Python for Cheminformatics & Bioinformatics

Labs: Hands-On Exercises

Nirajan Bhattarai

AI-Driven Drug Development Training

February 2026

Lab Exercises Overview

Lab 1: Variables & Data Types

Objective: Practice storing and converting molecular and sequence data.

Exercise 1.1 – Compound Data Storage

Create variables for a drug compound:

- Name (string): “Ibuprofen”
- SMILES (string): “CC(C)CC1=CC=C(C=C1)C(C)C(=O)O”
- Molecular weight (float): 206.28
- pIC50 (float): 6.1
- Is active (bool): True if pIC50 $\geq$ 6.0

Print all values using f-strings.

Lab 1: Variables & Data Types (cont.)

Exercise 1.2 – DNA Sequence

Store a DNA sequence: “ATGCGATCGATCGATCGATCG”
Calculate and print:

- Sequence length
- Number of adenines (A)
- Number of thymines (T)

Exercise 1.3 – Type Conversion

Given IC50 = “5.2” (string), convert to float and calculate pIC50.
Store the result as both float and formatted string (2 decimals).

Lab 2: Operators

Objective: Apply operators for molecular calculations and filtering.

Exercise 2.1 – IC50 Conversion

Write code to convert IC50 values from nM to pIC50:
IC50 values: 10.0, 100.0, 1000.0 (nM)
Formula: $\text{pIC50} = -\log_{10}(\text{IC50} \times 10^{-9})$
Hint: Use `import math; math.log10()`

Exercise 2.2 – Lipinski Check

Given: MW=450, LogP=3.5, HBD=2, HBA=8
Check if compound passes Rule of Five:
$(\text{MW} \leq 500)$ AND $(\text{LogP} \leq 5)$ AND $(\text{HBD} \leq 5)$ AND $(\text{HBA} \leq 10)$

Lab 2: Operators (cont.)

Exercise 2.3 – GC Content (Rosalind)

Calculate GC content percentage for: “AGCTATAG”
Formula: GC% $= (G + C) / \text{total} \times 100$
Expected output: “GC Content: XX.X%”

Exercise 2.4 – Activity Classification

Given pIC50 $= 7.2$, determine if compound is:

- “Highly potent” (pIC50 ≥ 8)
- “Potent” (pIC50 ≥ 7)
- “Moderate” (pIC50 ≥ 6)
- “Weak” (pIC50 < 6)

Use comparison and logical operators.

Lab 3: Strings

Objective: Manipulate SMILES and biological sequences.

Exercise 3.1 – DNA Transcription

Transcribe DNA to RNA:
Input: "ATGCGATCGATCG"
Replace all T with U
Expected: "AUGCGAUCGAUCG"

Exercise 3.2 – Reverse Complement (Rosalind REVC)

Generate the reverse complement of DNA:
Input: "AAAACCCGGT"
Complement: A↔T, G↔C
Then reverse the string
Expected: "ACCGGGTTTT"

Lab 3: Strings (cont.)

Exercise 3.3 – SMILES Analysis

Analyze SMILES: “CC(=O)OC1=CC=CC=C1C(=O)O” (Aspirin)

- Find if it contains a ring (digits indicate ring closure)
- Count the number of carbons (C)
- Count oxygen atoms (O)
- Check if it’s aromatic (contains lowercase letters)

Exercise 3.4 – Nucleotide Count (Rosalind DNA)

Count nucleotides in: “AGCTTTTCATTCTGACTGCAACGGGCAATA”
Print: “A:X T:X G:X C:X”

Lab 4: Conditionals

Objective: Implement decision logic for compound classification.

Exercise 4.1 – Drug-Likeness Checker

Create a program that checks Lipinski Rule of Five:
Input: MW, LogP, HBD, HBA
Output: Number of violations (0–4)
Print "Drug-like" if violations ≤ 1, else "Non-drug-like"

Exercise 4.2 – Codon Identifier

Given a 3-letter codon, identify if it's:

- Start codon: "ATG"
- Stop codon: "TAA", "TAG", "TGA"
- Other: any other codon

Use if/elif/else or match-case.

Lab 4: Conditionals (cont.)

Exercise 4.3 – Activity Classifier

Classify compound based on pIC50:

- $\text{pIC50} \geq 8$: "Highly Active"
- $7 \leq \text{pIC50} < 8$: "Active"
- $6 \leq \text{pIC50} < 7$: "Moderately Active"
- $5 \leq \text{pIC50} < 6$: "Weakly Active"
- $\text{pIC50} < 5$: "Inactive"

Test with values: 8.5, 7.2, 6.5, 5.3, 4.1

Lab 5: Loops

Objective: Process collections of compounds and sequences.

Exercise 5.1 – Batch IC50 Conversion

Convert list of IC50 values (nM) to pIC50:
IC50_list = [1.0, 10.0, 100.0, 1000.0, 10000.0]
Use a for loop to calculate and print each pIC50.

Exercise 5.2 – Nucleotide Counter (Rosalind DNA)

Count all nucleotides in a DNA sequence using a for loop:
seq = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGT"
Print counts for A, C, G, T separated by spaces.

