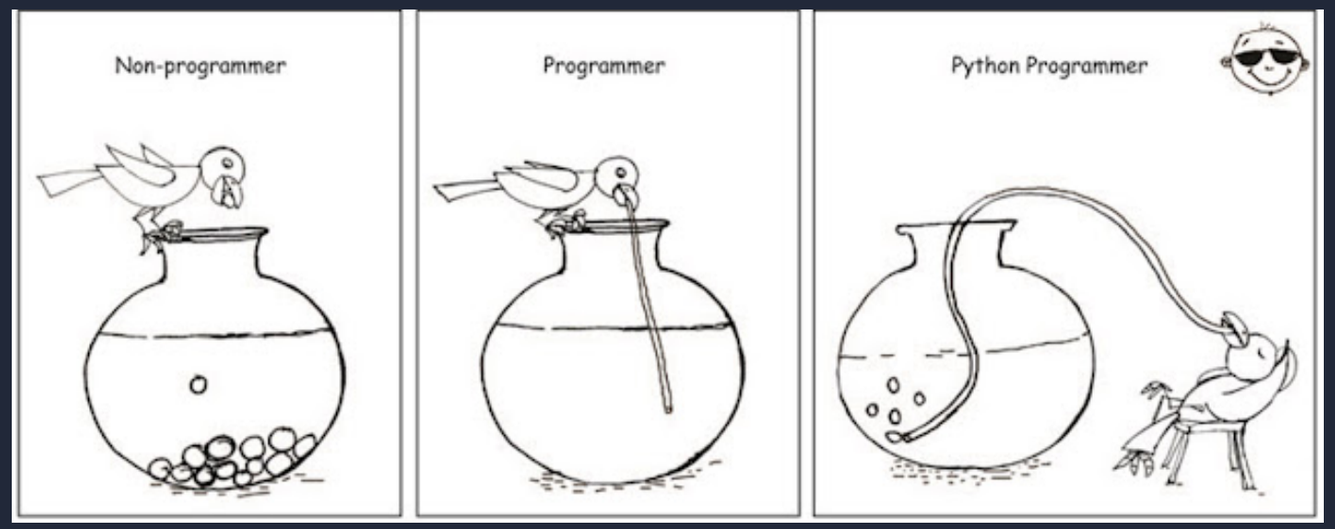
Introduction to Python

Computing 101 workshop - Summer 2024

Rafael Silva Coutinho

Physics Department



In collaboration with Dr. J. Eschle and F. De Almeida

The functional capabilities of Computers

Core Abilities:

- **Computations:** Executes up to a billion operations per second!
- **Memory:** Capable of storing hundreds of gigabytes of data.

Types of Operations:

- **Pre-programmed:** Integral to the programming language.
- **Custom:** Defined by the programmer based on specific needs.

Operational Principle: A computer operates strictly according to the instructions programmed by a user.

Types of knowledge

Declarative Knowledge: Factual statements

- **Example: "There are 118 elements in the periodic table."**

Imperative knowledge: step-by-step instructions or processes

1. Students memorize the first 10 elements of the periodic table.
2. They group elements based on their properties (metals, nonmetals, metalloids).
3. They practice using flashcards to enhance recall.
4. They test their knowledge through a quiz.

```python
# Declarative
small_nums = [x for x in range(20) if x < 5]

# Imperative
small_nums = []
for i in range(20):
    if i < 5:
        small_nums.append(i)
```

Programmable computer

Definition: type of computer where the programs (instructions) and data it needs are stored in its memory, allowing it to execute a wide variety of tasks (without hardware changes)

Instruction Composition: built from a basic set of operations:

1. Arithmetic and logic computations
2. Conducting simple tests
3. Data manipulation and movement

Execution Flow: Utilizes a special program (like an interpreter) to process each instruction in a defined sequence, altering control flow based on conditional tests and concluding upon task completion

Creating Recipes with Programming Languages

Foundation: Programming languages offer primitive operations similar to how languages use words

- Examples of Primitives:
  - English: Words
  - Programming: Numbers, strings, basic operators

Building Blocks: Complex expressions are crafted from these primitives, functioning like legal combinations that drive meaningful actions and calculations within the programming environment.

Purpose: These expressions and operations work together to create detailed, executable recipes in software development.

Why Python?

