PYTHON FULL STACK – PAYILAGAM SYLLABUS

Python Introduction, Installation

- Python Introduction
- Download Python
- Installing Python
- Verify the Installation
- Install a Text Editor or IDE (Optional)

Data Types

In Python, there are several built-in data types used to represent different kinds of values. Here are some of the fundamental data types.

- Numeric Types
- Text Type
- Sequence Types
- Set Types
- Mapping Type
- Boolean Type
- None Type

Operators

In Python, operators are symbols or keywords that perform operations on operands, which can be variables, values, or expressions. Here are some common types of operators.

- Arithmetic Operators
- Comparison Operators

- Logical Operators
- Assignment Operators
- Membership Operators
- Identity Operators
- Bitwise Operators

Flow Control Statements

In Python, flow control statements are used to manage the flow of execution in a program. They enable you to make decisions, create loops, and determine the order in which statements are executed. Here are the main flow control statements in Python.

- Conditional Statements
- Looping Statements
- Control Statements
- Exception Handling
- Pass Statement

List

In Python, a list is a versatile and widely used data type that represents an ordered, mutable collection of elements. Lists can contain elements of different data types, and you can easily modify, add, or remove elements from a list. Lists are created using square brackets [ ]. Here's an overview of working with lists

- Creating a List
- Accessing Elements
- Slicing
- Modifying Elements
- Adding Elements
- Removing Elements

- Sorting: Bubble Sort
- Searching: Binary Search

Mini Project: #1 STUDENT MANAGEMENT SYSTEM

Tuple

In Python, a tuple is another sequence data type that is similar to a list. However, unlike lists, tuples are immutable, meaning their elements cannot be changed or modified after the tuple is created. Tuples are created using parentheses ( ) and can contain elements of different data types. Here's an overview of working with tuples

- Creating a Tuple
- Accessing Elements
- Slicing
- Tuple Packing and Unpacking
- Immutable Nature

Mini Project: #1 FIND HIGHEST SCORER

#2 COORDINATE DISTANCE CALCULATOR #3 CAFE MENU ORDER APP

Set

In Python, a set is an unordered and mutable collection of unique elements. Sets are used to store multiple items in a single variable and are defined by enclosing the elements in curly braces { }. Here's an overview of working with sets

- Creating a Set
- Accessing Elements
- Adding Elements
- Removing Elements
- Set Operations

- Other Set Operations

Mini Project

#1 Duplicate Finder **#2 Unique Words from a book**

#3 Students Attendance Tracker

Dictionary

In Python, a dictionary is an unordered and mutable collection of key-value pairs. Dictionaries are defined using curly braces { } and consist of keys and their associated values separated by colons. Each key in a dictionary must be unique. Here's an overview of working with dictionaries

- Creating a Dictionary
- Accessing Values
- Modifying Values
- Adding New Key-Value Pairs
- Removing Key-Value Pairs
- Dictionary Operations
- Nested Dictionaries

Mini Project: #1 SIMPLE PHONE BOOK #2 STUDENT MARKS RECORD

#3 GROCERY PRICE CALCULATOR

Functions

Function Definition

- Function Call
- Return Statement
- Types of Parameters – Default Parameters, Variable Length Arguments
- Variable-Length Argument Lists

- Lambda Functions
- Recursion

Package

In Python, a package is a way of organizing related modules into a single directory hierarchy. It provides a mechanism for organizing large codebases and avoids naming conflicts between modules. A package is essentially a directory that contains a special file called __init__.py, which can be empty or contain Python code. This file is executed when the package is imported.

