arguments but 4 were given
>>> a = np.array([1, 2, 3, 4])  # RIGHT
```

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int64'
>>> a.itemsize
8
>>> a.size
15
>>> type(a)
<class 'numpy.ndarray'>
>>> b = np.array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
<class 'numpy.ndarray'>
```

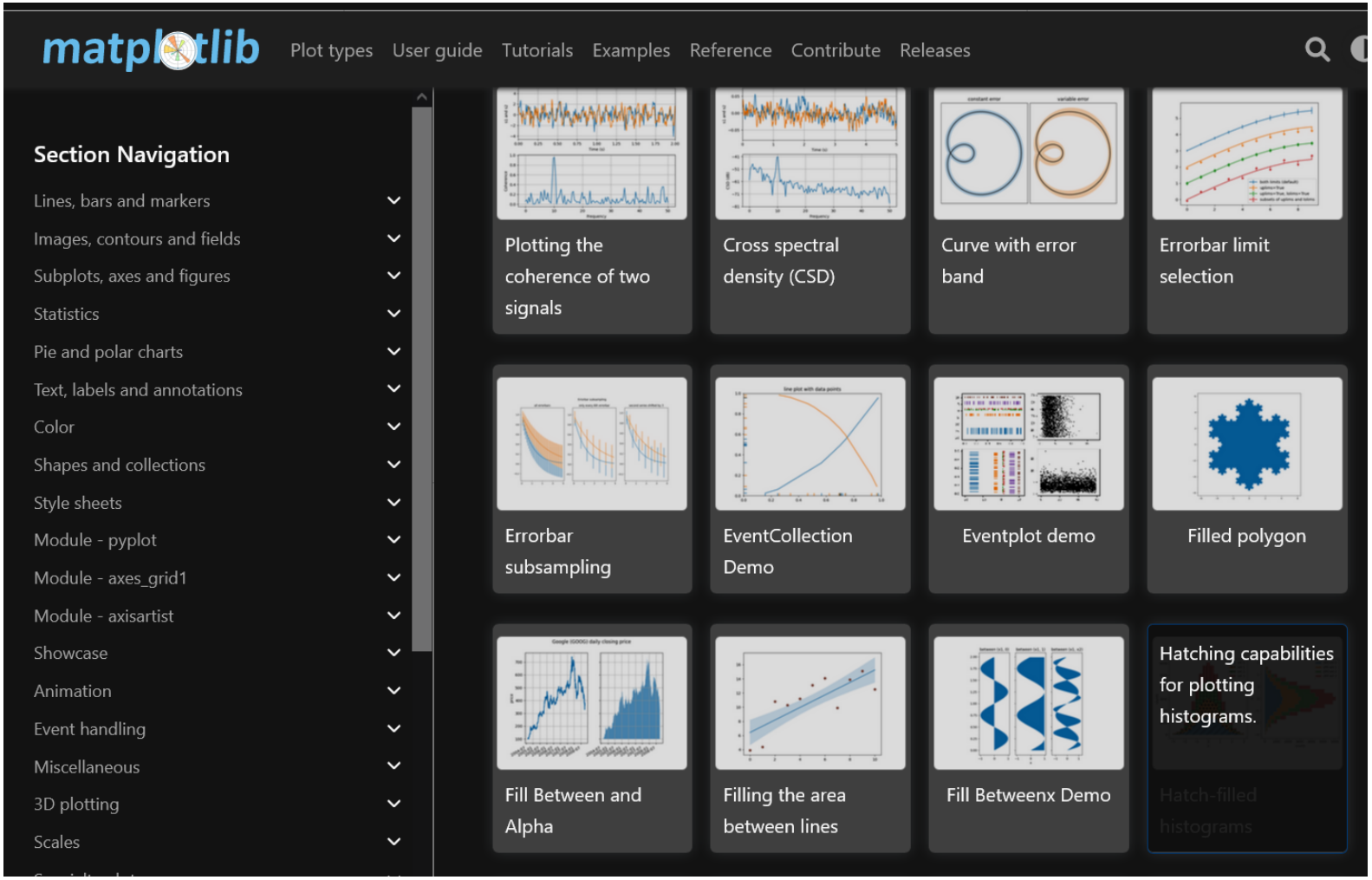
Python: must know

- **scipy**: Scientific/math codes (based on numpy) https://docs.scipy.org/doc/scipy/tutorial/index.html
  - Interpolation (fill missing data gaps): **scipy.interpolate**
  - Statistics: **scipy.stats** (T-test, chi-squared, etc)
  - Signal processing: **scipy.signal** (Filters, find peaks, spectra)
  - Optimization: **scipy.optimize** (curve fitting, parameter estimation)
  - Integration: **scipy.integrate** (Simpson, trapezoid, etc)
  - Sparse matrices: **scipy.sparse** (large data arrays with mostly zeros)
  - I/O: **scipy.io** (savemat, loadmat -> integration with MATLAB; easy data storage – better than numpy's, and more widely compatible)



