ChatGPT

7-Day Roadmap: ML for Chemistry, Quantum Chemistry & Quantum Computing

This 7-day workshop-style plan is designed to rapidly build hands-on experience in applying machine learning (ML) to chemistry, with an emphasis on molecular modeling, quantum chemistry, and an introduction to quantum computing for drug discovery. Each day includes a **learning track** (theory/tutorials) and a **project track** (practical notebooks or coding tasks), totaling about 8 hours. By the end of the week, you should have a working familiarity with deep learning techniques (GNNs, transformers, generative models) for molecular data, classical molecular modeling tools (docking, force fields), quantum chemistry methods (DFT and electronic structure), and quantum computing approaches (variational algorithms) relevant to chemistry. This will prepare you for a postdoc role in a lab like St. Jude's QAI4Bio that integrates **AI, quantum chemistry, and quantum computing** for ligand and drug discovery.

Why these domains? Modern drug discovery leverages ML to predict molecular properties and interactions, drastically accelerating workflows [1]. However, traditional molecular representations often ignore quantum-mechanical details, which are crucial for accuracy in complex tasks [2]. *Quantum chemistry* methods (like DFT) provide those details by simulating electronic structure, while *quantum computing* promises to eventually tackle molecular simulations beyond classical computing limits. In fact, combining quantum computing with ML is already showing promise – for example, augmenting an ML model with quantum computing helped researchers identify drug candidates for an "undruggable" target, outperforming classical methods [3] [4]. This roadmap emphasizes practical skills in molecular ML and quantum chemistry, with a glimpse into quantum computing applications, ensuring you can confidently contribute to interdisciplinary projects at the postdoc level.

Overview of the 7-Day Plan

- **Day 1 – ML & Cheminformatics Foundations:** Set up your environment and refresh chemistry ML basics. Learn how molecules are represented digitally (SMILES, graphs, descriptors) and do a simple property prediction exercise.
- **Day 2 – Deep Learning for Molecules (GNNs & Generative Models):** Dive into graph neural networks and transformers for chemical data. Build a graph convolution model for a molecular dataset, and explore a generative model (e.g. molecule generation with a GAN or transformer).
- **Day 3 – Molecular Modeling and Docking:** Learn about molecular mechanics and docking. Practice preparing protein/ligand structures and run a docking experiment (e.g. using AutoDock Vina/Gnina) to predict binding poses.
- **Day 4 – Quantum Chemistry Basics:** Study key quantum chemistry concepts (Hartree–Fock, DFT, basis sets) and use an open-source quantum chemistry package (Psi4/PySCF) to compute molecular properties (e.g. geometry optimization, single-point energy).
- **Day 5 – Quantum Chemistry + Machine Learning:** Apply ML to quantum chemistry data. For example, use a dataset like QM9 (DFT-calculated molecular properties) to train a model that predicts a quantum property. Understand how ML can accelerate quantum chemistry calculations.

- **Day 6 – Quantum Computing for Chemistry:** Get an introduction to quantum computing principles and quantum machine learning (QML). Learn about variational algorithms (VQE) for molecular simulations. Run a simple quantum chemistry simulation (e.g. find H_2 molecule's ground-state energy using Qiskit).
- **Day 7 – Integration and Advanced Topics:** Consolidate what you've learned. Explore advanced topics or recent research (e.g. read a paper from QAI4Bio or similar labs). Optionally, work on a mini-project that strings together multiple components (e.g. generate molecules, dock them, evaluate with a quantum chemistry method). Prepare next steps for continued learning.

Each day is structured with **Learning Track (theory, tutorials)** and **Project Work (hands-on)** sections. Open-access resources (tutorials, docs, GitHub repos) are provided for each topic for deeper exploration. Let's dive in!

Day 1: ML & Cheminformatics Foundations

Learning Track (4 hours): Begin by setting up your Python environment with key libraries: **RDKit** (cheminformatics toolkit), **DeepChem** (deep learning for molecules), and frameworks like PyTorch or TensorFlow for building models. *DeepChem* in particular will be useful throughout this week – it's an open-source toolkit that "aims to democratize deep-learning in drug discovery, materials science, quantum chemistry, and biology" [5].

- **Molecular Representations:** Review how molecules are represented in silico. Key formats include **SMILES strings** (text encoding of molecular structure), **molecular graphs** (atoms as nodes, bonds as edges), and **fingerprints/descriptors** (numerical feature vectors). A recent MolSSI workshop "*Python for Cheminformatics-Driven Molecular Docking*" provides an excellent introduction to digital molecular representations [6] – see Notebook 1 of that series for a tutorial on SMILES, 2D/3D structures, and calculating simple descriptors with RDKit. You should practice converting between representations (e.g., take a SMILES for a drug-like molecule, use RDKit to generate a 2D structure and some descriptors like Morgan fingerprints).