Lab 5: Loops (cont.)

Exercise 5.3 – Filter Active Compounds

Given pIC50 values: [5.2, 6.8, 7.3, 4.9, 8.1, 5.9, 6.2]
Use a for loop with continue to skip inactive (pIC50 < 6)
Print only active compounds.

Exercise 5.4 – Find First Potent (While Loop)

Given pIC50 values: [5.2, 5.8, 6.1, 7.5, 8.2, 6.8]
Use a while loop with break to find the first "highly potent" compound (pIC50 $\geq$ 7.5)
Print its index and value.
Hint: Use index variable i, increment i += 1

Lab 5: Loops (cont.)

Exercise 5.5 – Read Until Stop Codon (While Loop)

Given codons: [“ATG”, “CGA”, “TCG”, “GGC”, “TAA”, “AAA”]
Use a while loop to read codons and build a sequence string.
Stop when you encounter a stop codon (“TAA”, “TAG”, or “TGA”).
Print the sequence built before the stop codon.

Exercise 5.6 – Compound Screening (While Loop)

Simulate screening compounds until finding 3 active ones:
Given: [4.5, 5.2, 6.8, 5.1, 7.3, 4.9, 8.1, 5.9]
Use while loop to count actives (pIC50 $\geq$ 6), stop when count reaches 3.
Print how many compounds were screened total.

Lab 6: Functions

Objective: Create reusable molecular utility functions.

Exercise 6.1 – pIC50 Converter Function

Create function: `ic50_to_pic50(ic50_nm)`
Input: IC50 in nanomolar
Output: pIC50 value
Test with: 10, 100, 1000 nM

Exercise 6.2 – GC Content Function

Create function: `gc_content(sequence)`
Input: DNA sequence string
Output: GC percentage (float)
Test with: "AGCTATAG", "GCGCGCGC", "ATATAT"

Lab 6: Functions (cont.)

Exercise 6.3 – Lipinski Calculator

Create function: `check_lipinski(mw, logp, hbd, hba)`
Returns tuple: (passes: bool, violations: int)
Test with multiple compound property sets.

Exercise 6.4 – Reverse Complement Function

Create function: `reverse_complement(dna)`
Input: DNA sequence
Output: Reverse complement sequence
Test with Rosalind REVC sample: “AAAACCCGGT” $\rightarrow$ “ACCGGGTTTT”

Lab 6B: Error Handling

Objective: Build robust code that handles invalid inputs.

Exercise 6B.1 – Safe IC50 Conversion

Modify `ic50_to_pic50()` to handle:

- Negative IC50 values (raise ValueError)
- Zero IC50 (raise ValueError)
- Non-numeric input (catch TypeError)

Return None on error and print helpful message.

Exercise 6B.2 – SMILES Validator

Create function that validates SMILES:
Use RDKit: `Chem.MolFromSmiles(smiles)`
If returns None, raise ValueError with message
Handle with try/except and return valid/invalid status.

Lab 7: Lists

Objective: Manage compound libraries using lists.

Exercise 7.1 – Compound Library

Create a list of SMILES strings for 5 common drugs.
Perform operations:

- Add a new compound
- Remove a compound by value
- Insert at specific position
- Print first and last compounds

Exercise 7.2 – pIC50 Statistics

Given: [5.2, 6.8, 7.3, 4.9, 8.1, 5.9, 6.2, 7.8]
Calculate: min, max, sorted list, count of actives ($\geq$6)

Lab 7B: Tuples & Sets

Exercise 7B.1 – Compound Records

Create tuples for 3 compounds: (name, SMILES, pIC50)
Unpack each tuple and print formatted output.
Try to modify a tuple value – observe the error.

Exercise 7B.2 – Library Comparison

Library A: {“CMP001”, “CMP002”, “CMP003”, “CMP004”}
Library B: {“CMP003”, “CMP004”, “CMP005”, “CMP006”}
Find:

- All unique compounds (union)
- Common compounds (intersection)
- Compounds only in A
- Compounds only in B

Lab 8: List Comprehensions

Objective: Use comprehensions for elegant data transformations.

Exercise 8.1 – Filter Active Compounds

Given pIC50 $=$ [5.2, 6.8, 7.3, 4.9, 8.1, 5.9]
Use list comprehension to filter pIC50 $\geq$ 6.0

Exercise 8.2 – Batch Conversion

Convert IC50 list [10, 100, 1000] to pIC50 using:
a) List comprehension
b) map() with lambda

Exercise 8.3 – Conditional Comprehension

Create list of tuples: [(pIC50, "Active") if pIC50 $\geq$ 6 else (pIC50, "Inactive")]
for values [5.2, 6.8, 7.3, 4.9, 8.1]

Lab 9: Dictionaries

Objective: Build compound databases using dictionaries.

Exercise 9.1 – Compound Database

Create a dict of compounds with nested properties:
Key: compound name
Value: dict with SMILES, MW, pIC50, is_active

Exercise 9.2 – Codon Table (Rosalind)

Create a dict mapping codons to amino acids:
"ATG" → "M", "TGG" → "W", "TAA" → "Stop", etc.
Use to translate a short sequence.