- Creating a Package
- Importing Modules from a Package
- Importing the Whole Package
- Subpackages

Python – OOPs Introduction:

In Python Object-Oriented Programming (OOP) is a programming paradigm that uses objects–collections of data and methods that operate on that data–to design and organize code. Python is an object-oriented programming language that supports the core principles of OOP. Here's an introduction to OOP concepts in Python

- Class
- Object
- Attributes and Methods
- Encapsulation
- Inheritance
- Polymorphism

- Abstraction

Type of Methods:

In object-oriented programming (OOP), methods are functions that are associated with objects. They define the behavior of the objects and how they interact with each other. In Python, there are several types of methods, each serving a different purpose. Here are the main types of methods in Python

- Instance Methods
- Class Methods
- Static Methods
- Special Methods (Magic Methods or Dunder Methods)

Exception Handling:

Exception handling in Python allows you to handle runtime errors gracefully, preventing your program from crashing and providing a way to recover from unexpected situations. Python uses a try-except block to catch and handle exceptions. Here's an overview of exception handling in Python.

- Try-Except Block
- Handling Specific Exceptions
- Else and Finally Blocks
- Raising Exceptions
- Custom Exceptions

File Handling

File handling in Python allows you to work with files on your computer. You can read from and write to files using various methods and modes. Here's a basic overview of file handling in Python

- Opening a File
- Reading from a File
- Writing from a File
- Appending to a File
- Using with Statements
- File Modes
- Exception Handling for File Operations
- Working with Paths

Mini Project: #1 Simple Notepad, #2 Student Record System #3 TO DO using File

Regular Expression

Regular expressions (regex or regexp) are sequences of characters that define a search pattern. They are used for pattern matching within strings, making them a powerful tool for text processing and manipulation. Here's a brief overview of common regular expression concepts and usage in Python

- Basics of Regular Expressions
- Using Regular Expressions in Python

Mini Project: #1 Email Validator #2 Mobile Number Extractor #3 Password Validator

#4 Date Extractor #5 Username Extractor from Email #6 HTML Tag Remover

Multithreading

Multithreading in Python allows you to execute multiple threads (smaller units of a process) concurrently, improving the performance of certain types of programs. However, due to the Global Interpreter Lock (GIL) in CPython, the benefits of multithreading for CPU-bound tasks are limited. Multithreading is more effective for I/O-bound tasks where threads can wait for I/O operations to complete.

- Creating Threads
- Thread Synchronization
- Thread Communication
- Daemon Threads

Postgresql Essentials for Full Stack Developers

Module 1 : Introduction to Databases and PostgreSQL

- What is a Database? Types of databases
- RDBMS vs NoSQL (with examples)
- Why PostgreSQL for Full Stack development?
- Installing PostgreSQL and pgAdmin
- Introduction to PostgreSQL ecosystem (CLI tools, GUI, drivers)

Module 2 : SQL Basics – Table Operations & CRUD

- Filtering with WHERE, AND, OR, IN, BETWEEN, LIKE
- Sorting Results: ORDER BY
- Limiting Data: LIMIT, OFFSET
- Aggregate Functions: COUNT, SUM, AVG, MIN, MAX

- Grouping Data:
- GROUP BY, HAVING

Module 3 : Advanced Joins and Subqueries

- Types of Joins:
- INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN
- Writing Nested Queries and Subqueries
- Using subqueries in SELECT, WHERE, FROM
- Use cases in reporting and data analysis

Module 4 : Views, Indexes, and Query Optimization

- Creating and using VIEWs
- Materialized Views: use cases and refreshing
- Creating Indexes: B-tree, expression-based indexes
- Query optimization basics with EXPLAIN ANALYZE
- When and how indexes improve performance

Module 5 : Transactions and Data Integrity

- Transactions: BEGIN, COMMIT, ROLLBACK
- Ensuring ACID properties
- Savepoints and nested transactions
- Handling transaction errors
- Use cases: banking systems, form submissions

Module 6 : Functions, Stored Procedures, and Triggers

- Defining SQL & PL/pgSQL functions
- Returning values, variables, and conditional logic
- Stored Procedures
- Triggers:
- BEFORE and AFTER triggers for INSERT, UPDATE, DELETE

Module 7 : Relational Modeling and Database Design

- Normalization:
- 1NF, 2NF, 3NF with examples ⁕ Entity Relationship Diagrams (ERD)
- Implementing:
- One-to-One
- One-to-Many
- Many-to-Many (junction tables)
- Using UUID as Primary Keys