- **Chemistry Data and Datasets:** Familiarize yourself with common datasets in molecular ML (many are collected in the **MoleculeNet** benchmark suite). For example, the *ESOL* dataset (water solubility of compounds) or *Lipophilicity* could be good simple tasks. DeepChem's `molnet` module can fetch these easily. Also note more advanced sets like **QM9** (quantum-calculated properties) and **PDBbind** (protein-ligand binding affinities) for later in the week. Ensure you understand the task (regression vs classification) for each dataset. *MoleculeNet* is a great reference – it provides curated molecular datasets spanning physical chemistry and biophysics [7] [8].

- **Basic ML Pipeline:** With your data ready, review a simple ML workflow: splitting data, choosing features, training a model, and evaluating performance (RMSE, ROC-AUC, etc.). Since you have a strong ML background, try a quick experiment: use RDKit to compute molecular descriptors for a small dataset (e.g., ESOL) and train a scikit-learn model (like a Random Forest) to predict a property (solubility). This will refresh the basics of applying ML to chemical data (and highlight challenges like data distribution, evaluation metrics in chemistry).

Project Work (4 hours): Put theory into practice with a hands-on notebook:

- **Property Prediction with DeepChem:** Use a DeepChem tutorial or example to train a simple deep learning model on a molecular dataset. A good starting point is DeepChem's GraphConvolution model (a neural network that operates on molecular graphs). DeepChem's tutorial "Introduction to Graph Convolutions" notes that graph convnets are **"one of the most powerful deep learning tools for working with molecular data"** because molecules naturally form graph structures [9]. Follow a tutorial to implement a graph neural network that predicts a known property (for example, toxicity on the **Tox21** dataset or solubility on ESOL). DeepChem provides high-level API calls: you can load a dataset via `dc.molnet`, featurize molecules into graphs, and use `GraphConvModel` to train a model. Track your model's training (loss curves) and evaluate it on a test set.

- **Explore RDKit and Data Curation:** Simultaneously, practice some data handling skills. For instance, use **BindingDB** (a database of protein-ligand binding data) or another open source to find a small set of molecules and their activities. Clean the data (handle salts, remove duplicates) using RDKit. This can segue into Day 3's focus. (The MolSSI workshop's Notebook 2 covers *exploring BindingDB data with RDKit*, which you can skim or attempt if time permits [10]).

By the end of Day 1, you should have your environment ready, and you'll have built and evaluated a basic model on molecular data. You'll also be comfortable with how molecules are encoded for ML purposes. These foundations will be crucial for the more complex models and simulations later in the week.

Day 2: Deep Learning for Molecules (GNNs & Generative Models)

Learning Track (4 hours): Today, focus on advanced deep learning architectures used in drug discovery: **Graph Neural Networks, transformers, and generative models**.

- **Graph Neural Networks (GNNs):** Expand on yesterday's graph conv introduction. Read about message-passing neural networks and why they are well-suited for chemistry (they respect the graph structure and local chemical environments). A suggested read is the Medium article *"Building a Graph Convolutional Network for Molecular Property Prediction"* (Deshmukh, 2023), which conceptually compares GNNs with traditional chemistry models. Key takeaway: GNNs can learn complex structure–property relationships by iteratively aggregating information from neighboring atoms/bonds. Make sure you understand concepts like graph convolutions, message passing, and graph embeddings. Another resource is the DeepChem documentation and tutorials – for example, their GraphConv tutorial from Day 1 and others on interpreting model predictions. After reading, you should appreciate that GNNs have become a **workhorse for molecular ML**, achieving excellent accuracy on tasks like toxicity and activity prediction.

- **Transformers in Chemistry:** Next, explore how transformer models (the kind originally from NLP) are being applied to molecules. **Transformers** can treat molecular sequences (like SMILES strings) as a "chemical language" or even handle 3D coordinates with specialized attention mechanisms. Read a high-level blog such as *"Transformers in Drug Design"* [11] [12] or *Valence Labs' post on Transformers for Chemistry*. These explain that transformers, with their self-attention, can capture context in sequences of atoms or protein residues. For instance, **ChemBERTa** and **MolBERT** are transformer models trained on large chemical libraries to generate rich molecular embeddings. Key applications

of transformers include **molecular property prediction** (learn embeddings then fine-tune for tasks), **reaction prediction/retrosynthesis** (treating reactions as translation tasks), and **molecular generation** (sample new molecules by treating SMILES as a language). Even large language models are being adapted to design molecules. The takeaway: transformers offer powerful sequence modeling for chemistry, enabling **generation of new molecules, prediction of interactions, and multi-objective optimization** in drug design [13] [14].

- **Generative Models for Molecules:** Learn about common generative approaches: **variational autoencoders (VAEs)**, **generative adversarial networks (GANs)**, and emerging **diffusion models**. These models can create novel molecules with desired properties, which is key in ligand/drug discovery (virtual library generation). A notable example is **MolGAN**, a GAN that operates on molecular graphs. According to its introduction, MolGAN uses a GAN framework on graph data combined with a reinforcement learning objective to bias generation toward chemically desirable properties [15]. Also, familiarize yourself with the **Junction Tree VAE** (which generates molecules fragment by fragment to ensure validity) and recent **diffusion models** that have achieved state-of-the-art in molecule generation. Given time constraints, focus on understanding the **goals** (e.g. generate molecules that are drug-like or active against a target) and **constraints** (valid chemical structures) of these models rather than all mathematical details.

