ADVANCED PYTHON PROGRAMMING FOR DATA SCIENCE

ENCT 325

Lecture : 3 **Year : III**
Tutorial : 2 **Part : I**
Practical : 1

Course Objectives:
The objective of this course is to develop advanced proficiency in Python programming for data science applications. It focuses on efficient coding practices, sophisticated data manipulation, statistical analysis, and data visualization using modern Python libraries. Students will also learn fundamental data engineering and pipeline design concepts to automate and scale real-world data workflows.

1 Advanced Python Concepts and Best Practices (7 hours)

1.1 Review of Python essentials and coding conventions
1.2 Advanced data structures: Collections, iterators, generators, and decorators
1.3 Functions and lambda expressions
1.4 Object-Oriented Programming for data science applications
1.5 Exception handling, debugging, and logging
1.6 Working with modules and packages

2 Data Sources and APIs (7 hours)

2.1 Reading and writing structured/unstructured data (CSV, JSON, Excel, text)
2.2 Database access with relational database and non-relational database
2.3 Accessing and processing data from APIs (REST, SOAP)
2.4 Web scraping using requests and BeautifulSoup
2.5 Handling large datasets with chunking and lazy evaluation

3 Advanced Data Wrangling and Transformation (9 hours)

3.1 Advanced Pandas operations: Merging, joining, reshaping, pivoting
3.2 Handling missing, categorical, and time-series data
3.3 Feature transformation, scaling, and encoding
3.4 Memory optimization and efficient data processing
3.5 Building a reusable data-cleaning pipeline
3.6 Introduction to data pipeline components (Ingestion, transformation, storage)

4 Applied Statistics and Exploratory Analysis (7 hours)

4.1 Statistical measures: Correlation, covariance, skewness, kurtosis
4.2 Probability review, sampling, and hypothesis testing
4.3 Regression and trend analysis using stats models
4.4 Exploratory data analysis (EDA) using descriptive and inferential methods
4.5 Automation of EDA workflows using Python

5 Data Visualization and Storytelling (7 hours)

5.1 Principles of effective visualization and dashboard design
5.2 Visualization with Matplotlib: Line, bar, histogram, scatter, subplots
5.3 Seaborn for statistical visualization: Box plot, pair plot, heat map
5.4 Interactive visualization using Plotly
5.5 Visualization driven insight generation
5.6 Case study: End-to-end visualization and reporting project

6 Data Engineering and Automation (8 hours)

6.1 Overview of data engineering in applied data science
6.2 Designing and implementing ETL pipelines
6.3 Automating workflows with schedulers (CRON, schedule)
6.4 Logging, monitoring, and error handling in pipelines
6.5 Data storage and retrieval strategies for pipelines
6.6 Automated report generation (Excel, HTML, PDF)
6.7 Case study: End-to-end automated analytics pipeline

Tutorial (30 hours)

1. Python refresher and best practices: Review of Python syntax, indentation, and PEP8 coding conventions through short exercises
2. Iterators, generators, and decorators: Writing small programs using generators for data streaming and decorators for function modification
3. Object-oriented programming (OOP): Designing simple class-based programs and demonstrating inheritance and encapsulation in python
4. Data access and integration: Reading data from CSV, Excel, JSON, and APIs; Discussion on best practices for data ingestion
5. Database and SQL interaction: practice using SQLite and SQLAlchemy to query and manipulate structured datasets
6. Web scraping practice: Extracting tabular data using requests and BeautifulSoup; Handling exceptions and encoding issues
7. Advanced Pandas operations: Hands-on merging, reshaping, pivoting, and group-by operations for complex data manipulation
8. Data cleaning and transformation: Exercises on handling missing values, encoding categorical data, and normalization techniques
9. Statistical computation and EDA: Performing descriptive analysis, correlation, and hypothesis testing using Python libraries

10. Data visualization practice: Creating comparative plots using matpLotlib, Seaborn, and Plotly; Customizing themes and layouts
11. Pipeline and automation concepts: Designing pseudocode and flow diagrams for ETL data pipelines and discussing error handling strategies
12. Mini case study discussion: Guided review of a small end-to-end applied data pipeline from ingestion to visualization and reporting

Practical **(15 hours)**

1. Setting up Python environment for applied data workflows and writing modular programs using OOP and functions
2. Collecting data via APIs and web scraping
3. Building advanced data cleaning and transformation pipelines using Pandas
4. Conducting exploratory data analysis and statistical summaries
5. Developing interactive visualizations using Plotly, matplotlib and Seaborn
6. Automating ETL tasks and data refresh using Python schedulers
7. Generating summary dashboards and automated analytical reports
8. Mini Project: Build a complete applied data pipeline from ingestion to visualization and reporting on a real-world dataset

Final Exam

The questions will cover all the chapters in the syllabus. The evaluation scheme will be as indicated in the table below:

Chapter	Hours	Marks distribution*
1	7	9
2	7	9
3	9	12
4	7	10
5	7	9
6	8	11
Total	**45**	**60**

* There may be minor deviation in marks distribution.