Module 8 : Access Control and Security

- Creating database users and roles
- Granting and revoking permissions:
- GRANT, REVOKE
- Role-based access control (RBAC)
- Best practices for securing PostgreSQL in production
- Schema-level and table-level permissions

HTML Essentials for Full Stack Developers

Module 1 : HTML Elements

- Introduction to HTML syntax
- Block-level vs Inline elements
- Common elements: , , <p>, <a>, <img>,
, <hr>
- Nesting elements correctly *Outcome: Mini Project #1*

Module 2 : HTML Attributes

- Global attributes: id, class, title, style, lang
- data Specific attributes: href, src, alt, type, value, placeholder
- Boolean attributes: checked, disabled, readonly, required

Outcome: Mini Project #2

Module 3 : HTML Forms

- <form> element and its attributes (action, method, etc.)
- Input fields: <input>, <textarea>, <select>, <option>, <button>
- Fieldsets, legends, and labels
- Form validation (required, pattern, minlength, maxlength)

Outcome: Mini Project #3

Module 4 : HTML Lists

Ordered Lists: <ol>,

- Unordered Lists: <ul>,
- Description Lists: <dl>, <dt>, <dd>
- Nesting and styling lists

Outcome: Mini Project #4

Module 5 : HTML Tables

- Creating tables with <table>, <tr>, <th>, <td>
- Table structure: <thead>, <tbody>, <tfoot>
- Merging cells with colspan and rowspan
- Table accessibility and semantics *Outcome: Mini Project #5*

Module 6 : Semantic HTML

- Importance of semantic elements for SEO & accessibility
- Elements: <header>, <nav>, <main>, <section>, <article>, <aside>, <footer>
- When and where to use semantic tags

Outcome: Mini Project #6

Module 7 : HTML Media

- Images: <img>, srcset, alt, picture
- Video: <video>, controls, autoplay, loop, muted
- Audio: <audio>, controls, autoplay, loop, preload
- Embedding YouTube and other media

Outcome: Mini Project #7

Module 8 : Input Types

- Text-based: text, email, password, search, url, tel
- Choice-based: checkbox, radio
- Date/time: date, datetime-local, time, month, week
- Others: file, color, range, number, hidden

Outcome: Mini Project #8

CSS Essentials for Full Stack Developers

Module 1 : CSS Selectors

- Basic selectors: element, class, ID
- Grouping and combining selectors
- Attribute selectors
- Combinators: descendant, child (>), adjacent sibling (+), general sibling (~)

Outcome: Mini Project #1

Module 2 : Box Model

- Understanding content, padding, border, and margin
- Using box-sizing: border-box
- Visualizing box dimensions with dev tools
- Margin collapsing

Outcome: Mini Project #2

Module 3 : Display Property block, inline, inline-block, none flex

- grid
- visibility: hidden vs display: none

Outcome: Mini Project #3

Module 4 : Positioning

- Static, Relative, Absolute, Fixed, Sticky
- Using top, right, bottom, left with positioning
- Z-index and stacking context

Outcome: Mini Project #4

Module 5 : Flexbox

- Introduction to display: flex

- Main axis vs cross axis
- Properties: justify-content, align-items, align-content, flex-wrap
- Flex item properties: flex, flex-grow, flex-shrink, flex-basis, order

Outcome: Mini Project #5

Module 6 : CSS Grid

- Introduction to grid layout
- Creating columns and rows using grid-template-columns and grid-template-rows
- Placing items with grid-column, grid-row
- Grid gap, alignment, and nested grids

Outcome: Mini Project #6

JavaScript Essentials for Full Stack Developers

Module 1 : Variables & Data Types

- Understand var, let, and const when and why to use each
- Explore JavaScript's powerful dynamic typing system
- Work with strings, numbers, booleans, null, undefined, and more

Module 2 : Operators & Logic

- Arithmetic, assignment, comparison, and logical operators
- Master the building blocks of computation and condition

Module 3 : Conditionals & Control Flow

- Decision–making with if, else, else if, and switch
- Write clean, readable branching logic

Module 4 : Loops & Iteration

- for, while, and do...while loops
- Introduction to forEach() for working with arrays

Module 5 : Functions

- Function declarations vs expressions
- Arrow functions: cleaner syntax, smarter scopes

  Parameters, return values, and function composition

Module 6 : Events & Interaction

- Add interactivity with addEventListener()
- Handle clicks, inputs, mouse events

React Essentials for Full Stack Developers

1. JSX – JavaScript XML

Understand JSX syntax and how it integrates HTML-like code within JavaScript.

