Group Project Guide - Python Programming & Version Control

Overview

The group project is a core component of this course where you'll apply everything you've learned to create a Python application. You'll work in teams of 4-5 students to design, implement, and present a scientific computing project.

Project Duration: Weeks 1-6
Presentation: Week 6 (15 minutes + 5 minutes Q&A)
Weight: 30% of final grade

Learning Objectives

By completing this project, you will:

Apply object-oriented programming principles to a problem
Use Git and GitHub for collaborative development
Analyze and visualize scientific data using pandas and matplotlib
Write clean, documented, and well-structured code
Present technical work to an audience
Work effectively in a team on a coding project

Project Requirements

1. Technical Requirements

Your project must include:

Object-Oriented Design

- At least 3 custom classes with meaningful relationships
- Use of inheritance OR composition (or both)
- Implementation of appropriate magic methods (e.g., `__str__`, `__repr__`, `__eq__`)
- Properties for data validation/encapsulation where appropriate

Data Handling

- Read data from files (CSV, TXT, JSON, or other formats)
- Use pandas DataFrames for data analysis
- Handle potential errors (file not found, invalid data, etc.)

Visualization

- At least 2-3 different plot types using matplotlib

- Clear labels, titles, and legends on all plots
- Plots should communicate scientific insights

Version Control

- Code hosted on GitHub
- Regular commits throughout the project (not all at the end!)
- Meaningful commit messages
- Evidence of collaboration (multiple contributors)

Documentation

- Docstrings for all classes and methods
- README file with:
  - Project description
  - Installation instructions
  - Usage examples
  - Team member contributions
- Comments for complex code sections

Testing

- Code should run without errors
- Test with sample data to verify correctness
- Handle edge cases (empty data, invalid inputs, etc.)

---

2. Scope Guidelines

Appropriate Scope:

- Achievable in 4-6 weeks with team of 4-5 students
- Focuses on demonstrating course concepts
- Has a clear scientific application
- Complex enough to showcase skills, but not overwhelming

Too Small:

- Single class with basic functions
- No data analysis or visualization
- Can be completed in 2-3 hours

Too Large:

- Machine learning model with extensive training
- Web application with database backend
- Requires external APIs or services not covered in course

Rule of Thumb: If you can implement the core functionality in Week 2, your scope is too small. If you're not sure you can finish by Week 5, your scope is too large.

Project Timeline

Week 1: Form Teams & Choose Topic

Action Items:

- Form groups of 4-5 students
- Brainstorm project ideas
- Choose your project topic
- Identify what data you'll need
- Fill out the group contract
- Set up GitHub repository

Deliverable: Upload your group contract on Canvas

Week 2: Design & Setup

Action Items:

- Design your class structure (draw a class diagram)
- Start README
- Find/create sample data for testing
- Implement core classes and methods

Week 3: Core Implementation

Action Items:

- Begin data reading functionality
- Write tests for basic functionality
- Regular commits to GitHub

Milestone: Core classes working with sample data

Week 4: Data Analysis & Visualization

Action Items:

- Implement data analysis features
- Create visualizations with matplotlib
- Add error handling
- Begin documentation

Milestone: Full functionality working, first plots generated

Deliverable: Submit your GitHub repository for peer review

Week 5: Finalization

Action Items:

- Complete all features
- Polish documentation (README, docstrings)
- Create presentation slides
- Practice presentation as a team
- Final testing and debugging
- Integrate the recieved feedback

Deliverable:

- Final code committed to GitHub
- README complete
- Presentation slides ready

Week 6: Presentations

Action Items:

- Present your project (15 min)
- Answer questions (5 min)
- Watch other teams present

Deliverable: Live presentation + working demo

Deliverable: Submit GitHub Link for your finalised poject (as release)

Project Ideas by Scientific Domain

Biology / Life Sciences

1. DNA/Protein Sequence Analyzer

- Classes: Sequence (parent), DNA, RNA, Protein
- Features: GC content, ORF finding, translation, motif search
- Data: FASTA files
- Plots: GC distribution, sequence length histograms, motif locations

2. Population Dynamics Simulator

- Classes: Species, Population, Ecosystem

- Features: Logistic growth, predator-prey models, competition
- Data: Initial populations, parameters from literature
- Plots: Population over time, phase space diagrams

3. **Phylogenetic Tree Builder**
    - Classes: Species, TreeNode, PhylogeneticTree
    - Features: Distance calculation, tree construction, visualization
    - Data: Sequence alignments or distance matrices
    - Plots: Tree visualization, distance heatmaps

Chemistry

1. **Chemical Reaction Network Simulator**
    - Classes: Molecule, Reaction, ReactionNetwork
    - Features: Concentration tracking, rate equations, equilibrium
    - Data: Reaction definitions, initial concentrations
    - Plots: Concentration vs time, reaction rates
2. **Molecular Property Calculator**
    - Classes: Atom, Bond, Molecule
    - Features: Molecular weight, formula, structure properties
    - Data: SMILES strings or molecular formulas
    - Plots: Property distributions, structure comparisons
3. **Acid-Base Titration Analyzer**
    - Classes: Solution, Titration, Buffer
    - Features: pH calculation, equivalence point, buffer capacity
    - Data: Titration data (volume, pH)
    - Plots: Titration curves, derivative plots

