Complete Programming Syllabi

Python Programming Syllabus

Module 1: Python Fundamentals

- Introduction to Python and programming concepts
- Installing Python and setting up development environment
- Python syntax, indentation, and code structure
- Variables, data types, and type conversion
- Input/output operations and basic formatting

Task: Create a simple calculator program that performs basic arithmetic operations and handles user input validation.

Module 2: Control Structures and Operations

- Arithmetic, comparison, and logical operators
- Conditional statements (if, elif, else)
- Loops (for, while) and loop control statements
- Understanding scope and variable lifetime
- Exception handling basics

Task: Build a number guessing game where the computer generates a random number and the user has to guess it with hints.

Module 3: Data Structures

- Lists: creation, indexing, slicing, and methods

- Tuples: immutable sequences and use cases
- Dictionaries: key-value pairs and operations
- Sets: unique collections and set operations
- String manipulation and formatting techniques

Task: Create a student grade management system that stores student information and calculates statistics using various data structures.

Module 4: Functions and Modules

- Defining and calling functions
- Parameters, arguments, and return values
- Local vs global variables and scope
- Lambda functions and functional programming concepts
- Creating and importing modules
- Package management with pip

Task: Develop a modular text analyzer program with separate functions for word count, character count, and frequency analysis.

Module 5: File Handling and I/O

- Opening, reading, and writing files
- File modes and context managers
- Working with CSV and JSON data
- Directory operations and path handling
- Error handling in file operations

Task: Build a contact management system that saves and loads contact information from JSON files with search functionality.

Module 6: Object-Oriented Programming

- Classes and objects fundamentals
- Attributes and methods
- Constructors and destructors
- Inheritance and method overriding
- Encapsulation and polymorphism
- Special methods and operator overloading

Task: Create a bank account management system using classes with different account types and transaction handling.

Module 7: Advanced Python Concepts

- Decorators and their applications
- Generators and iterators
- Context managers and the 'with' statement
- Regular expressions for pattern matching
- Working with dates and times

Task: Develop a log file analyzer that uses decorators, generators, and regex to process and analyze server logs.

Module 8: Libraries and Frameworks

- NumPy for numerical computing
- Pandas for data manipulation

- Matplotlib for data visualization
- Requests library for HTTP operations
- Introduction to web scraping with BeautifulSoup

Task: Create a data analysis project that fetches data from an API, processes it with pandas, and creates visualizations.

Module 9: Testing and Debugging

- Unit testing with unittest module
- Debugging techniques and tools
- Code profiling and optimization
- Best practices for code quality
- Documentation and commenting standards

Task: Write comprehensive unit tests for previously created projects and optimize their performance.

Python Mini Project: Personal Finance Tracker

Build a complete personal finance application that includes:

- Income and expense tracking with categories
- Data visualization of spending patterns
- Budget management with alerts
- Data export/import functionality
- Simple GUI using tkinter
- Database integration with SQLite
- Report generation and financial insights

Django Web Framework Syllabus

Module 1: Django Introduction and Setup

- Understanding web development and MVC architecture
- Django framework overview and philosophy
- Installing Django and project setup
- Understanding Django project structure
- Virtual environments and dependency management

Task: Create a basic Django project with a simple "Hello World" application and explore the project structure.

Module 2: Django Fundamentals

- Creating Django applications
- URL configuration and routing
- Views: function-based and class-based
- Templates and template inheritance
- Static files management
- Django admin interface basics

Task: Build a simple blog homepage that displays static content using templates and proper URL routing.

Module 3: Models and Database Integration

- Django ORM fundamentals
- Model definition and field types

- Database migrations and schema management
- QuerySet API and database operations
- Model relationships (OneToOne, ForeignKey, ManyToMany)
- Custom model methods and properties

Task: Create a library management system with Book and Author models, including proper relationships and admin interface.

Module 4: Advanced Models and Database Operations

- Model inheritance strategies
- Database indexing and optimization
- Custom managers and QuerySets
- Database transactions
- Raw SQL queries when necessary
- Database backend configuration

Task: Extend the library system with advanced queries, custom managers, and database optimization techniques.

Module 5: Views and URL Patterns

- Function-based views in detail
- Class-based views and mixins
- Generic views for common patterns
- URL namespacing and reverse URL lookup
- Handling different HTTP methods
- View decorators and middleware

Task: Implement CRUD operations for the library system using both function-based and class-based views.

Module 6: Templates and Frontend Integration

- Django template language syntax
- Template filters and tags
- Custom template tags and filters
- Template inheritance and inclusion
- Integrating CSS frameworks
- JavaScript integration strategies

Task: Create responsive templates for the library system with custom filters and Bootstrap integration.

Module 7: Forms and User Input

- Django forms framework
- Form validation and error handling
- ModelForms for database integration
- Formsets for multiple forms
- File uploads and media handling
- CSRF protection and security

Task: Add book submission forms with validation, image uploads, and user feedback mechanisms.

Module 8: User Authentication and Authorization

- Django authentication system
- User registration and login/logout

- Password management and reset
- User profiles and custom user models
- Permissions and groups
- Decorators for access control

Task: Implement user authentication system with role-based access control for librarians and members.

