Python for Cheminformatics & Bioinformatics: Homework Assignment

AI-Driven Drug Development Training

February 2026

Instructions

- Complete all problems in a single Python file named `homework_solutions.py`
- Use comments to separate each problem (e.g., `# Problem 1`)
- Include docstrings for all functions
- Test your code before submission
- Total Points: 100
- Estimated Time: 3-4 hours

---

1 Part 1: Bioinformatics - DNA/RNA/Protein (25 points)

Inspired by Rosalind.info problems

Problem 1: Counting DNA Nucleotides (5 points)

(Rosalind ID: DNA)

Given a DNA string, count the occurrences of each nucleotide.

```
dna = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC"
```

Implement `count_nucleotides(dna)` that returns a dictionary:

a) Count of A, C, G, T

b) Validate input contains only valid nucleotides

c) Handle both uppercase and lowercase input

d) Return counts in order: A, C, G, T

Expected output: `{"A": 20, "C": 12, "G": 17, "T": 21}`

Problem 2: Transcribing DNA to RNA (5 points)

(Rosalind ID: RNA)

Create a function `transcribe(dna)` that converts DNA to RNA:

a) Replace all occurrences of 'T' with 'U'

b) Validate input is valid DNA

c) Handle edge cases (empty string, invalid characters)

```
dna = "GATGGAACTTGACTACGTAAATT"
# Expected: "GAUGGAACUUGACUACGUAAAUU"
```

Problem 3: Reverse Complement (5 points)

(Rosalind ID: REVC)

Create `reverse_complement(dna)` that returns the reverse complement:

a) A $\leftrightarrow$ T, C $\leftrightarrow$ G

b) Reverse the complemented string

c) Handle invalid input gracefully

```
dna = "AAAACCCGGT"
# Expected: "ACCGGGTTTT"
```

Problem 4: Computing GC Content (5 points)

(Rosalind ID: GC)

Create `gc_content(dna)` that calculates the GC percentage:

a) GC content = (G + C) / total length $\times$ 100

b) Return as percentage with 2 decimal places

c) Also implement `highest_gc(fasta_dict)` that takes multiple sequences and returns the one with highest GC content

```
sequences = {
    "Rosalind_0001": "AGCTATAG",
    "Rosalind_0002": "GCGCGCGC",
    "Rosalind_0003": "ATATATATAT"
}
```

Problem 5: Translating RNA to Protein (5 points)

(Rosalind ID: PROT)

Create `translate(rna)` using the codon table:

```
codon_table = {
    "UUU": "F", "UUC": "F", "UUA": "L", "UUG": "L",
    "UCU": "S", "UCC": "S", "UCA": "S", "UCG": "S",
    "UAU": "Y", "UAC": "Y", "UAA": "*", "UAG": "*",
    "UGU": "C", "UGC": "C", "UGA": "*", "UGG": "W",
    "CUU": "L", "CUC": "L", "CUA": "L", "CUG": "L",
```

```
    "CCU": "P", "CCC": "P", "CCA": "P", "CCG": "P",
    "CAU": "H", "CAC": "H", "CAA": "Q", "CAG": "Q",
    "CGU": "R", "CGC": "R", "CGA": "R", "CGG": "R",
    "AUU": "I", "AUC": "I", "AUA": "I", "AUG": "M",
    "ACU": "T", "ACC": "T", "ACA": "T", "ACG": "T",
    "AAU": "N", "AAC": "N", "AAA": "K", "AAG": "K",
    "AGU": "S", "AGC": "S", "AGA": "R", "AGG": "R",
    "GUU": "V", "GUC": "V", "GUA": "V", "GUG": "V",
    "GCU": "A", "GCC": "A", "GCA": "A", "GCG": "A",
    "GAU": "D", "GAC": "D", "GAA": "E", "GAG": "E",
    "GGU": "G", "GGC": "G", "GGA": "G", "GGG": "G"
}
```

a) Translate RNA to protein sequence

b) Stop at stop codons (*, UAA, UAG, UGA)

c) Handle sequences not divisible by 3

2 Part 2: Cheminformatics - Molecules & Properties (25 points)

Problem 6: Molecular Formula Parser (5 points)

Create `parse_formula(formula)` that parses a molecular formula:

```
formulas = ["H2O", "C6H12O6", "C2H5OH", "NaCl", "Ca(OH)2"]
```

a) Return dictionary of element counts

b) Handle parentheses (e.g., Ca(OH)2 = Ca:1, O:2, H:2)

c) Handle subscripts correctly

Problem 7: Molecular Weight Calculator (5 points)

Create `calculate_mw(formula)` using atomic weights:

```
atomic_weights = {
    "H": 1.008, "C": 12.011, "N": 14.007, "O": 15.999,
    "S": 32.065, "P": 30.974, "Na": 22.990, "Cl": 35.453,
    "Ca": 40.078, "Fe": 55.845, "Mg": 24.305, "K": 39.098
}
```

a) Calculate MW from parsed formula

b) Round to 3 decimal places

c) Test with: Aspirin (C9H8O4), Caffeine (C8H10N4O2), Glucose (C6H12O6)