Project Work (4 hours): Implement and experiment with one discriminative model and one generative model:

- **GNN Hands-On – Property Prediction:** Using either **PyTorch Geometric** or DeepChem, train a more custom GNN on a dataset. For example, try a **Graph Attention Network (GAT)** or **message-passing neural network** on the **HIV dataset** (MoleculeNet) for molecular activity prediction. This will involve writing your own PyTorch dataset for molecules or leveraging DeepChem's API. Aim to understand the model's performance and what structural features it's learning. You can use a Weights & Biases integration or tensorboard to monitor training. By doing this, you reinforce GNN concepts in code.

- **Molecular Generation Experiment:** Pick a generative model to try. One accessible route is to use the **MolGAN** tutorial from DeepChem [15]. The DeepChem example trains MolGAN on a subset of the *Tox21* chemical dataset to generate new molecules optimized for certain properties. Follow that notebook: it will guide you through model setup, training, and then sampling new molecules from the generator. After training, generate a set of molecules and visualize a few (RDKit can depict molecules from generated adjacency matrices). Evaluate basic properties of generated molecules (are they realistic? what is their predicted property from the discriminator or an external predictor?). If MolGAN is too heavy to train from scratch, an alternative is to use a pre-trained model: for instance, download a pre-trained VAE for molecules (such as the one from the MOSES benchmark) and sample molecules from it. The goal is to get familiarity with the process of *creating new chemical structures with AI*.

By the end of Day 2, you'll have practical experience with **graph neural networks** and **generative modeling** in chemistry. You should grasp how transformers and GNNs complement each other (sequence-based vs graph-based views) and how generative models can propose novel compounds – all vital skills for modern AI-driven drug discovery.

Day 3: Molecular Modeling and Docking

Learning Track (4 hours): Today shifts to **classical molecular modeling**, focusing on molecular mechanics and **docking** – essential tools for evaluating ligand-target interactions.

- **Force Fields & Energy Minimization:** Start with a quick overview of molecular mechanics. Unlike quantum chemistry, which treats electrons explicitly, molecular mechanics uses empirical **force fields** (e.g., AMBER, OPLS) to model the energy of molecules (bonds, angles, van der Waals, etc.). Read a short introduction (for example, a chapter from a computational chemistry online resource) to understand terms like potential energy surface, local minima, and the role of force fields in modeling conformations. Practically, try a simple task: take a molecule (e.g., a drug candidate from Day 2's generation or a known ligand) and use RDKit or Open Babel to **energy-minimize** its 3D geometry with a force field (RDKit has UFF/MMFF94 implementations). This will give you an intuition of molecular geometry optimization at the classical level, which is much faster than quantum methods.

- **Molecular Docking Theory:** Next, focus on **docking**, which predicts how a small molecule (ligand) binds to a protein (receptor). Docking involves sampling ligand poses in the binding site and scoring them. The DeepChem documentation succinctly states that docking finds a "*binding pose*" where the small molecule fits in the protein's binding pocket [16]. Classic docking programs (AutoDock Vina, DOCK, etc.) use scoring functions to approximate binding affinity. Read the AutoDock Vina paper (Trott & Olson 2010) or a docking tutorial to understand the basics of the scoring function and search algorithm. Key concepts: **binding pocket**, **scoring function** (evaluating intermolecular interactions), and **pose ranking**. Note that docking is an approximation – it's fast but may not fully capture entropic effects or detailed physics, which is why more accurate methods (like free energy perturbation or quantum chemistry) might follow up in a drug discovery project.

- **Docking Tools and Data:** Identify open tools you can use. **AutoDock Vina** is free and reasonably easy to use; **Gnina** is a fork of Vina that incorporates CNN scoring (useful if available). Also, get a protein structure for testing – the **Protein Data Bank (PDB)** is the repository of crystal structures. You might choose a well-studied target like **acetylcholinesterase** or a kinase where many ligands are known. The MolSSI docking workshop (from Day 1) provides a step-by-step guide: Notebook 3 covers preparing structures (protein and ligand) for docking, and Notebook 4 demonstrates running a docking with Gnina [17]. Skim through their notes to understand the process: typically, one must add hydrogens to the protein, remove waters, define a grid box for docking, etc.

Project Work (4 hours): Perform a **docking experiment** to predict ligand binding:

- **Protein-Ligand Docking Practical:** Follow the MolSSI **"Molecular Docking using Gnina"** notebook or a similar tutorial [17]. Concretely:
- Select a protein target (e.g., use PDB ID `6VJJ` which is KRAS G12C with a ligand, or any target of interest). Retrieve the PDB file and identify the binding pocket (the co-crystallized ligand indicates where to dock).
- Prepare the protein using a tool like Open Babel or PyMOL (or RDKit for basic steps): remove nonessential molecules, add hydrogens, and save it in PDBQT format for AutoDock. Similarly, prepare a ligand (if you have a known binder or use one of your Day 2 generated molecules) by generating 3D coordinates and converting to PDBQT.

