Lecture #2: Python for atomistic materials modeling

Slides by Artem Dembitskiy and Dmitry Aksenov

Previously on:

- What is materials informatics
- Python crash course
- HW0 announcement

Goals

- Introduce python libraries for materials modeling
- Explain the advantages of using programming over the manual approach

Agenda

- Materials modeling toy example
- Why use python?
- ASE and Pymatgen

Toy example: Band gap vs. X in cubic LiX, X = F, Cl, Br, I

Task

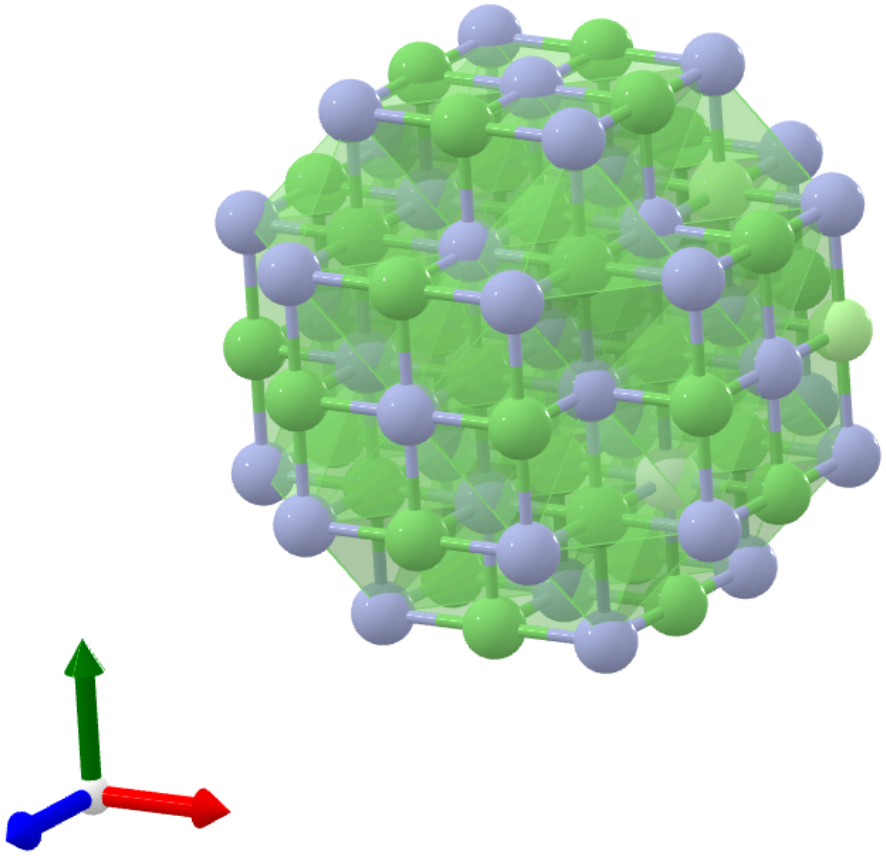
- Calculate a band gap of LiX using a software "Y"
- Perform correlation analysis
- Explain the trend

A software "Y" takes the following input files:

- structure file
- parameters of the calculation
- other specific files

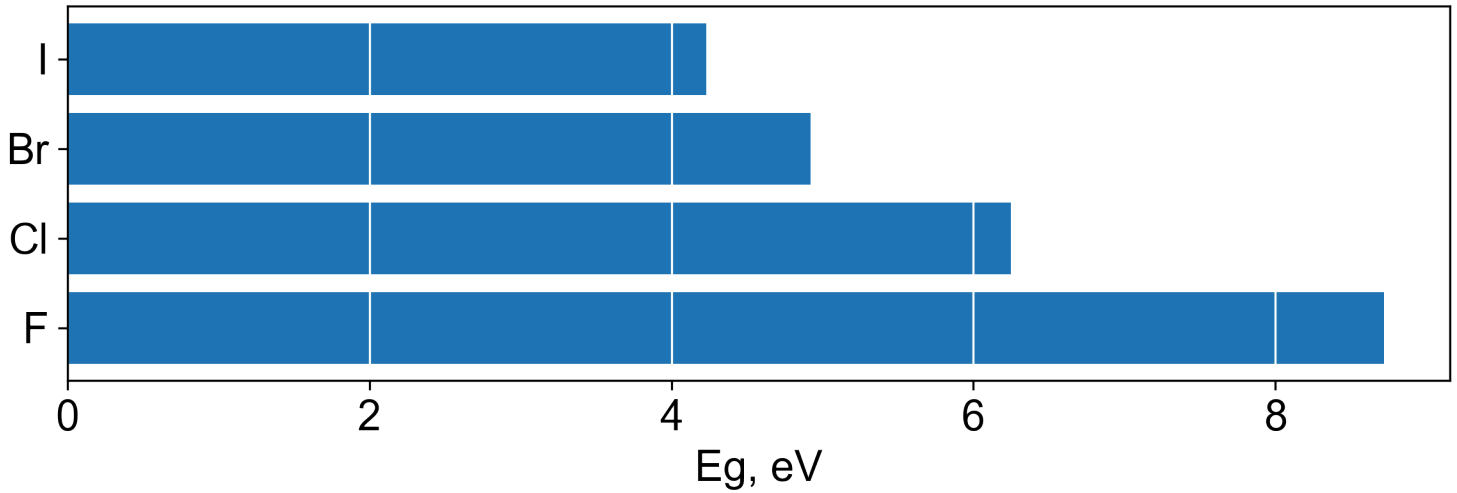
... and outputs:

- 100,000 lines of text
- and one of these contains the Eg value



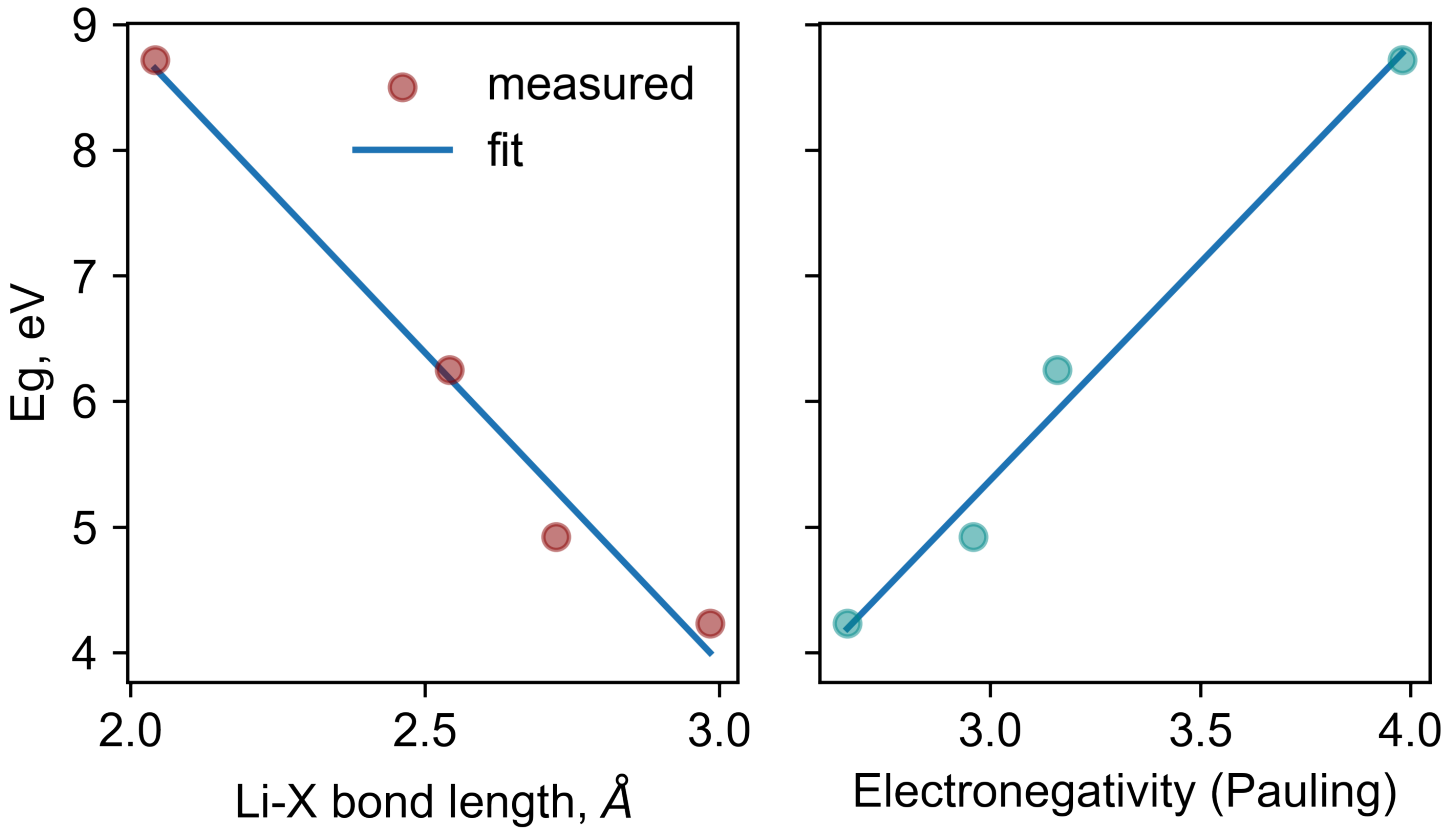
The trend

- The higher atomic number the lower Eg



Correlation analysis

- Li-X bond length vs. Eg
- Electronegativity (X) vs. Eg



Workflow

- Get(Create) structure file
- Prepare input files
- Run calculations
- Parse output
- Repeat the above steps for other structures
- Collect observables
- Analyze correlations
- Make conlusions

Which step takes the longest?

Which can be automated?

Manual vs. Automation

Manual

- typing input files by hands
- getting output data by opening files in text editors and copy/paste numbers
- analysis of output data using Excel and Origin-like programs
- relying on programs with graphical interface (in some cases its is efficient but in many cases you need to press many buttons every time to accomplish routine tasks)

Slow, easy to make a mistake, boring for repeating tasks

Manual vs. Automation

Automation

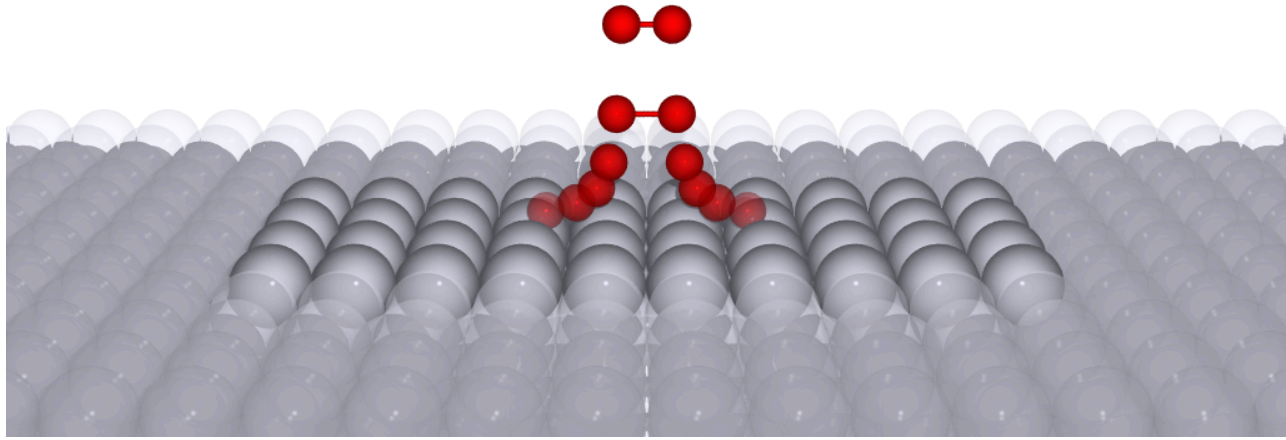
- all input files are created by scripts (only small changes are needed) files
- comprehensive and very flexible analysis specific for your needs easy reuse of code for routine tasks
- Information can be stored in a database
- Takes more time in the beginning, but then is much faster

Fast, error-proof, fun

Python libraries for atomistic materials modeling: ASE

ASE - is an Atomic Simulation Environment written in the Python programming language with the aim of setting up, steering, and analyzing atomistic simulations.

