This code is creating a **simple web app** using **Streamlit** that allows users to enter a **SMILES string** (a text-based representation of a chemical molecule) and **see its structure as an image**.

**Step-by-Step Explanation:**

1. **Setting Up the App** 🏗️
   * The app **displays a title**: *"🧬 Molecular Visualization App"*.
   * It shows a small instruction: *"Enter a SMILES string to visualize the molecule!"*
2. **User Input (SMILES Code)** 🧪
   * The user **types a SMILES string** in a text box.
   * Example:
     + "CCO" (for ethanol, alcohol).
     + "C1=CC=CC=C1" (for benzene).
3. **Converting the SMILES Code to a Molecule** 🔄
   * The app **uses RDKit** to convert the SMILES text into a chemical structure.
   * If the conversion works, it **draws the molecule as an image**.
4. **Displaying the Molecular Image** 🖼️
   * The molecule's **2D structure** is **displayed on the screen**.
   * If the user enters an **incorrect SMILES code**, the app shows an error message:
     + ❌ *"Invalid SMILES string. Please try again."*
5. **Handling Errors Gracefully** ⚠️
   * If something goes wrong, the app **displays an error message** instead of crashing.

**What is this app useful for?**

* **Chemists & Students** can quickly **visualize molecules** by entering their SMILES codes.
* **Researchers in Drug Discovery** can **explore and validate** chemical structures before running experiments.
* **Beginners in Computational Chemistry** can **learn about molecular representations interactively**.

**Basic Organic Molecules:**

1. **Water** (H₂O) → O
2. **Methane** (CH₄) → C
3. **Ethanol** (Alcohol, C₂H₆O) → CCO
4. **Acetone** (Nail polish remover, C₃H₆O) → CC(=O)C
5. **Benzene** (Aromatic ring, C₆H₆) → c1ccccc1

**Medicinal & Drug Molecules:**

1. **Aspirin** (Pain reliever) → CC(=O)OC1=CC=CC=C1C(=O)O
2. **Caffeine** (Stimulant) → CN1C=NC2=C1C(=O)N(C(=O)N2C)
3. **Paracetamol** (Tylenol) → CC(=O)NC1=CC=C(C=C1)O
4. **Ibuprofen** (Painkiller) → CC(C)CC1=CC=C(C=C1)C(C)C(=O)O
5. **Morphine** (Pain medication) → CN1CC[C@]23C4=CC5=C(C=C4O)O[C@@H]2[C@H]1CC[C@@]3(O5)C6=CC=CC=C6O

**Interesting & Unique Molecules:**

1. **Glucose** (Sugar) → C(C1C(C(C(C(O1)O)O)O)O)O
2. **Cholesterol** (Lipid) → CC(C)CCCC(C)C1CCC2C3CCC4=CC(O)CCC4(C)C3CCC12C
3. **Serotonin** (Neurotransmitter) → CC(=O)NC1=CC=C(C=C1)O
4. **Capsaicin** (Spicy molecule in chili) → CC(C)CCNC(=O)C1=CC=C(C=C1)OC1=CC=C(C=C1)O
5. **THC** (Active compound in cannabis) → CC1=C(C2=C(C3=C(O1)C=C(O)C=C3)C(CCCCC)=C2)O

MOLProp2.py

This code creates a **web app** using **Streamlit** that allows users to:  
✅ **Enter a molecule's SMILES code**  
✅ **See its structure as an image**  
✅ **Calculate important molecular properties** (such as molecular weight, hydrophobicity, and hydrogen bonding)

