Computational Space Biochemistry and Human Health: A Review of Python Libraries and Biological Databases for Astronaut Health Research.

Space Biotechnology Research Assistant Internship.
Task 04 | August 6, 2026 to August 28, 2026

Report Prepared by:
Masuma Shaikh
Biochemistry Student

Submitted to:
Sagar Sakalley
Founder & Director
MacroEdtech

Date of Submission:
August 28, 2026

Table of Contents.

1. Introduction.

Computational space biochemistry is an interdisciplinary field that connects space biochemistry with computer science to deal with the problems faced by the astronauts during long duration missions. The astronauts in their long missions encounter difficult conditions, including microgravity and galactic cosmic radiation, which affect the biochemical mechanisms of the human body. To understand how these processes take place, one should analyze large biological datasets called multi-omics, for instance genomics help in detecting the way the radiation can affect the DNA molecules and cause mutations, proteomics studies the impact of microgravity on the proteins, while metabolomics allows studying blood samples to detect the signs of stress and inflammation. However, conducting experiments through the wet-lab technique is quite difficult because it is too expensive to send laboratory equipment on spaceships and also there is not enough physical space. This is the crucial barrier which is overcome by computational biology. The main benefit of this computing method is the incredible speed with which it can process data, identify biomarkers, and develop new drugs. Rather than spending time on the slow process of conducting experiments in laboratories, complex algorithms can analyze multiple-omics data and identify any early warning signs or biomarkers that may indicate an astronaut's potential illness before he or she begins exhibiting symptoms. Moreover, computers will also be able to simulate how molecules deform under zero gravity conditions and test millions of chemical compounds.

The use of computational space biochemistry depends largely on the Python programming language. Python has clear and concise syntax which is similar to plain English. Its simple nature makes it possible to write codes quickly without having extensive knowledge of computer science. While other programming languages have proven to be very powerful, Python is preferred by many due to the perfect combination between usability and scientific capabilities. For example, C++ and Java languages are extremely powerful and quick but their syntax is hard to write and learn. In conclusion, Python is selected due to its ability to integrate all the data, math, molecular modeling and AI altogether.

2. Literature Review: Effects of Spaceflight on Human Health.

Microgravity is an environment where astronauts experience near-weightlessness because they are continuously orbiting Earth, and without Earth's gravity, bones and muscles can no longer support body weight due to which mechanical loading on bones decreases, osteoblast (bone - forming activity) decreases, osteoclast (bone- resorbing activity) increases, calcium is released from bones into bloodstream and the reduced sunlight exposure lowers the vitamin D synthesis which in turn reduces the calcium absorption and osteoblast process does not occur properly. Thus, the concentration of Vitamin D and Bone Density decreases and creatine kinase level in blood might increase due to muscle and protein breakdown. Moreover, fluid in the body moves upward from the legs to the chest and head, resulting in facial swelling, blocked sinuses, and increased pressure within the skull. This pressure causes deformation of the eye socket, which in turn affects vision. On the other hand, since there is no need for the body to work against gravity anymore, the muscular and circulatory system becomes less strained.

Unlike on Earth, astronauts in space receive greater exposure to galactic cosmic rays and solar radiation because they are no longer fully protected by Earth's atmosphere and magnetic field and these high-energy particles can penetrate and damage the body cells and tissues. Cosmic radiation increases the production of Reactive Oxygen Species (ROS) which is called Oxidative Stress during which excess ROS can damage lipids in cell membrane, proteins, enzymes, DNA , mitochondria and cell structure. This biological stress causes severe DNA mutations and makes healthy cells age quickly or transform themselves. Also, due to radiation, the immune system of the body is also affected and thus WBC count decreases. Moreover, the cosmic radiation damages the endothelium of blood vessels, leading to faster development of heart diseases, kills neurons, deteriorates memory, and destroys immune cells and leaves the body defenseless against various infections.

3. Python Libraries for Computational Space Biochemistry.

Python libraries are simply the sets of ready-made modules, functions, and other resources which make it easier for developers to do certain tasks without having to write code from scratch. In computational biology and space biochemistry, these Python libraries provide an opportunity to study biological data, visualize proteins, run molecular simulation, construct machine learning models, and process big data. Rather than inventing complicated algorithms on their own, the scientists may use the libraries developed by the scientific community.

Why are Python Libraries Important?
Python libraries allow researchers to do the following:

- Sequence analysis of DNA, RNA, and proteins
- Study of 3D protein structure
- Molecular docking studies
- Molecular dynamics simulation
- Processing of big biological data
- Graphical representation of scientific data
- Machine learning model building
- Drug discovery
- Analysis of data from NASA GeneLab and other biological databases

Benefits of using Python Libraries

- Open source and freely available
- User friendly
- Abundant documentation
- Facilitates AI and machine learning
- Compatible with biological databases
- Popular in research laboratories

Limitations

- Some libraries are difficult to install.
- Some libraries need programming skills.
- Simulations at molecular level might need fast computers or GPUs.
- Some computational tasks take a lot of time.

Each Python library has a unique role that addresses a certain scientific challenge. As a result, there is an arrangement of libraries based on what their main application is. Such grouping enables scientists to choose the best software depending on their particular research task.

3.1 Bioinformatics Libraries.

There are specific Python libraries called bioinformatics libraries, which have been designed specifically for the analysis of biological information such as DNA sequences, RNA sequences, protein sequences, protein structures, and biological data sets. In space biochemistry computations, these libraries can be applied effectively in order to study the impact of microgravity and space radiation on genes, proteins, and biological pathways. These can be used to conduct genomic, proteomic, transcriptomic, structural biology, and biomarker discoveries through information found in databases such as NASA GeneLab, UniProt, and PDB.

Library	Reason for Inclusion
Biopython	Sequence analysis, DNA, RNA, proteins, biological files format
scikit-bio	Biological sequence analysis, diversity analysis, phylogenet
BioPandas	Biological structure data processing (PDB files) using Pandas

1. Biopython

Biopython is a software library that was developed for use in bioinformatics using Python programming language. The library contains various utilities for dealing with biological sequences, proteins, biological databases, alignments, phylogenetic trees, and biological file formats. It is the most popular bioinformatics library in laboratories and educational institutions.

Biopython assists the researchers in:

- Reading and writing of DNA, RNA, and protein sequences.
- Sequence alignments.
- Databases in biology.
- FASTA, GenBank, and PDB files.
- Translation of DNA to protein sequences.
- Computational analysis of biological sequences.

Key Features

- DNA sequence analysis
- RNA sequence analysis
- Protein sequence analysis
- Sequence alignment
- Translation of DNA to proteins

- Reading FASTA files
- Reading GenBank files
- Protein structure parsing
- Access to NCBI databases through Entrez
- Phylogenetic tree analysis

Advantages

- Open source software.
- Easy to use.
- Works with multiple biological data files.
- Well integrated with NCBI databases.

Limitations

- Not intended for molecular docking studies.
- Has limited machine learning functionality.
- Large datasets can need optimization.
- Certain analyses need the use of other packages alongside it.

Applications in Computational Space Biochemistry

Using Biopython, we may:

- Analyze genes impacted by microgravity.
- Study proteins whose sequences have been modified due to spaceflight.
- Gather biological information from NCBI.
- Compare proteins that play a role in bone metabolism.
- Analyze genes that cause muscle atrophy.
- Protein analysis for oxidative stress.
- Study immune response proteins.
- Process omics data from NASA GeneLab.

Installation

```
!pip install biopython
```

Google Colab Walkthrough:

Biopython was utilized to convert a DNA sequence to a protein sequence according to the standard genetic code. This represents a simple example of sequence analysis employed in bioinformatics. The practical execution and output are included in the attached Google Colab file.

2. scikit-bio

Scikit-bio is an open-source library written in Python that is meant for bioinformatics applications, microbial ecology, evolutionary biology, and biological sequence analysis. It contains analytical tools for diversity analysis, DNA sequences, proteins, phylogenies, and ecological data.

Scikit-bio assists the researchers in:

- Analysis of DNA and protein sequences.
- Biodiversity studies.
- Computation of biological diversity indexes.
- Phylogenetic tree construction.
- Sequence alignments.
- Statistical analysis of biological data.

Key Features

- Sequence analysis
- DNA/RNA/protein manipulation
- Phylogenetic analysis
- Biological statistics
- Diversity analysis
- Distance matrix calculations
- Sequence alignment tools

Advantages

- Excellent for biological statistics.
- Strong support for phylogenetics.
- Good documentation.
- Open source.
- Integrates well with NumPy and Pandas.

Limitations

- Less suitable for structural biology.
- Not intended for molecular docking.

Applications in Computational Space Biochemistry

Using Scikit-bio, we can do:

- Analysis of microbial community structure in spacecraft.
- Comparison of DNA sequences.
- Microbiome of space.
- Microbial diversity analysis.
- Evolutionary analysis of microorganisms in space.

Installation

!pip install scikit-bio

Google Colab Walkthrough:
The scikit-bio package was used to carry out analysis of a DNA sequence through determination of its sequence length, GC content, and AT content. This is an example of basic genomic composition analysis that can be useful for analyzing the properties of DNA.

3. BioPandas

BioPandas is a Python package which merges the features of Pandas and structural biology. BioPandas enables researchers to work with protein structure files like PDB and mmCIF formats using Pandas DataFrame.

BioPandas assists the researchers in:
- Read protein structure files.
- Translates protein structure information to DataFrames.
- Helps in analyzing protein data.
- Facilitates filtering and manipulating protein coordinates easily.
- Provides compatibility with Pandas library.

Key Features
- Read PDB files
- Read mmCIF files
- Protein coordinate analysis
- DataFrame-based manipulation
- Structural biology support
- Easy filtering of atoms and residues

Advantages
- Very easy to use for anyone familiar with Pandas.
- Excellent for protein structure analysis.
- Integrates with existing data analysis workflows.
- Simplifies handling of PDB files.

Limitations
- Focused mainly on structural data.
- Less comprehensive than Biopython for sequence analysis.
- Not designed for molecular dynamics or docking.

Applications in Computational Space Biochemistry

With BioPandas one can:

- Study structures of proteins responsible for bone metabolism.
- Study structural alterations of proteins impacted by space flight.
- Process protein data for molecular docking studies.
- Download and analyze structures of proteins from the Protein Data Bank.
- Research structural biomarkers connected with radiation effects and immunological response.

Installation

!pip install biopandas

Google Colab Walkthrough:
BioPandas was used to fetch the structure of the protein with PDB ID 1CRN and convert the coordinates of the atoms of the structure into a Pandas DataFrame. This is an illustration of how protein structural data can be transformed into tabular data for further analysis. The implementation and result can be seen in the attached Google Colab notebook.

Summary of Bioinformatics Libraries.

Library	Primary Purpose
Biopython	Sequence analysis and biological databases
scikit-bio	Biological statistics and phylogenetics
BioPandas	Protein structure data analysis

3.2 Molecular Docking Libraries.

Molecular docking is the name of the computer-aided technique which helps in predicting the binding interaction between the small molecules and biological macromolecules like proteins.

Workflow of docking study is as follows:
Chemical Structure → Ligand preparation → Receptor preparation → Docking → Binding poses/scores → Visualization & analysis

Tool	Main role in docking workflow
RDKit	Chemical structure handling, molecular descriptors, ligand preparation and cheminformatics
Open Babel	Molecular-format conversion and chemical file preparation
AutoDock Vina	Performs the actual molecular docking calculation
Meeko	Preparation of ligands/receptors into formats suitable for AutoDock Vina

1. RDKit

RDKit is a cheminformatics toolkit that is open-source and has interfaces for Python and C++. RDKit is developed for handling chemical data and molecular structures. Some of its capabilities include molecular structure manipulation, fingerprints, molecular descriptors, similarity calculations, substructure search, and molecular visualization.

RDKit is especially helpful prior to and during docking since scientists have to:

- Analyze chemical structures.
- Transform between molecule formats.
- Deal with SMILES.
- Create molecular fingerprints.
- Compute molecular descriptors.
- Find compounds that have similar structures.
- Screen compounds based on chemical features.
- Create molecule diagrams.
- Prepare ligand molecules.

For instance, when a scientist receives hundreds of compounds from PubChem or ChEMBL, RDKit may assist in organizing and screening them prior to docking.

Key Features

- SMILES handling
- Molecular structure representation
- Molecular fingerprints
- Molecular similarity
- Substructure searching
- Molecular descriptors
- Chemical property calculations

- 2D molecular drawing
- 3D molecular operations
- Chemical data processing

Advantages

- Open source.
- Powerful cheminformatics functionality.
- Very useful for drug-discovery workflows.
- Excellent Python integration.
- Supports molecular fingerprints and descriptors.
- Can process large chemical datasets.
- Widely used in computational chemistry.

Limitations

- It is not itself a molecular docking engine.
- Some advanced features require greater programming knowledge.
- Understanding chemical representations such as SMILES and molecular fingerprints requires some chemistry background.
- RDKit results still require biological interpretation.

