032CM - 2026

PROGRAMMING FOR COMPUTATIONAL CHEMISTRY

Python classes

Gianluca Levi

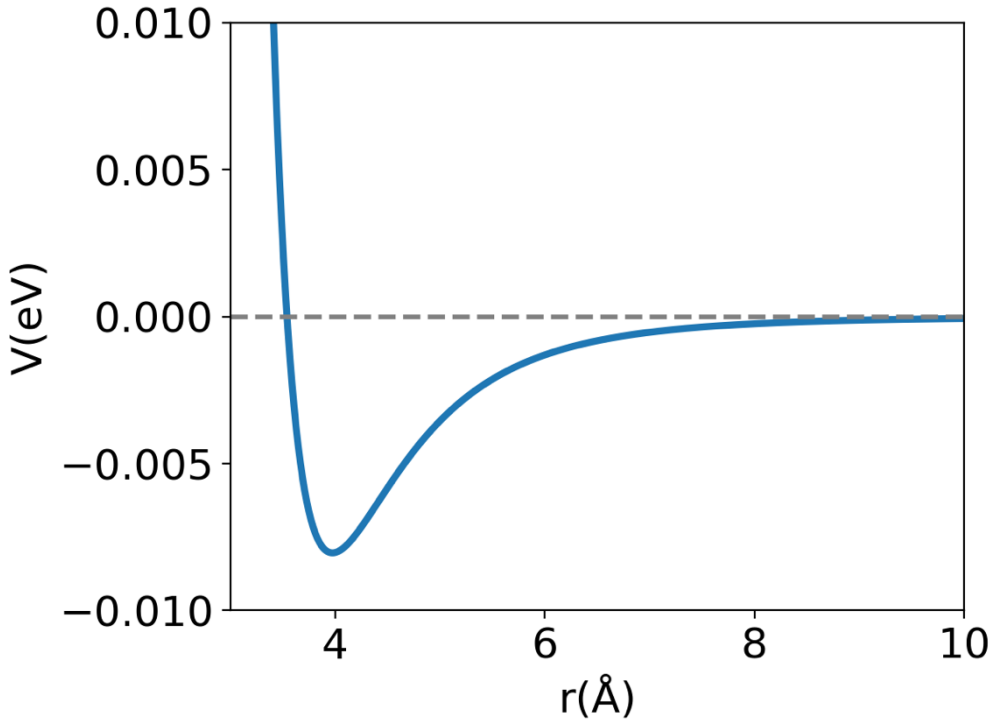
gianluca.levi@units.it, giale@hi.is

Office: Building C11, 3rd floor, Room 329

Fall 2026

Lennard-Jones potential

$$V(r) = 4\varepsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right]$$



Simple **pairwise model** for interatomic interactions

Short-range repulsion + **dispersion attraction**

Used to model short-range interactions in **molecular dynamics simulations**

Specific vs. general implementation

Python class

- Defines a **data type**
- Objects made from the class are **instances** of that type

Bundles data and behavior

- Packs data (attributes) and functions (methods) operating on them
- Avoids setting global variables (and related side effects)
- Optimal for multiple parameter sets
- Improves **reusability** and **readability**

Example: a NumPy array

```
import numpy as np

a = np.array([1.0, 2.0, 3.0, 4.0])

a.size     # Attribute: number of elements → 4
a.mean()   # Method: compute the average → 2.5
```

General structure of a class

```
def Class_Name:
    """
    Description of the class
    """
    def __init__(self, p1, p2, ...):
        self.var1 = p1
        self.var2 = p2
            ...
            ...

    def function_name(self, fp1, ...):
        <expression>

        return result_1, result_2, ...
```