- Run a docking program: if using **AutoDock Vina**, set up a config (specify center of binding site and grid size) and run it to produce docked poses. If using **Gnina (via the provided Colab)**, you can simply run the notebook which automates many steps.

- Examine the results: the docking tool will output predicted poses and scores. Use a viewer (PyMOL or RDKit visualization) to see how the ligand is positioned in the pocket. Note the top score pose and the interactions (e.g., hydrogen bonds, hydrophobic contacts) it suggests.

- **Analysis of Docking Results:** Interpret the docking score (while keeping in mind it's in arbitrary units or rough binding energy estimates). If you docked multiple ligands (you could try docking a **small library** of, say, 5 molecules to compare), see which one was predicted best and hypothesize why. This is a good place to reflect on how reliable (or not) docking predictions are and how ML might improve this (e.g., ML-based scoring functions, or using docking poses as features for ML models).

By the end of Day 3, you will have practical knowledge of *molecular docking*: preparing real biological structures, using a docking engine, and analyzing the output. You'll also better appreciate how **molecular modeling** complements ML – for instance, docking can be used to label data for ML (binding vs non-binding), or ML can be used to fast-filter compounds before docking. This sets the stage for integrating physics-based and learning-based approaches in the coming days.

Day 4: Quantum Chemistry Basics

Learning Track (4 hours): The focus now shifts to **quantum chemistry** – understanding how to compute molecular electronic properties from first principles (important for accuracy and for generating training data for ML).

- **Quantum Chemistry Theory Refresher:** Since you have a chemistry background, spend the morning reviewing the foundations of electronic structure theory:
- **Hartree–Fock (HF):** the basic quantum chemical method that approximates the multi-electron Schrödinger equation by assuming a single Slater determinant (mean-field approach).
- **Density Functional Theory (DFT):** a widely used approach that includes electron correlation through functionals of electron density. Recall that DFT uses functionals like B3LYP, etc., and is usually more accurate than HF for chemistry while still relatively efficient.
- **Basis Sets:** e.g., STO-3G, 6-31G*, cc-pVDZ – functions used to represent atomic orbitals. Understand the trade-off: larger basis sets = higher accuracy but more computational cost.
- Key terms like *optimization (geometry optimization)* vs *single-point energy calculation*, *frequency calculation* (for IR/Raman spectra and thermochemistry), etc.

Use a resource like the free ebook chapters or a reputable online course in computational chemistry. The goal isn't to derive equations, but to know what these methods do and their inputs/outputs (e.g., you feed a molecular geometry, choose a method and basis, and get energies, orbitals, etc.). For instance, **MolSSI's Quantum Mechanics Tools** workshop modules demonstrate tasks like geometry optimization using Psi4 [18] – glancing through these can solidify concepts and show how theory connects to practice.

- **Setting Up Quantum Chem Tools:** Choose an open-source quantum chemistry package to install and use:

- **Psi4** (Python-integrated, user-friendly for beginners),
- or **PySCF** (Python library, great for customizing computations, widely used in ML contexts).

Spend time installing and testing one of these on your HPC environment. If using Psi4, test it with a simple input: e.g., compute the energy of a water molecule at HF/STO-3G to ensure it's working. Familiarize yourself with input structure (in Psi4 you can use a Python API, define molecule geometry, then call energy computations). A snippet from a MolSSI tutorial: to calculate an energy, you define a molecule with `psi4.geometry(...)` and then call a method with a basis set, e.g. HF with cc-pVDZ [19]. This should yield a single-point energy.

Project Work (4 hours): Apply quantum chemistry tools to a small problem:

- **Geometry Optimization and Property Calculation:** Select a small molecule relevant to drug discovery (e.g., a fragment or a small ligand like benzene, ethanol, or a candidate you generated earlier). Using Psi4 or PySCF:
- Perform a **geometry optimization** at a reasonably robust level of theory (for example, DFT with B3LYP/6-31G*). This will refine the molecular structure to a local minimum. Monitor the optimization steps and final energy. Save the optimized geometry.
- Compute an electronic property of interest. For instance, calculate the molecule's **dipole moment** or **HOMO-LUMO gap** using the optimized geometry (these often come for free in output, or can be obtained via an extra calculation or parsing the wavefunction). If time permits, you could also do a single-point energy at a higher level (e.g., MP2 or a higher basis) to see the difference.

This exercise gives hands-on experience with quantum chemistry software and a sense of the computational cost. Try using your HPC's parallel capability if supported by the software (Psi4 can utilize multiple threads for DFT).

- **Analysis and Link to Experiment/ML:** Reflect on the results. How does the optimized bond lengths or angles compare to expected values (maybe compare against an X-ray crystal data if available)? Note how much time the calculation took for this small molecule – it highlights why more complex molecules (with many atoms or requiring a protein environment) become infeasible, motivating the need for ML approximations or even quantum computing in the future. If you computed a property like dipole moment, consider how that property could be used in ML modeling (e.g., predicting polarity of molecules).

