**Beginner-Friendly Quantum & Chemistry Projects (Python-Based)**

These projects will help you build intuition, hands-on skills, and portfolio-worthy artifacts:

**🧪 1. Visualize Quantum Wavefunctions**

* **Tool**: qmsolve
* **Project Idea**: Simulate and plot 1D/2D Schrödinger equation solutions (e.g., particle in a box, harmonic oscillator).
* **Portfolio Angle**: Show interactive Jupyter notebooks with visualizations and physical interpretations.

**⚛️ 2. Quantum Chemistry with Deep Learning**

* **Tool**: DeepChem
* **Project Idea**: Predict molecular properties (e.g., solubility, toxicity) using pretrained models.
* **Portfolio Angle**: Build a mini pipeline from SMILES to prediction, comparing classical vs quantum-inspired models.

**🧠 3. Quantum Machine Learning**

* **Tool**: PennyLane
* **Project Idea**: Train a quantum neural network to classify simple molecular data.
* **Portfolio Angle**: Showcase hybrid quantum–classical models and explain the quantum circuit logic.

**🔬 4. Parse and Analyze Quantum Chemistry Logs**

* **Tool**: cclib
* **Project Idea**: Extract energies, orbitals, and geometries from Gaussian or ORCA outputs.
* **Portfolio Angle**: Build a parser that visualizes convergence behavior or compares basis sets.

**🧮 5. Quantum Monte Carlo Simulations**

* **Tool**: deepqmc
* **Project Idea**: Use neural networks to approximate electron distributions.
* **Portfolio Angle**: Explain how deep learning enhances traditional quantum methods.

**🧰 Top Open-Source Python Libraries for Quantum & Chemistry**

| **Tool** | **Focus Area** | **Highlights** |
| --- | --- | --- |
| QuTiP | Quantum mechanics | Simulate open/closed quantum systems |
| PennyLane | Quantum computing & chemistry | Hybrid models, integrates with PyTorch & TensorFlow |
| DeepChem | Drug discovery & quantum chemistry | Pretrained models, SMILES support |
| qmsolve | Schrödinger equation | Visual wavefunction simulations |
| cclib | Computational chemistry logs | Parses Gaussian, ORCA, etc. |
| deepqmc | Quantum Monte Carlo | Deep learning for electron distributions |
| torchani | Neural potentials | ANI models for molecular energies |
| DFTB+ | Tight-binding DFT | Fast semi-empirical simulations |

**🧠 Portfolio-Worthy Project Ideas**

Here are a few curated ideas that blend your cheminformatics background with quantum learning:

**1. Quantum-Enhanced Ligand Screening**

* Use PennyLane or DeepChem to compare classical vs quantum models on ligand–receptor binding scores.

**2. Wavefunction Explorer**

* Build a Jupyter dashboard using qmsolve to visualize quantum states and explain physical intuition.

**3. Quantum Log Analyzer**

* Use cclib to parse ORCA/Gaussian outputs and generate automated reports on convergence, HOMO–LUMO gaps, etc.

**4. Neural Potential Explorer**

* Use torchani to predict molecular energies and compare with DFTB+ results.

**🧭 Next Steps**

* Start with **visual simulations** (qmsolve) to build intuition.
* Move to **data-driven models** (DeepChem, torchani) to connect with your cheminformatics workflows.
* Explore **quantum circuits** via PennyLane once you're comfortable with basic quantum concepts.

**What Is Psi4?**

Psi4 is an open-source suite for **ab initio quantum chemistry**, designed to compute molecular properties with high accuracy. It supports a wide range of methods including:

* **Hartree–Fock (HF)**
* **Density Functional Theory (DFT)**
* **Møller–Plesset Perturbation Theory (MP2)**
* **Coupled Cluster (CCSD, CCSD(T))**
* **Symmetry-Adapted Perturbation Theory (SAPT)**
* **Geometry optimization, vibrational analysis, and more**

It’s written in C++ but has a **Python API (PsiAPI)** that lets you script everything interactively—ideal for Jupyter notebooks and reproducible workflows.

**🧪 What You Can Do with Psi4**

Here are some beginner-to-intermediate project ideas:

**1. Compute Molecular Energies**

python

import psi4

psi4.set\_memory('500 MB')

psi4.set\_num\_threads(4)

molecule = psi4.geometry("""

0 1

O

H 1 0.96

H 1 0.96 2 104.5

""")

energy = psi4.energy('scf/6-31G(d)', molecule=molecule)

print("HF Energy:", energy)

* Try different basis sets and methods (e.g., mp2, b3lyp).
* Compare energies across conformers or isomers.