References

1. McKinney, W. (2022). Python for data analysis. O'Reilly Media.
2. VanderPlas, J. (2016). Python data science handbook. O'Reilly Media.
3. Beazley, D. (2021). Python cookbook. O'Reilly Media.
4. Grus, J. (2022). Data science from scratch. O'Reilly Media.

COMPILER DESIGN

ENCT 326

Lecture : 3 **Year : III**
Tutorial : 2 **Part : I**
Practical : 1

Course Objectives:
The objective of this course is to develop a clear understanding of the structure, components, and design principles of a compiler. It focuses on the key phases of compilation, including lexical analysis, parsing, semantic analysis, and code generation. The course also introduces optimization techniques and error-handling strategies to enhance the efficiency and reliability of compilers.

1 Introduction (5 hours)

1.1 Compiler and Interpreter
1.2 Compiler structure: Analysis and synthesis model of compilation, different sub-phases within the analysis and synthesis phases
1.3 Basic concepts like preprocessor, macros, symbol table, and error handler

2 Lexical Analysis (9 hours)

2.1 The role of the lexical analyzer
2.2 Specification and recognition of tokens
2.3 Input buffering
2.4 Finite automata and their relevance to compiler construction
2.5 Design of a lexical analyzer generator
2.6 Error handling in lexical analysis

3 Syntax Analysis (Parsing) (8 hours)

3.1 Role of syntax analysis
3.2 Context-free grammars and parse trees
3.3 Parsing techniques: Top-down (recursive descent, LL(1)) and bottom-up (LR(1))
3.4 Operator precedence
3.5 Handling issues like left recursion and ambiguity

4 Syntax-Directed Translation and Semantic Analysis (9 hours)

4.1 Syntax-directed definitions (SDD) and syntax-directed translation (SDT)
4.2 Synthesized and inherited attributes
4.3 Syntax tree construction

4.4 Intermediate code generation
4.5 Semantic analysis, including static and dynamic checks
4.6 Type checking and type systems
4.7 Symbol table construction, attributes, and data structures for symbol table
4.8 Run–time storage management

5 Code Generation (7 hours)

5.1 Factors affecting a code generator
5.2 Basic blocks and flow graphs
5.3 Optimization of basic blocks
5.4 Register allocation and assignment
5.5 Dynamic programming code-generation algorithm

6 Code Optimization (7 hours)

6.1 Loops in flow graphs, constant propagation, partial redundancy elimination
6.2 Data-flow analysis and solving data-flow equations
6.3 Storage organization and allocation strategies (Stack, heap)
6.4 Need and criteria of code optimization and basic optimization techniques
6.5 Basic block optimization
6.6 Case studies of some compilers, like the c compiler, the C++ compiler

Tutorial (30 hours)

1. Description of various phases and structures of a compiler.
2. Understanding of regular expressions
3. DFA exercises
4. NFA, Thomson's construction
5. Bottom-up parsing
6. Top down parsing
7. Building LR(1) and LL(1) tables
8. Creating graphical intermediate code generations
9. Creating linear intermediate code generations
10. Building hash tables
11. Basic exercise on designing a code generator
12. Peephole optimization
13. Designing flow graphs
14. Data flow analysis
15. Solving data-flow equation

Practical (15 hours)

1. Setting lexical analysis
2. Introduction to Lex/Flex tools
3. Implementing context-free grammars
4. Designing a basic recursive descent parser, computing first and follow values, predictive parsing table construction and LL (1) grammar

5. Syntax analysis with YACC/Bison
6. Semantic analysis and intermediate code generation

Final Exam

The questions will cover all the chapters in the syllabus. The evaluation scheme will be as indicated in the table below:

Chapter	Hours	Marks distribution*
1	5	7
2	9	12
3	8	11
4	9	12
5	7	9
6	7	9
Total	**45**	**60**

* There may be minor deviation in marks distribution.

References

1. Aho, A. V., Lam, M. S., Sethi, R., Ullman, J. D. (2006). Compilers: Principles, techniques, and tools. Pearson.
2. Cooper, K. D., Torczon, L. (2011). Engineering a compiler. Morgan Kaufmann.
3. Hopcroft, J. E., Motwani, R., Ullman, J. D. (2006). Introduction to automata theory, languages, and computation. Addison-Wesley.
4. Muchnick, S. (1997). Advanced compiler design and implementation (Latest Publication). Morgan Kaufmann.
5. Thain, D. (2020). Introduction to compilers and language design. Douglas Thain.