Physics

1. **Particle Physics Simulator**
    - Classes: Particle, Force, System
    - Features: Motion under gravity, electrostatics, collisions
    - Data: Initial positions, velocities, masses
    - Plots: Trajectories, energy over time, phase space
2. **Wave Phenomena Analyzer**
    - Classes: Wave, Medium, Interference
    - Features: Superposition, standing waves, diffraction
    - Data: Wave parameters (frequency, amplitude)

- Plots: Wave patterns, interference patterns, spectra

3. Thermodynamics Calculator

- Classes: System, Process, Cycle
- Features: Work, heat, entropy calculations for various processes
- Data: State parameters (P, V, T)
- Plots: PV diagrams, process cycles

Environmental Science / Earth Science

1. Climate Data Analyzer

- Classes: Station, Measurement, ClimateDataset
- Features: Temperature trends, anomaly detection, statistics
- Data: Historical climate data (CSV)
- Plots: Time series, trend lines, anomaly maps

2. Water Quality Monitor

- Classes: Sample, Parameter, WaterBody
- Features: Quality indices, trend analysis, threshold alerts
- Data: Water quality measurements
- Plots: Parameter trends, quality scores, spatial distributions

---

Mathematics / Statistics

1. Statistical Distribution Analyzer

- Classes: Distribution (parent), Normal, Binomial, etc.
- Features: Parameter estimation, hypothesis testing, simulation
- Data: Experimental data or generated samples
- Plots: PDFs, CDFs, Q-Q plots, histograms

2. Numerical Methods Library

- Classes: Function, DifferentialEquation, Integrator
- Features: Root finding, integration, ODE solving
- Data: Function definitions, initial conditions
- Plots: Solution curves, error analysis, convergence

Interdisciplinary

1. Scientific Instrument Data Processor

- Classes: Instrument, Measurement, Calibration

- Features: Data import, calibration, quality control
- Data: Instrument output files
- Plots: Calibration curves, measurement distributions

2. Experiment Planner & Tracker

- Classes: Experiment, Trial, Protocol
- Features: Design tracking, result recording, statistics
- Data: Experimental parameters and results
- Plots: Result comparisons, success rates, parameter effects

---

Team Roles (Suggested)

You don't need strict roles, but consider dividing responsibilities:

Project Manager

- Organizes meetings
- Tracks deadlines
- Ensures all tasks are completed

Lead Developer

- Oversees code structure
- Reviews pull requests
- Ensures coding standards

Data Specialist

- Handles data acquisition/generation
- Implements pandas analysis
- Validates results

Visualization Specialist

- Creates matplotlib plots
- Ensures clear communication of results
- Prepares presentation visuals

Documentation Lead

- Writes/reviews docstrings
- Creates README
- Prepares presentation content

Note: Everyone should code! These are coordination roles, not exclusive responsibilities. Document what you have agreed upon in the group contract.

GitHub Workflow

Setting Up

1. One team member creates a repository on GitHub
2. Add other team members as collaborators
3. Everyone clones the repository
4. Create a `.gitignore` file (for Python projects)

Daily Workflow

```bash
# Start of work session
git pull origin main  # Get latest changes

# Do your work, then:
git add .
git commit -m "Descriptive message about what you changed"
git push origin main

# If you get conflicts, resolve them and commit
```

Best Practices

Commit often - Multiple small commits better than one huge commit
Write clear commit messages - "Fix bug in GC calculation" not "fixed stuff"
Pull before you push - Always get latest changes first
Test before committing - Make sure your code runs
Don't commit data files - Add large data files to `.gitignore`

Presentation Guidelines

Structure (15 minutes)

1. Introduction (3 minutes)

- Team members
- Problem/motivation
- Project goals

2. Technical Overview (3 minutes)

- Class structure (show diagram)
- Key design decisions
- Technologies used

3. Implementation Highlights (4 minutes)

- Show interesting code snippets

- Explain challenging problems you solved
- Discuss OOP concepts you applied

4. **Demo (4 minutes)**
   - Run your code live
   - Show data analysis
   - Display visualizations

5. **Conclusion (1 minute)**
   - What you learned
   - Future improvements
   - Acknowledgments

Presentation Tips

Practice together - Do a full run-through at least twice
Everyone speaks - Each team member presents a section
Prepare backup - Have screenshots if live demo fails
Show, don't just tell - Demos and visuals > slides of text
Time yourselves - Stay within 15 minutes

See the detailed presentation rubric on Canvas for grading criteria.