General structure of a class

Docstring

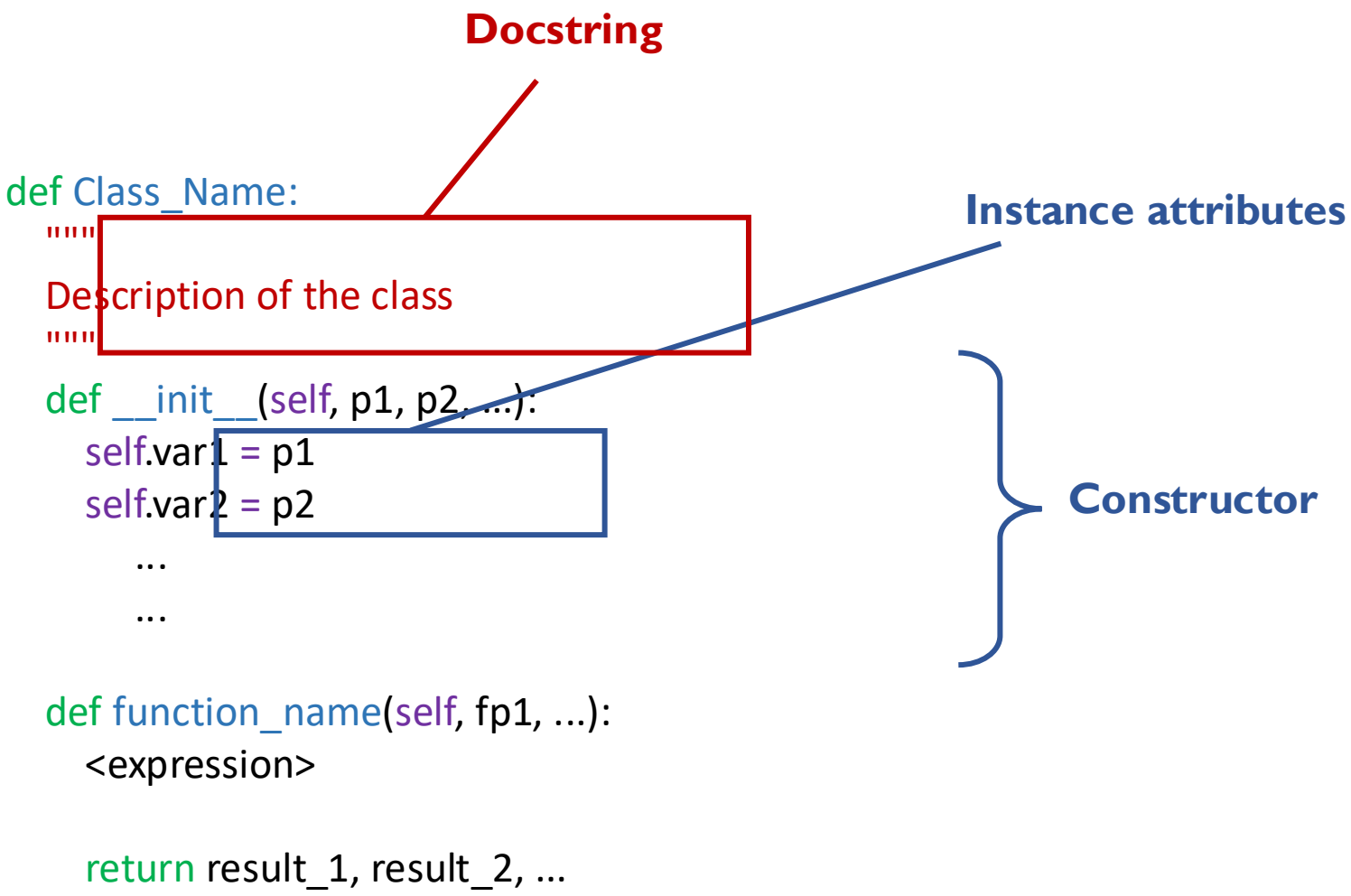
```
def Class_Name:
    """
    Description of the class
    """
    def __init__(self, p1, p2, ...):
        self.var1 = p1