JAVA PROGRAMMING

ENCT 327

Lecture : 3 **Year : III**
Tutorial : 2 **Part : I**
Practical : 1

Course Objectives:
The objective of this course is to equip students with comprehensive knowledge and practical skills in Java programming, enabling them to design and develop robust, modern applications. Emphasis is placed on using the Spring Boot framework to build scalable, production-ready enterprise solutions.

1 Java Fundamentals (7 hours)

- 1.1 Overview of Java ecosystem (JDK, JVM, JRE)
- 1.2 Evolution of Java
- 1.3 Modern development tools (Maven, Gradle, IntelliJ, VS Code)
- 1.4 Review of core Java OOP, collections and exception handling
- 1.5 Functional programming: Lambda, expressions and streams API
- 1.6 Introduction to modern features like records and sealed classes

2 Java I/O and Data Handling (6 hours)

- 2.1 Streams and file handling
  - 2.1.1 Byte and character streams
  - 2.1.2 Reading/writing text files
  - 2.1.3 Object streams and serialization
  - 2.1.4 New I/O (NIO.2) file operations
- 2.2 JSON and data Interchange
  - 2.2.1 Handling JSON data with libraries like Jackson or Gson
  - 2.2.2 JSON vs. XML in REST APIs

3 Database and Web Communication (10 hours)

- 3.1 Java database connectivity (JDBC)
  - 3.1.1 JDBC architecture and drivers
  - 3.1.2 Connection, statement, and result set management
  - 3.1.3 Prepared statements for secure and efficient queries
- 3.2 Network programming and rest
  - 3.2.1 Client-server model and basics of socket communication
  - 3.2.2 Consuming restful APIs using the HTTPClient from Java 11+
  - 3.2.3 Overview of microservices architecture

4 Spring Boot and Modern Web Development (16 hours)

4.1 Spring Boot fundamentals; Dependency injection and inversion of control; Creating RESTful APIs with Spring Boot
4.2 Data Access with Spring
  4.2.1 Overview of ORM and JPA
  4.2.2 Building a repository layer with spring data JPA
  4.2.3 CRUD operations and transactional management
4.3 API Development and Security
  4.3.1 REST API design principles
  4.3.2 Handling exceptions in spring boot
  4.3.3 Overview of spring security for authentication and authorization
4.4 Deployment and Project Work
  4.4.1 Docker and containerization
  4.4.2 Packaging and deploying a Spring Boot application

5 Concurrency and Performance (6 hours)

5.1 Overview of multithreading
5.2 Thread lifecycle and synchronization
5.3 Executor framework and thread pools
5.4 Completable future and parallel streams
5.5 Best practices for thread-safe programming
5.6 Performance of multi-threaded program

Tutorial (30 hours)

1. Practice Lambda expressions and Stream API operations
2. Implement functional interfaces and method references
3. Perform file I/O operations and object serialization
4. Implement multithreading and synchronization mechanisms
5. Develop a Java program for CRUD operations using JDBC
6. Create and configure a Spring Boot project using Spring Initializer
7. Implement dependency injection using constructor, setter, and field injection
8. Develop restful CRUD APIs using Spring Boot and JPA
9. Create custom database queries using derived methods and the @query annotation
10. Implement centralized exception handling and API error responses
11. Configure authentication and authorization using Spring Security
12. Consume third-party rest APIs using WebClient or RestTemplate
13. Configure environment-specific properties and external database connections
14. Containerize and deploy a spring boot application using Docker
15. Write and execute unit and integration tests using JUnit 5 and MockMvc
16. Project work: Apply learned concepts by building a small, secure, full-stack REST service

Practical **(15 hours)**

1. Programs using Lambda expressions and Streams API
2. Implementation of file I/O operations
3. Development of a simple client-server application
4. Implementation of a Spring Boot application with a data access layer
5. Containerization of a Spring Boot application using Docker

Final Exam

The questions will cover all the chapters in the syllabus. The evaluation scheme will be as indicated in the table below:

Chapter	Hours	Marks distribution*
1	7	8
2	6	8
3	10	14
4	16	22
5	6	8
Total	**45**	**60**

* There may be minor deviation in marks distribution.

References

1. Bloch, J. (2017). Effective Java. Addison-Wesley Professional.
2. Schildt, H. (2024). Java: The Complete Reference. McGraw-Hill Education.
3. Goetz, B., Peierls, T., Bloch, J., Bowbeer, D., Holmes, J., Lea, D. (2006). Java Concurrency in Practice. Addison-Wesley Professional.
4. Walls, C. (2016). Spring Boot in Action. Manning Publications.