Applications in Computational Space Biochemistry

RDKit could support astronaut-health research by helping researchers:

- Screen compounds against proteins involved in bone metabolism.
- Study candidate drugs for muscle wasting.
- Analyze compounds with antioxidant properties.
- Filter compounds before docking.
- Calculate chemical descriptors for machine-learning models.
- Search for structurally similar compounds.
- Prepare chemical datasets for AI-based drug discovery.

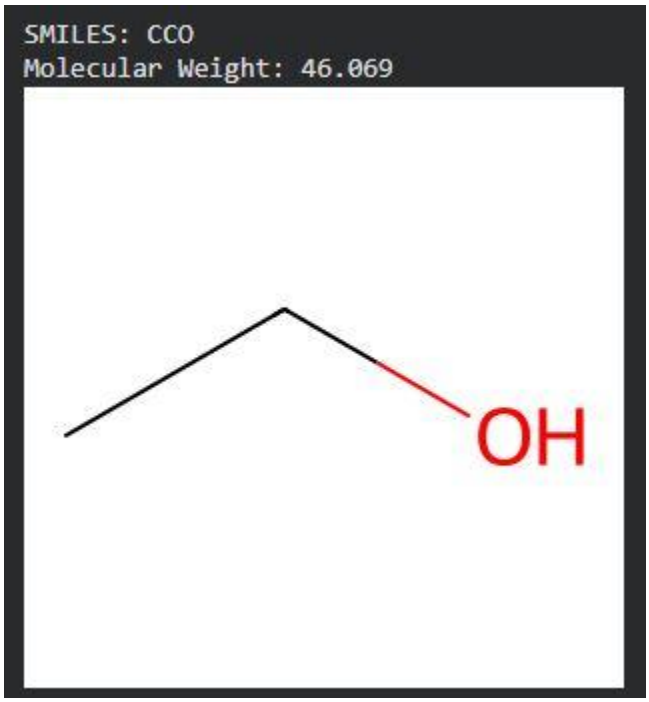
Installation

```
!pip install rdkit
```

Google Colab Walkthrough:

The use of RDKit was applied for transforming the SMILES string 'CCO' into a computable molecular structure, recognizing the structure as ethanol, calculating the molecular weight of the molecule, and rendering it. This serves as an example of the digital representation and analysis of chemical molecules using cheminformatics software. RDKit can be used to perform compound screening and molecular analysis in computational biochemistry prior to performing docking. The

actual code implementation and results can be found in the Google Colab notebook.



2. Open Babel

OpenBabel is an open-source chemistry toolbox that enables scientists to handle various file formats of chemicals. The use of OpenBabel is especially significant when structures of molecules retrieved from various databases or applications are available in various formats.

What is the use of Open Babel?
There are different file formats for various computational chemistry applications.
For instance:
SMILES → SDF → MOL2 → PDB → PDBQT

Open Babel helps in the conversion and manipulation of molecular files in compatible file formats. This comes in handy during the preparation of molecules for docking studies.

Key Features

- Chemical file conversion
- Molecular format handling
- Molecular structure processing
- Command-line tools

- Python bindings
- Molecular property manipulation
- Support for many chemical file formats

Advantages
- Open source.
- Supports many chemical formats.
- Useful for interoperability between programs.
- Helpful for molecular preparation.
- Can be used from command line and Python.

Limitations
- It is not a docking engine.
- Installation can be more complicated than pure Python packages.
- Different chemical formats may store information differently, so conversion should always be checked.
- A file conversion does not automatically guarantee that the molecule is scientifically ready for docking.

Applications in Computational Space Biochemistry

Open Babel can help:
- Prepare compounds obtained from chemical databases.
- Convert molecular files for docking programs.
- Process libraries of candidate compounds.
- Support automated drug-screening workflows.
- Convert structures used in astronaut-health drug discovery.

Installation

```
!apt-get update -qq
!apt-get install -y openbabel
```

Google Colab Walkthrough:
Conversion of a molecule from SMILES notation to SDF format was done using the software Open Babel, and this shows how molecular structures can be converted from one format to another in order to make them workable in various computational chemistry packages. In the computational space biochemistry domain, Open Babel will help prepare and convert molecules that have been extracted from databases like PubChem and ChEMBL for further docking studies in drug discovery.

3. AutoDock Vina

AutoDock Vina is a free molecular docking software that predicts possible interactions of small ligands with the target protein and their affinity with the help of a scoring function. This is the actual docking engine from amongst the four programs mentioned here.

Why is it used?
AutoDock Vina is used for:

- Prediction of the pose of ligand-binding.
- Calculation of docking score.
- Studying the protein-ligand interaction.
- Evaluation of various compounds.
- Virtual screening.
- Drug candidates investigation.

For instance:
Target protein + drug candidate → Vina → docking poses + docking score

Key Features

- Molecular docking
- Protein–ligand pose prediction
- Binding-score estimation
- Virtual screening
- Multiple ligand docking
- Python bindings
- Parallel computation
- Different scoring functions

Advantages

- Open source.
- Widely used.
- Relatively fast.
- Suitable for virtual screening.
- Python bindings available.
- Can process many ligands.

Limitations
Most importantly:

- A docking score is not proof that a drug will work biologically.

Other limitations include:

- Docking uses approximations.

- Protein flexibility is limited compared with molecular dynamics.
- Solvent effects may be simplified.
- Scoring functions are imperfect.
- Results depend strongly on receptor preparation and docking-box selection.
- Experimental validation is ultimately required.

Applications in Computational Space Biochemistry

Vina could be used to investigate candidate compounds targeting proteins associated with:

- Bone remodeling
- Muscle wasting
- Oxidative stress
- Radiation response
- DNA damage
- Immune dysfunction

For example:
Astronaut health problem → Identify biological target → Obtain protein structure from PDB → Obtain candidate compounds from PubChem/ChEMBL → Prepare receptor and ligand → AutoDock Vina → Compare predicted binding poses/scores → Select candidates for further investigation

Installation

```
!pip install vina
```

Google Colab Walkthrough:
AutoDock Vina was used to initialize the docking environment and create a Vina docking object, demonstrating that the Python interface was successfully loaded. This particular example is a part of the first step of the molecular docking pipeline; a full-fledged docking experiment would require preparing a receptor, ligand, docking box, and all the parameters. Within the computational biochemistry field, AutoDock Vina could be employed to research possible interactions of the selected molecules with the protein involved in bone resorption, muscle atrophy, oxidative stress, DNA damage, and immune response.

4. Meeko

Meeko is a software tool written in Python language that helps in preparation of molecules and receptors for the docking process using AutoDock. In simple words: Meeko prepares the molecules but Vina docks them.

Why is it used?
Docking programs require molecules to be prepared in specific formats. Meeko helps with tasks such as:

- Ligand preparation
- Receptor preparation
- PDBQT generation
- Assigning appropriate molecular information required for docking
- Preparing structures for Vina

Key Features

- Ligand preparation
- Receptor preparation
- PDBQT generation
- Integration with RDKit
- Integration with AutoDock Vina
- Support for modern docking workflows

Advantages

- Designed specifically for AutoDock workflows.
- Works well with RDKit.
- Automates important preparation steps.
- Reduces manual preparation.
- Useful for reproducible docking pipelines.

Limitations

- It does not perform the docking calculation itself.
- Primarily useful within AutoDock-related workflows.
- Requires understanding of molecular structures and docking preparation.
- Installation and dependency management can be more difficult for beginners.

Applications in Computational Space Biochemistry

Meeko could be used to prepare:

- Drug candidates targeting bone-related proteins.
- Compounds targeting muscle-associated proteins.
- Antioxidant compounds.
- Candidate radioprotective compounds.
- Compounds targeting immune-related proteins.

A possible workflow is:
PubChem/ChEMBL → RDKit → Meeko → AutoDock Vina → docking results

Installation

```
!pip install -q meeko rdkit gemmi
```

Google Colab Walkthrough:
Meeko was used to prepare the ethanol molecule generated with RDKit and convert it into the PDBQT format required for AutoDock Vina docking workflows. This demonstrates how a molecular structure can be prepared with atom types, partial charges, three-dimensional coordinates, and flexibility information for docking. In computational space biochemistry, Meeko can help prepare candidate compounds for studying potential interactions with targets associated with bone loss, muscle loss, oxidative stress, radiation damage, and immune-system changes. The practical implementation and output are provided in the accompanying Google Colab notebook.

3.3 Molecular Dynamics Libraries.

Molecular dynamics (MD) is a computational modeling tool that simulates the behavior and movement of atoms and molecules over time. Instead of analyzing proteins as fixed structures, molecular dynamics could be employed to examine their flexibility, conformational change, stability, and interaction with other molecules. MD is especially valuable for computational biochemistry because biological macromolecules tend to be flexible. A protein might alter its conformation in response to binding a ligand, a change in environment, or physical/chemical perturbation. Some Python packages and tools facilitate different steps of an MD workflow. Some of them could be primarily applied for simulations, whereas others for trajectory analysis and interpretation.

The following is a simplified workflow:
Protein structure → System setup → MD simulation → Trajectory → Analysis → Biological interpretation

Library	Main Use
OpenMM	Running molecular dynamics simulations
MDAnalysis	Analyzing structures and trajectories
MDTraj	Calculating structural properties from trajectories
ProDy	Protein motions, flexibility and normal modes

1. OpemMM

OpenMM is an open-source software application designed for conducting molecular dynamics simulations. It uses force field models to simulate molecules by applying laws of physics. This software has the ability to conduct computations using either CPU or GPU and hence suitable for performing complex molecular simulations. OpenMM is especially useful in cases where one wants to conduct molecular dynamics simulations for proteins, nucleic acids, ligands, membranes, solvents among others.

Why is it used?
OpenMM is used when researchers want to study questions such as:

- How stable is a protein structure?
- How does a protein move over time?
- Does ligand binding affect protein structure?
- Which regions of a protein are flexible?
- How does a mutation affect protein dynamics?
- How does a molecular system behave under different conditions?

Unlike molecular docking, which generally provides a predicted binding pose, MD simulations can examine how the protein–ligand system behaves over time.

Key Features

- Molecular dynamics simulations
- GPU acceleration
- CPU-based simulation
- Multiple force-field options
- Protein and ligand simulation
- Periodic boundary conditions
- Energy calculations
- Integration with Python
- Simulation trajectory generation

Advantages

- Powerful molecular simulation capabilities
- GPU acceleration
- Python-friendly
- Highly customizable
- Suitable for research-scale simulations
- Can be integrated with other computational biology tools

Limitations

- Full MD simulations can be computationally expensive.
- Requires understanding of molecular mechanics and force fields.
- Preparing a biologically meaningful simulation can be complex.
- Simulation results depend on the quality of the system setup and force field.
- A short simulation does not necessarily reproduce all biologically relevant processes.

Applications in Computational Space Biochemistry

OpenMM could potentially be used to study:

- **Bone health** - Where researchers could simulate proteins involved in: Bone remodeling, Osteoblast activity, Osteoclast activity and Calcium regulation. This could help investigate how structural changes in proteins may contribute to molecular pathways associated with microgravity-induced bone loss.
- **Muscle health** - MD simulations could help investigate proteins involved in: Muscle contraction, Muscle protein degradation, Cellular signaling, Energy metabolism and Radiation effects. .
- **Drug discovery** - A compound identified through docking could be subjected to MD simulation to investigate whether its interaction with a target remains stable over time.

A simplified workflow is:
Docking → OpenMM simulation → trajectory → stability analysis

Installation

```
!pip install -q openmm
```

Google Colab Walkthrough:
OpenMM was used for the demonstration of a basic molecular dynamics simulation process where a simple system was defined, physical properties were assigned, an integrator was created, and the simulation time was advanced. Even though this is an example of a single particle being used and not a biological system, it is an illustration of how molecular dynamics simulations work in computational space biochemistry. This can even be done in the context of larger protein-ligand systems to understand the stability of proteins and other molecular interaction issues in astronauts' health. The code and results can be found in the attached Google Colab notebook.

2. MDAnalysis

MDAnalysis is a Python software library that is mainly used to analyze molecular structures and molecular dynamics simulations. It handles numerous molecular simulation file formats and enables the extraction of information about the structure and dynamics of molecular systems.

Why is it used?
MDAnalysis comes handy after performing an MD simulation. It might happen that thousands of molecular structures have been produced by the simulation at various points of time, and manual inspection becomes impossible. Thus, MDAnalysis can be helpful for calculation of:

- Distances
- Angles
- RMSD
- RMSF
- Radius of gyration
- Hydrogen bonds
- Atom selection
- Protein-ligand interactions

Key Features

- Trajectory analysis
- Structure analysis
- Atom selection
- RMSD calculation
- RMSF calculation
- Distance analysis
- Radius of gyration
- Hydrogen-bond analysis
- Support for many molecular file formats
- Integration with NumPy and other scientific Python tools

Advantages

- Powerful trajectory-analysis capabilities
- Python-based
- Works with many molecular formats
- Useful for large simulation datasets
- Highly extensible
- Can be combined with NumPy, Pandas and Matplotlib

Limitations

- It does not perform the docking calculation itself.