Python: must know

- **Matplotlib**: MATLAB-like figure plotting capabilities
  https://matplotlib.org/stable/gallery/index.html



Python: must know

- **pandas**: data science toolkit…
  CSV and Excel file I/O (main features to me :P)
  https://pandas.pydata.org/docs/user_guide/10min.html

pandas.read_excel

```
pandas.read_excel(io, sheet_name=0, *, header=0, names=None, index_col=None,
usecols=None, dtype=None, engine=None, converters=None, true_values=None,
false_values=None, skiprows=None, nrows=None, na_values=None,
keep_default_na=True, na_filter=True, verbose=False, parse_dates=False,
date_parser=_NoDefault.no_default, date_format=None, thousands=None, decimal='.',
comment=None, skipfooter=0, storage_options=None,
dtype_backend=_NoDefault.no_default, engine_kwargs=None)                [source]
```

Further learning…

- Our git repo:
  https://github.com/Lascon2024/python-tutorial

- Scientific python lectures (Scipy, numpy examples)
  https://github.com/jrjohansson/scientific-python-lectures/tree/master

- Beginner concepts:
  The way of the program:
  https://greenteapress.com/thinkpython2/html/thinkpython2002.html
  Variables, expressions, statements:
  https://greenteapress.com/thinkpython2/html/thinkpython2003.html

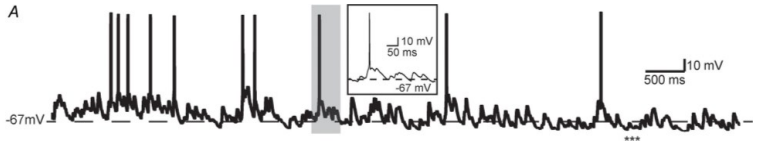
- Intro to python:
  https://cs111.wellesley.edu/lectures/lec_python_intro/

Example…

J Physiol 601.19 (2023) pp 4397–4422

Adaptive spike threshold dynamics associated with sparse spiking of hilar mossy cells are captured by a simple model

Anh-Tuan Trinh[1,2], Mauricio Girardi-Schappo[3,4], Jean-Claude Béïque[2,5,6], André Longtin[4,5,6] and Leonard Maler[2,5,6]

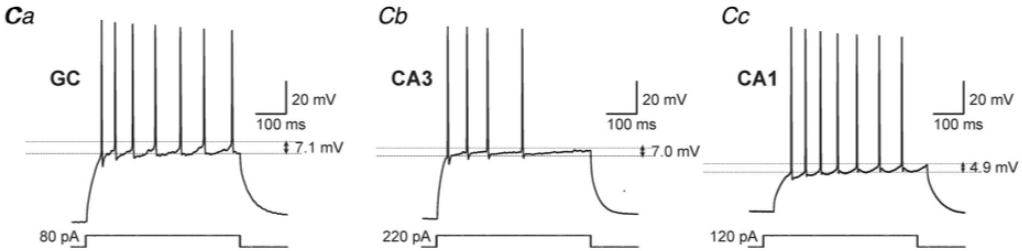


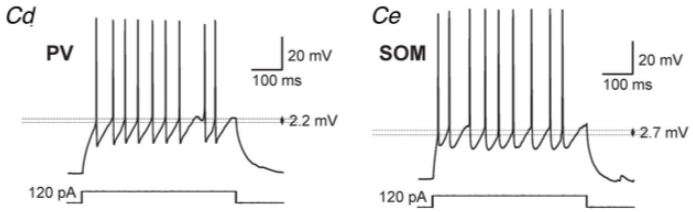
Leaky IF neuron: $\tau \frac{dV}{dt} = V_b - V + RI$

If $V = \theta$

$V(t) = 60\ mV$

$V(t + dt) = V_0$





Simplest possible model (ODE-based) that **captures spiking**

How to go about coding it:

- Determine the problem
  - Integrate an ODE
- Identify parameters
- Identify variables

Example…

J Physiol 601.19 (2023) pp 4397–4422

Adaptive spike threshold dynamics associated with sparse spiking of hilar mossy cells are captured by a simple model

Anh-Tuan Trinh[1,2], Mauricio Girardi-Schappo[3,4], Jean-Claude Béïque[2,5,6], André Longtin[4,5,6] and Leonard Maler[2,5,6]

Quadratic IF neuron:

$$C\frac{dV}{dt} = G(V - V_r)(V - V_c) + I$$

If $V = \theta$

$V(t) = 60\ mV$

$V(t + dt) = V_0$

- Determine the problem
- Identify parameters
- Identify variables