Problem 8: SMILES String Analyzer (5 points)

Create functions to analyze SMILES strings (simplified, no RDKit):

```
smiles_list = [
    "CCO",            # Ethanol
    "CC(=O)O",        # Acetic acid
    "c1ccccc1",       # Benzene
    "CC(=O)Oc1ccccc1C(=O)O"  # Aspirin
]
```

a) `count_atoms(smiles)`: Count C, N, O, S atoms (simplified)

b) `has_ring(smiles)`: Check if molecule has a ring (contains digits)

c) `count_double_bonds(smiles)`: Count '=' occurrences

d) `is_aromatic(smiles)`: Check for lowercase letters (aromatic atoms)

Problem 9: Drug-likeness Calculator (5 points)

Implement Lipinski's Rule of Five checker:

```
# Drug properties dictionary
drugs = {
    "Aspirin": {"MW": 180.16, "LogP": 1.19, "HBD": 1, "HBA": 4},
    "Ibuprofen": {"MW": 206.29, "LogP": 3.97, "HBD": 1, "HBA": 2},
    "Metformin": {"MW": 129.16, "LogP": -1.43, "HBD": 3, "HBA": 5},
    "Atorvastatin": {"MW": 558.64, "LogP": 6.36, "HBD": 4, "HBA": 7}
}
```

a) `check_lipinski(properties)`: Check all 5 rules

- MW $\leq$ 500
- LogP $\leq$ 5
- HBD (H-bond donors) $\leq$ 5
- HBA (H-bond acceptors) $\leq$ 10

b) Return number of violations

c) `classify_drugs(drugs_dict)`: Classify all as "Drug-like" or "Not Drug-like"

Problem 10: IC50 to pIC50 Converter (5 points)

Create functions for bioactivity data:

```
ic50_values = [0.5, 1.0, 10.0, 100.0, 1000.0, 5000.0]  # in nM
```

a) `ic50_to_pic50(ic50_nm)`: Convert IC50 (nM) to pIC50

- pIC50 = -log10(IC50 in M) = 9 - log10(IC50 in nM)

b) `classify_potency(pic50)`: Classify activity

- pIC50 $\geq$ 8: "Highly Active"
- pIC50 $\geq$ 6: "Active"
- pIC50 $\geq$ 5: "Moderately Active"
- pIC50 $<$ 5: "Inactive"

c) `analyze_series(ic50_list)`: Return statistics (min, max, mean pIC50)

3 Part 3: Advanced Sequence Analysis (25 points)

Problem 11: Finding Motifs in DNA (5 points)

(Rosalind ID: SUBS)

Create `find_motif(dna, motif)` that finds all occurrences:

```
dna = "GATATATGCATATACTT"
motif = "ATAT"
# Expected positions: [2, 4, 10] (1-indexed)
```

a) Return all starting positions (1-indexed, as in Rosalind)

b) Handle overlapping matches

c) Case-insensitive matching

Problem 12: Hamming Distance (5 points)

(Rosalind ID: HAMM)

Create `hamming_distance(s1, s2)` to count point mutations:

```
s1 = "GAGCCTACTAACGGGAT"
s2 = "CATCGTAATGACGGCCT"
# Expected: 7
```

a) Count positions where sequences differ

b) Handle sequences of different lengths (raise error)

c) Also implement `similarity_percentage(s1, s2)`

Problem 13: FASTA File Parser (5 points)

(Rosalind ID: GC)

Create `parse_fasta(fasta_string)` to parse FASTA format:

```
fasta_string = """>Rosalind_6404
CCTGCGGAAGATCGGCACTAGAATAGCCAGAACCGTTTCTCTGAGGCTTCCGGCCTTCCC
TCCCACTAATAATTCTGAGG
>Rosalind_5959
CCATCGGTAGCGCATCCTTAGTCCAATTAAGTCCCTATCCAGGCGCTCCGCCGAAGGTCT
ATATCCATTTGTCAGCAGACACGC"""
```

a) Return dictionary: {sequence_id: sequence}

b) Handle multi-line sequences

c) Remove newlines from sequences

d) Validate FASTA format

Problem 14: Protein Mass Calculator (5 points)

(Rosalind ID: PRTM)

Calculate protein mass from amino acid sequence:

```
aa_mass = {
    "A": 71.04, "R": 156.10, "N": 114.04, "D": 115.03, "C": 103.01,
    "E": 129.04, "Q": 128.06, "G": 57.02, "H": 137.06, "I": 113.08,
    "L": 113.08, "K": 128.09, "M": 131.04, "F": 147.07, "P": 97.05,
    "S": 87.03, "T": 101.05, "W": 186.08, "Y": 163.06, "V": 99.07
}
```

a) `protein_mass(sequence)`: Calculate total mass

b) Handle invalid amino acids

c) Return mass rounded to 3 decimal places

Test with: `"SKADYEK"` (Expected: 821.392 Da)