- Primarily useful within AutoDock-related workflows.
- Requires understanding of molecular structures and docking preparation.
- Installation and dependency management can be more difficult for beginners.It is mainly an analysis library rather than a molecular dynamics simulation engine.
- Beginners may initially find trajectory analysis difficult.
- Understanding molecular dynamics concepts is necessary for interpreting results correctly.
- Large trajectories may require substantial memory and computational resources.

Applications in Computational Space Biochemistry

MDAnalysis could help researchers analyze simulations related to:

- Protein structural changes during spaceflight
- Bone-related protein dynamics
- Muscle-related proteins
- Radiation-associated molecular damage
- Protein–ligand interactions
- Drug stability
- Astronaut-health targets

For example: OpenMM simulation → trajectory → MDAnalysis → RMSD/RMSF → protein stability and flexibility

Installation

```
!pip install -q MDAnalysis
```

Google Colab Walkthrough:
MDAnalysis was used to load the ubiquitin protein structure (PDB ID: 1UBQ) and programmatically obtain its basic structural information, including the number of atoms, residues, and segments. This demonstrates how protein structures can be converted into computational objects for systematic analysis. In computational space biochemistry, the same approach can be extended to analyze molecular dynamics trajectories and investigate structural changes in proteins associated with bone loss, muscle loss, oxidative stress, DNA repair, and immune responses. The practical implementation and output are provided in the accompanying Google Colab notebook.

```
Number of atoms: 660
Number of residues: 134
Number of segments: 1
```

3. MDTraj

MDTraj is a Python library for the analysis of molecular dynamics trajectories. It offers tools for computing structural information from molecular simulations.

Why is it used?
MDTraj is an analysis tool that enables the extraction of quantitative information from MD trajectories. For instance, MDTraj can be utilized to calculate:

- RMSD
- RMSF
- Distance
- Dihedral Angles
- Radius of Gyration
- Hydrogen Bonds
- Contacts
- Secondary Structure

These values facilitate the conversion of trajectory data into biologically meaningful information.

Key Features

- Molecular trajectory analysis
- RMSD calculation
- RMSF analysis
- Distance calculations
- Dihedral-angle analysis
- Secondary-structure analysis
- Hydrogen-bond analysis
- Efficient trajectory processing
- Support for common trajectory formats

Advantages

- Easy integration with Python
- Efficient trajectory analysis
- Many useful structural calculations
- Works well with NumPy
- Suitable for large trajectory datasets
- Simple syntax for common analyses

Limitations

- Primarily an analysis library rather than a simulation engine.
- Requires molecular dynamics trajectory data for meaningful analysis.
- Some advanced analyses require additional programming knowledge.

- Correct interpretation requires knowledge of molecular simulation concepts.

Applications in Computational Space Biochemistry

MDTraj could be used to investigate:

- Protein stability
- Protein flexibility
- Ligand-induced conformational changes
- Structural effects of mutations
- Protein–drug interactions
- Molecular mechanisms associated with astronaut health

For example: Protein + ligand → MD simulation → MDTraj → RMSD/RMSF → structural interpretation

Installation

```
!pip install -q mdtraj
```

Google Colab Walkthrough:
MDTraj was used to load a protein structure as a trajectory object and examine its basic structural information. The demonstration contains a single frame because a static PDB structure was used, while real molecular dynamics trajectories can contain thousands of frames for analyzing structural changes over time. MDTraj can subsequently be used to calculate properties such as RMSD, RMSF, distances, angles, contacts, hydrogen bonds, and secondary structure. The practical implementation and output are provided in the accompanying Google Colab notebook.

```
Number of frames: 1
Number of atoms: 660
Number of residues: 134
```

4. ProDy

ProDy is a software package that deals with protein structure, dynamics, and conformational changes in Python. It is especially valuable in examining protein movement and flexibility.

Why is it used?
Proteins are flexible molecules. They exhibit various kinds of motions that may have biological significance. ProDy implements computational methods to investigate:

- Protein dynamics

- Normal mode analysis
- Principal component analysis
- Structural fluctuations
- Protein flexibility
- Conformational changes
- Protein interactions

Key Features

- Protein structure analysis
- Normal Mode Analysis (NMA)
- Principal Component Analysis (PCA)
- Protein dynamics
- Structural alignment
- Protein flexibility analysis
- Analysis of molecular dynamics trajectories
- Visualization of protein motions

Advantages

- Particularly strong for protein dynamics
- Useful for studying collective protein motions
- Supports normal mode analysis
- Python-based
- Can work with PDB structures and trajectories
- Useful for structural biology research

Limitations

- Mainly focused on proteins and structural dynamics.
- Does not replace a full molecular dynamics engine.
- Interpretation of normal modes requires knowledge of protein dynamics.
- Some analyses may require understanding of mathematical concepts such as covariance matrices and eigenvectors.

Applications in Computational Space Biochemistry

ProDy could be useful for studying how proteins associated with astronaut health may move or change their conformations.
Potential applications include:

- **Bone health** - Studying flexibility and collective motions of proteins involved in Bone remodeling, Calcium signaling and Osteoblast/osteoclast pathways.
- **Muscle health** - Investigating protein motions involved in Muscle contraction, Muscle signaling and Protein degradation.

- **Radiation response** - Protein structural dynamics can be studied for proteins involved in DNA repair, Oxidative-stress response and Cellular stress pathways.
- **Drug discovery** -ProDy can help investigate whether a ligand or mutation may affect protein flexibility and collective motions.

Installation

!pip install -q prody

Google Colab Walkthrough:
ProDy was used to retrieve a protein structure from the Protein Data Bank and represent it in a computationally analyzable form. This demonstrates how protein structural information can be accessed and prepared for further analysis of molecular motions and flexibility. ProDy can also support advanced analyses such as normal mode analysis and principal component analysis, which are useful for studying protein dynamics. In computational space biochemistry, these approaches can help investigate proteins involved in bone remodeling, muscle function, DNA repair, oxidative stress, immune signaling, and potential astronaut-health drug targets. The practical implementation and output are provided in the accompanying Google Colab notebook.

```
Protein structure loaded successfully.
Number of atoms: 660
Number of residues: 134
```

3.4 Data Analysis Libraries.

Data analysis libraries are Python tools used to organize, process, calculate, analyze, and interpret scientific data. In computational biochemistry, researchers often work with large datasets obtained from experiments, biological databases, molecular simulations, gene-expression studies, and spaceflight research.

The three libraries included here have complementary roles:
- NumPy → numerical calculations and multidimensional arrays
- Pandas → organizing, cleaning, filtering, and analyzing tabular data
- SciPy → advanced scientific and statistical calculations

Together, they form an important foundation for computational research.

A simplified workflow is: Raw biological data → NumPy/Pandas → SciPy analysis → Results → Visualization → Biological interpretation.

1. NumPy

NumPy (Numerical Python) is an essential package in the Python programming language for numerical and scientific computations. The key element of NumPy is the NumPy array which helps store and manipulate large sets of numerical data. NumPy includes mathematical operations that are extensively used in other scientific packages in Python.

Why is it used?
NumPy is used when researchers need to perform calculations on numerical datasets.It can be used for:

- Mathematical calculations
- Array operations
- Matrix operations
- Statistical calculations
- Linear algebra
- Random number generation
- Numerical transformations
- Processing scientific datasets

In computational biochemistry, numerical data may represent:

- Protein coordinates
- Molecular simulation data
- Gene-expression values
- Biochemical measurements
- Radiation-response measurements
- Physiological measurements

Key Features

- Multidimensional arrays
- Fast numerical operations
- Mathematical functions
- Linear algebra
- Statistical functions
- Random number generation
- Matrix calculations
- Foundation for Pandas, SciPy, Matplotlib and many other libraries

Advantages

- Very fast numerical calculations
- Efficient handling of arrays
- Simple syntax

- Widely used across scientific Python
- Works with many other computational biology libraries
- Essential foundation for scientific computing

Limitations

- Mainly designed for numerical data
- Not specifically designed for biological data
- Large datasets can require considerable memory
- More advanced statistical analysis requires libraries such as SciPy or specialized packages

Applications in Computational Space Biochemistry

NumPy can support calculations involving:

- **Bone health** - Processing numerical measurements related to Bone mineral density, Calcium-related measurements and Bone biomarkers.
- **Muscle health** - Analyzing Muscle-mass measurements, Protein-expression data and Metabolic measurements.
- **Radiation response** -Processing numerical data related to Radiation exposure, DNA-damage measurements and Oxidative-stress markers.

Installation

```
!pip install -q numpy
```

Google Colab Walkthrough:

NumPy was used to store protein-expression measurements as a numerical array and calculate basic statistical properties, including the mean, range, and standard deviation. This demonstrates how biological numerical data can be efficiently processed and summarized using Python. Although the demonstration uses hypothetical values, the same approach can be applied to experimental data such as bone and muscle biomarkers, oxidative-stress measurements, DNA-damage indicators, and immune-related data. The practical implementation and output are provided in the accompanying Google Colab notebook.

```
Protein expression data: [12 15 18 14 20]
Mean: 15.8
Maximum: 20
Minimum: 12
Standard deviation: 2.85657137141714
```

2. Pandas

Pandas is a Python library designed for data manipulation and analysis, particularly structured and tabular datasets. It provides two important data structures:

- Series → one-dimensional labelled data
- DataFrame → two-dimensional table-like data

A Pandas DataFrame is similar to a spreadsheet, making it especially useful for biological datasets.

Why is it used?

Pandas can be used to:

- Import datasets
- Organize data
- Filter rows
- Select columns
- Sort data
- Handle missing values
- Calculate statistics
- Group data
- Combine datasets
- Export results

Biological and space-biological research frequently produces data in tables.
For example:

Sample	Microgravity	Bone Marker	Oxidative Stress
A	O days	15.2	4.1
B	7 days	13.8	5.3
C	14 days	11.6	6.8

Key Features

- DataFrames
- Data cleaning
- Data filtering
- Missing-value handling
- Sorting
- Grouping
- Statistical summaries
- CSV/Excel data handling
- Combining datasets

- Integration with NumPy and Matplotlib

Advantages
- Very easy to work with tabular data
- Similar to spreadsheet-based data handling
- Powerful filtering and grouping
- Handles missing data
- Supports CSV and Excel files
- Integrates well with NumPy and visualization libraries
- Useful for both small and large datasets

Limitations
- Large datasets can consume significant memory
- Not specifically designed for biological data
- Very large-scale datasets may require specialized technologies
- Complex statistical analyses may require SciPy or other libraries

Applications in Computational Space Biochemistry

Pandas can be particularly useful for organizing data from:
- NASA spaceflight experiments
- Gene-expression studies
- Microgravity experiments
- Radiation experiments
- Bone-health studies
- Muscle-health studies
- Immune-system studies
- Metabolomics and proteomics datasets
- Molecular docking results
- Molecular dynamics analyses

For example: NASA GeneLab dataset → Pandas → clean data → filter experimental groups → calculate statistics → visualize results

Installation

!pip install -q pandas

Google Colab Walkthrough:
Pandas was used to organize biological measurements into a structured DataFrame, calculate the average bone-marker value, and filter samples based on oxidative-stress measurements. This demonstrates how Pandas can be used to

organize, filter, and analyze biological datasets. Although the example uses hypothetical data, the same approach can be applied to real datasets from NASA GeneLab, gene-expression studies, and biochemical experiments. The practical implementation and output are provided in the accompanying Google Colab notebook.

3. SciPy

SciPy (Scientific Python) is an open-source Python library that provides advanced mathematical, scientific and statistical functions. It builds on NumPy and extends it with tools for scientific computing.

Why is it used?
SciPy is useful when basic numerical calculations are not enough and researchers need more advanced scientific methods.It provides tools for:

- Statistics
- Optimization
- Signal processing
- Numerical integration
- Interpolation
- Linear algebra
- Spatial analysis
- Scientific mathematics

Key Features
Important SciPy modules include:

- **scipy.stats** → statistics
- **scipy.optimize** → optimization
- **scipy.integrate** → integration
- **scipy.linalg** → linear algebra
- **scipy.signal** → signal processing
- **scipy.spatial** → spatial calculations
- **scipy.interpolate** → interpolation

Advantages

- Extensive scientific functions
- Powerful statistical tools
- Works directly with NumPy
- Useful for optimization and mathematical modelling
- Widely used in scientific research
- Integrates with Pandas and visualization libraries

Limitations

- Requires understanding of the statistical or mathematical method being used
- Incorrect statistical assumptions can produce misleading conclusions
- Not specifically designed for biological research
- More advanced analysis may require specialized statistical or bioinformatics packages

Applications in Computational Space Biochemistry

SciPy can support research involving:

- **Microgravity research** - Statistically comparing measurements between Earth control vs. microgravity
- **Bone health** - Testing whether bone-related biomarkers differ between experimental groups.
- **Muscle health** - Comparing muscle-related biochemical measurements between control and spaceflight conditions.
- **Radiation research** - Analyzing relationships between: Radiation exposure → oxidative stress → DNA damage
- **Immune research** - Statistically analyzing differences in immune-related measurements between experimental conditions.
- **Molecular research** - SciPy can also support: optimization of models, numerical calculations, distance calculations and statistical analysis of molecular data.