By the end of Day 4, you should be comfortable setting up and running basic quantum chemical calculations and interpreting their output. This skill is important for a role that requires understanding electronic structure (for example, computing accurate labels for an ML model, or evaluating a drug candidate's properties beyond what classical methods provide). You've also laid groundwork to use quantum data for machine learning on Day 5.

Day 5: Quantum Chemistry + Machine Learning

Learning Track (4 hours): Today you will bridge quantum chemistry and machine learning – a critical synergy in modern computational chemistry (sometimes called **quantum machine learning** in the context of using ML to replicate QC calculations). The idea is to use ML to **learn patterns from quantum data**,

achieving (partial) quantum accuracy with far less computation. This is especially relevant for the QAI4Bio lab's mission of deep learning for electronic structure.

- **ML on Quantum Data – Concepts:** Read about how ML models have been used to emulate quantum chemistry. A classic example is the **QM9 dataset** – ~134k small molecules with DFT-calculated properties (energies, enthalpies, HOMO/LUMO energies, dipole moments, etc.) [8] . Models like **SchNet, PhysNet, MPGNNs** have been trained on QM9 to predict these quantum properties with high accuracy, effectively reproducing DFT results instantaneously. Skim a reference describing one of these models (e.g., SchNet paper by Schütt et al. 2017, or a blog summary of it) to understand how they incorporate physical knowledge (SchNet uses continuous-filter convolutions and is aware of spatial geometry). Note: these models treat atoms as nodes with distances – bridging the gap between graph networks and physics.

Also be aware of techniques like **Δ-learning (Delta learning)**: where an ML model learns to predict the difference between a cheap method and an expensive method (e.g., learn the correlation energy missing from Hartree-Fock to reach DFT accuracy). This approach is used to systematically improve cheaper calculations with ML. It could be something you encounter in a research context.

- **Quantum Chemistry in Model Inputs:** Learn how quantum chemical information can augment molecular representations for ML. For example, the CMU study (Gomes *et al.*, 2025) introduced a representation that includes **quantum-chemical interactions** in molecular graphs to improve ML predictions [2] . The idea is that by encoding orbitals or partial charges (from quantum calcs) into features, ML models can achieve better accuracy on challenging prediction tasks. Keep this in mind: one way your role might use quantum chemistry is to generate richer features for deep learning models or to serve as a source of ground truth for training data.

- **Tools and Libraries:** Familiarize with libraries that combine QC and ML. DeepChem has some capabilities to load QM9 and train models; also, **ASE (Atomic Simulation Environment)** with ML potentials (like ASE's interface to ML force fields) might be relevant if doing materials or drug design (force-field replacement) work. There's also an emerging library **QML** (by Montavon et al.) for kernel-based learning of quantum properties, and libraries like **TorchANI** (for neural network potentials). While you may not use these today, be aware of their existence. It underscores the trend that **quantum data + ML** is a hot area.

Project Work (4 hours): Conduct a mini-project to apply ML on quantum chemistry data:

- **Train a Model on QM9:** Use DeepChem or PyTorch to train a model on a subset of the QM9 dataset. For example, predict the **molecular dipole moment** or **heat of formation (energy)** of molecules. DeepChem can load QM9 via `dc.molnet.load_qm9()`, giving you a dataset of molecules (likely represented by Coulomb matrices or you can switch to graph featurization). If using DeepChem's built-in models, try a **SchNetModel** or **MPNNModel** which are designed for such data. Alternatively, if you prefer PyTorch Geometric or DGL, you could use their example of QM9 (they often have a built-in small dataset loader and a model example). Train the model (leveraging your GPU resources for speed). Monitor the training and see how close you can get to chemical accuracy (for context, chemical accuracy is about 1 kcal/mol $\approx$ 0.043 eV error for energy predictions).

- **Evaluate and Interpret:** Once trained, evaluate the model on a validation set. Calculate metrics like MAE (mean absolute error) which is common for energy prediction. You might find that your model can predict, say, heats of formation within a few kcal/mol of DFT values – pretty good! This demonstrates how ML can replicate quantum results. If you've integrated a physics-based model like SchNet, reflect on how it's leveraging pairwise distances and physics inspired features. If time allows, test the model on a few *new* molecules (e.g., draw a simple molecule not in the training set, compute its properties with your model and compare to a quick Psi4 calculation to see if they agree).

- **Optional - Delta-ML experiment:** If you are curious and have time, a quick experiment: perform a cheap calculation (like semi-empirical PM6 or HF/STO-3G) for a small set of molecules, and use your trained model to correct it to approximate a higher level (DFT). For instance, take 20 molecules, compute HF energies and compare with your ML-predicted DFT energies. This mimics a Δ-learning approach in a simplistic way and shows how ML and QC can integrate (with ML filling the accuracy gap).