Interpreted Language: Executes line-by-line without pre-compilation, facilitating rapid development and debugging

User-Friendly: Known for its clean, intuitive syntax, making it ideal for both beginners and experts

Versatile in Data Analysis:
The go-to language for data science

General-Purpose: Suitable for various applications, from web development to machine learning.

Python programming essentials

What is a Python Program?

- A Python program is composed of **definitions** and **commands**:
  - **Definitions** set up variables and functions
  - **Commands** direct the Python interpreter to perform specific tasks

Execution in Python:

- Commands are executed by the Python interpreter either in an interactive shell or read from a script file for more complex operations
- The interpreter processes these commands sequentially, performing actions or computations as instructed

Python in a nutshell

Python is easy (!) or (?)

- Simple to learn, just use it
  - "Driving a car is easy. Just push the power. Any 5 year old can do…"

Accident ↔ bugs

Python is slow (!) or (?)

- Unfortunately, therefore C++ must be used for large data
  - “A Ferrari is slow ….. if you transport goods (compared to a truck)

Python in a nutshell

Python is beautiful

- Clean, powerful and well designed
  - ... and yes, less to care about

Python is a fast, high level language

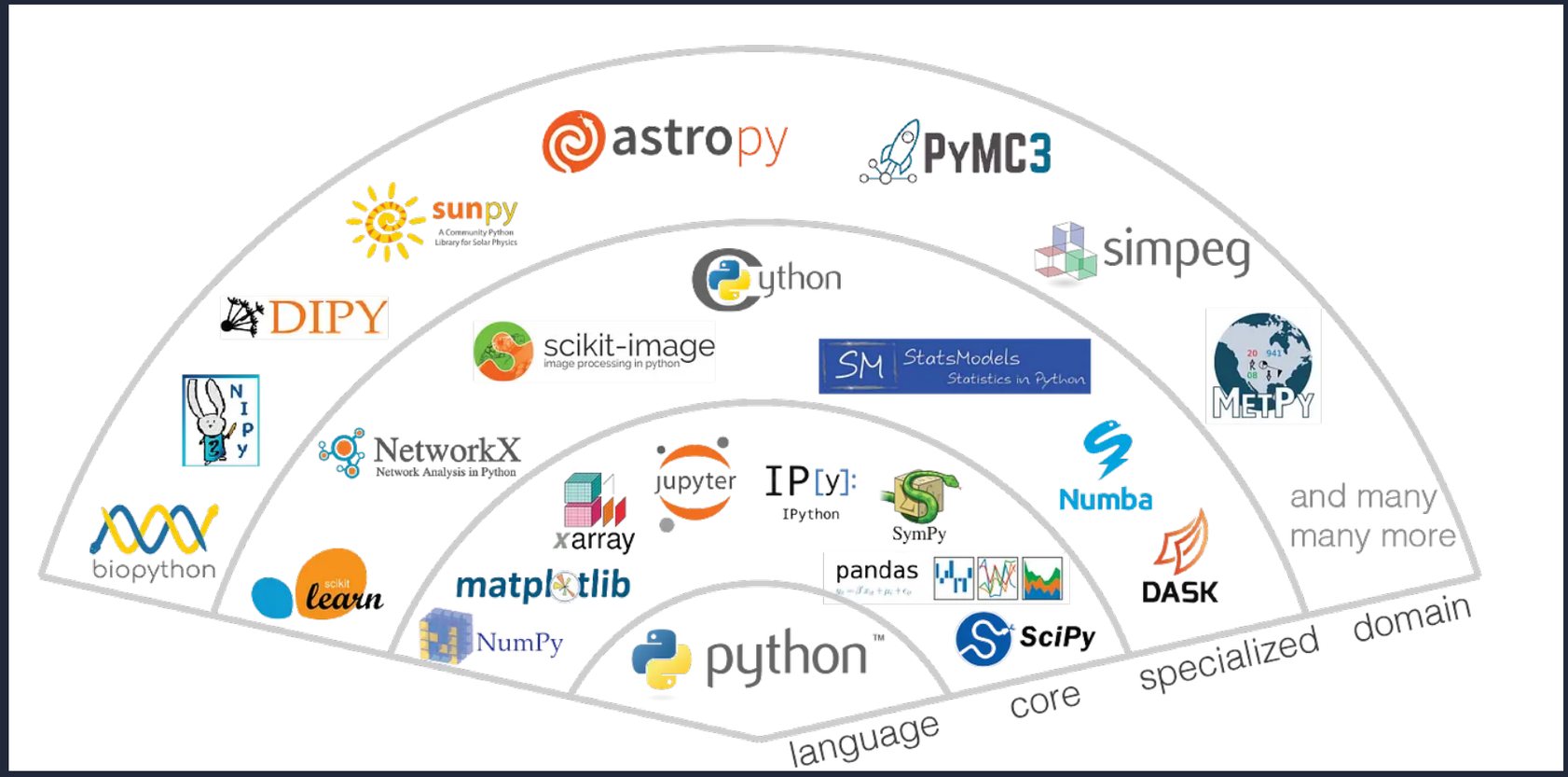
- It can use C++/Fortran code for computation
- No need for manual implementation

