Python Bootcamp 2025-26

(*Statistics and Data Analysis* course)

Teachers: [Jorge Carretero](), [Pau Tallada]() & [Francesc Torradeflot]()

https://mastercosmosbcn.cat/

Prerequisites

- Laptop with:
  - Anaconda (~5GB of storage!)
    - Python environment with the following libraries installed:
      - NumPy • SciPy • Pandas • Astropy • Matplotlib • Scikit-learn • (PyTorch)
  - Git
  - VSCode

- For communicating during classes:
  - Connection details: https://meet.google.com/qgt-gdwb-nqt

PIC port d'informació científica

General information

- Master official information
- In the Virtual Campus: *Estadística i Anàlisi de Dades* [MO79240]
    - **Part 0: Python transversal course**
- Master Schedule
- 6 classes; 2 hours each, always from 10:00 - 12:00
    - **Sep 23, 26, 30, Oct 3, 7, 10**
- Classroom: C7/029
- Contact:
    - Jorge Carretero (carretero@pic.es) - Physicists
    - Pau Tallada (tallada@pic.es) - Research Software Engineer (RSE)
    - Francesc Torradeflot (torradeflot@pic.es) - Mathematician & RSE

Course structure

- **Project-oriented course (group work)**
  - Includes some theoretical sections
- Main goal: **Train a machine learning algorithm using images**
- Sessions:
  - Session 1 (23 Sept): Introduction and Setup
  - Session 2 (26 Sept): Data Exploration
  - Session 3 (30 Sept): Plotting
  - Session 4 (3 Oct): Advanced Programming Concepts
  - Session 5 (7 Oct): Miscellaneous Topics
  - Session 6 (10 Oct): Project Presentation & Feedback

Evaluation Criteria

- Active participation during the course (80%):
  - Attending the sessions
  - Presenting group work
    - At least 3 groups each week
  - Asking and answering questions
- Effort (20%):
  - Going beyond the suggested implementations
  - Proposing innovative solutions
  - Other significant contributions
- Overall grade: 10% of the total mark of the course

Practical note: Post-its will be given to people who contribute; they should be filled in with your name(s)

Today's course

- Quick self-introductions (very very brief, please!)
  - Example: “I’m Jorge Carretero, a physicist and I'm interested on galaxy surveys.”
- Project Overview and Introduction
- Last part:
  - Confirm that everyone has the proper setup
  - **Define the configuration of the different groups (3 people per group)**
    - **Can also be done during the mid-course break**
  - Propose the first exercise for the next session
- Open Meet connection to be able to share screens:
  - Video call link: https://meet.google.com/qgt-gdwb-nqt

Welcome & Overview (I)

- Purpose of the Bootcamp
  - Equip you with **modern programming tools used in research and industry**
  - Build habits for **reproducibility, collaboration, and scalability**
- Why Python?
  - Ubiquitous in science, data analysis, and AI
  - Huge ecosystem of open-source libraries for astronomy, simulation, and ML
  - Easy to learn, readable, and powerful enough for large-scale computation

Welcome & Overview (II)

- Learning Outcomes
  - Set up a productive development environment (IDEs, notebooks)
  - Use Git & GitHub for version control and collaboration
  - Automate workflows with CI/CD
  - Write reliable code through testing & TDD
  - Leverage LLM coding assistants (Copilot, Gemini)
- Course Themes
  - Reproducibility: code & data pipelines that others can trust
  - Collaboration: working together effectively with modern tools
  - Productivity: using automation and best practices to move faster

Python in Scientific Computing

- Why Python Is the Researcher’s Language
  - Ecosystem for Science
    - NumPy • SciPy • Pandas • Astropy • Matplotlib • PyTorch • Scikit-learn
  - Performance
    - Interoperates with C/Fortran via Numba, Cython
  - Community
    - Open-source, peer-reviewed, collaborative culture
  - Adoption
    - Standard in astronomy, cosmology, and data science

Intro to Programming Languages

What Is a Programming Language?

- **Purpose**: Tool to give precise instructions to a computer
- Levels:
  - High-level: Python, R, Julia – human-readable
  - Low-level: C, Fortran, Assembly – close to hardware
- **Domain-Specific**: IDL, Mathematica – specialized for a field

Compiled vs Interpreted Languages

How Code Becomes Execution

- **Compiled**: C, Fortran — translated to machine code before running → very fast
- **Interpreted**: Python, R — executed line by line → easier to prototype
- **Performance Tradeoff**: Fast runtime vs. fast development cycle
- **JIT Compilation**: Numba, PyPy bring compiled speed to Python

IDEs & Development Environments

- **IDE** = Integrated Development Environment
  - VS Code, PyCharm — autocomplete, linting, debugging, testing, tooling
- **JupyterLab**: Interactive notebooks for science & exploration
- **Terminal** Tools: Run quick scripts, manage environments
- Recommended Setup:
  - Python 3.12+, VS Code, Jupyter, Git, venv/Conda

PIC port d'informació científica

Jupyter Notebooks for Research

- Literate Programming
    - Combine code, text (Markdown), LaTeX, plots
- Use Cases
    - Data exploration, simulations, reproducible workflows
- Sharing
    - Easy to save, share, and rerun experiments
- Tools
    - JupyterLab • VS Code notebooks • Google Colab

Version Control with Git

- Version Control
  - Save history, revert changes, experiment safely
- Core Commands
  - init, clone, commit, push, pull, branch
- Collaboration
  - GitHub / GitLab for hosting & teamwork
- Best Practices
  - Small commits
  - Clear messages
  - .gitignore

Collaborating on Code

Teamwork and Open Science

- Collaboration Tools:
  - Pull Requests • Code Reviews • GitHub Issues/Discussions
- Project Structure:
  - Modular code • Clear directories • Inline documentation
- Open Science & Licensing:
  - MIT, GPL, and other open-source licenses
- Best Practices:
  - Review carefully, communicate clearly, document changes

PIC
port d'informació
científica

Agile & Scrum (for Scientists!)