**2. Geometry Optimization**

python

psi4.optimize('b3lyp/6-31G(d)', molecule=molecule)

* Useful for preparing ligands or fragments before docking or simulation.

**3. Vibrational Analysis**

python

freqs = psi4.frequency('b3lyp/6-31G(d)', molecule=molecule)

* Identify IR-active modes or validate optimized geometries.

**4. SAPT Analysis**

* Decompose interaction energies between two fragments (e.g., ligand–receptor).
* Great for understanding binding contributions: electrostatics, dispersion, induction.

**🧠 Portfolio Project Ideas with Psi4**

| **Project** | **Description** | **Skills Highlighted** |
| --- | --- | --- |
| **Ligand Optimization Pipeline** | Optimize ligand geometries before docking | Basis set selection, geometry optimization |
| **Binding Energy Comparison** | Use SAPT to compare ligand–receptor interactions | Fragment definition, energy decomposition |
| **Conformer Energy Landscape** | Generate and rank conformers using HF/DFT | SMILES parsing, energy profiling |
| **Quantum Descriptors for ML** | Extract HOMO–LUMO gaps, dipole moments for ML models | Feature engineering, quantum–ML integration |

**🔧 Installation Tips**

Psi4 can be installed via Conda:

bash

conda install psi4 -c psi4

Or use their GitHub repo for source builds

### What is PSI4?

PSI4 is a massively powerful, open-source ab initio computational chemistry package. "Ab initio" means it solves the electronic Schrödinger equation from first principles, without empirical parameters. It's used by academics and industry researchers worldwide for tasks like calculating:

* Molecular energies
* Molecular structures (bond lengths, angles)
* vibrational frequencies
* interaction energies between molecules
* and much more.

### Core Concepts You'll Learn (The "What")

1. **The Schrödinger Equation:** You'll learn what it means to "solve" it, even approximately, for a molecule.
2. **Basis Sets:** You'll learn what they are (sets of functions used to describe electron orbitals) and how the choice of basis set (cc-pVDZ, 6-31G\* etc.) affects your results.
3. **Electronic Structure Methods:** This is the hierarchy of approximations:
   * **Hartree-Fock (HF):** The starting point. You must understand this first.
   * **Density Functional Theory (DFT):** The workhorse of modern computational chemistry. You'll learn about different **functionals** (e.g., B3LYP, ωB97X-D) which are the "recipes" for calculating the energy.
   * **Post-Hartree-Fock Methods:** More accurate (and expensive) methods like **MP2** or **CCSD(T)** (the "gold standard") for when DFT isn't good enough.

### How to Start with PSI4: A Learning Path

The goal is to start simple and automate the process, which is where your Python skills will shine.

#### Phase 1: Installation and Absolute Basics

1. **Installation:** The easiest way is via Conda.

bash

conda create -n psi4env python=3.10

conda activate psi4env

conda install psi4 -c psi4

1. **Run a Simple Input File:** PSI4 traditionally uses input files. Create a file called h2o\_energy.dat:

python

*# h2o\_energy.dat*

memory 1 GB

molecule h2o {

O

H 1 0.96

H 1 0.96 2 104.5

}

set basis sto-3g

set scf\_type pk

energy('scf')

Run it from the terminal: psi4 h2o\_energy.dat. This calculates the Hartree-Fock energy of a water molecule with a minimal basis set. Congrats, you've done your first QC calculation!

#### Phase 2: Integrate with Python (The Powerful Part)

PSI4 has a superb Python API. This is where you can build scripts, analyze results, and create automated workflows. **This is the skill to showcase in your portfolio.**

Create a Python script instead of a .dat file.