QUANTUM COMPUTING

ENCT 328

Lecture : 3 **Year : III**
Tutorial : 2 **Part : I**
Practical : 1

Course Objectives:
The objective of this course is to introduce the principles, mathematical foundations, and computational models of quantum computing. It emphasizes on algorithmic insights, and offer practical experience with quantum programming environments. By the end of the course, students will be able to represent and manipulate quantum states and gates, design and simulate basic quantum circuits, implement fundamental quantum algorithms and their speedup mechanisms, and explore applications of quantum machine learning (QML) techniques.

1 Fundamentals of Quantum Concepts (6 hours)

1.1 Classical versus Quantum computation
1.2 Qubits, superposition, measurement
1.3 Postulates of quantum mechanics
1.4 Dirac notation and Bloch sphere

2 Quantum Gates and Circuits (6 hours)

2.1 Quantum gates (X, Y, Z, H, Phase, CNOT)
2.2 Unitary operations and reversibility
2.3 Multi-qubit systems and tensor products
2.4 Circuit design and visualization

3 Entanglement and Quantum Communication (5 hours)

3.1 Entangled states and Bell theorem
3.2 EPR paradox
3.3 Quantum teleportation and superdense coding
3.4 Quantum key distribution protocols: BB84, E91

4 Quantum Algorithms (10 hours)

4.1 Quantum parallelism
4.2 Deutsch and Deutsch–Jozsa algorithms
4.3 Bernstein–Vazirani
4.4 Grover’s algorithm
4.5 Shor’s algorithm

5 Quantum Hardware and NISQ Era (6 hours)

5.1 Physical realization (Ion trap, superconducting, photonic)
5.2 Noise, decoherence, and qubit metrics
5.3 NISQ model and limitations
5.4 Quantum circuit execution on IBM quantum simulators

6 Quantum Error and Correction (4 hours)

6.1 Bit-flip and phase-flip errors
6.2 Simple quantum error detection
6.3 Shor's 9-qubit code
6.4 Surface code overview

7 Quantum Annealing, Optimization Algorithms (4 hours)

7.1 Principles of quantum annealing
7.2 Relationship with adiabatic quantum computation
7.3 Optimization problem formulation (QUBO, Ising models)
7.4 Quantum annealer architecture

8 Quantum Machine Learning (4 hours)

8.1 Introduction to QML and its importance
8.2 Quantum data representation and feature encoding
8.3 Variational Quantum Circuits (VQC)
8.4 Simple quantum classifiers and QNN concepts
8.5 Hybrid quantum-classical learning models

Tutorial (30 hours)

1. Classical versus quantum computation; Qubit basics, superposition, and measurement
2. State vectors, normalization and probability amplitudes
3. Dirac notation and Bloch sphere representation with simple simulations
4. Single-qubit gates (X, Y, Z, H, Phase); Matrix representation and simulation
5. Multi-qubit systems and two-qubit gates such as CNOT; Tensor product exercises and simulation
6. Entanglement and Bell states; Measurement correlation analysis
7. Simulation of quantum key distribution protocols, including BB84
8. Quantum algorithms I: Deutsch and Deutsch–Jozsa; Algorithm formulation and coding
9. Quantum algorithms II: Bernstein–Vazirani and Grover search; Coding and simulation exercises
10. Introduction to quantum hardware; Physical qubit types, noise, and decoherence; Basic modeling
11. NISQ devices; Build and execute simple circuits on IBM Quantum or equivalent simulator and analyze results

12. Quantum error modeling including bit-flip and phase-flip; Simple error detection and correction exercises
13. Quantum annealing and optimization; Formulation of QUBO and Ising problems
14. Quantum machine learning and variational circuits; Data encoding, simple VQC construction

Practical **(15 hours)**

1. IBM qiskit / quantum composer labs
2. Implement Grover's or teleportation circuit
3. Implement Shor's algorithm
4. Applications in combinatorial optimization (E.g., scheduling, routing, network optimization)
5. Hands-on simulation using quantum annealers
6. Mini-project demonstration

Final Exam

The questions will cover all the chapters in the syllabus. The evaluation scheme will be as indicated in the table below:

Chapter	Hours	Marks distribution*
1	6	8
2	6	8
3	5	7
4	10	12
5	6	8
6	4	6
7	4	5
8	4	6
Total	**45**	**60**

* There may be minor deviation in marks distribution.

References

1. Nielsen, M. A., Chuang, I. L. (2002). Quantum computation and quantum information. Cambridge University Press.
2. Bernhardt, C. (2019). Quantum computing for everyone. MIT Press.
3. Schuld, M., Petruccione, F. (2021). Machine learning with quantum computers. Springer.
4. Sahni, V. (2007). Quantum computing. Tata McGraw-Hill Publishing Company.
5. Akama, S. (2014). Elements of quantum computing: History, theories and engineering applications. Springer International Publishing.
6. Wichert, A. (2014). Principles of quantum artificial intelligence. World Scientific Publishing Co.