        self.var2 = p2
            ...
            ...

    def function_name(self, fp1, ...):
        <expression>

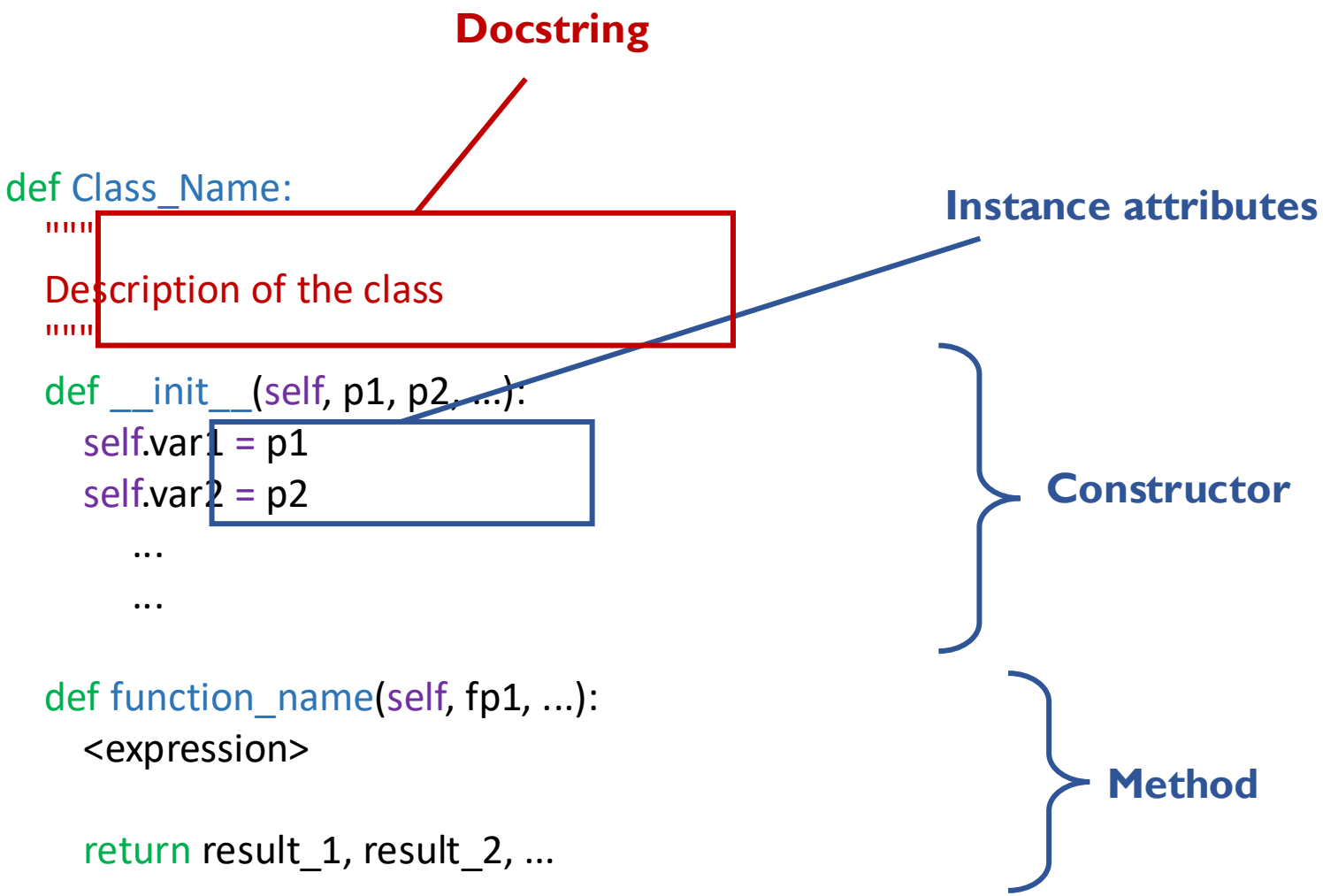
        return result_1, result_2, ...
```