Installation

```
!pip install -q scipy
```

Google Colab Walkthrough:

SciPy was used to perform a statistical comparison between hypothetical control and microgravity groups and evaluate whether their measured values differed significantly. The example demonstrates how statistical tests can be applied to biological datasets and how p-values can be used to assess evidence for differences between experimental groups. Although the data are artificial and cannot establish an actual effect of microgravity, the same approach can be applied to real space-biology datasets comparing conditions such as Earth control versus microgravity, different radiation exposures, or pre- and post-flight measurements. The practical implementation and output are provided in the accompanying Google Colab notebook.

How NumPy, Pandas and SciPy Work Together

These three libraries should not be treated as completely separate tools. They often form a single data-analysis workflow.

Example: Spaceflight biological dataset
Suppose a researcher obtains a dataset containing:
Mission duration
Bone marker
Muscle marker
Oxidative stress
DNA damage
Immune marker

The workflow could be:
Pandas - Import and organize the dataset
↓
NumPy - Perform numerical calculations
↓
SciPy - Perform statistical tests
↓
Matplotlib / Plotly - Visualize the results
↓
Biological interpretation

3.5 Scientific Visualization Libraries.

Scientific visualization libraries are Python tools used to convert numerical, biological, chemical, and structural data into visual representations that are easier to understand and interpret. In computational biochemistry, visualization is important because researchers often work with complex information such as:

- Gene-expression measurements
- Protein structures
- Molecular docking results
- Molecular dynamics trajectories
- Protein–ligand interactions
- Radiation and oxidative-stress data
- Bone and muscle biomarkers
- Large biological datasets

Visualization can help identify patterns, trends, differences, structural features, and relationships that may not be obvious from numerical data alone.

The four tools in this section have different purposes:

Library	Main purpose	Type of visualization
Matplotlib	Scientific graphs and plots	Mainly 2D
Plotly	Interactive data visualization	2D and 3D
PyMOL Python API	Molecular/protein visualization	3D molecular structures
py3Dmol	Interactive molecular visualization in notebooks	3D molecular structures

1. Matplotlib

The Matplotlib package is one of the most popular visualization libraries in Python. It can plot numerous types of graphs, including lines, scatterplots, bars, histograms, among others. It is normally used in conjunction with NumPy and Pandas.

Why is it used?
Matplotlib is used to convert numerical data into graphs so that researchers can identify:

- Trends
- Differences between groups
- Relationships between variables
- Distributions
- Changes over time

For example, a researcher could plot:
Mission duration vs. bone-marker level - to visually examine how the measurement changes with time.

Key Features

- Line graphs
- Bar charts
- Scatter plots
- Histograms
- Box plots
- Error bars
- Multiple datasets
- Figure customization
- Scientific-quality figures
- Export to different formats

Advantages

- Free and open-source
- Very widely used
- Highly customizable
- Suitable for scientific publications
- Works well with NumPy and Pandas
- Supports many types of plots

Limitations

- Interactive visualization is more limited than Plotly.
- Highly customized figures can require more code.
- It is primarily focused on graphs rather than molecular 3D visualization.

Applications in Computational Space Biochemistry

Matplotlib can be used to visualize:

- Bone-loss measurements
- Muscle-loss measurements
- Oxidative-stress data
- DNA-damage measurements
- Immune-system biomarkers
- Gene-expression data
- Radiation-response data
- Molecular-dynamics measurements
- Docking scores

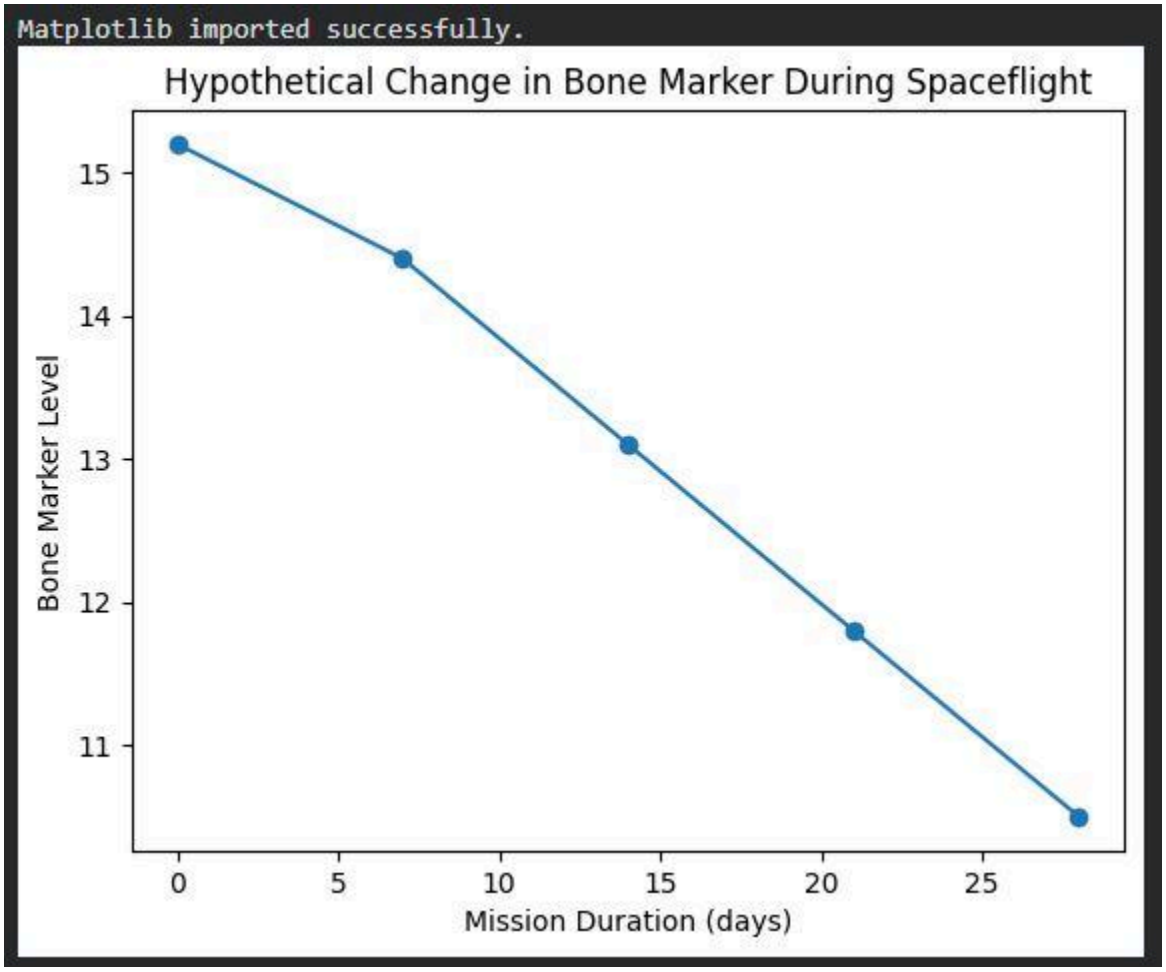
For example:
Pandas → analyze NASA GeneLab dataset → Matplotlib → plot gene-expression changes

Installation

```
!pip install -q matplotlib
```

Google Colab Walkthrough:
Matplotlib was used to create a line graph showing the relationship between mission duration and a hypothetical bone-marker value. The visualization makes it easier to identify trends and patterns in biological data than by examining numerical values alone. Although the example uses hypothetical data and does not represent an actual spaceflight finding, the same approach can be applied to

visualize real datasets related to bone loss, muscle loss, oxidative stress, DNA damage, immune-system changes, radiation exposure, and gene expression.



2. Plotly

Plotly is an interactive Python visualization library that can create a wide range of graphs, including scientific, statistical and 3D visualizations. Plotly's Python library supports interactive figures that can be displayed in notebooks and web-based environments

Why is it used?
Plotly is particularly useful when researchers want to interactively explore data. For example, users can often:

- Hover over data points
- Zoom
- Pan
- Examine individual measurements

- Interactively explore 3D plots

This is useful for large biological datasets where static graphs may hide important details.

Key Features
- Interactive graphs
- Line charts
- Scatter plots
- Bar charts
- Histograms
- Box plots
- Heatmaps
- 3D scatter plots
- 3D surface plots
- Interactive scientific visualization

Advantages
- Interactive
- Supports many chart types
- Excellent for exploratory data analysis
- Supports 3D visualization
- Works well in Jupyter/Google Colab
- Useful for presentations and dashboards

Limitations
- Interactive graphs can be more complex to customize.
- Some advanced features require additional understanding of Plotly's figure structure.
- Static publication workflows may require additional configuration.

Applications in Computational Space Biochemistry

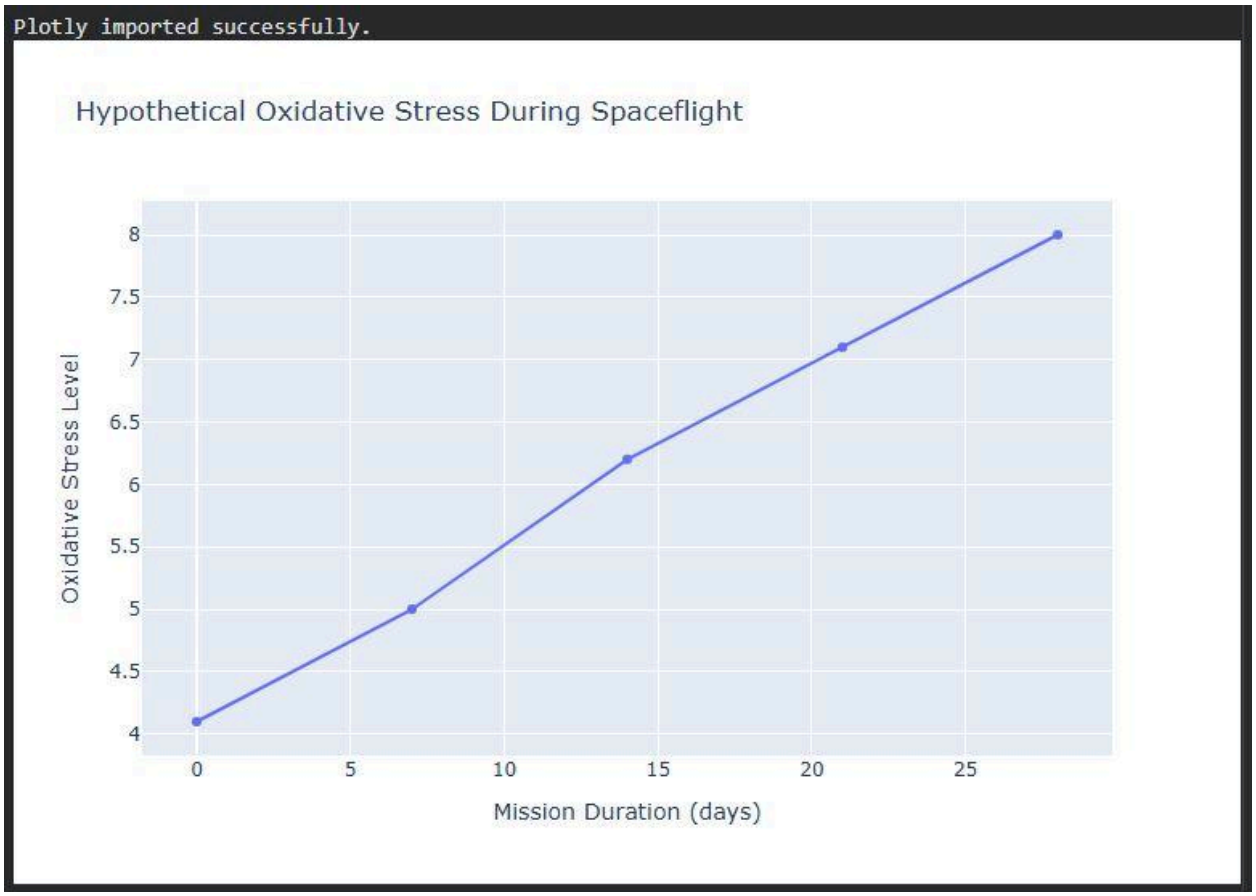
Plotly could be used to interactively explore:
- Spaceflight gene-expression datasets
- Radiation-response datasets
- Bone and muscle biomarkers
- Metabolomics data
- Molecular-dynamics measurements
- Docking-score comparisons
- Multivariable biological datasets

For example, a 3D scatter plot could potentially display relationships between:
Mission duration × radiation exposure × oxidative stress

Installation

!pip install -q plotly

Google Colab Walkthrough:
Plotly was used to create an interactive graph of hypothetical oxidative-stress measurements, allowing individual data points and their values to be explored interactively. This demonstrates how Plotly can make biological datasets easier to inspect and interpret, particularly when working with large numbers of observations. Although the example uses hypothetical data, the same approach can be applied to datasets from spaceflight experiments, NASA GeneLab, radiation studies, gene-expression experiments, metabolomics, proteomics, and bone and muscle research.