- Good documentation with tutorials
- Interfaces for most popular modeling simulation codes for density functional theory, molecular dynamics, etc
- Main features:
  - Atoms object for handling molecules/crystals
  - Read/write input files | Parse output files
  - Optimizers
  - Calculators
  - Visualization
  - Surface modeling

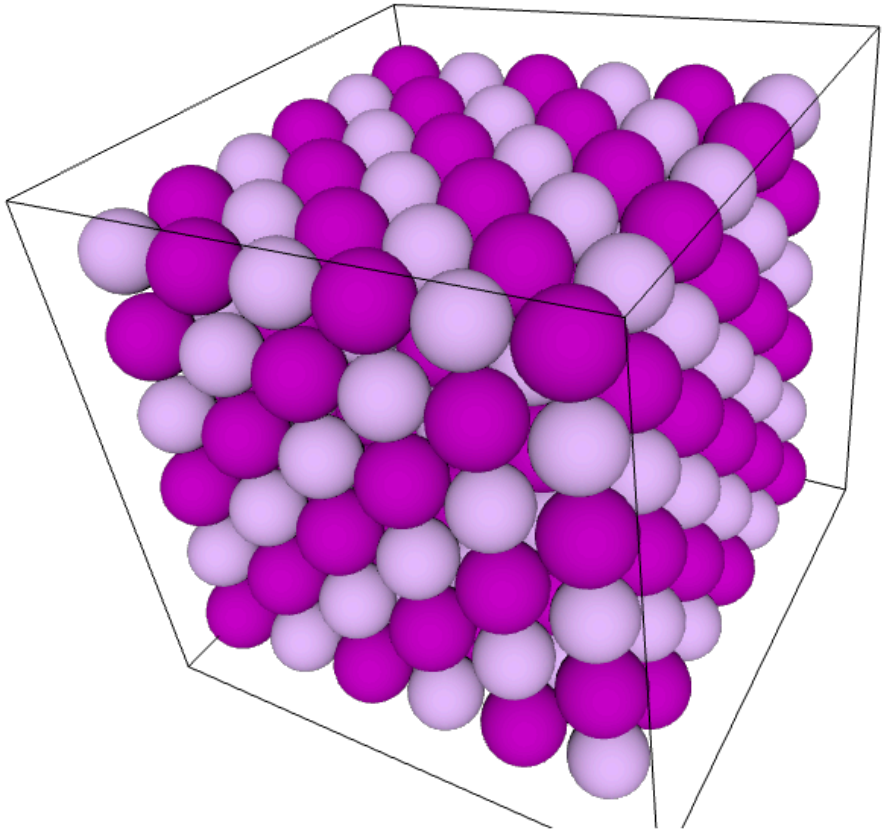


ASE minimum usage example

- read .cif file
- create supercell
- visualize
- save supercell
- more at the seminar

```
from ase.io import read, write
from ase.visualize import view
from ase.build import make_supercell

atoms = read('data/LiI.cif')
sc = make_supercell(atoms,
  [[3, 0, 0 ],
   [0, 3, 0],
   [0, 0, 3]]
   )
view(sc, viewer='x3d')
write('data/LiI_3x3x3.cif', sc)
```