By the end of Day 5, you'll have **experience using ML to learn from quantum chemistry data**. You should appreciate that ML can drastically speed up predictions (once trained) and how quantum-accurate data can improve ML models. This is highly relevant to projects where you can't run expensive QC for every candidate molecule and need an ML surrogate. You have effectively practiced a core aspect of the postdoc role: combining deep learning with quantum chemistry insights.

Day 6: Quantum Computing for Chemistry

Learning Track (4 hours): Today is an introduction to **quantum computing (QC)**, specifically as it applies to chemistry. Since QC is a vast field, we will focus on the essentials needed to understand research like QAI4Bio's and to start tinkering with quantum algorithms for molecules.

- **Quantum Computing Basics:** Review the fundamentals of how quantum computers work: qubits (quantum bits) instead of bits, superposition (qubits can be in a combination of 0 and 1), entanglement (qubits can be correlated in ways classical bits cannot), and interference. A good beginner-friendly explanation is to picture qubits on a Bloch sphere (states as vectors) and how operations (gates) rotate these states. IBM's online Qiskit textbook or YouTube tutorials by PennyLane provide accessible introductions. You don't need to become an expert in quantum gates, but understand key ones like X, Z, H (Hadamard), CNOT, and rotation gates, since these often appear in quantum algorithms.

- **Variational Quantum Algorithms (VQAs):** The most relevant quantum computing method for chemistry in the NISQ (noisy intermediate-scale quantum) era is the **Variational Quantum Eigensolver (VQE)**. *Why VQE?* Because it's designed to find minimum eigenvalues of a Hamiltonian – essentially solving for molecular ground state energies, which is a quantum chemistry problem. Go through a conceptual explanation of VQE. For example, the Medium article *"Simulating the Hydrogen Molecule (H_2) with VQE in Qiskit"* (Sekar, 2025) provides a quick overview [20]. It explains: we prepare a parameterized quantum state (an **ansatz** circuit with adjustable parameters θ), measure its energy with respect to the molecular Hamiltonian, and use a classical optimizer to tweak θ to minimize the energy. In essence, **VQE is a hybrid algorithm** – quantum hardware evaluates the energy for a given state, and a classical routine adjusts the state parameters [21]. The end result approximates the ground-state energy of the molecule.

Ensure you grasp terms like **Hamiltonian** (in chemistry, often expressed in second-quantized form with Pauli matrices), **ansatz** (a guessed form of wavefunction circuit, e.g. UCC – Unitary Coupled Cluster – ansatz is popular), and the role of a **classical optimizer**. Also recognize limitations: current quantum hardware is noisy and limited in qubits, so these simulations are mostly for very small molecules (H_2, LiH, BeH_2, etc.) as proofs of concept.

- **Quantum Machine Learning (QML):** Beyond VQE, note that quantum computing can also intersect with machine learning in other ways, like quantum classifiers or quantum kernels. Given the time, briefly read about one example (e.g., *variational quantum classifiers* or *quantum kernel methods*) just to know the buzzwords. However, the priority for chemistry is quantum simulation. Another concept is **QAOA (Quantum Approximate Optimization Algorithm)** for optimization problems, but for drug discovery the direct app is less clear than VQE. Keep in mind QML is still very nascent – as the St. Jude article noted, it's an exciting frontier to *augment ML with quantum computing* for possibly better results [22].

- **Tools: Qiskit and PennyLane:** Install a quantum computing development library. **Qiskit** (by IBM) is a comprehensive framework which includes chemistry modules (Qiskit Nature). **PennyLane** (by Xanadu) is another that is very user-friendly for creating quantum ML models and has plugins for chemistry. For learning purposes, Qiskit has extensive documentation and an open textbook; PennyLane has many example notebooks. Choose one and skim how to define a simple circuit and run a simulation. For instance, look at how to create a 2-qubit circuit in Qiskit, or how PennyLane's `qml.Hamiltonian` can define a molecular Hamiltonian from an electronic structure problem. Installing Qiskit might take a bit, but it's straightforward with pip and it comes with Aer (simulator backend), which you will use.

Project Work (4 hours): Run a basic quantum computing simulation for a chemistry problem:

- **Simulate H_2 with VQE (Quantum Chemistry on a Quantum Simulator):** Following the example from earlier, perform a VQE for the hydrogen molecule. You can use Qiskit's tutorial or code. Qiskit Nature, for example, can generate the qubit Hamiltonian for H_2 automatically and even provide a pre-built ansatz. But to learn, you might try a more manual approach:
- Define the problem: H_2 has two electrons. Use Qiskit Nature or an electronic structure package to get the Hamiltonian (this involves mapping molecular orbitals to qubits via techniques like Jordan-Wigner transformation – the tutorial covers this so you don't have to derive it).
- Choose an ansatz circuit, like a two-qubit UCC ansatz or even a simpler heuristic ansatz (a few rotation gates and an entangler).
- Use Qiskit's VQE routine: it will require the ansatz, an initial set of parameters, an optimizer (e.g., COBYLA or SLSQP), and a quantum backend (use the statevector simulator or QASM simulator with a few shots for realism).
- Run the VQE optimization to find the minimum energy. Monitor the convergence (some tutorials print intermediate energies). You should get a result close to the known ground energy of H_2 (around -1.14 hartree for the chosen bond distance).