Exercise 9.3 – Dict Comprehension

Filter the compound database to only active compounds
using dict comprehension: {k:v for k,v in ... if ...}

Lab 10: File Handling

Objective: Read and write compound and sequence files.

Exercise 10.1 – Write Compound CSV

Create a CSV file with columns: name, SMILES, pIC50
Write data for 5 compounds using `with open()`.

Exercise 10.2 – Read FASTA

Create a simple FASTA parser:
Read file, extract header (lines starting with $>$)
Concatenate sequence lines
Return dict: {header: sequence}

Exercise 10.3 – Filter and Export

Read compound CSV, filter active compounds (pIC50 $\geq$ 6)
Write filtered results to new file “actives.csv”

Lab 11: NumPy

Objective: Use NumPy for numerical molecular data.

Exercise 11.1 – Descriptor Matrix

Create a 2D array of molecular descriptors:
Rows = compounds, Columns = [MW, LogP, HBD, HBA]
Calculate mean and std for each descriptor (column).

Exercise 11.2 – Normalization

Normalize descriptor values to 0-1 range:
Formula: (x - min) / (max - min)
Use vectorized operations (no loops).

Exercise 11.3 – Boolean Filtering

Filter compounds where MW $<$ 500 AND LogP $<$ 5
Use boolean indexing.

Lab 11B: Pandas

Objective: Analyze compound datasets with Pandas.

Exercise 11B.1 – Create Compound DataFrame

Create DataFrame with: Name, SMILES, MW, LogP, pIC50
Add 5+ compounds. Add column for activity class.

Exercise 11B.2 – Data Analysis

Calculate: mean pIC50, count by activity class
Filter drug-like compounds (MW $<$ 500, LogP $<$ 5)
Sort by pIC50 descending.

Exercise 11B.3 – GroupBy Analysis

Group compounds by activity class
Calculate mean MW and LogP per group
Export results to CSV.

Lab 12: JSON & Regex

Exercise 12.1 – Parse PubChem JSON

Parse JSON compound data:

`{"CID": 2244, "name": "Aspirin", "MW": 180.16}`

Extract and print each field.

Exercise 12.2 – Find Restriction Sites

Use regex to find all occurrences of “GAATTC” (EcoRI site) in sequence: “ATGAATTCGCGAATTCTA”

Print positions of each match.

Exercise 12.3 – SMILES Validation

Use regex to check if SMILES contains:

- Aromatic ring (lowercase c, n, o, s)
- Ring closure (digits)
- Double bond (=)

Rosalind Challenges

Complete these Rosalind.info problems:

1. **DNA** – Counting DNA Nucleotides
2. **RNA** – Transcribing DNA into RNA
3. **REVC** – Complementing a Strand of DNA
4. **GC** – Computing GC Content
5. **HAMM** – Counting Point Mutations
6. **PROT** – Translating RNA into Protein
7. **SUBS** – Finding a Motif in DNA
8. **CONS** – Consensus and Profile

Submission: Upload your solutions to GitHub with:
- Clear function documentation
- Test cases with sample data
- README explaining approach

Mini-Project: QSAR Data Prep Pipeline

Objective: Build a complete data preparation pipeline. **Tasks:**

1. Read compound CSV with SMILES and IC50 values
2. Validate all SMILES using RDKit
3. Convert IC50 (nM) to pIC50
4. Calculate Lipinski descriptors (MW, LogP, HBD, HBA)
5. Add activity class column (Active/Inactive)
6. Filter drug-like compounds
7. Export clean dataset to CSV
8. Generate summary statistics

Deliverables:

- Python script with documented functions
- Output CSV file
- Summary report (mean/std of descriptors)

Mini-Project: Hints

Useful RDKit Functions:

- `Chem.MolFromSmiles(smiles)` – parse SMILES
- `Descriptors.MolWt(mol)` – molecular weight
- `Descriptors.MolLogP(mol)` – LogP
- `Descriptors.NumHDonors(mol)` – H-bond donors
- `Descriptors.NumHAcceptors(mol)` – H-bond acceptors

Pandas Operations:

- `pd.read_csv()` – read input
- `df.apply()` – apply function to column
- `df.describe()` – summary statistics
- `df.to_csv()` – export results

Lab Summary

Basics Labs (1–6B):

- Variables, operators, strings for molecular data
- Conditionals for classification
- Loops for batch processing
- Functions for reusable utilities
- Error handling for robust code

Collections Labs (7–12):

- Lists, tuples, sets, dicts for compound storage
- Comprehensions for elegant transformations
- File I/O for CSV and FASTA
- NumPy/Pandas for data analysis
- JSON/Regex for APIs and pattern matching

Projects:

- Rosalind bioinformatics challenges

Resources

Practice Platforms:

- Rosalind.info – Bioinformatics problems
- LeetCode – General programming
- Kaggle – Data science competitions

Cheminformatics Data:

- ChEMBL – Bioactivity database
- PubChem – Chemical compounds
- ZINC – Virtual screening library

Documentation:

- RDKit: rdkit.org
- Pandas: pandas.pydata.org
- NumPy: numpy.org

Good luck with your exercises!