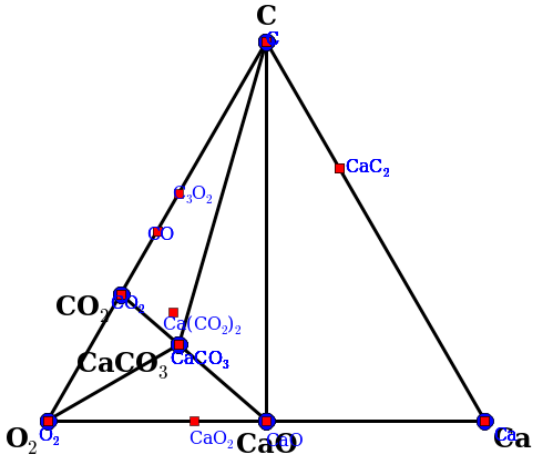


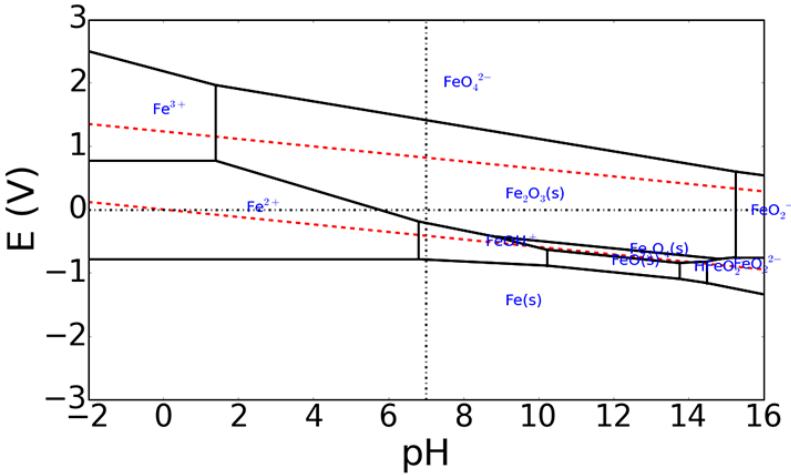
Python libraries for atomistic materials modeling: Pymatgen

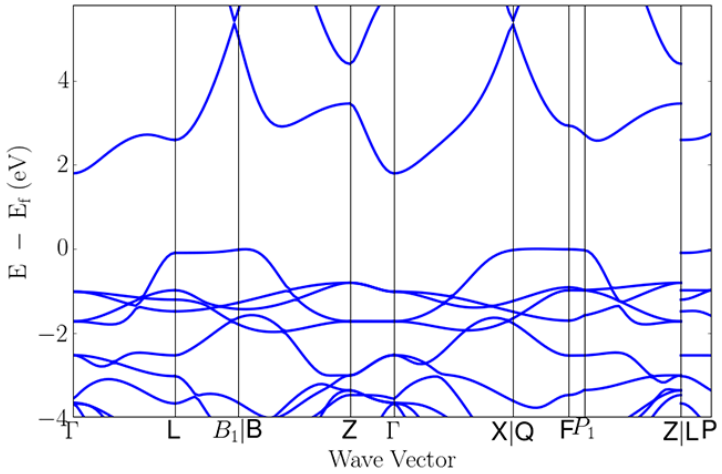
Pymatgen (Python Materials Genomics) - a robust, open-source Python library for materials analysis

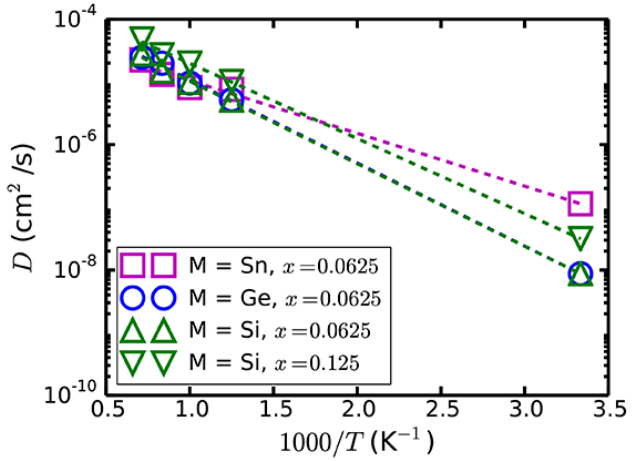
Main features

- Highly flexible classes for the representation of Element, Site, Molecule, Structure objects.
- Powerful analysis tools, including generation of phase diagrams, Pourbaix diagrams, diffusion analysis, reactions, etc.
- Electronic structure analysis
- Integration with The Materials Project REST API (next lecture)