3. PyMOL Python API

PyMOL is a molecular visualization system which provides visualization of three-dimensional molecules. This tool can be used to visualize proteins, ligands, nucleic acids, and molecular interactions. One of the main features of PyMOL is Python scripting support. Users can create Python scripts to perform molecule visualization and analysis. One of the key differences of PyMOL is that it is mainly a molecular visualization tool with Python support, not a conventional Python plotting library.

Why is it used?
PyMOL can be used to visually examine:

- Protein structures
- Ligand-binding sites
- Protein–ligand complexes
- Molecular surfaces
- Residues
- Hydrogen-bond interactions
- Structural changes
- Docking poses

This is especially important after molecular docking because researchers need to visually inspect the predicted interaction between a ligand and its target protein.

Key Features

- 3D protein visualization
- Ligand visualization
- Molecular surfaces
- Cartoon representation
- Stick and sphere representations
- Residue selection
- Structural comparison
- Molecular interaction visualization
- Python scripting
- Image generation

Advantages

- Powerful molecular visualization
- Excellent for protein structures
- Useful for structural biology
- Strong molecular interaction visualization
- Python scripting allows automation
- Useful for preparing scientific figures

Limitations

- More difficult for complete beginners than simple plotting libraries.
- Installation in Google Colab can be more complicated.
- It is not primarily a statistical data-analysis tool.
- Some PyMOL features depend on the specific distribution/license.

Applications in Computational Space Biochemistry

PyMOL can be used to visualize:

- Proteins involved in bone metabolism
- Muscle-related proteins
- DNA-repair proteins
- Proteins involved in oxidative-stress responses
- Immune-system proteins
- Protein–drug docking poses
- Structural changes observed in computational studies

For example: AutoDock Vina → docking pose → PyMOL → visualize protein–ligand interaction
This can help a researcher inspect which residues are located near a docked ligand.

Installation

```
!pip install -q pymol-open-source
```

Google Colab Walkthrough:
The PyMOL Python API was used to retrieve the ubiquitin protein structure (PDB ID: 1UBQ), control its molecular representation through Python, and generate a 3D visualization of the protein. This demonstrates how PyMOL can be controlled programmatically for protein-structure visualization and analysis. The same approach can later be extended to visualize protein–ligand interactions and docking results in computational space-biochemistry research.

4. py3Dmol

Py3Dmol is a Python and Jupyter interface that helps embed 3Dmol.js, which is a molecular viewer software, in Jupyter notebooks. This tool facilitates the visualization of interactive 3D molecular structures in notebook formats. Py3Dmol is particularly suitable for Google Colab and Jupyter notebooks since 3D structures of molecules can be viewed without launching a molecular viewing application.

Why is it used?
py3Dmol can be used to:

- Display proteins in 3D
- Display ligands
- Show molecular surfaces
- Highlight residues
- Display different molecular representations
- Examine protein–ligand complexes
- Interactively rotate and zoom molecular structures

3Dmol.js supports common molecular formats such as PDB, SDF, MOL2 and XYZ and provides representations including cartoon, stick, sphere and surface styles.

Key Features

- Interactive 3D visualization
- Works inside Jupyter/Google Colab
- Protein visualization
- Ligand visualization
- Cartoon representation
- Stick representation
- Surface representation
- Atom/residue selection
- Molecular structure highlighting

Advantages

- Very easy to use in Google Colab
- Interactive
- No separate molecular-visualization application required
- Excellent for notebooks and educational demonstrations
- Useful for proteins and ligands
- Supports multiple molecular representations

Limitations

- Primarily a visualization tool rather than a molecular-analysis or simulation engine.
- It does not perform molecular docking or molecular dynamics.
- Advanced visualization may require knowledge of molecular structure and selection syntax.
- Its notebook-based interaction model is different from a full molecular visualization application.

Applications in Computational Space Biochemistry

py3Dmol can be used to visualize:

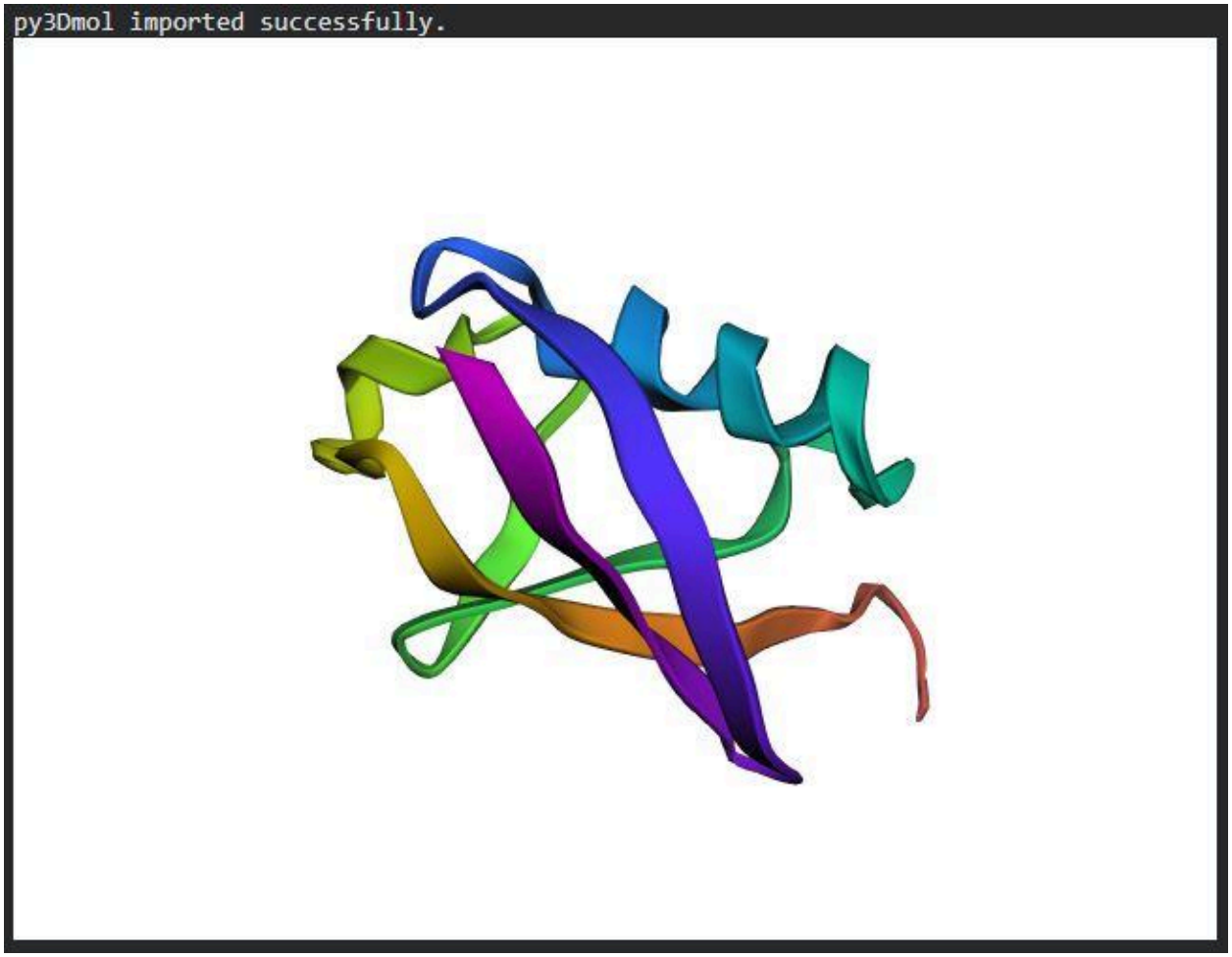
- Space-health-related proteins
- Protein structures obtained from PDB
- Docking results
- Protein–ligand interactions
- Molecular dynamics snapshots
- Structural changes
- Potential drug targets

For example: PDB → protein structure → py3Dmol → interactive visualization
or:
AutoDock Vina → docking pose → PDBQT/PDB → py3Dmol → visualize predicted binding

Installation

```
!pip install -q py3Dmol
```

Google Colab Walkthrough:
py3Dmol was used to display the three-dimensional structure of ubiquitin directly within Google Colab using structural data obtained from the Protein Data Bank (PDB). This demonstrates how py3Dmol can connect biological structural data with interactive 3D molecular visualization. Unlike Matplotlib and Plotly, which primarily visualize numerical or graphical data, py3Dmol allows researchers to visually examine the physical three-dimensional structure of molecules. The same approach can be used to visualize proteins involved in spaceflight-related biological processes, potential drug targets, protein–ligand complexes, docking results, molecular-dynamics structures, and structural changes in proteins. The practical implementation and output are provided in the accompanying Google Colab notebook.



3.6 Machine Learning , Artificial Intelligence & Network Libraries.

The Machine Learning (ML) and Artificial Intelligence (AI) libraries are Python tools which enable scientists to create machine learning algorithms that learn patterns in biological, chemical, structural, or experimental data.

In computational space biochemistry, these tools can be useful because long-duration space missions generate complex datasets involving:

- Gene expression
- Protein expression
- Metabolites
- Radiation exposure
- Bone and muscle biomarkers
- Immune-system changes
- Molecular structures
- Drug–target interactions
- Protein networks

Instead of examining every variable manually, machine-learning methods can identify patterns, relationships, classifications, and predictions from large datasets.

The five tools in this section have different purposes:

Library	Main purpose
Scikit-learn	Classical machine learning and statistical modeling
TensorFlow	Deep learning and neural networks
PyTorch	Deep learning and flexible neural-network development
DeepChem	Machine learning specifically for chemistry and drug discovery
NetworkX	Network and graph analysis

Important: NetworkX is not really a machine learning/artificial intelligence library such as Scikit-learn, TensorFlow, PyTorch, or DeepChem. NetworkX is actually a graph/network analysis package. NetworkX has been mentioned here because the biological processes, molecule interactions, gene-protein interactions, and drug-target interactions can all be described in terms of graphs/networks.

1. Scikit-learn

Scikit-learn is an open-source machine learning library written in Python for supervised and unsupervised learning, preprocessing, model selection, and evaluation. It is one of the easiest machine learning libraries for beginners.

Why is it used?
Scikit-learn can be used to:

- Classify biological samples
- Predict numerical values
- Cluster similar samples
- Reduce dataset dimensions
- Detect patterns
- Evaluate predictive models

For example, a model could be trained to classify biological samples according to whether they represent normal conditions or spaceflight-associated changes.

Key Features

- Linear regression
- Logistic regression
- Decision trees
- Random forests
- Support vector machines
- k-nearest neighbors
- k-means clustering
- Principal Component Analysis (PCA)
- Data preprocessing
- Model evaluation
- Cross-validation

Advantages

- Beginner-friendly
- Large collection of classical ML algorithms
- Good documentation
- Easy integration with NumPy and Pandas
- Useful for small and medium-sized datasets
- Provides model-evaluation tools

Limitations

- Not primarily designed for advanced deep learning
- May be less suitable for very large unstructured datasets
- Requires appropriate feature selection and preprocessing
- Model performance depends strongly on the quality of the input data

Applications in Computational Space Biochemistry

Scikit-learn could be used for:

- Classifying spaceflight biological samples
- Predicting bone-health changes
- Identifying patterns in muscle-related biomarkers
- Classifying radiation-response profiles
- Clustering gene-expression samples
- Predicting biochemical measurements
- Reducing the dimensionality of large omics datasets

Example NASA GeneLab data → Pandas → preprocessing → Scikit-learn → classification/clustering → biological interpretation

Installation

```
!pip install -q scikit-learn
```

Google Colab Walkthrough:
Scikit-learn was used to train a simple classification model on hypothetical biological data and predict the class of a new input value. The model classified the new value, 7, as belonging to class 1, demonstrating the basic workflow of training a machine-learning model and making predictions. Because the dataset is very small and hypothetical, the result is only a demonstration of the method and is not a scientifically validated biological prediction. The same workflow could be applied to larger datasets to classify spaceflight versus ground-control samples, different radiation-response groups, or biological samples with different bone and muscle profiles. The practical implementation and output are provided in the accompanying Google Colab notebook.

2. TensorFlow

TensorFlow is a freely available machine learning library used for creating and training machine learning and deep learning models.