Learn JSX rules, expressions, and embedding JavaScript logic into markup.

2. Components: Functional & Class

Build UI with reusable functional components (primary focus).

Introduction to class components and when they are used.

Understand component structure, export/import, and nesting.

3. Props and State

Pass dynamic data to components using props.

Manage component data using state in functional components.

Understand unidirectional data flow and state lifting.

4. Conditional Rendering

Render elements based on logic using if-else, ternary, &&

operators. Show loading states, error messages, or user-specific

views.

5. List Rendering with map()

Use JavaScript's map() to dynamically generate lists.

Assign unique keys to improve rendering performance. Render nested data or display

lists with interaction.

6. Event HandlingReact Essentials for Full Stack DevelopersAttach event listeners like

onClick, onChange, onSubmit.

Write custom event handlers and pass data using functions.

Use synthetic events in Reacts cross-browser environment.

7. Component Lifecycle (Class Components)

Learn lifecycle methods: componentDidMount, componentDidUpdate, componentWillUnmount.

Use lifecycle concepts to manage effects like API calls or cleanup.

Understand lifecycle phases: mounting, updating, unmounting.

8. React Hooks

useState: manage local component state. useEffect: run side effects like fetching data or subscriptions. useContext: access global state without prop drilling. useRef: persist values and access DOM elements.

9. React Router

Set up routing using BrowserRouter, Routes, and Route.

Navigate between pages with Link and useNavigate.

Create dynamic routes using URL parameters.React Essentials for Full Stack Developers

10. Fetching Data (APIs)

Fetch data from backend APIs using fetch() or Axios.

Display data using useEffect and handle loading/error states.

Create reusable API utility functions.

Use API data to populate components dynamically.

Django Project-Oriented Training

Module 1: Django MVT Architecture

- What is a server? Understanding basic client-server architecture
- HTTP methods: GET, POST, PUT, DELETE

- Introduction to Django and its advantages
- MVT (Model–View–Template) Architecture in Django
- Request–response flow in Django
- **Assignment:**
- Explain MVT architecture with a diagram.
- Create a Django project and one app called core.
- Display "Hello, Django!" on the homepage using a view and template.

Module 2: pip and Virtualenv Concepts

- Creating and managing Python virtual environments
- Installing Django using pip
- Understanding Django project and app structure
- Importance of requirements.txt
- Difference between .py files, apps, and packaged libraries

Assignment:

- Set up a virtual environment and install Django.
- Create a requirements.txt and demonstrate installing from it.
- Write the purpose of manage.py, settings.py, and urls.py.

Module 3: Registration Form Using Django

- Django Form vs ModelForm
- Creating models for user registration
- Using built-in User model or custom AbstractUser
- Adding validations in forms

- Rendering templates using Django Template Language (DTL)
- Using Crispy Forms (optional for styling)

Assignment:

- Create a registration form with name, email, username, and password.
- Add validations (e.g., required, email format, min password length).
- Display success message after submission.

Module 4: Color Picker using Django

- Creating color input form in HTML
- Capturing user input via views
- Sending and displaying dynamic data (list of colors) in template
- Using Django templates to render lists dynamically

Assignment:

- Create a form with a color input.
- Display the selected color below the form.
- Maintain a list of recently selected colors and display them.

Module 5: Simple Calculator with Django

- Building a calculator form with two inputs
- Performing arithmetic logic in the view
- Displaying calculated result on the same page

Assignment:

- Build a calculator that accepts two numbers and an operation (+, −, ★, /).
- Show result on the same page without redirection.
- Handle division by zero error.