Constructor

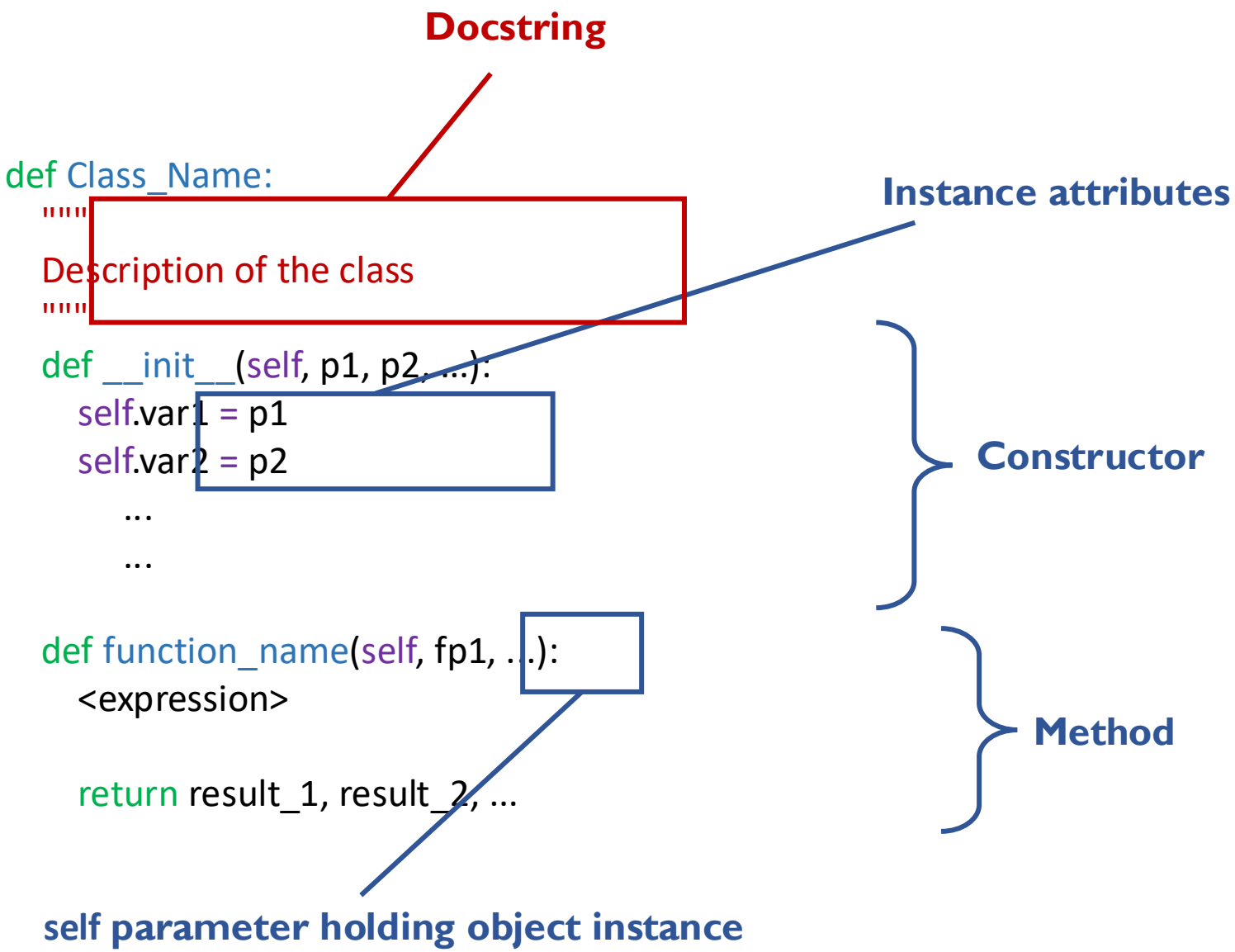
General structure of a class



General structure of a class



General structure of a class



Assignment 5

Problem 1

The wave function of the hydrogen-like $3p_z$ atomic orbital in spherical polar coordinates is given by:

$$\psi_{3p_z}(r,\theta,\phi) = R_{31}(r)Y_{10}(\theta,\phi)$$

where $R_{31}(r)$ is the radial part:

$$R_{31}(r) = \frac{1}{27}\left(\frac{2Z}{3a_0}\right)^{3/2} \rho\,(6-\rho)\,e^{-\rho/3}, \qquad \rho = \frac{Zr}{a_0},$$

and $Y_{10}(\theta,\phi)$ is the angular part (spherical harmonic):

$$Y_{10}(\theta,\phi) = \sqrt{\frac{3}{4\pi}}\cos\theta.$$

Here, Z is the atomic number and a_0 is the Bohr radius. Throughout this problem, consider the hydrogen atom ($Z = 1$).