**Step-by-Step Explanation:**

1. **App Setup & Title Display** 🏗️
   * The app displays a **title**: *"🧬 Molecular Property Calculator with Visualization"*.
2. **User Input (SMILES Code)** 🧪
   * The user **types a SMILES string** in a text box.
   * Example Inputs:
     + "CCO" → **Ethanol**
     + "C1=CC=CC=C1" → **Benzene**
3. **Convert the SMILES Code to a Molecule** 🔄
   * Uses **RDKit** to **convert the text-based SMILES representation into a molecular structure**.
4. **Calculate & Display Molecular Properties** 📊
   * If the molecule is valid, the app **calculates key properties**:
     + **Molecular Weight** (size of the molecule in grams per mole)
     + **LogP (Partition Coefficient)** (measures how well the molecule dissolves in fat vs. water)
     + **Hydrogen Donors** (how many hydrogen atoms the molecule can give in bonding)
     + **Hydrogen Acceptors** (how many hydrogen atoms the molecule can attract in bonding)
5. **Visualize the Molecular Structure** 🖼️
   * The app **draws the molecule’s 2D structure** and displays it as an image.
6. **Error Handling** ⚠️
   * If the SMILES string is incorrect, the app **shows an error message**:
     + ❌ *"Invalid SMILES string! Please enter a valid one."*

**What is this app useful for?**

* **Chemists & Students** 🧑‍🔬 can quickly analyze molecular properties.
* **Drug Discovery Research** 💊 helps determine if a molecule is a good candidate for medicine.
* **Chemical Informatics** 📈 useful for predicting how molecules behave in different environments.

DRUGLIKENESS

This code creates a **web app** using **Streamlit** that helps users determine if a molecule is likely to be a **drug-like compound** based on its chemical properties.

**Step-by-Step Explanation:**

1. **App Setup & Title** 🏗️
   * The app displays a **title**: *"💊 Drug-Likeness Predictor with Visualization"*
2. **User Input (SMILES Code)** 🧪
   * The user **enters a SMILES string** (a text representation of a molecule).
   * Example Inputs:
     + "CCO" → **Ethanol**
     + "CC(C)CC1=CC=C(C=C1)C(C)C(=O)O" → **Ibuprofen**
3. **Convert SMILES Code to a Molecule** 🔄
   * Uses **RDKit** to convert the text into a molecular structure.
4. **Visualize the Molecular Structure** 🖼️
   * The app **draws the 2D structure** of the molecule and displays it.
5. **Calculate Key Molecular Properties** 📊
   * The app extracts **four important molecular properties**:
     + **Molecular Weight** (size of the molecule in grams per mole)
     + **LogP (Partition Coefficient)** (measures fat solubility)
     + **Hydrogen Bond Donors** (atoms that donate hydrogen in bonding)
     + **Hydrogen Bond Acceptors** (atoms that attract hydrogen in bonding)
6. **Predict Drug-Likeness** 💊
   * Uses a **simple rule**:
     + If **Molecular Weight < 500 g/mol**, the molecule is **"✅ Likely Drug-like"**
     + Otherwise, it’s **"🚫 Not Drug-like"**
   * This rule is based on **Lipinski's Rule of Five**, which helps determine if a molecule is a good candidate for an oral drug.
7. **Display Results** 🔎
   * The app **shows the drug-likeness prediction** and the **calculated properties**.
8. **Error Handling** ⚠️
   * If the user enters an **invalid SMILES string**, the app displays:
     + ❌ *"Invalid SMILES string! Please try again."*

**What is this app useful for?**

* **Chemists & Pharmacologists** 🧑‍🔬 can quickly check if a molecule follows basic drug-likeness rules.
* **Drug Discovery Researchers** 💊 can use it as a **fast pre-screening tool** for potential drug candidates.
* **Students & Learners** 📚 can interactively explore how chemical properties affect drug-likeness.

