Southwestern University of Finance and Economics

Fundamentals of Computer Programming

Course Syllabus · Fall Semester 2025–2026 · Research Institute of Economics and Management

1. Course Information

Course Title	Fundamentals of Computer Programming	**Audience**	Undergraduate Students (Econ/Mgmt)
Instructor	Luping Zhang, Ph.D. (Associate Professor)	**Email**	lpzhang@swufe.edu.cn
Office	Room 110, Gezhi Building, Liulin Campus	**Office Hours**	By appointment via email
Teaching Assistants	Course TA Team (TBD)	**Credit / Hours**	27 Contact Hours (3 hrs/session, 9 weeks)
Lecture Schedule	Weeks 1–9: Wed. Periods 5–7 (Aft.), 10–12 (Eve.)	**Location**	Per University Registrar schedule
Course Resources	Textbooks, handouts, empirical case studies	**Environment**	Python 3.10+ / Personal laptop

2. Course Description and Learning Objectives

2.1 Course Description

Empirical economic, financial, and managerial analysis relies on quantitative computational methods. This course teaches the core principles, syntax, and practices of computer programming using Python. Designed for students without prior coding experience, the curriculum builds computational thinking and modular program design habits. Students advance from basic syntax, data types, and control flow to compound collections, functional abstraction, object-oriented concepts, file persistence, and introductory data analysis packages. Practical exercises teach students to construct clean, verifiable code to address empirical questions.

2.2 Learning Objectives

By the end of this course, students will be able to:

1. **Master Core Syntax and Program Logic**: Understand the Python execution model and write structured, readable scripts using variables, fundamental data types, conditional branches, iteration, functions, and module imports.
2. **Organize and Process Structured Data**: Select and use built-in compound collections (lists, tuples, dictionaries, and sets). Apply regular expressions for text cleaning and extraction. Handle disk file input/output, and use NumPy and Pandas to inspect, clean, and compute descriptive statistics on tabular data.
3. **Formulate Computational Solutions**: Translate practical questions from economics and business into concrete algorithmic steps, implement programmatic solutions, and verify computational results.

3. Course Materials and References

3.1 Recommended Textbooks

1. Junxiu An. *Python 3 from Beginner to Master* (in Chinese). Beijing: People's Posts and Telecommunications Press. (Primary textbook)
2. Brian Overland and John Bennett. *Python Without Fear: A Beginner's Guide that Makes You Feel Smart* (Chinese edition translated by Publishing House of Electronics Industry). (Supplementary reading)
3. Luciano Ramalho. *Fluent Python* (2nd Edition). Beijing: People's Posts and Telecommunications Press. (Advanced reference for language internals and design patterns)

3.2 Online References

- W3Schools Python Tutorial: `https://www.w3schools.com/python/default.asp`
- Runoob Python 3 Tutorial: `https://www.runoob.com/python3/python3-tutorial.html`
- Official Python Documentation: `https://docs.python.org/3/`

4. Course Schedule and Weekly Plan

Week	Module	Core Concepts and Mechanics	Learning Outcomes
Week 1	**Introduction and Environment Setup**	Language ecosystem, interpreter mechanics, interactive shell, script execution, comments and structure	Configure the local Python environment, run scripts via the interpreter, and write standalone programs
Week 2	**Basic Syntax and Core Data Types**	Variable assignment, dynamic typing, numeric operations, boolean logic, type conversion, string formatting	Apply variable binding rules, evaluate numeric and boolean expressions, and format text output cleanly
Week 3	**Control Flow and Conditional Logic**	Standard I/O, conditional branches (`if-elif-else`), iterative loops (`while`, `for`), `break`/`continue` control	Implement branching and iterative logic to direct program execution and process repetitive tasks
Week 4	**Built-in Data Collections**	Lists, tuples, dictionaries, and sets: creation, indexing, slicing, common methods, and comprehensions	Choose appropriate data collections to organize, query, and transform multi-attribute records
Week 5	**Functional Abstraction and Modules**	Function definition, arguments (positional, keyword, default), variable scope, lambda expressions, module imports	Decompose procedures into cohesive functions, organize reusable code into modules, and import libraries
Week 6	**File I/O and Data Persistence**	File lifecycle, context managers (`with` block), text and CSV file reading/writing, path operations, cursor positioning	Read local data files from disk, parse structured records, modify contents, and persist outputs safely
Week 7	**Object-Oriented Programming**	Classes and instances, constructor (`__init__`), instance attributes and methods, encapsulation, inheritance	Model domain entities with classes, encapsulate internal state and operations, and instantiate interacting objects
Week 8	**Regular Expressions and Text Extraction**	Metacharacters, quantifiers, character sets, greedy vs. non-greedy matching, capture groups, `re` methods	Formulate regular expressions to validate patterns, clean raw text, and extract key fields from documents
Week 9	**Data Analysis Tools: NumPy and Pandas**	`ndarray` vector operations; Pandas `DataFrame` indexing, filtering, missing values, aggregation; empirical demo	Construct arrays and DataFrames, apply vector transformations, and compute summary statistics on economic tables

Total Instructional Hours: 27 Contact Hours (3 hours per week across 9 weeks)

5. Course Policies and Academic Integrity

1. **Academic Integrity and Code Ethics**: Academic honesty is fundamental to this course. All individual homework assignments and quizzes must represent each student's independent work. For the collaborative course project, teams must document the division of responsibilities and individual contributions explicitly. When using open-source libraries or third-party algorithms, students must provide clear citations in code comments or project reports. Plagiarism, unauthorized copying, and submitting uncredited third-party code are strictly prohibited.
2. **Reading and Preparation**: Students are expected to complete the assigned readings before each session. Class sessions include unannounced discussions and designated student presentations to assess comprehension of programming principles.
3. **Homework Submissions and Feedback**: Homework must be submitted to the teaching assistants before the stated deadline. The instructor and teaching assistants review common errors and code quality standards in class every two weeks.
4. **Final Team Project**: Students work in teams of 3–4 to build an end-to-end program that addresses an empirical problem in economics, finance, or business management. Deliverables include documented source code, a technical report, and an in-class presentation in Week 9.
5. **Attendance and Classroom Conduct**: Regular attendance is required. Students who must miss a class due to documented medical or personal emergencies must submit formal verification in advance. Students must maintain active attention and respect classroom decorum.

6. Assessment and Grading Policy

The final course grade consists of the final team project and continuous term assessments, weighted as follows:

Assessment Component	Scope and Deliverables	Evaluation Method	Weight
Final Team Project	Empirical project (instructor-assigned or approved topic)	Written technical report + oral presentation	60%
Continuous Assessment	In-class discussions, code exercises, homework, attendance	Quizzes and regular homework reviews	40%

Grading Policy Notes:

- There is no midterm examination in this course.
- Attendance is recorded through unannounced roll calls and in-class participation checks.
- Accumulating three unexcused absences results in a grade of zero for the attendance component and disqualifies the student from the final project defense and grade evaluation.