COURSE TITLE: Object-Oriented Programming - Python ¬ TARGET AUDIENCE: 3rd-year students ¬ PREREQUISITES: To start using Python, you must have some basic knowledge of the following technologies:

1. HTML5/CSS3: Since Django is used to develop web applications, a basic understanding of HTML and CSS is necessary.
2. SQL: Django uses an ORM (Object-Relational Mapping) to interact with the database.
3. Command Line: django-admin is Django's command-line utility for administrative tasks. Additionally, manage.py is automatically created in every Django project. It does the same thing as django-admin but also sets the DJANGO\_SETTINGS\_MODULE environment variable to point to the project's settings.py file.
4. JavaScript: Although not strictly necessary to use Django, a basic understanding of JavaScript can be very helpful, especially if you plan to add client-side interactivity to your web applications.

¬ GENERAL OBJECTIVE The general objective of this course is not only to enable the student to master object-oriented programming in Python but also to be able to take control of the DJANGO framework, i.e., to master its principles and best practices. ¬ SPECIFIC OBJECTIVES

1. Model problems using OOP concepts.
2. Describe basic OOP concepts.
3. Distinguish between classes and objects.
4. Know how to interface Django with a database;
5. Develop dynamic views and templates in Django;
6. Manage Django forms;
7. Implement major advanced functions

DURATION: • Hours: 30 hours • Lecture: 10 hours • Tutorial: 05 hours • Practical work: 15 hours • Timeframe: Semester 5

EVALUATION Students are evaluated in two ways

* Continuous assessment: The teacher may assess their students through projects, presentations, or written evaluations
* Exams: At the end of the semester, the teacher will conduct a written exam according to the academic calendar.

KEYWORDS: Python, Django, CSRF. ¬ TEACHING METHOD (PEDAGOGICAL APPROACH) 1st point: Highlight the main themes to be mastered by learners at the beginning of the course. 2nd point: Lecture, the teacher should focus the lecture on the essentials. Not delve too much into details; emphasis should be more on practice. 3rd point: Accompany the main themes of the course with practical work 4th point: Conduct written exams at the end of the semester ¬ UNIVERSAL VALUES THAT THE TEACHER SHOULD ADHERE TO DURING THE COURSE

* Discipline (examples: not allowing a student to enter after you unless the reasons for going outside are regulatory, proper dress code,
* Respect
* Punctuality

¬ TEACHER'S BEHAVIOR The teacher must have self-control in front of the students. They must demonstrate a good mastery of their course. They should be calm and attentive to their students. They should politely correct them. They should be able to manage their class.

¬ DOCUMENTATION courses.programming.in <https://roadmap.sh/python> <https://www.programiz.com/python-programming> <https://learn.microsoft.com/fr-fr/training/modules/python-object-oriented-programming/> <https://docs.djangoproject.com/fr/5.0/>

COURSE PLAN

Chapter 1: Review of basic concepts

1. Syntax of a Python code
2. Lists, tuples, sets, dictionaries
3. Functions
4. If elif else statements
5. For / while loops
6. Modules
7. Package managers o PyPI o Pip o Conda
8. Regular expressions

Chapter 2: Classes and objects

Chapter 3: Constructors and Destructors

Chapter 4: Inheritance, types of inheritance, and Polymorphism

Chapter 5: Operator overloading

Chapter 6: Iterators and Generators

Chapter 7: Closures and Decorators

Chapter 8: Exceptions

Chapter 9: Testing

1. Pytest
2. Doctests
3. UnitTest / pyUnit
4. nose

Chapter 10: Frameworks

1. Differences between asynchronous and synchronous frameworks
2. Django

2.1 Django Overview

2.2 Installing Django

2.3 Initializing a Django project

2.4 Creating a Django application

2.5 Django ORM

2.6 Django Login

2.7 Models layer

2.7.1 Models

2.7.2 QuerySets

2.7.3 Instance methods and accessing related objects

2.7.4 Existing databases and supported databases

2.8 Views layer

2.8.1 URLConfs

2.8.2 View functions

2.8.3 Shortcuts

2.8.4 Included views

2.8.5 Request-response objects and templateResponse objects

2.8.6 Generating PDFs and CSVs

2.8.7 Included middleware classes

2.9 Templates layer

2.9.1 Included tags and filters

2.9.2 Custom tags and filters

2.9.3 Template API

2.9.4 Custom template engine

2.10 Forms

2.10.1 Forms API

2.10.2 Included fields / included components

2.10.3 Model forms

2.10.4 Customizing validation

2.11 Django admin interface

2.12 Security

2.12.1 Published security advisories

2.12.2 Protection against clickjacking

2.12.3 Protection against Cross-Site Request Forgery (CSRF) Token

(Note: This Chapter 10 can be given as Projects to Students because there are several Packages in Django 10).