The Medium blog mentioned earlier provides code and explanation for this [20], and it links to a GitHub repository with the full code if you need reference. By actually coding or running the notebook yourself, you'll get a feel for how quantum algorithms are implemented and how they differ from classical code.

- **Analyze Quantum Results:** Once VQE finishes, examine the output: the estimated ground-state energy. Compare it to a classical calculation (you could quickly run a Psi4 calculation for H_2 at a similar geometry to see the expected exact energy). If there's a discrepancy, note sources of error (e.g., ansatz may be too simple, or optimizer got stuck in a local minimum, or simulation noise if you added it). This teaches you about the challenges in quantum computing. Also, note how many iterations and evaluations it took – this is essentially the "cost" of the hybrid algorithm.

- **(Optional)** If you are curious, you can also try a tiny **quantum machine learning** task: for example, PennyLane has a tutorial where a quantum circuit learns to classify simple data. This is optional, but it can be eye-opening to see a quantum neural network in action. Another quick experiment could be using a quantum kernel in an SVM for a very small dataset (Qiskit has demos on that). These are more ML-focused uses of QC, whereas VQE was physics-focused.

By the end of Day 6, you will have a basic understanding of **quantum computing for chemistry** and firsthand experience running a quantum algorithm (on a simulator). You should be able to discuss variational algorithms and their potential in drug discovery – for instance, how one might use a quantum computer to evaluate certain molecular properties or screen compounds in ways classical computers struggle with. This knowledge is cutting-edge and will let you engage in discussions and projects that aim to use quantum advantage in the future of drug discovery.

Day 7: Integration and Advanced Topics

Learning Track (4 hours): The final day is about synthesis and looking forward. Use the morning to fill any knowledge gaps and to explore advanced resources that tie everything together.

- **Interdisciplinary Case Studies:** Read one or two recent research articles that exemplify the integration of deep learning, quantum chemistry, and possibly quantum computing in drug discovery. A top pick is the St. Jude study mentioned earlier: the Nature paper where Gorgulla *et al.* used quantum computing alongside ML for drug discovery. The St. Jude press release highlights that *"by augmenting our machine learning model with quantum computing, we outperformed similar purely classical models in identifying promising compounds"* [4]. Try to get the main points of the paper: what was the workflow? (It likely involved generating or screening compounds with ML, then using a quantum algorithm to refine the results or explore chemical space, followed by experimental validation.) Another example is the Nature Machine Intelligence 2025 paper by Gomes *et al.* on quantum-informed molecular representations [2] – skim the abstract/introduction to see how they infused quantum insights into ML to improve predictions. These readings will show you how the pieces you learned fit together in cutting-edge research.

- **Emerging Tools and Trends:** Take note of any tools you haven't tried yet but might want to in future:

- **Docking + AI combos:** e.g., **MASIF** (ML for protein-ligand binding sites) or **DiffDock** (diffusion model for docking poses). These might be relevant as an evolution of classical docking.

- **Automated pipelines:** Platforms like **DeepChem** or **OpenChem** can integrate steps (featurization, model, evaluation) – explore their docs for any end-to-end drug discovery pipeline examples.
- **Quantum computing developments:** Keep an eye on libraries like Qiskit Nature (which might streamline what you did manually) and any increase in qubit counts that could handle bigger molecules soon.

- **Datasets and Challenges:** Check if there are public challenges or datasets released recently (for example, the Quantum Computing for Drug Discovery Challenge 2023). These often highlight current capabilities and limitations.

- **Plan Next Steps:** Since a postdoc is about doing new research, think about what areas you want to deepen. Identify a few topics or skills to pursue after this week:

- For instance, **Molecular dynamics simulations** (using tools like OpenMM or GROMACS) to complement docking with flexibility and thermodynamics.
- **Advanced generative models** like reinforcement learning for drug design (e.g., using a package like REINVENT for goal-directed generation).
- **Hardware experience**: if possible, running a job on an actual quantum processor (IBM offers cloud access to small quantum chips) – this could be a valuable experience to mention.
- Keep a list of references: e.g., the **DeepChem book** (open source) for deep learning, and perhaps a standard textbook for quantum chemistry (like Szabo & Ostlund or Jensen's book) for deeper theory when needed.

Project Work (4 hours): Use the afternoon to **integrate and demonstrate** what you've learned in a mini-project or summary output:

- **Mini-Project – AI-Driven Drug Discovery Pipeline:** Design a simple pipeline that goes from molecular design to evaluation:
- **Generate** a small library of molecules (perhaps using your Day 2 generative model or even manually picking some diverse molecules).
- **Predict** a property for them using your Day 5 ML model (for example, predict which might have a high dipole moment or desired quantum property, or use a predictive model like a toxicity or activity model if you built one).
- **Dock** the top candidates to a protein target of interest (from Day 3's skills) to evaluate their binding potential.
- **Refine** one candidate with a quantum chemistry single-point energy or optimization (from Day 4 skills) to get a more accurate assessment of properties like binding pocket partial charges or interaction energy in a small cluster model.