Python for data analysis

Several features that significantly aid in data analysis

- Large open-source community
- Big data boom in recent years is a strong driver

"Python ecosystem"



Editors

Collin has already introduced the topic, but it is essential for more complex examples the usage of editors

- Vim vs Emacs
- Simpler editors like nano, gedit, nedit (require less predefined knowledge)
- IDE (integrated development environment)
  - Offers great support for many things and cool features
  - Basically two (similar options)
    - PyCharm
    - VSCode

No particular preference for these exercises

Languages

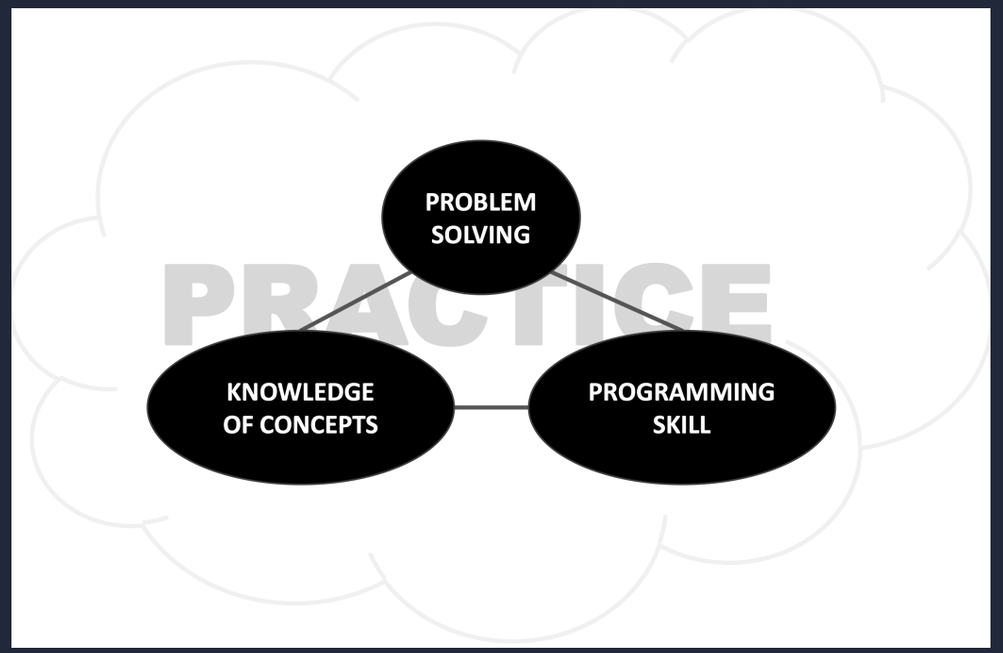
Programming is similar to natural language

How to learn?

- Grammar/Syntax + Rules
- Practice
- Time

Be Curious!

- Ask yourself: what exactly is happening?
- Try out: what if I do this?



Conventions

I am not a book author ... (Although I speak English)

- A book needs a story and a language to "implement it"
- Book author ↔ software engineer

Knowing how to program ≠ knowing the language

We don't need to know that all!

- OOP, coupling, interfaces, protocols, CI/CD, unittests, responsibilities, stateful, VCS, code review, legacy, forward/backwards compatibility, DRY, etc

Be aware of your limits/goals, don't need to reinvent programming!

Installing Python

Recommended: use anaconda/miniconda

- Always use within a given environment
- Package installer that can handle way more:
  - e.g. Multiple python and package versions

Attention:

- can grow big (~GB)! Install to "data" folder with still fast I/O rate, not home
- Check Session 2, day 1 from workshop for details

First step: clone the “project” from workshop webpage