Problem 15: Consensus and Profile (5 points)

(Rosalind ID: CONS)

Given multiple DNA strings of equal length, find the consensus:

```
sequences = [
    "ATCCAGCT",
    "GGGCAACT",
    "ATGGATCT",
    "AAGCAACC",
    "TTGGAACT",
    "ATGCCATT",
    "ATGGCACT"
]
```

a) `build_profile(sequences)`: Create profile matrix (count of each nucleotide at each position)

b) `consensus_sequence(profile)`: Return consensus string

c) Handle ties by choosing alphabetically first

4 Part 4: Scientific Data Analysis (25 points)

Problem 16: Bioactivity Data Processing (5 points)

Process a bioactivity dataset:

```
bioactivity_data = [
    {"compound": "CPD001", "target": "EGFR", "IC50_nM": 5.2, "MW": 423.5},
    {"compound": "CPD002", "target": "EGFR", "IC50_nM": 120.0, "MW": 389.2},
    {"compound": "CPD003", "target": "VEGFR", "IC50_nM": 8.7, "MW": 512.3},
    {"compound": "CPD004", "target": "EGFR", "IC50_nM": 2.1, "MW": 445.6},
    {"compound": "CPD005", "target": "VEGFR", "IC50_nM": 450.0, "MW": 378.9},
]
```

a) Add pIC50 values to each entry

b) Filter compounds by target

c) Find most potent compound per target

d) Calculate average pIC50 per target

Problem 17: Sequence Statistics with NumPy (5 points)

Analyze multiple sequences using NumPy:

```
import numpy as np
sequences = ["ATGCGATCGATCG", "ATGCATGCATGCA", "GCGCGCGCGCGCG"]
```

a) Convert sequences to numerical encoding (A=0, C=1, G=2, T=3)

b) Create a 2D NumPy array of encoded sequences

c) Calculate nucleotide frequencies per position

d) Find positions with highest variability

Problem 18: Compound Library Analysis with Pandas (5 points)

Create and analyze a compound library:

```
import pandas as pd
compounds = pd.DataFrame({
    "ID": ["CPD001", "CPD002", "CPD003", "CPD004", "CPD005"],
    "MW": [423.5, 389.2, 512.3, 445.6, 378.9],
    "LogP": [3.2, 2.8, 4.5, 3.8, 2.1],
    "HBD": [2, 1, 3, 2, 1],
    "HBA": [5, 4, 6, 5, 3],
    "IC50_nM": [5.2, 120.0, 8.7, 2.1, 450.0]
})
```

a) Add pIC50 column

b) Add Lipinski_Violations column

c) Filter drug-like compounds (violations ≤ 1)

d) Rank by potency (pIC50)

e) Export filtered results to CSV

Problem 19: Dose-Response Analysis (5 points)

Analyze dose-response data:

```
doses = [0.001, 0.01, 0.1, 1, 10, 100, 1000]  # uM
responses = [2, 5, 15, 45, 78, 95, 99]  # % inhibition
```

a) Plot dose-response curve (log scale for dose)

b) Estimate IC50 by interpolation (dose at 50% response)

c) Calculate Hill slope (steepness)

d) Classify as: steep (>1.5), normal (0.8-1.5), shallow (<0.8)

Problem 20: Molecular Descriptor Analysis (5 points)

Analyze descriptor correlations:

```
descriptors = pd.DataFrame({
    "MW": [423, 389, 512, 445, 378, 520, 410, 395],
    "LogP": [3.2, 2.8, 4.5, 3.8, 2.1, 5.1, 3.0, 2.5],
    "TPSA": [78, 65, 95, 82, 55, 102, 70, 60],
    "RotBonds": [5, 4, 8, 6, 3, 9, 5, 4],
    "pIC50": [8.3, 6.9, 8.1, 8.7, 6.3, 5.8, 7.5, 7.0]
})
```

a) Calculate correlation matrix

b) Find descriptors most correlated with pIC50

c) Identify highly correlated descriptor pairs ($|r| > 0.7$)

d) Create scatter plot of best predictor vs pIC50

e) Simple linear regression to predict pIC50

Submission Guidelines

1. Submit a single Python file: `homework_solutions.py`
2. Include all required imports at the top (numpy, pandas, math, re)
3. Use comments to clearly label each problem
4. Include docstrings for all functions
5. Test all functions before submission
6. Include sample output as comments where appropriate

Resources

- Rosalind.info - Bioinformatics problems: `http://rosalind.info`
- NCBI - Nucleotide database: `https://www.ncbi.nlm.nih.gov`
- ChEMBL - Bioactivity database: `https://www.ebi.ac.uk/chembl/`

Grading Rubric

- **Correctness (60%)**: Code produces correct output for test cases
- **Code Quality (20%)**: Clean, readable, well-organized code
- **Documentation (10%)**: Clear comments and docstrings
- **Error Handling (10%)**: Appropriate validation and try/except

---

Due Date: One week from assignment date
Late Policy: 10% deduction per day late
Questions: Contact instructor via email or office hours