This pipeline is primitive and won't be fully automated in a day, but even doing it for one example ligand is instructive. Document each step – this could even form the basis of a report or presentation to demonstrate your interdisciplinary skill set. For instance, you could take a known drug molecule, modify it (generate analogs), predict which analog might be more polar (via ML) and still bind, then dock it, and finally compute a quantum-based score for its binding site interaction (e.g., optimize a ligand–sidechain complex at low level theory to estimate interaction energy).

- **Knowledge Consolidation:** As you work, consolidate your findings and learning. It might help to write a short summary of each day's key outcomes and how it applies to the QAI4Bio postdoc role.

This could be in the form of a blog post or an outline for an interview presentation. Emphasize how **deep learning models (Day 1-2)** can drastically speed up predictions in drug discovery, how **molecular modeling (Day 3)** provides physical insight and validation, how **quantum chemistry (Day 4-5)** offers accuracy and data for training AI, and how **quantum computing (Day 6)** is an emerging tool that could eventually handle problems intractable to classical methods. This synthesis will solidify your readiness and give you talking points to discuss in group meetings or interviews.

By the end of Day 7, you will have tied together the threads of the week. You'll be able to see the "big picture" of an AI-driven drug discovery project: from generating ideas with ML, validating them with physics-based methods, and even envisioning quantum computing in the loop for the hardest problems. You should feel confident in your ability to contribute to an interdisciplinary research project at the intersection of **deep learning, quantum chemistry, and quantum computing**.

Finally, keep this momentum going: the fields of AI in chemistry and quantum computing are rapidly evolving. Continue to leverage open resources (many we used are free and community-driven) and maybe even contribute to them. Good luck with your postdoctoral journey, and enjoy pushing the boundaries of what's possible in computational drug discovery!

Sources:

- St. Jude Research News – *Quantum computing makes waves in drug discovery* [3] [4]
- CMU News – *Enhancing Molecular ML with Quantum-Chemical Insight* [1] [2]
- DeepChem Library (open-source toolkit for deep learning in drug discovery) [5]
- MolSSI Workshop – *Python for Cheminformatics-Driven Molecular Docking* (Notebooks on molecular representations and docking) [6] [17]
- DeepChem Tutorial – *Introduction to Graph Convolutions* [9]
- SilicoGene Blog – *Transformers in Drug Design* (role of transformers in molecule generation and prediction) [13] [14]
- DeepChem Tutorial – *Generating Molecules with MolGAN* (GAN-based molecular generation) [15]
- DeepChem Docs – *Docking* (overview and Autodock Vina reference) [16]
- MolSSI Quantum Mechanics Tools – *Geometry Optimization with Psi4* (Psi4 example objectives) [18] [19]
- DeepChem MoleculeNet – description of QM9 quantum chemistry dataset [8]
- Medium Blog – *Simulating H_2 with VQE in Qiskit* (quantum computing in chemistry and VQE intro) [20] [21]

---

[1] [2] Enhancing Molecular Machine Learning with Quantum-Chemical Insight - Mellon College of Science - Carnegie Mellon University
https://www.cmu.edu/mcs/news-events/2025/0605_machine-learning-quantum-chemical-insight.html

[3] [4] [22] Quantum computing makes waves in drug discovery | St. Jude Research
https://www.stjude.org/research/progress/2025/quantum-computing-makes-waves-in-drug-discovery.html

[5] GitHub - deepchem/deepchem: Democratizing Deep-Learning for Drug Discovery, Quantum Chemistry, Materials Science and Biology
https://github.com/deepchem/deepchem

6 10 17 GitHub - MolSSI-Education/iqb-2025: This repository contains notebooks for the workshop Python for Cheminformatics-Driven Molecular Docking held on May 1, 2025 in collaboration with the RCSB Protein Data Bank.
https://github.com/MolSSI-Education/iqb-2025

7 8 MoleculeNet — deepchem 2.8.1.dev documentation
https://deepchem.readthedocs.io/en/latest/api_reference/moleculenet.html

9 DeepChem
https://deepchem.io/tutorials/introduction-to-graph-convolutions/

11 12 13 14 Transformers in Drug Design - SilicoGene
https://silicogene.com/blog/transformers-in-drug-design/

15 DeepChem
https://deepchem.io/tutorials/generating-molecules-with-molgan/

16 Docking — deepchem 2.8.1.dev documentation
https://deepchem.readthedocs.io/en/latest/api_reference/docking.html

18 19 Quantum Mechanics Tools: Geometry optimization
https://education.molssi.org/qm-tools/01-geom-opt/index.html

20 21 Simulating the Hydrogen Molecule (H_2) with VQE in Qiskit | by Anirudh Sekar | May, 2025 | Medium
https://medium.com/@anirudhsekar2008/simulating-the-hydrogen-molecule-h%E2%82%82-with-vqe-in-qiskit-ee5f6fe35299