Trying

### ****1️⃣ Paracetamol (Acetaminophen) – Common Pain Reliever****

**SMILES:** CC(=O)NC1=CC=C(C=C1)O  
✅ Expected: **Likely Drug-like**

### ****2️⃣ Ibuprofen – Anti-inflammatory Drug****

**SMILES:** CC(C)CC1=CC=C(C=C1)C(C)C(=O)O  
✅ Expected: **Likely Drug-like**

### ****3️⃣ Caffeine – Stimulant in Coffee****

**SMILES:** CN1C=NC2=C1C(=O)N(C(=O)N2C)  
✅ Expected: **Likely Drug-like**

### ****4️⃣ Cholesterol – Important Lipid (Not a Drug)****

**SMILES:** CC(C)CCCC(C)C1CCC2C3CCC4=CC(O)CCC4(C)C3CCC12C  
🚫 Expected: **Not Drug-like (High Molecular Weight)**

### ****5️⃣ Paclitaxel (Taxol) – Cancer Drug****

**SMILES:** CC1=C(C(=O)OC2C(CC3(C2C(C(=O)O1)O)OC4C(C3OC(C4O)(C5=CC=CC=C5)O)C)O)O  
🚫 Expected: **Not Drug-like (Very Large & Complex Molecule)**

This **web app** helps compare **two molecules** based on their **chemical similarity** using **fingerprint-based similarity scores**. It allows users to:  
✅ **Enter two SMILES (chemical structures in text form)**  
✅ **Generate molecular fingerprints (unique chemical signatures)**  
✅ **Compare how similar they are** using a **numerical similarity score** (between 0 and 1)  
✅ **Visually display both molecules side by side**

**Step-by-Step Explanation**

1. **App Setup & Title Display** 🏗️
   * The app **displays a title**: *"🧬 Molecular Similarity for Drug Screening"*
2. **User Input (SMILES Codes for Two Molecules)** 🧪
   * Users enter two **SMILES strings** (chemical representations).
   * Example Inputs:
     + "CCO" → **Ethanol**
     + "CCN" → **Ethylamine**
3. **Convert SMILES to Molecule Objects** 🔄
   * Uses **RDKit** to convert the text-based **SMILES representation** into molecular structures.
4. **Generate Molecular Fingerprints** 🔬
   * The app creates **fingerprints** for both molecules.
   * **Fingerprint = unique chemical signature** based on **atoms, bonds, and structure**.
   * This is like a **barcode** that represents molecular properties numerically.
5. **Calculate Molecular Similarity** 📊
   * Uses **Tanimoto Similarity** (a common measure in cheminformatics).
   * **Similarity Score (0 to 1)**:
     + **1.00 → Identical molecules**
     + **0.80 → Very similar**
     + **0.50 → Moderately similar**
     + **0.20 → Very different**
     + **0.00 → Completely different**
6. **Display Similarity Score & Molecules** 🖼️
   * Shows **a similarity score** (e.g., **0.75 = moderately similar**).
   * **Displays both molecules side by side** using RDKit’s molecule visualization tools.
7. **Error Handling** ⚠️
   * If invalid SMILES are entered, the app **shows an error message**:
     + ❌ *"Invalid SMILES entered."*

**Example Outputs**

**Example 1: Comparing Ethanol (CCO) & Ethylamine (CCN)**

* 🔎 **Similarity Score:** **0.75** (Moderately similar, both have a small structure)

**Example 2: Comparing Benzene (c1ccccc1) & Aspirin (CC(=O)OC1=CC=CC=C1C(=O)O)**

* 🔎 **Similarity Score:** **0.35** (Somewhat similar, but aspirin has extra functional groups)

**Example 3: Comparing Water (O) & Caffeine (CN1C=NC2=C1C(=O)N(C(=O)N2C))**

* 🔎 **Similarity Score:** **0.02** (Completely different)

**What is this app useful for?**

✅ **Drug Discovery** 💊 – Helps find similar molecules for drug candidates.  
✅ **Cheminformatics & Chemistry Research** 🔬 – Quickly compare molecular structures.  
✅ **Educational Purposes** 📚 – Helps students understand molecular similarity concepts.

Protein Ligand

This **web app** helps researchers **prepare small molecules (ligands) for protein-ligand docking** in drug discovery. It takes a **SMILES string** (a text representation of a molecule), converts it into a **3D optimized structure**, and allows users to **download the structure for docking simulations**.