**Project 1: The Molecular Energy Calculator**

python

*# molecule\_energy.py*

import psi4

import matplotlib.pyplot as plt

*# Initialize a list of molecules and their names*

molecules = {

"Water": "O; H 1 0.96; H 1 0.96 2 104.5",

"Methane": """

C

H 1 1.09

H 1 1.09 2 109.5

H 1 1.09 2 109.5 3 120

H 1 1.09 2 109.5 3 240

""",

"Ethylene": "C; C 1 1.33; H 1 1.09 2 121.5; H 1 1.09 2 121.5 3 180; H 2 1.09 1 121.5 3 180; H 2 1.09 1 121.5 3 0"

}

*# Set the calculation method*

method = 'scf/sto-3g' *# Hartree-Fock with a small basis set*

energies = {}

*# Calculate the energy for each molecule*

for name, geom in molecules.items():

mol = psi4.geometry(geom)

e = psi4.energy(method, molecule=mol)

energies[name] = e

print(f"{name} Energy: {e:.8f} Hartree")

*# (Optional) Plot the results*

plt.bar(energies.keys(), energies.values())

plt.ylabel('Energy (Hartree)')

plt.title('HF/STO-3G Energies of Small Molecules')

plt.xticks(rotation=45)

plt.tight\_layout()

plt.savefig('molecule\_energies.png')

**What you learn:** How to set up a molecule, run a basic calculation, and handle output programmatically.

#### Phase 3: Dive into DFT and Basis Sets

**Project 2: Benchmarking DFT Functionals**  
A classic project is to see how different functionals perform at a simple task, like calculating the bond length of a molecule.

python

*# dft\_benchmark.py*

import psi4

import numpy as np

*# Define the molecule (Carbon Monoxide)*

mol = psi4.geometry("""

C

O 1 {r}

symmetry c1

""")

*# Define a list of functionals to test*

functionals = ['B3LYP', 'WPBE', 'wB97X-D']

basis = 'cc-pVDZ' *# A better basis set*

*# Range of bond lengths to scan (in Angstrom)*

r\_vals = np.linspace(1.0, 1.3, 20)

*# Dictionary to store results*

results = {func: [] for func in functionals}

*# For each functional, calculate energy at different bond lengths*

for r in r\_vals:

mol.set\_variable('r', r) *# Update the bond length*

mol.update\_geometry()

for func in functionals:

e = psi4.energy(f'{func}/{basis}', molecule=mol)

results[func].append(e)

*# Now find the minimum energy and corresponding bond length for each functional*

for func, energy\_list in results.items():

min\_index = np.argmin(energy\_list)

min\_energy = energy\_list[min\_index]

min\_r = r\_vals[min\_index]

print(f"{func}: Optimal R = {min\_r:.4f} Å, Energy = {min\_energy:.8f} H")

**What you learn:** The core concept of a **potential energy surface**, how to perform a geometry scan, and that different DFT functionals give slightly different answers.

#### Phase 4: Portfolio-Worthy Projects

1. **Calculate Reaction Energies:**
   * **Concept:** Calculate the energy change of a chemical reaction, e.g., CH4 -> CH3 + H. This requires calculating the energy of the reactants and products separately.
   * **Why it's great:** This is a real application of QC. You can compare your calculated reaction energy to experimental values from databases like the NIST Chemistry WebBook.
2. **Calculate Vibrational Frequencies:**
   * **Concept:** Use psi4.frequency('scf/6-31g\*') to compute the IR spectrum of a molecule like CO₂. You'll get the frequencies and intensities of its vibrations.
   * **Why it's great:** It's highly visual. You can plot a simulated IR spectrum and assign peaks to specific molecular vibrations (bending, stretching).
3. **Intermolecular Interaction Energy (e.g., a Water Dimer):**
   * **Concept:** Calculate the strength of the hydrogen bond between two water molecules. This requires using a **high-level method** like CCSD(T) with a large basis set as a benchmark and comparing it to cheaper DFT methods.
   * **Why it's great:** It shows you understand a critical concept in chemistry (non-covalent interactions) and the importance of method selection. You can create a beautiful table or plot comparing methods.

### How to Present This in Your Portfolio

* **GitHub is Key:** Create a repo for each substantial project.
* **Jupyter Notebooks are Perfect:** Use them to combine your Python code, detailed explanations (in Markdown cells), and the beautiful plots/result tables you generate.
* **Explain the "Why":** For each project, write a brief introduction:
  + \*"This project benchmarks three common DFT functionals (B3LYP, WPBE, wB97X-D) to determine which most accurately predicts the bond length of carbon monoxide compared to high-level experimental data."\*
* **Visualize Results:** Always include plots (energy curves, spectra, molecular structures drawn with psi4.plotting or a tool like nglview).