Module 9: Advanced Django Features

- Django REST framework basics
- API development and serialization
- Caching strategies and implementation
- Internationalization and localization
- Email integration
- Custom management commands

Task: Build REST API endpoints for the library system with proper serialization and caching.

Module 10: Testing and Deployment

- Unit testing Django applications
- Integration testing and test databases
- Test-driven development practices
- Deployment preparation and settings management
- Production deployment strategies
- Performance monitoring and optimization

Task: Write comprehensive tests for the library application and prepare it for production deployment.

Django Mini Project: E-Commerce Platform

Develop a complete e-commerce web application featuring:

- Product catalog with categories and search functionality
- Shopping cart and checkout system
- User registration and authentication
- Order management and tracking
- Payment integration (sandbox)
- Admin dashboard for inventory management
- Product reviews and ratings
- Email notifications
- Responsive design with modern UI/UX
- RESTful API for mobile app integration

C++ Programming Syllabus

Module 1: C++ Fundamentals

- Introduction to C++ and its history
- Setting up development environment (IDE/compiler)
- Basic program structure and compilation process
- Variables, constants, and data types
- Input/output with iostream library

- Comments and code documentation

Task: Create a program that calculates compound interest with proper input validation and formatted output.

Module 2: Operators and Control Flow

- Arithmetic, relational, and logical operators
- Bitwise and assignment operators
- Conditional statements (if, switch)
- Looping constructs (for, while, do-while)
- Break, continue, and goto statements
- Operator precedence and associativity

Task: Build a menu-driven program that performs various mathematical operations and number pattern generation.

Module 3: Functions and Program Structure

- Function declaration and definition
- Function parameters and return types
- Function overloading
- Default parameters and inline functions
- Recursion and recursive algorithms
- Scope and storage classes
- Header files and separate compilation

Task: Develop a mathematical library with functions for factorial, Fibonacci, prime checking, and recursive algorithms.

Module 4: Arrays and Strings

- One-dimensional and multi-dimensional arrays
- Array initialization and manipulation
- C-style strings and string functions
- Introduction to C++ string class
- Character arrays vs string objects
- Dynamic memory allocation for arrays

Task: Create a student records system using arrays to store and manipulate student data with string operations.

Module 5: Pointers and References

- Understanding memory addresses
- Pointer declaration and initialization
- Pointer arithmetic and array-pointer relationship
- References and their differences from pointers
- Passing by value, reference, and pointer
- Dynamic memory allocation (new/delete)

Task: Implement a dynamic array class that can resize itself and demonstrate pointer manipulation techniques.

Module 6: Object-Oriented Programming Basics

- Classes and objects concept
- Data members and member functions
- Constructors and destructors

- Access specifiers (private, public, protected)
- Static members and functions
- Friend functions and classes

Task: Design a bank account class with proper encapsulation, constructors, and member functions for transactions.

Module 7: Advanced OOP Concepts

- Inheritance and types of inheritance
- Function overriding and virtual functions
- Polymorphism and virtual destructors
- Abstract classes and pure virtual functions
- Multiple inheritance and virtual inheritance
- Operator overloading

Task: Create a shape hierarchy with inheritance, virtual functions, and operator overloading for area calculations.

Module 8: Memory Management and Advanced Features

- Dynamic memory management best practices
- Copy constructors and assignment operators
- Move semantics and rvalue references
- Smart pointers (unique_ptr, shared_ptr)
- Exception handling (try, catch, throw)
- Namespaces and scope resolution

Task: Implement a smart string class with proper memory management, copy semantics, and exception handling.

Module 9: Standard Template Library (STL)

- Introduction to templates
- STL containers (vector, list, map, set)
- STL iterators and their categories
- STL algorithms and function objects
- String processing with STL
- Custom comparators and functors

Task: Build a text processing application using STL containers and algorithms for word frequency analysis.

Module 10: Advanced C++ Topics

- Template specialization and metaprogramming
- Lambda expressions and closures
- File I/O and stream manipulation
- Multithreading basics with std::thread
- Modern C++ features (C++11/14/17/20)
- Best practices and code optimization

Task: Create a file compression utility using templates, file I/O, and modern C++ features.

Module 11: Data Structures and Algorithms

- Implementing common data structures
- Linked lists, stacks, and queues

- Trees and binary search trees
- Hash tables and collision resolution
- Graph representations and algorithms
- Sorting and searching algorithms

Task: Implement a complete binary search tree with insertion, deletion, traversal, and search operations.

Module 12: Project Development and Best Practices

- Software design principles
- Code organization and project structure
- Debugging techniques and tools
- Unit testing frameworks
- Version control integration
- Performance profiling and optimization
- Code review and documentation standards

Task: Refactor and optimize previous projects following software engineering best practices and create comprehensive documentation.

C++ Mini Project: Hospital Management System

Develop a comprehensive hospital management application including:

- Patient registration and medical records management
- Doctor and staff scheduling system
- Appointment booking and management
- Inventory management for medical supplies

- Billing and payment processing
- Report generation and analytics
- File-based data persistence
- Multi-threaded operations for concurrent access
- Template-based generic data structures
- Complete error handling and logging system
- Modern C++ features implementation
- Performance optimization and memory management