(a) Write a Python function that computes the ψ_{3p_z} wave function for the hydrogen atom for given values of the radial coordinate r and polar angle θ. Use atomic units, where the Bohr radius $a_0 = 1$.

(b) Plot the radial wave function $R_{31}(r)$ for $r \in [0, 25]$ Bohr, and create a 2D contour plot of ψ_{3p_z} in the xz-plane for $x, z \in [-15, 15]$ Bohr. How many radial nodes can you identify? Compare with the number of radial nodes obtained using the formula $n - l - 1$, where n and l are the principal and angular momentum quantum numbers, respectively. *Note*: the point at $r = 0$ is not considered a node.

Hint: For the 2D plot, convert Cartesian to spherical coordinates using

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\theta = \arccos\left(\frac{z}{r}\right).$$

(c) Find the approximate location of the radial node between $r = 2.5$ and 7.5 Bohr using the bisection and the Newton-Raphson methods. Use $[2.5, 7.5]$ Bohr as the initial interval for the bisection method and $r_0 = 2.5$ Bohr as the initial guess for the Newton method.

(d) For each method, save the error at each iteration. For the bisection method, use the length of the interval as the error. For the Newton-Raphson method, use the absolute value of the radial wave function evaluated at the current approximation, $\epsilon_i = |R_{31}(r_i)|$. Then, plot the error ϵ_i versus the iteration number using a logarithmic scale for the y-axis and create a log-log plot of ϵ_{i+1} versus ϵ_i. Exclude zero error values from the logarithmic plots, if necessary. Roughly estimate the slope for each method from the log-log plot. The slope is the *order of convergence*, which measures the rate of convergence of the algorithm. Comment on the differences between the two methods.

Hint: To set the log scale in Matplotlib use `plt.xscale('log')/plt.yscale('log')`.

(e) Find the approximate location of the maximum of the radial wave function $R_{31}(r)$ using the Newton-Raphson method with an initial guess of $r_0 = 1$ Bohr. What happens if you use $r_0 = 7$ Bohr? Comment.

Assignment 5

Problem 2

The paiwise Lennard-Jones potential can be used to model the interatomic interactions of noble-gas atoms. The Lennard-Jones interaction energy between atoms is given by:

$$E_{LJ} = \sum_{i=1}^{N} \sum_{j>i}^{N} 4\epsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

where N is the number of atoms and r_{ij} the distance between atoms i and j. The positive r^{-12} term describes the repulsive part of the potential, while the negative r^{-6} term describes the attractive part.

(a) Create a Python class called `Atoms` that represents a cluster of atoms with attribute the (x, y, z) coordinates of the atoms in the cluster (NumPy array). Include a `set_positions` method to update the atom positions and a `get_pairwise_distances` method that computes and returns a vector of pairwise distances between all atoms.

(b) Create a `LennardJonesCalculator` class that computes the Lennard-Jones energy of the cluster of atoms with attributes `epsilon` and `sigma`. The class should have a `calculate_energy` method that takes the `Atoms` object and returns the total Lennard-Jones energy.

(c) Calculate the energy of the following two configurations (atomic coordinates in Å) of a cluster of four Ar atoms ($\epsilon = 8.0$ meV and $\sigma = 3.54$ Å).
Linear chain:

$$\texttt{positions} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 4.0 & 0.0 & 0.0 \\ 8.0 & 0.0 & 0.0 \\ 12.0 & 0.0 & 0.0 \end{bmatrix}$$

Square arrangement:

$$\texttt{positions} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 4.0 & 0.0 & 0.0 \\ 0.0 & 4.0 & 0.0 \\ 4.0 & 4.0 & 0.0 \end{bmatrix}$$

Which configuration has the lowest energy?