Managing Research Like a Software Project

- Agile Mindset
  - Iterative, flexible, feedback-driven progress
- Scrum Basics
  - Sprints • Stand-ups • Retrospectives
- Adapted for Science
  - Weekly goals • Research backlogs • Milestone tracking
- Tools
  - Trello • GitHub Projects • Notion

PIC port d'informació científica

Test-Driven Development (TDD)

Trust Your Code, Trust Your Results

- Why Test?
  - Catch bugs early, ensure reproducibility
- Test Types:
  - Unit • Integration • Regression
- TDD Workflow:
  - Write test → Write code → Validate
- Tools:
  - pytest • unittest • doctest

CI / CD

- CI/CD Concepts:
  - CI (Continuous Integration): Run tests on every code change
  - CD (Continuous Delivery/Deployment): Automate releases or reproducible pipelines
- Automation Tasks
  - Testing • Linting • Formatting
- Tools
  - GitHub Actions • GitLab CI • CircleCI
- Example
  - Python repo automatically runs tests and linters on each push

Package & Environment Management

- Why It Matters:
  - Reproducibility • Dependency isolation
- Tools & Approaches:
  - venv + pip — lightweight environment + package management
  - conda — science-focused packages, cross-platform support
- Tracking Dependencies:
  - requirements.txt • environment.yml • pyproject.toml
- Advanced Reproducibility:
  - Binder • Docker

Using LLMs & Co-Agents

- What They Do:
  - Suggest code, boilerplate, docstrings, test cases
- Use Cases:
  - Debugging, refactoring, algorithm ideas, documentation
- Tools:
  - GitHub Copilot • Gemini • ChatGPT
- Pitfalls:
  - Always review suggestions, **don't trust blindly**
- Ethics:
  - Maintain code quality, credit sources, ensure reproducibility

PIC port d'informació científica

Distributed & Parallel Computing

Scale Your Computations

- Why Parallelize?
  - Handle large datasets, run heavy simulations faster
- Tools:
  - Dask: Scales NumPy/pandas for parallel processing
  - Ray: Python-native distributed tasks
  - Spark / PySpark: Big data analytics across clusters
- Cluster Environments
  - HPC basics, SLURM, MPI, job scripts

PIC port d'informació científica

Documentation

Make Your Code Understandable and Reusable

- Why Document?
  - Ensure reproducibility • Shareable knowledge • Maintainable code
- Docstring Standards:
  - NumPy/SciPy style — describe inputs, outputs, purpose, examples
- Tools:
  - help() in Python • Sphinx • MkDocs for auto-generated documentation

Code Quality & Style

Keep Your Code Clean and Consistent

- Linting:
    - flake8, pylint — detect errors, enforce style
- Formatting:
    - ruff, black, isort — automatic code formatting and import sorting
- Pre-commit Hooks:
    - Run checks before every commit
- Principle:
    - Write clean, readable code for yourself and collaborators

Summary & Setup Instructions

Recap & Get Started

- Recap of Key Tools:
  - IDEs, Git, Testing, CI/CD, Notebooks, LLMs
- Recommended Setup:
  - Python 3.12+, VS Code, JupyterLab, Git, pytest
- Support & Resources
  - GitHub repo, Slack/Discord channels, install guides
- Next Steps
  - Build your development environment, follow setup instructions in shared repo

PIC port d'informació científica

Configuration Setup – Troubleshooting

- Define the configuration of the different groups
- Verify that everyone has the correct setup (troubleshooting)
    - Identify common configuration issues

Exercise 1: Documentation & Data retrieval

- Data Sources – Galaxy Zoo 1
  - Where to access it: SDSS Skyserver
- For the next session, we need two datasets:
  - Galaxy morphology classifications (table "zooSpec")
    - Labels: elliptical or spiral (used as the target for training the ML model)
  - Photometry data for galaxies with morphology classifications (table "PhotoObjDR7")
    - Used to select objects of interest and as features for the ML model
- Hint – SQL query (JOIN between those two tables)

Exercise 1: Documentation & Data retrieval

- Data Sources – Galaxy Zoo 1
  - Where to access it: SDSS Skyserver (account needed!)
- For the next session, we need two datasets:
  - Galaxy morphology classifications (table "zooSpec")
    - Labels: elliptical or spiral (used as the target for training the ML model)
  - Photometry data for galaxies with morphology classifications (table "PhotoObjDR7")
    - Used to select objects of interest and as features for the ML model
- Hint – SQL query (JOIN between those two tables)
  - Submit a CAS job with the SQL query below and download in csv format:

```
SELECT ZooSpec.*, PhotoObjDR7.* into MyDB.ZooSpecPhoto
FROM ZooSpec INNER JOIN PhotoObjDR7
ON PhotoObjDR7.dr7objid = ZooSpec.dr7objid
```