**Step-by-Step Explanation**

1. **App Setup & Title Display** 🏗️
   * The app **displays a title**: *"🔬 Protein-Ligand Preparation Tool"*
2. **User Input (SMILES String for Ligand)** 🧪
   * The user **enters a SMILES string** (text-based molecular representation).
   * Example Inputs:
     + "CCO" → **Ethanol**
     + "CC(C)CC1=CC=C(C=C1)C(C)C(=O)O" → **Ibuprofen**
3. **Convert SMILES to a Molecule** 🔄
   * Uses **RDKit** to convert the **SMILES representation** into a molecule object.
4. **Prepare the Ligand for Docking** 🔬
   * Adds **hydrogen atoms** (important for accurate modeling).
   * Generates a **3D molecular structure** (embedding).
   * **Optimizes the molecular geometry** using **UFF (Universal Force Field)** to find a stable conformation.
5. **Visualize the 3D-Optimized Ligand** 🖼️
   * Displays the **optimized molecular structure** as an image.
6. **Allow Downloading of Ligand in SDF Format** 💾
   * The app **exports the prepared ligand** in **SDF format**, which is commonly used in **docking software like AutoDock Vina**.
   * Users can **download the SDF file** and use it for further computational analysis.
7. **Error Handling** ⚠️
   * If an **invalid SMILES string** is entered, the app displays an error message:
     + ❌ *"Invalid SMILES entered."*

----------------------------------RNA Velocity----------------------------------------------------

This **web app** is designed for **analyzing single-cell RNA sequencing (scRNA-seq) data** using **trajectory inference and RNA velocity analysis**. It allows users to:

✅ **Upload their own dataset** or choose from **preloaded example datasets**  
✅ **Process and clean the data** to remove noise  
✅ **Estimate RNA velocity** (how genes change over time in single cells)  
✅ **Visualize cell trajectories** using different methods (e.g., scVelo, Monocle, Slingshot, PAGA)  
✅ **Generate interactive plots** to explore RNA velocity and gene expression

**Step-by-Step Explanation**

**1️⃣ Choosing a Dataset 📊**

* Users can **upload their own scRNA-seq dataset** (CSV file) or **select a built-in dataset** (like pancreas, pbmc3k, or lung cells).
* The dataset is converted into **AnnData format** (a special data format for single-cell analysis).

**2️⃣ Preprocessing the Data 🧹**

* The app **filters out poor-quality cells and genes** to ensure reliable analysis.
* It **normalizes** the data (so that large differences in cell size don’t affect results).
* **Principal Component Analysis (PCA)** is applied to reduce the dataset’s complexity.
* **UMAP is computed** to visualize the data in 2D space.

**3️⃣ RNA Velocity Estimation 🚀**

* **RNA velocity** predicts how gene expression is changing over time in single cells.
* The app **computes spliced and unspliced mRNA counts** to infer gene activity.
* A **velocity graph** is built, showing how cells transition from one state to another.
* **Pseudotime analysis** is performed to arrange cells along a timeline of biological progression.

**4️⃣ Visualization & Analysis 🎨**

* The app allows users to **select a method for trajectory inference**:  
  ✅ **scVelo** (default method, based on spliced/unspliced RNA)  
  ✅ **Velocyto** (another RNA velocity method)  
  ✅ **Monocle** (infers developmental trajectories)  
  ✅ **Slingshot** (connects clusters along a trajectory)  
  ✅ **PAGA** (probabilistic inference of cell transitions)
* Users can **choose how they want to visualize their data**:  
  ✅ **RNA Velocity Streamplot** – Shows how cells are moving in gene expression space.  
  ✅ **Velocity Graph** – Displays directional gene activity changes.  
  ✅ **Gene Expression Heatmap** – Highlights expression patterns over pseudotime.