CH40208: TOPICS IN COMPUTATIONAL CHEMISTRY

FUNCTIONS & MODULAR CODE

FUNCTIONS

- Functions save programmers time and makes code simpler
- Essentially it is a way stop you having to write the same thing time and time again
- We have already encountered the use of functions
  - `print()`
  - `np.sum()`

FUNCTIONS

- We define a function, can pass it arguments and possibly receive information returned
- Not all functions need arguments or necessary return an object
- It is considered good form to add a *docstring* to all of your functions

FUNCTIONS

DEMO

FUNCTIONS

- Not all arguments are the same
- Those we have already seen included *required* arguments
- There is also *default* arguments, where the argument will default to a given value if not defined
- And *variable-length* arguments, essentially a list that is passed that can be any length

FUNCTIONS

DEMO

MODULES

- Another way to simplify your code is to store frequently used functions in *modules*
- Modules are files (with the file format `.py`) where we can store a lot of functions that we need to use
- Like libraries (such as NumPy) it is necessary to `import` any modules that we want to use in our code

MODULES

DEMO

PROBLEM

- The interaction energy between two atoms may be estimated using the Lennard-Jones potential model

$$E_{LJ} = \epsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right]$$

- Where ϵ and r_m are constants describing the shape of the interaction energy surface

PROBLEM

- Add a function to calculate the interaction energy between two atoms to your `atom_helper.py` module

$$E_{LJ} = \epsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right]$$

$\epsilon = -2.2\,\mathrm{kJ\,mol^{-1}}$

$r_m = 3.7\,\text{Å}$

PROBLEM

- Use this function to calculate the interaction energy between each of the pairs of atoms from Week 2

$$E_{LJ} = \epsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right]$$

PROBLEM

- A common aim in computational chemistry is to fit the energetically optimised structure of a given chemical system
- To find this, we must find the energy minimum for all of the interactions
- This is achieved using an optimisation algorithm

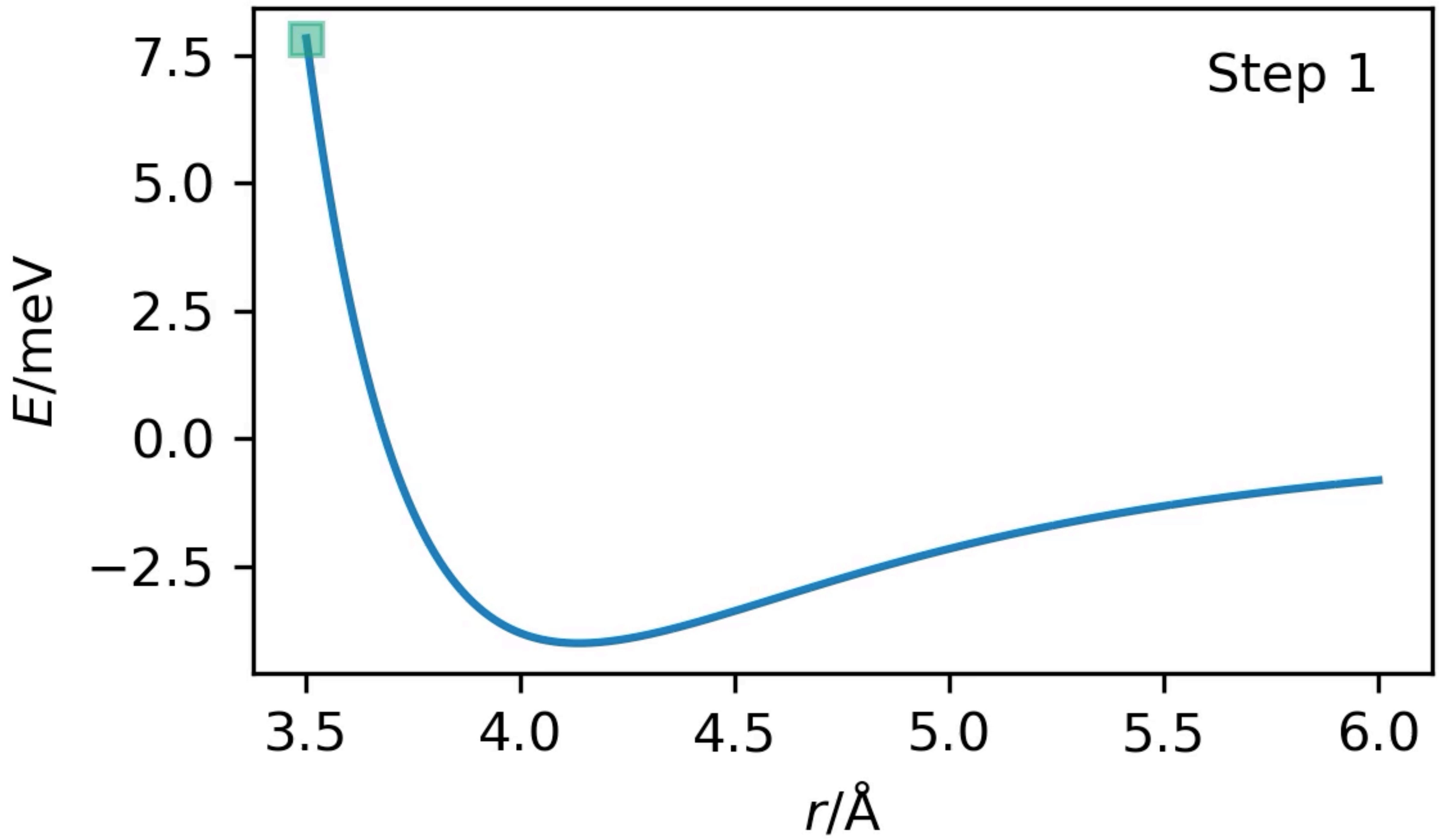
PROBLEM

- A popular, and simple, optimisation algorithm is the Newton-Raphson Method

$$r_1 = r_0 - \frac{E'}{E''}$$

- The distance between the two atoms is updated iteratively based on the first and second derivatives of the energy

PROBLEM



PROBLEM

- A popular, and simple, optimisation algorithm is the Newton-Raphson Method

$$r_1 = r_0 - \frac{E'}{E''}$$

- The distance between the two atoms is updated iteratively based on the first and second derivatives of the energy (given in the handout)

PROBLEM

- Write an implementation of the Newton-Raphson Method that will perform 10 iterations
  - Hint: it is not straightforward to implement with this to take advantage of NumPy optimisation
- Calculate the energy-minimum distance for the two sets of A and B

SCIPY.OPTIMIZE.MINIMIZE

DEMO