Evaluation Criteria

Your project will be evaluated on:

Code Quality (40%)

- Object-oriented design
- Code organization and structure
- Proper use of Python features
- Error handling

Functionality (25%)

- Features work as intended
- Data analysis is correct
- Visualizations are meaningful
- Appropriate scope

Documentation (15%)

- Clear docstrings
- Comprehensive README
- Code comments where needed

- Usage examples

Version Control (10%)

- Regular commits
- Meaningful commit messages
- Evidence of collaboration
- Proper repository structure

Presentation (10%)

- Clear communication
- Effective demo
- Good teamwork
- Time management

Common Pitfalls to Avoid

Starting too late - Projects take longer than you think!
Poor communication - Communicate delays early
Uneven contribution - Everyone should code and commit
Overambitious scope - Start simple, add features if time permits
No testing - Test as you go, don't wait until the end
Forgetting documentation - Write docstrings while coding
One person does everything - This is a team project!
No backup for demo - Live demos can fail; have screenshots ready

---

Resources

Python Libraries

- **pandas**: Data manipulation and analysis
- **matplotlib**: Plotting and visualization
- **numpy**: Numerical computing
- **scipy**: Scientific computing

Sample Data Sources

- **Kaggle**: https://www.kaggle.com/datasets
- **UCI Machine Learning Repository**: https://archive.ics.uci.edu/ml/
- **Our World in Data**: https://ourworldindata.org/
- **NCBI**: https://www.ncbi.nlm.nih.gov/ (biology)
- **Or create your own synthetic data!**

Learning Resources

- Course materials on Canvas
- Python documentation: https://docs.python.org/3/
- Pandas documentation: https://pandas.pydata.org/docs/
- Matplotlib gallery: https://matplotlib.org/stable/gallery/
- Real Python: https://realpython.com/

Getting Help

Stuck on your project?

1. **Check course materials** - Review relevant exercises and examples
2. **Search online** - Stack Overflow, Python docs, library documentation
3. **Ask your team** - Two heads are better than one!
4. **Post on Canvas forum** - Share the problem (not your solution)
5. **Email instructor** - For urgent issues or private questions

When asking for help: - Describe what you're trying to do - Explain what you've tried - Share the specific error message - Provide a minimal code example

---

Submission Checklist

Before your Week 6 presentation:

GitHub Repository

☐ All code committed and pushed
☐ Repository is public (or instructor has access)
☐ No sensitive data committed
☐ `.gitignore` file present

README.md

☐ Project title and description
☐ Team members listed
☐ Installation instructions
☐ Usage examples with sample data
☐ Dependencies listed
☐ Screenshots or example outputs

Code

☐ All classes have docstrings
☐ Methods have docstrings

☐ Code runs without errors
☐ No hardcoded file paths
☐ Tested with sample data

Documentation

☐ Comments for complex sections
☐ Example data included or linked
☐ Requirements.txt or dependencies listed

Presentation

☐ Slides prepared
☐ Demo tested and working
☐ Each team member knows their part
☐ Backup screenshots ready
☐ Presentation is 15 minutes or less

Example Project Structure

```
my-science-project/

  README.md                 # Project overview and usage
  requirements.txt          # Python dependencies
  .gitignore                # Files to ignore in Git

  src/                      # Source code
      __init__.py
      classes.py            # Your main classes
      analysis.py           # Analysis functions
      visualization.py      # Plotting functions

  data/                     # Data files (or link to external data)
      sample_data.csv
      README.md             # Data description

  examples/                 # Usage examples
      example_usage.py

  tests/                    # Test scripts (optional but recommended)
      test_classes.py

  docs/                     # Additional documentation (optional)
      design.md             # Design decisions, class diagrams
```

Frequently Asked Questions

Q: How do we choose team members?
A: Form groups of 4-5. You can self-select or I can help match students with similar interests.

Q: Can we use external libraries not covered in class?
A: Yes, but you must be able to explain why you chose them. Core functionality should use course concepts.

Q: What if someone isn't contributing?
A: Document contributions via GitHub commits. Contact me if issues arise. Peer evaluation forms will be provided.

Q: Can we change our project idea after Week 1?
A: Small changes are fine. Major changes should be discussed with me first.

Q: How complex should our classes be?
A: Each class should have 3-5 methods and meaningful attributes. Focus on good OOP design over complexity.

Q: Do we need to use real data?
A: Real data is great but not required. Synthetic/simulated data is perfectly acceptable.

Q: What if our demo doesn't work during the presentation?
A: Have backup screenshots/videos. You won't lose all points, but preparation matters.

Q: Can we work on the project during class time?
A: Weeks 1-4 have dedicated project work time after exercises. Week 5 is mostly project time after the exam.

Academic Integrity

You may: - Discuss ideas with other teams - Use code from official documentation or course materials - Search for help online for specific techniques

You may not: - Copy code from other teams - Submit code you didn't write (without attribution) - Have someone outside the class write code for you

When in doubt: Ask me! It's always better to ask than risk academic integrity issues.

If you use code from online sources, cite it:

```python
# Adapted from: https://stackoverflow.com/questions/12345/...
def my_function():
    pass
```