Why is it used?
TensorFlow is particularly useful when researchers need to develop deep neural networks capable of learning complex patterns from large datasets. It can be used for:

- Classification
- Regression
- Image analysis
- Sequence analysis
- Pattern recognition
- Deep learning

Key Features

- Neural networks
- Deep learning
- Convolutional Neural Networks (CNNs)
- Recurrent/sequence models
- Model training
- GPU acceleration
- Model evaluation
- Deployment tools
- Integration with Keras

Advantages

- Powerful deep-learning framework
- Good GPU support
- Suitable for large datasets
- Strong ecosystem
- Supports model deployment
- Keras makes model development relatively accessible

Limitations

- More complex than Scikit-learn
- Deep-learning models can require substantial computational resources
- Requires more data for many applications
- Model development and tuning can be time-consuming

Applications in Computational Space Biochemistry

TensorFlow could potentially be used for:

- Predicting biological responses to spaceflight
- Gene-expression classification
- Radiation-response prediction
- Image-based analysis of cellular changes
- Predicting biochemical biomarkers
- Drug-response prediction
- Deep-learning-based molecular analysis

For example: Spaceflight omics dataset → preprocessing → TensorFlow neural network → prediction of biological response

Installation

```
!pip install -q tensorflow
```

Google Colab Walkthrough:
TensorFlow was used to create a simple neural-network model, demonstrating the basic structure of a deep-learning workflow. The practical confirms that the neural-network architecture was successfully created, although the model was not trained on real biological data. With appropriate experimental datasets, similar TensorFlow models could be developed for gene-expression classification, radiation-response prediction, biological-response prediction, image-based cellular analysis, biomarker prediction, and drug-response modeling. The practical

implementation and output are provided in the accompanying Google Colab notebook.

3. PyTorch

PyTorch is an open-source machine learning library that is commonly employed to create and train deep learning models.

Why is it used?
PyTorch is useful for developing flexible deep-learning models and is widely used in research. It can be applied to:

- Classification
- Regression
- Image analysis
- Sequence modeling
- Representation learning
- Scientific machine learning

Key Features

- Tensor computation
- Automatic differentiation
- Neural networks
- GPU acceleration
- Flexible model architecture
- Dynamic computational graphs
- Research-oriented ecosystem

Advantages

- Flexible
- Strong deep-learning capabilities
- GPU support
- Easy to experiment with custom models
- Large research community

Limitations

- More difficult for beginners than Scikit-learn
- Requires understanding of neural networks
- Large deep-learning models can require significant computational resources
- Model development can become complex

Applications in Computational Space Biochemistry

PyTorch could be used for:

- Predicting biological responses to microgravity
- Radiation-response modeling
- Omics-data analysis
- Protein-related machine learning
- Drug-response prediction
- AI-based drug discovery
- Deep-learning models for astronaut-health datasets

Installation

```
!pip install -q torch
```

Google Colab Walkthrough:
PyTorch was used to construct a simple neural-network model, demonstrating the basic structure of a deep-learning workflow. The practical confirms that the neural network was successfully created; however, the model was not trained on biological data and therefore does not produce meaningful biological predictions. With suitable datasets, PyTorch can be applied to advanced computational research involving omics data, radiation-response prediction, biological image analysis, protein-related machine learning, drug-response prediction, and AI-based drug discovery. The practical implementation and output are provided in the accompanying Google Colab notebook.

4. DeepChem

The DeepChem framework is an open-source Python library created for using machine learning and deep learning in chemistry, biology, material science, and drug discovery domains. Different from typical machine learning libraries, the DeepChem library has specialized datasets and algorithms that support molecular applications.

Why is it used?
DeepChem can be used for:

- Molecular property prediction
- Drug discovery
- Toxicity prediction
- Drug–target interaction modeling
- Molecular representation learning
- Chemical machine learning
- Graph-based molecular modeling

This makes it particularly relevant to AI-based drug discovery.

Key Features

- Molecular datasets
- Molecular featurization
- Graph neural networks
- Molecular property prediction
- Toxicity prediction
- Drug-discovery workflows
- Integration with deep-learning frameworks
- Chemical machine learning

Advantages

- Designed specifically for chemistry and biology
- Strong relevance to drug discovery
- Provides molecular featurization methods
- Supports graph-based learning
- Integrates with deep-learning frameworks
- Useful for molecular property prediction

Limitations

- More specialized than general ML libraries
- Installation and dependencies can sometimes be challenging
- Requires understanding of molecular representations
- Advanced workflows require knowledge of chemistry and machine learning

Applications in Computational Space Biochemistry

DeepChem has particularly strong potential for:

- Astronaut drug discovery
- Molecular property prediction
- Drug-target interaction prediction
- Toxicity prediction
- Radiation-protective compound screening
- Predicting compounds that may help manage spaceflight-associated health problems

For example: PubChem/ChEMBL → molecular structures → DeepChem → ML model → candidate-compound prediction
This could eventually support the identification of compounds for astronaut health research.

Installation

```
!pip install -q rdkit deepchem
```

Google Colab Walkthrough:
DeepChem was used to convert the SMILES representation of ethanol (CCO) into a 1024-dimensional molecular fingerprint**.** This demonstrates how a chemical structure can be transformed into numerical features that can be used as input for machine-learning models in molecular property prediction and drug-discovery research. The output shape, **(1, 1024)**, indicates that one molecule was processed and represented using 1024 numerical features. Although the demonstration uses only ethanol, the same approach can be applied to large collections of drug-like molecules for molecular screening, property prediction, and identification of potential compounds relevant to astronaut health and space-drug-discovery research. The practical implementation and output are provided in the accompanying Google Colab notebook.

5. NetworkX

NetworkX is a Python library for creating, manipulating and analyzing networks and graphs.

A graph consists of:
- Nodes — entities
- Edges — relationships between entities

For biological research:
- Protein = node
- Protein interaction = edge

Therefore, complex biological systems can be represented as networks.

Why is it used?
NetworkX can help researchers analyze:
- Protein–protein interaction networks
- Gene regulatory networks
- Metabolic networks
- Drug–target networks
- Biological pathways
- Molecular interaction networks

Key Features
- Graph creation

- Network analysis
- Node and edge manipulation
- Shortest-path analysis
- Centrality measures
- Community detection
- Network visualization
- Directed and undirected graphs

Advantages

- Easy to use
- Excellent for biological network analysis
- Supports many graph algorithms
- Works well with scientific Python libraries
- Useful for visualizing relationships between biological entities

Limitations

- Not a conventional machine-learning library
- Very large networks can become computationally demanding
- Network analysis requires appropriate biological interpretation
- Visualization of extremely large networks can become difficult

Applications in Computational Space Biochemistry

NetworkX can be used to study:

- Protein–protein interaction networks
- Immune-system pathways
- Oxidative-stress pathways
- DNA-repair networks
- Metabolic networks
- Drug–target relationships
- Biological pathways affected by microgravity or radiation

For example: Gene-expression data → identify altered genes → construct interaction network → NetworkX → identify important biological nodes

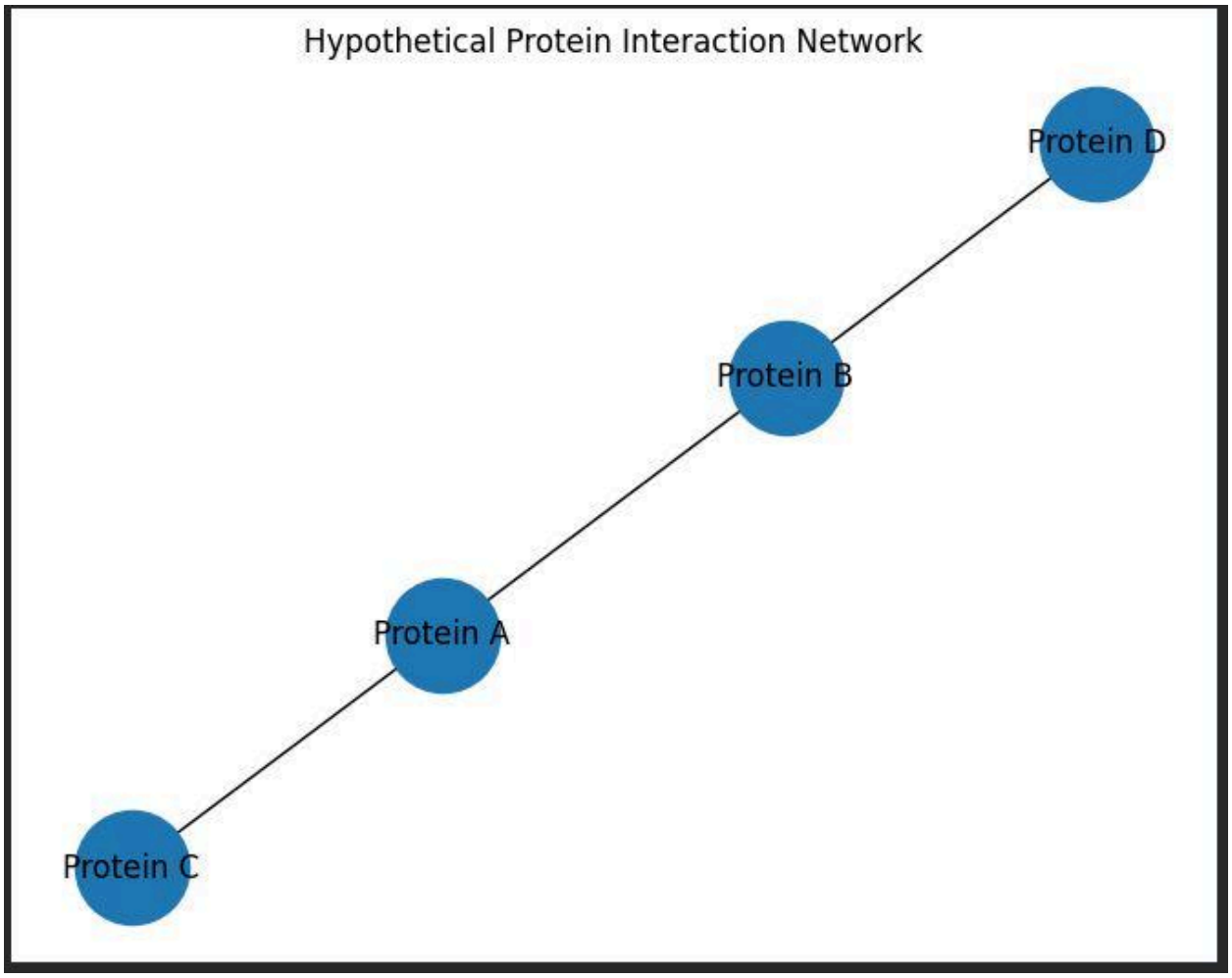
This could help researchers identify key proteins or pathways affected during spaceflight.

Installation

```
!pip install -q networkx
```

Google Colab Walkthrough:
NetworkX was used to create and visualize a simple biological interaction network containing four hypothetical proteins and three interactions. The demonstration shows how proteins can be represented as nodes and their interactions as edges, allowing biological relationships to be analyzed computationally. Although the example uses a small hypothetical network, the same approach can be applied to much larger biological networks involving protein–protein interactions, immune-system networks, DNA-repair networks, oxidative-stress pathways, metabolic networks, drug–target relationships, and pathways affected by microgravity or radiation.

```
Number of nodes: 4
Number of edges: 3
```



4. Biological and Chemical Databases for Computational Space Biochemistry.

A database refers to a systematic set of data that is structured, stored, organized, and processed in such a way that the user can search, retrieve, compare, and analyze the information. In the biological and chemical sciences, databases contain vast amounts of scientific data that would be hard to collect and process manually. Such tools make it possible for researchers to access the pre-existing experimental and computational data and perform additional analysis on it.

Biological databases are specialized databases that store information related to living organisms and biological systems. They may contain information about genes, DNA and RNA sequences, proteins, gene expression, biological pathways, molecular interactions, and biological experiments.

Chemical databases store information about chemical compounds and their properties. They may provide molecular structures, chemical names, molecular formulas, SMILES, biological activities, drug information, and compound–target relationships.

Hence, these databases act as significant tools for bioinformatics, computational biology, structural biology, drug discovery, and systems biology.

Depending on the purpose of a database, it may contain:

- DNA and RNA sequences
- Gene and genome information
- Protein sequences
- Protein three-dimensional structures
- Gene-expression data
- Molecular and chemical structures
- Drug and compound information
- Protein–ligand interactions
- Protein–protein interactions
- Biological pathways
- Metabolic pathways
- Disease-related information
- Experimental datasets
- Spaceflight- and microgravity-related biological datasets

For instance, the researcher who is working to understand the impact of microgravity on bone cells needs information about genes from GeneLab, proteins

from UniProt, proteins structure from Protein Data Bank, chemicals from PubChem/ChEMBL, and pathways from KEGG or Reactome.

Today's biological experiments generate huge amounts of data and it is not possible for scientists to manually gather, curate, and analyze all the data that is generated. Thus, Biological and chemical databases make information accessible through uniform interfaces. Scientists can search databases for sequences, structures, molecules, experimental datasets, or pathways, and conduct analyses using computational techniques. Scientists can also correlate data available from different sources using databases. For example, a protein detected through a gene expression study can be correlated with its sequences, 3-D structures, biological functions, interacting molecules, and pathways.

Moreover, a computational interface between the scientific database and analysis is provided by the Python language. Rather than manually downloading the data from the database, Python allows you to interact with the databases using Application Programming Interface, Web Services, or downloaded data files. The downloaded data can be analyzed using Python modules like Pandas, NumPy, Biopython, SciPy, Matplotlib, and Plotly.