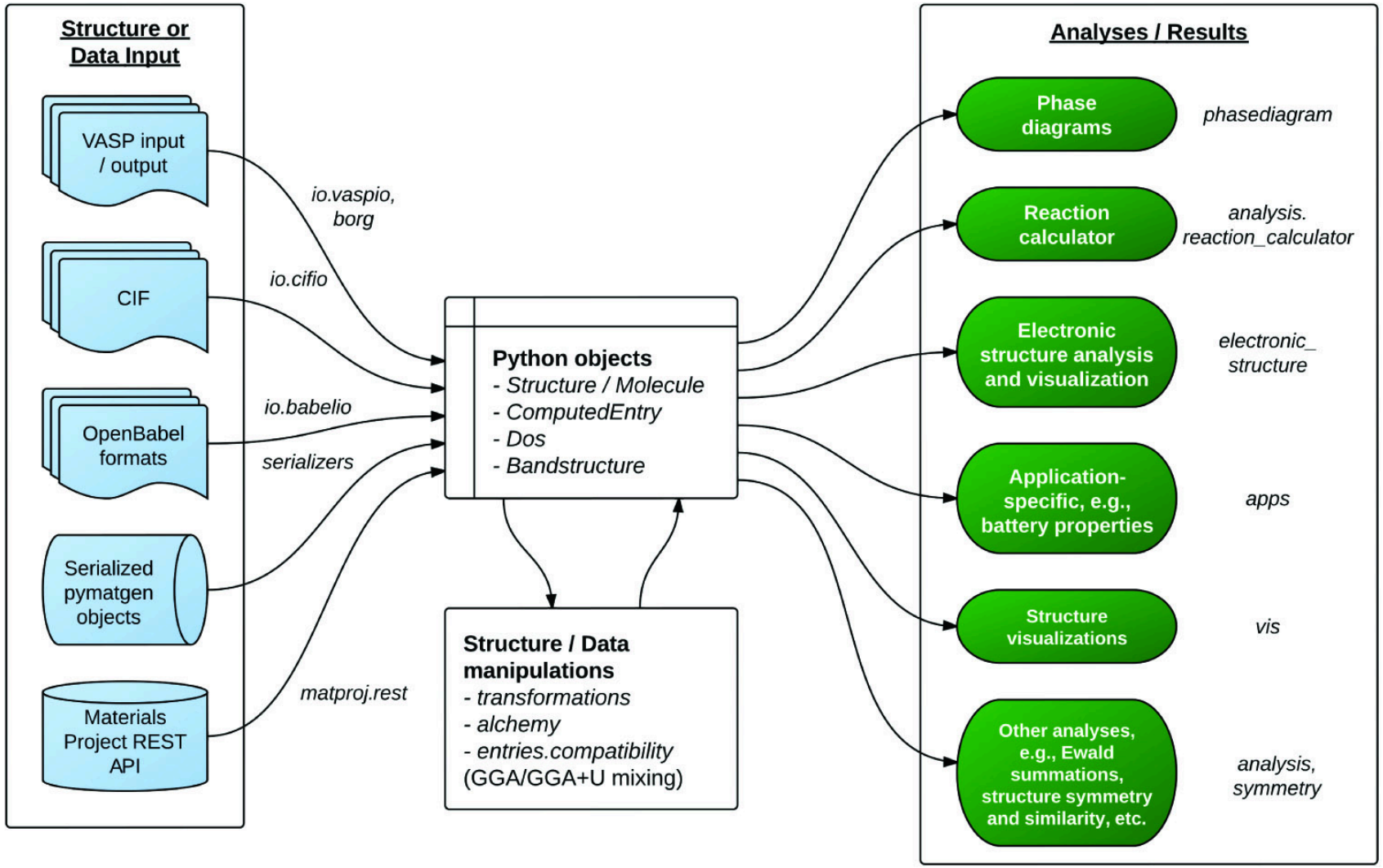








Basic workflow with pymatgen



Take home message

Automation:

- saves your time
- gives you the opportunity (tools) to realize your ideas
- reduces number of mistakes (typos) made