Module 6: Product Catalogue for E-Commerce Site

- Creating Product model

- Using Django ORM to store and retrieve products
- Creating product list and detail views
- Displaying products using templates

Assignment:

- Create a Product model with name, price, description.
- Display all products in a grid view.
- Create a product detail page using dynamic URL routing.

Module 7: Quiz Application using Django

- Creating models for questions and answers
- Displaying quiz using Django forms
- Capturing user answers
- Calculating and displaying result
- Optional: Frontend using React

Assignment:

- Create Question and Answer models (OneToMany).
- Display quiz questions as a form.
- Save the score and show a results page.

Module 8: To-Do List

- Creating Task model
- Using Django sessions to track user tasks
- Add, delete individual tasks
- Delete all tasks
- Optional: React frontend

Assignment:

- Create a Task model with task name, status (done/pending).

- Add new tasks via a form.
- Display and delete individual tasks.
- Add "Clear All" functionality.

Module 9: REST API Integration with Django

- Introduction to Django REST Framework
- Creating simple API endpoints
- Consuming external APIs (e.g., OpenWeatherMap) using requests
- Parsing and displaying weather data in view

Assignment:

- Use requests to call the OpenWeatherMap API.
- Show weather for a user-input city.
- Handle invalid city name error.

Module 10: API Documentation

- Adding Swagger/OpenAPI docs using drf-yasg or drf-spectacular
- Visualizing and testing APIs via Swagger UI

Assignment:

- Create simple APIs using Django REST Framework (GET, POST).
- Install and configure Swagger using drf-yasg.
- Test your API endpoints via Swagger UI and Postman.

Module 11: Django Exception Handling

- Handling Django's built-in exceptions
- Creating custom exception classes
- Displaying error messages gracefully in templates

Assignment:

- Raise a custom exception when a product price is negative.
- Display user-friendly messages using templates.
- Create custom 404 and 500 error pages.

Module 12: Logging with Python's logging module

- Configuring logging in settings.py
- Adding log messages in views and models
- Writing logs to files

Assignment:

- Configure logging in settings.py.
- Log every view access and form submission.
- Log form validation errors.

Module 13: Django ORM and Advanced Queries

- Using Model.objects to query the database
- Difference between basic CRUD and advanced ORM
- Writing custom queries with Q() and annotations
- Using select_related, prefetch_related for JOINs

Assignment:

- Write ORM queries using filter, exclude, Q, and annotate.
- Create a model with ForeignKey and perform JOIN queries.
- Show related data in templates.

Module 14: PostgreSQL Integration

- Installing PostgreSQL and psycopg2
- Configuring DATABASES in settings.py
- Running migrations on PostgreSQL

- Connecting Django to remote PostgreSQL DB (optional)

Assignment:

- Install PostgreSQL and connect it with Django.
- Run migrations and verify data is saved in PostgreSQL.
- Export and import PostgreSQL database using pg_dump and pg_restore.

Module 15: Django Authentication and Authorization

- Using built-in User model and auth system
- Creating login, logout, and registration views
- Using decorators like @login_required
- Defining user roles with Groups and Permissions
- Restricting access to views/templates based on roles

Assignment:

- Create login, logout, and register views.
- Restrict access to dashboard for only logged-in users.
- Add admin, user roles and show different menus per role.

Module 16: E-Commerce Project

- Creating products as JSON (if needed for API)
- Product List Page
- Add to Cart Functionality (using session or DB)
- Saving Cart to Database
- Delete From Cart
- View Cart with total price

Assignment:

- Display all products with "Add to Cart" button.
- Use session to store cart items.

- Implement cart view, delete from cart, and display total.
- Optional: Save orders to database.

CONTACT:

PAYILAGAM

7 VIJAYA NAGAR FIRST MAIN ROAD

VELACHERY CHENNAI 600042

https://payilagam.com

+91 8344777333 / +91 8883775533

info@payilagam.com