A simplified computational workflow is:

Scientific Database → Retrieve/Download Data → Python → Data Cleaning & Processing → Analysis → Visualization → Biological Interpretation

For example: PDB → Protein structure → Python/BioPandas/MDAnalysis → Structural analysis → Visualization

Or

PubChem/ChEMBL → Chemical compounds → Python/RDKit → Molecular analysis → Molecular filtering → Docking/Drug discovery

Furthermore, scientific databases are especially useful in space biochemistry because of the complexity of biological and molecular data resulting from spaceflight experiments.

Databases allow researchers to study alterations in:

- Microgravity
- Space radiation
- Bone loss
- Muscle loss
- Oxidative stress

- DNA damage
- Immune-system changes
- Metabolic changes
- Drug responses

For example, a simplified space-biochemistry workflow could be:

NASA GeneLab → Spaceflight biological data → Python analysis → Identify altered genes/proteins → Pathway analysis → Biological interpretation

In this project, the following eight databases are studied:

Database	Main Information
NASA GeneLab / OSDR	Spaceflight and space-biology datasets
Protein Data Bank (PDB)	3D structures of proteins, nucleic acids and complexes
UniProt	Protein sequences and functional information
PubChem	Chemical compounds and molecular properties
ChEMBL	Bioactive molecules, drug-related information and biological activities
KEGG	Biological and metabolic pathways
Reactome	Biological pathways and molecular reactions
NCBI GEO	Gene-expression and functional genomics datasets

1. NASA GeneLab / NASA Open Science Data Repository (OSDR)

NASA GeneLab is a NASA resource for accessing biological data generated from spaceflight and space-relevant experiments. GeneLab has been incorporated into the broader **NASA Open Science Data Repository (OSDR)**, which contains multimodal space-life-science data. OSDR includes omics, physiological, behavioral, biomedical, bioimaging, environmental and other experimental data from humans, animals, plants, microbes and other biological models. The repository is particularly valuable for studying how spaceflight conditions such as microgravity and radiation affect biological systems.

NASA OSDR/GeneLab contain:

- Gene-expression datasets
- RNA-sequencing data
- Genomic and transcriptomic information
- Proteomic and metabolomic data
- Physiological and phenotypic measurements
- Experimental metadata
- Data from humans, animals, plants and microorganisms
- Spaceflight and ground-control experimental data
- Raw and processed datasets

The OSDR Biological Data API provides programmatic access to datasets, metadata and data through REST and query interfaces, with tabular outputs available through the query interface.

Why is it useful?

The GeneLab/OSDR from NASA helps scientists use data obtained from expensive and hard-to-repeat spaceflight experiments. Rather than repeating a spaceflight experiment for each research question, scientists can use existing datasets to study how biology reacts in the conditions of spaceflight.

For example, researchers can investigate:

- Microgravity-induced changes in gene expression
- Bone-loss mechanisms
- Muscle loss
- Oxidative stress
- DNA damage
- Immune-system alterations
- Radiation responses
- Metabolic changes

Accessing NASA OSDR using Python

NASA provides a Biological Data API that allows researchers to programmatically access OSDR data. The API can be accessed using Python libraries such as `requests`.

A simplified workflow is: **OSDR API → retrieve dataset information → obtain data/file information → Python → process data → analyze → visualize**

The API supports REST-based access to datasets and query-based access to metadata and data.

Relevance to computational space biochemistry

NASA OSDR is the most directly relevant database for this project because it contains biological data generated from spaceflight and space-relevant experiments.

A possible workflow is: NASA OSDR → spaceflight dataset → Python/Pandas → statistical analysis → visualization → biological interpretation

This can help researchers identify molecular and biochemical changes associated with microgravity, radiation and other spaceflight conditions.

Google Colab Walkthrough: The NASA OSDR API was accessed successfully using Python, and information about the available datasets was retrieved. The first 20 available dataset identifiers were then displayed programmatically. **Attempts to retrieve a specific gene-expression file were unsuccessful because subsequent file-specific API requests returned errors.** Therefore, the Colab demonstration was limited to **API access and dataset discovery**, rather than downloading and analyzing a gene-expression dataset.

```
NASA OSDR API Status Code: 200
NASA OSDR API is accessible successfully.

Total datasets retrieved: 631

First 20 available datasets:
OSD-1
OSD-2
OSD-3
OSD-4
OSD-5
OSD-6
OSD-7
OSD-8
OSD-9
OSD-11
OSD-12
OSD-13
OSD-14
OSD-15
OSD-16
OSD-17
OSD-18
OSD-19
OSD-20
OSD-21
```

2. Protein Data Bank (PDB)

The Protein Data Bank (PDB) is a major archive of experimentally determined three-dimensional structures of biological macromolecules and their complexes. These include proteins, nucleic acids and complexes containing small molecules such as ligands, cofactors and ions.

PDB provides structural information including:

- Protein structures
- DNA and RNA structures
- Protein–ligand complexes
- Protein–protein complexes
- Atomic coordinates
- Chains and residues
- Ligands
- Cofactors
- Structural annotations
- Experimental information

The database represents molecular structures at different hierarchical levels, from atoms and residues to chains and biological assemblies.

Why is it useful?

PDB is essential for structural biology and computational drug discovery.

Researchers can use PDB structures to:

- Visualize proteins
- Study active sites
- Analyze protein structure
- Investigate protein–ligand interactions
- Perform molecular docking
- Perform molecular dynamics simulations
- Study structural changes associated with disease

Accessing PDB using Python

The Protein Data Bank (PDB) can be accessed programmatically in Google Colab using **Biopython** and **py3Dmol**. In this demonstration, the PDB ID **1UBQ**, corresponding to ubiquitin, was specified and Biopython's `PDBList` was used to retrieve the protein structure file from the PDB. The downloaded PDB file was then

read using Python and passed to py3Dmol, which generated an interactive three-dimensional visualization of the protein structure. This demonstrates how researchers can retrieve structural data directly from a biological database and visualize it computationally, providing a foundation for further protein-structure analysis, molecular docking, and molecular dynamics studies.

Relevance to computational space biochemistry

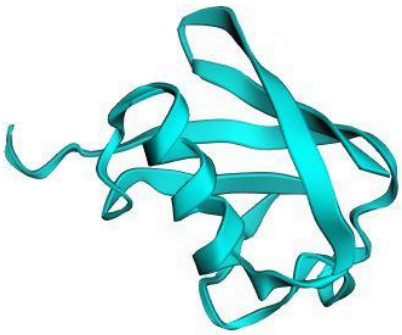
PDB structures can be used to study proteins involved in:

- Bone metabolism
- Muscle metabolism
- Oxidative stress
- DNA repair
- Immune responses
- Radiation responses

A possible workflow is: PDB → protein structure → structural analysis → docking/MD simulation → biological interpretation

Google Colab Walkthrough: In the Colab notebook, a PDB structure such as 1UBQ (ubiquitin) can be retrieved and loaded into Python. The structure can then be examined using BioPandas or visualized using PyMOL/py3Dmol. This demonstrates how a protein structure can move from a public structural database into a computational analysis workflow.

```
Downloading PDB structure '1ubq'...
PDB structure downloaded successfully.
PDB ID: 1UBQ
```



3. UniProt

UniProt is a major protein-information resource providing protein sequences and functional annotations. It is widely used to obtain information about proteins from different organisms, including their sequences, functions and associated biological information.

UniProt provides information such as:

- Protein sequences
- Protein names
- Gene names
- Organisms
- Protein functions
- Subcellular locations
- Protein families
- Domains
- Enzyme information
- Sequence features
- Cross-references to other databases
- Evidence supporting protein annotations

UniProt provides REST APIs through which individual entries and query results can be retrieved programmatically in formats including FASTA, XML, RDF, GFF and TSV.

Why is it useful?

UniProt is useful when researchers need to understand what a protein is and what it does.

For example, after identifying a gene from a gene-expression dataset, researchers can use UniProt to determine:

Gene → protein → function → biological pathway

This makes UniProt particularly useful for interpreting genomic and transcriptomic results.

Accessing UniProt using Python

UniProt was accessed using Python's `requests` library and the UniProt REST API. The accession number **P0A7V0** was used to retrieve the FASTA sequence of the E. *coli* uS2 ribosomal protein. The output successfully displayed the protein's

identification information followed by its amino acid sequence, demonstrating that protein sequence data can be retrieved programmatically from UniProt.

Relevance to computational space biochemistry

UniProt can help characterize proteins associated with:

- Bone remodeling
- Muscle function
- Oxidative stress
- DNA repair
- Immune responses
- Radiation responses

A possible workflow is: NASA GeneLab → altered gene → UniProt → corresponding protein → functional interpretation

Google Colab Walkthrough: The UniProt API was accessed using Python to retrieve the protein sequence with accession number P0A7V0. The output displayed the protein's UniProt accession, entry name, organism, gene name, evidence level, sequence version, and amino-acid sequence in FASTA format. This demonstrates how protein sequence information can be retrieved programmatically from UniProt and subsequently analyzed using Python. Such protein sequences can be used in computational studies of proteins involved in bone metabolism, muscle biology, DNA repair, oxidative stress, and immune responses.

4. PubChem

PubChem is a chemical database maintained by the National Center for Biotechnology Information (NCBI). It provides information about chemical substances and compounds. It is widely used for retrieving chemical structures and molecular properties.

PubChem provides information including:

- Chemical names
- Molecular formulas
- Molecular weights
- Chemical structures
- SMILES
- InChI
- InChI Keys
- Compound identifiers

- Chemical properties
- Biological activities
- Links to related scientific information

Why is it useful?

PubChem provides easily accessible chemical information for computational chemistry and drug-discovery workflows.

Researchers can obtain a compound's structure and then use Python tools such as RDKit or Open Babel for further computational analysis.

For example: PubChem → chemical structure → RDKit → molecular properties → drug screening

Accessing PubChem using Python

PubChem was accessed using Python's `requests` library and the PubChem PUG REST API. In this demonstration, **ethanol** was selected as the compound, and its molecular formula, molecular weight, and SMILES representation were retrieved programmatically. The JSON response was processed to extract these chemical properties, demonstrating how molecular information can be obtained directly from PubChem for further computational analysis.

Relevance to computational space biochemistry

PubChem can provide candidate compounds for computational studies involving astronaut health. For example, compounds could be investigated for potential applications related to:

- Bone loss
- Muscle loss
- Oxidative stress
- Radiation protection
- Inflammation
- Immune-system changes

A possible workflow is: PubChem → compound → RDKit/Open Babel → molecular preparation → docking → candidate evaluation

Google Colab Walkthrough: The PubChem API was accessed using Python to retrieve chemical information for **ethanol**. The output provided its molecular formula, molecular weight, and SMILES representation in a machine-readable format. The SMILES representation can then be used with tools such as RDKit and

DeepChem for molecular analysis and machine-learning applications. This demonstrates how chemical information can be retrieved from PubChem and incorporated into computational drug-discovery workflows relevant to astronaut health.

```
Compound: ethanol
Molecular Formula: C2H6O
Molecular Weight: 46.07
SMILES: CCO
```

5. ChEMBL

ChEMBL is a database of bioactive molecules containing information about chemical compounds and their biological activities. It is particularly valuable for drug discovery and chemical biology because it connects compounds with biological targets and experimental activity information.

ChEMBL contains information such as:

- Chemical compounds
- Molecular structures
- Compound identifiers
- Drug-related information
- Biological targets
- Bioactivity measurements
- Assay information
- Mechanisms of action
- Clinical information
- Compound–target relationships

Why is it useful?

ChEMBL helps researchers investigate the relationship between: Chemical compound → biological target → activity

This makes it particularly useful for:

- Drug discovery
- Virtual screening
- Molecular docking
- Compound prioritization
- Structure–activity relationship studies

Accessing ChEMBL using Python

ChEMBL was accessed using Python's `requests` library and the ChEMBL REST API. In this demonstration, the ChEMBL ID **CHEMBL25**, corresponding to **aspirin**, was used to retrieve molecular information. The program successfully retrieved the molecule's ChEMBL ID, molecule type, preferred name, molecular structure, and maximum clinical phase. This demonstrates how chemical and drug-related information can be obtained programmatically from ChEMBL for further computational drug-discovery and molecular analysis.

Relevance to computational space biochemistry

ChEMBL can be used to identify compounds with known biological activity that may be relevant to astronaut health.

For example: ChEMBL → bioactive compounds → target selection → RDKit → ligand preparation → AutoDock Vina → docking analysis

This could support research into potential therapeutic strategies for spaceflight-associated biological changes.

Google Colab Walkthrough: The ChEMBL API was accessed using Python to retrieve information for **CHEMBL25**, corresponding to aspirin. The output confirmed successful data retrieval and provided details such as the molecule's preferred name, molecular structure, canonical SMILES, InChI, InChI Key, and maximum clinical phase. This demonstrates how ChEMBL can be used programmatically to obtain structured chemical and drug-related information for cheminformatics, molecular analysis, drug discovery, and computational space-biochemistry research.

6. KEGG

KEGG (Kyoto Encyclopedia of Genes and Genomes) is a biological database system used to understand biological functions and pathways at the molecular level. It connects genes, proteins, compounds, reactions and biological pathways.

KEGG provides information about:

- Metabolic pathways
- Cellular processes
- Biological pathways
- Genes
- Proteins
- Enzymes
- Chemical compounds

- Reactions
- Disease-related pathways
- Organism-specific pathways

Why is it useful?

KEGG is useful for understanding how individual genes or proteins participate in larger biological systems.

For example, instead of studying one protein independently, researchers can determine which pathway it belongs to and what other biological components interact with it. This is particularly useful in systems biology and pathway analysis.

Accessing KEGG using Python

KEGG was accessed using Python's `requests` library and the KEGG REST API. In this demonstration, the human **Cell Cycle pathway (hsa04110)** was selected and its pathway information was retrieved programmatically. The output provided details such as the pathway name, description, classification, and pathway map information, demonstrating how biological pathway data can be obtained from KEGG for further computational analysis.

Relevance to computational space biochemistry

KEGG can help researchers interpret biological changes identified in spaceflight datasets.

For example: Altered genes → KEGG → affected pathways → biological interpretation

Pathway analysis can help investigate processes associated with:

- Cell cycle
- DNA damage
- Oxidative stress
- Energy metabolism
- Immune responses
- Bone metabolism

This makes KEGG useful for connecting individual molecular changes with larger biological processes

Google Colab Walkthrough: The KEGG REST API was accessed using Python to retrieve information about the human **Cell Cycle pathway (hsa04110)**. The output

confirmed successful retrieval and provided pathway identifiers, description, classification, and information about associated biological processes such as DNA replication, mitosis, cyclin-CDK regulation, and DNA-damage responses. This demonstrates how pathway information can be obtained programmatically and used in computational space biochemistry to investigate cellular responses potentially affected by microgravity and space radiation.

7. Reactome

Reactome is a curated database of biological pathways and molecular reactions. It provides detailed information about how proteins, genes and other biological molecules participate in cellular processes.

Reactome includes information about:

- Biological pathways
- Molecular reactions
- Proteins
- Genes
- Complexes
- Cellular processes
- Metabolic pathways
- Signaling pathways
- Disease-related processes
- Species-specific biological processes

Why is it useful?

Reactome helps researchers move from individual molecular findings to biological pathway interpretation. For example, if a spaceflight experiment identifies several genes with altered expression, Reactome can help determine whether these genes are concentrated within a particular biological pathway.

Accessing Reactome using Python

Reactome was accessed using Python's `requests` library and the Reactome Content Service API. In this demonstration, the Reactome identifier **R-HSA-109581** was used to retrieve information about a specific human biological pathway or reaction. The returned JSON data was processed to obtain the pathway name and stable Reactome ID, demonstrating how curated biological pathway information can be accessed programmatically for further computational analysis.

Relevance to computational space biochemistry

Reactome can be used to interpret molecular changes associated with:

- Microgravity
- Radiation exposure
- Oxidative stress
- DNA damage
- Immune-system changes
- Cellular signaling
- Metabolic adaptation

A possible workflow is: Gene-expression data → altered genes → Reactome → affected pathways → biological interpretation

Reactome therefore complements databases such as UniProt and KEGG by providing detailed pathway-level context.

Google Colab Walkthrough: The Reactome API was accessed using Python to retrieve information for the selected human Reactome entry **R-HSA-109581**. The output successfully displayed the pathway name and its stable Reactome identifier, demonstrating that curated biological pathway information can be retrieved programmatically. Such information can support computational studies of immune responses, DNA repair, metabolism, cellular signaling, and oxidative-stress pathways relevant to spaceflight research.

```
Reactome data retrieved successfully.
Pathway name: Apoptosis
Pathway ID: R-HSA-109581
```

8. NCBI GEO

NCBI Gene Expression Omnibus (GEO) is a public repository for functional genomics and gene-expression data. It contains datasets generated from experiments such as microarray and high-throughput sequencing studies.

GEO contains information such as:

- Gene-expression datasets
- Microarray data
- RNA-sequencing-related datasets
- Sample information
- Experimental conditions
- Gene-expression measurements
- Study metadata

- Organism information
- Processed and submitted experimental data

Why is it useful?

GEO allows researchers to access previously published gene-expression experiments and perform independent computational analyses.

Researchers can compare:

- Control vs experimental samples
- Healthy vs diseased samples
- Different treatments
- Different environmental conditions
- Different time points

This makes GEO valuable for validating findings and performing comparative studies.

Accessing NCBI GEO using Python

NCBI Gene Expression Omnibus (GEO) was accessed using Python with the **GEOparse** library. In this demonstration, the GEO dataset **GSE1007** was downloaded and loaded programmatically into Google Colab. The code then displayed the dataset ID, number of samples, and number of platforms available in the dataset. This demonstrates how gene-expression datasets can be accessed and examined using Python for downstream analysis.

Relevance to computational space biochemistry

GEO can provide complementary gene-expression datasets that can be compared with spaceflight-related findings.

For example: NASA GeneLab dataset → altered gene → GEO → comparable terrestrial experiment → comparison

This could help determine whether molecular changes observed during spaceflight are also associated with conditions such as:

- Oxidative stress
- Inflammation
- Radiation exposure
- Muscle loss
- Bone-related processes

- Cellular stress

Google Colab Walkthrough: The NCBI Gene Expression Omnibus (GEO) dataset **GSE1007** was successfully loaded into Google Colab using the GEOparse Python library. The output displayed the dataset ID along with the number of samples and experimental platforms included in the study. This demonstrates how publicly available gene-expression datasets can be retrieved programmatically and prepared for further analysis in Python. GEO can be used to investigate gene-expression changes associated with biological stress, immune responses, oxidative stress, muscle biology, and other physiological processes, and can complement spaceflight-specific resources such as NASA GeneLab.

5. Recommendations for Learning and Research.

From a **Life Sciences (Biochemistry) student perspective**, computational biology should be learned as a complementary skill that connects biological knowledge with programming, data analysis, and computational research. Students do not need to become expert programmers initially; instead, they should gradually develop computational skills while applying them to biological problems.

1. Strengthen Basic Biological Concepts

Students should first develop a strong understanding of **cell biology, molecular biology, genetics, biochemistry, physiology, and microbiology**. These concepts help students understand and interpret computational results correctly.

2. Learn Basic Programming

It is recommended that students start by learning **basics of Python** such as variables, loops, functions, data structures, file operations, and basics of data handling. **Google Colab** is a good platform for learning Python without the need to install any complicated software.

3. Develop Data-Analysis Skills

Students should learn commonly used Python libraries such as **NumPy, Pandas, SciPy, and Matplotlib**. These tools can be used to organize biological data, perform statistical analysis, identify trends, and create scientific visualizations.

4. Become Familiar with Biological Databases

Students should learn how to use databases such as **UniProt, PDB, PubChem, ChEMBL, KEGG, Reactome, NCBI GEO, and NASA OSDR/GeneLab**. Understanding

what information each database contains and how to retrieve it using Python is an important computational biology skill.

5. Explore Bioinformatics and Structural Biology

After developing basic computational skills, students can explore **Biopython, BioPandas, RDKit, Open Babel, PyMOL, and py3Dmol**. These tools can help students work with biological sequences, protein structures, chemical compounds, and molecular visualizations.

6. Progress Towards Advanced Computational Methods

Students interested in computational research can gradually move towards **molecular docking, molecular dynamics, network analysis, and machine learning** using tools such as AutoDock Vina, OpenMM, MDAnalysis, MDTraj, NetworkX, scikit-learn, TensorFlow, PyTorch, and DeepChem.

7. Learn Through Small Research Projects

Instead of trying to learn many tools simultaneously, students should complete **small, well-defined projects**. Examples include analysing a gene-expression dataset, comparing protein structures, studying a biological pathway, analysing molecular properties, or creating a protein–ligand visualization.

8. Connect Computational Skills with Biological Questions

Computational tools should be used to answer biological questions rather than being learned only as software packages. A useful approach is:

Biological question → Database → Data retrieval → Python analysis → Visualization → Biological interpretation

9. Develop Reproducible Research Practices

Students should maintain organized **Google Colab notebooks**, document their code, record database identifiers and sources, explain their outputs, and clearly distinguish between real experimental data and hypothetical examples. These practices improve the reliability and reproducibility of computational research.

10. Explore Interdisciplinary Research Areas

Once the fundamentals are established, Life Sciences students can explore areas such as **bioinformatics, computational biology, drug discovery, structural biology,**

systems biology, personalized medicine, biotechnology, and space biology according to their interests.

Recommended Learning Path:

Life Sciences Foundation → Python → Data Analysis → Statistics & Visualization → Biological Databases → Bioinformatics → Structural Biology → Docking/MD → Machine Learning → Independent Research Project

Thus, for Life Sciences students, computational biology should be viewed as an **extension of biological knowledge rather than a replacement for laboratory science**. Developing the ability to combine biological understanding with programming, databases, data analysis, and computational modelling can help students participate in modern interdisciplinary research and prepare for further study or careers in **bioinformatics, biotechnology, computational biology, drug discovery, systems biology, and space-related life-science research**.

6. Proposed Google Colab Workflow for Computational Space Biochemistry Research.

Based on the databases and Python libraries explored in this work, a simple computational workflow is proposed for future space-biochemistry studies. The workflow would begin with selecting a biological question related to spaceflight, such as changes in gene expression, protein structure, oxidative stress, or bone and muscle biology. Relevant data could then be obtained from databases such as NASA OSDR/GeneLab, NCBI GEO, UniProt, PDB, PubChem, or KEGG and processed using Python in Google Colab.

Basic workflow:

Biological question → Database → Data retrieval → Data processing → Statistical analysis → Visualization → Biological interpretation

Different Python tools could be selected depending on the type of data. For example, **Pandas and NumPy** could be used for data handling, **SciPy** for statistical analysis, **Matplotlib and Plotly** for visualization, and **KEGG or Reactome** for pathway interpretation. For structural or drug-discovery studies, **PDB, RDKit, AutoDock Vina, and OpenMM** could be incorporated.

Example application: A protein associated with **bone-related biological processes** could be selected for computational investigation. Information about the protein could first be obtained from **UniProt**, followed by retrieval of its experimentally determined three-dimensional structure from the **Protein Data Bank (PDB)**, where

available. The structure could then be visualized using **py3Dmol**. Potential compounds could subsequently be obtained from **PubChem or ChEMBL** and examined using **RDKit**. In a more advanced study, the selected protein and compounds could be prepared for molecular docking using **Meeko and AutoDock Vina**.

This workflow illustrates how biological databases and computational tools could be connected to investigate potential molecular targets relevant to spaceflight-associated health problems such as bone loss. However, this is only a proposed computational workflow and not a completed research experiment. It does not establish that any particular compound is effective against bone loss or that a selected protein is a validated therapeutic target.

7. Conclusion.

Through this project, I was able to learn more about computational biology and its potential application in space biochemistry. Various biological and chemical databases including NASA OSDR/GeneLab, NCBI GEO, UniProt, PDB, PubChem, ChEMBL, KEGG, and Reactome were reviewed. Multiple Python libraries that could be applied to sequences, proteins, chemical compounds, biological data, statistics, visualization, molecular structures, and machine learning were also looked into.

The work showed that computational tools can help Life Sciences students collect, organize, analyze, and visualize biological information without relying only on manual analysis. These methods could be useful for studying topics related to spaceflight such as bone loss, muscle changes, oxidative stress, DNA damage, immune-system changes, and drug discovery. **However, most demonstrations in this project were basic examples, and some used hypothetical data, so they should not be considered as actual experimental findings.**

In general, this project was useful for showing how biology, databases, Python, and computing could all be brought together. This is a good start for Life Science students interested in computational biology and who wish to slowly build their knowledge to tackle more complex topics like space biochemistry, bioinformatics, and drug discovery.

8. References.

1. The Human Body in Space - NASA
2. Fundamental Biological Features of Spaceflight: Advancing the Field to Enable Deep Space Exploration - PMC

3. Integrating bioinformatic strategies in spatial life science research - PMC
4. Biopython
5. scikit-bio
6. Pandas
7. RDKit
8. Open Babel
9. AutoDock Vina
10. meeko documentation
11. MDAnalysis
12. MDTraj
13. OpenMM
14. PyMOL
15. 3Dmol.js
16. NumPy
17. Pandas
18. SciPy
19. Matplotlib
20. Open Source Python Graphing & Data Visualization Library | Plotly
21. Scikit-learn
22. TensorFlow
23. PyTorch
24. DeepChem
25. NetworkX
26. About OSDR - NASA Science
27. RCSB PDB
28. UniProt
29. PubChem
30. ChEMBL
31. KEGG: Kyoto Encyclopedia of Genes and Genomes
32. Reactome Pathway Database
33. Home - GEO - NCBI

9. Google Colab Link

Google Colab Practical Notebook: Computational Space Biochemistry by MAS...