Course title: Computational Economics			
Course code: UEO 402	**No. of credits**: 4	**L-T-P:** 30-12-36	**Learning hours:** 78
Pre-requisite course code and title: UEO103: Introductory Mathematical Methods for Economics, UEO102: Basic Mathematics for Economics, UDS103: Mathematics for Data Science, Introductory UEO 303: Econometrics, Intermediate Microeconomics, Intermediate Macroeconomics			
Department: Department of Policy and Management Studies			
Course coordinator: Dr. Priyanka. Arora			**Course instructor**: Guest Faculty
Contact details: priyanka.arora@terisas.ac.in			
Course type: Core			**Course offered in:** 7th Semester

Course description: This course gives students a gentle, hands-on introduction to using the computer to explore economic ideas, working in the Python programming language. No prior programming experience is assumed. Building on their earlier study of economics, mathematics, and statistics, students learn to turn simple economic ideas into short computer programs, run them, and make sense of the results. The course covers the basics of Python programming, working with matrices on the computer, randomness and simulation, and a simple way of describing how economies change over time (Markov chains). The emphasis throughout is on doing and understanding rather than on advanced theory or proofs. Applications are drawn from development economics, environmental economics, and sustainability — themes central to the TERI SAS curriculum. The final module is a capstone project in which students build a small computational economics application on a topic of their choice. The course draws on Stachurski (2022), Economic Dynamics: Theory and Computation, 2nd edition (MIT Press) and the open-access QuantEcon Python lecture series by Sargent and Stachurski, using mainly the introductory lectures from these resources at a level suited to undergraduate students.

Course objective:

1. To equip students with Python programming skills for scientific computing in economics.
2. To develop the ability to translate economic models into computable algorithms and simulations.
3. To understand core computational techniques — working with matrices on the computer, simple simulation, and Markov chains — in the context of economic applications.
4. To apply computational economics methods to problems in development, sustainability, and environmental policy relevant to the Indian and global context.

Course contents

S.No	Topics	L	T	P	
1	**Foundations: Python for Economists** Why computation is useful in economics, explained with simple examples. Getting started with Python: installing and running Python, using Jupyter Notebooks and Google Colab. The building blocks of programming: variables, data types, doing arithmetic, lists, if-statements, and loops. Writing simple functions. A first look at NumPy arrays for handling numbers in bulk. Making basic plots with Matplotlib, such as a supply-and-demand diagram or a simple time series. This module assumes no prior programming experience and moves at a gentle pace, with plenty of practice time.	10	3	6	

2	**Working with Matrices in Python** This module shows how to use the matrix ideas students have already met in their mathematics courses, this time on a computer. Creating and working with vectors and matrices in NumPy. Basic operations: addition, multiplication, and the transpose. Using the computer to solve a system of linear equations. A worked economic example: setting up and solving a small input-output (Leontief) model in Python. The focus is on doing, not on theory or proofs; the underlying mathematics is taken as given.	6	2	4	
3	**Randomness and Simulation** A gentle, computer-based look at randomness. What random variables and probability distributions are, explained through examples rather than formulas. Generating random numbers in Python. The idea of the Law of Large Numbers seen by experiment: as we take more random draws, the average settles down. Running simple simulations and reading the results. Applications kept simple: simulating dice and coin tosses, and building a simple income distribution to see how unequal it is. Monte Carlo Simulations.	6	3	7	
4	**How Economies Change Over Time: Markov Chains** A simple way of describing how something moves between different situations, or "states," over time. The idea of states and the chances of moving from one state to another, set out in a table. Worked economic examples kept intuitive: a worker moving between employment and unemployment, or a household moving between income groups. Using Python to follow these movements step by step and to see where things tend to settle in the long run. Application: thinking about poverty traps as a state that is hard to leave. The treatment is hands-on and example-led; no heavy theory. Introduction to Dynamic Programming.	8	4	7	
5	**Capstone Project: Applications to Development and Sustainability** A project-based module. Working in pairs, students bring together what they have learned to build a small computational economics application on a topic of their choice. Suggested topics, kept at an achievable level: (i) a simple simulation of how a household's income changes over time; (ii) using a Markov chain to describe a shift towards renewable energy; (iii) simulating how a shared resource (such as a village pond or forest) is used; (iv) a simple model of people moving from villages to cities; (v) exploring a real economic dataset and presenting clear charts; (vi) labour market outcomes — for example, simulating how workers move between employment, unemployment, and informal work as a Markov chain, or exploring how wages and employment differ across groups using a real labour-force dataset; (vii) technical change — for example, a simple simulation of how a new technology spreads across firms or households over time (an adoption / diffusion curve), or a Markov chain describing the gradual replacement of an older technology by a newer one. Deliverable: a Jupyter Notebook containing the code, a short written explanation, and charts, presented in the final week.	0	0	12	
	Total	**30**	**12**	**36**	

Pedagogical approach: Lectures introduce economic motivation and algorithmic intuition. Tutorials involve pen-and-paper problem sets, pseudocode exercises, and worked examples. Practicals are hands-on Python coding sessions using Jupyter Notebooks / Google Colab. The capstone project integrates skills across modules and connects to TERI SAS focus areas in environment and development.

Evaluation criteria:
Minor 1: Written Examination - 20% [Syllabus: 1-3, Learning outcomes: 1-3]
Minor 2: Coding Assignments (4 assignments, one per module cluster) - 30% [Learning outcomes: 1-4]
Class Participation and Practical Attendance: 10% [Learning outcomes: 1-4]
Major: Final Capstone Project (Jupyter Notebook + Presentation) - 40% [Syllabus: Complete course, Learning outcomes: 1-4]
Additional Notes: The capstone project requires students to apply computational methods to a development or sustainability problem, producing a reproducible Jupyter Notebook with code, analysis, and visualisations, followed by a 10-minute presentation. Coding assignments will be submitted as Jupyter Notebooks via the course portal.

Learning outcomes:
At the end of the course, the students will be able to:
1. Write Python programs to solve, simulate, and visualise economic models. [Modules 1-5]
2. Apply computational techniques — matrices, simple simulation, and Markov chains — to standard economic problems. [Modules 2-4]
3. Take a simple economic idea, write a short program to explore it, and explain in plain terms what the results mean. [Modules 2-4]
4. Design and execute an independent computational economics project on a topic related to development, sustainability, or environmental policy, producing a reproducible research notebook. [Module 5]

Core readings:
1. Stachurski, J. (2022). *Economic Dynamics: Theory and Computation*, 2nd ed. MIT Press. [EDTC]
2. Sargent, T.J. and Stachurski, J. *Quantitative Economics with Python*. https://python.quantecon.org [QE]
3. Sargent, T.J. and Stachurski, J. *Python Programming for Economics and Finance*. https://python-programming.quantecon.org [PPE]
Additional readings:
4. Miranda, M.J. and Fackler, P.L. (2002). *Applied Computational Economics and Finance*. MIT Press.
5. Judd, K.L. (1998). *Numerical Methods in Economics*. MIT Press.
6. Wooldridge, J.M. *Introductory Econometrics: A Modern Approach*. Cengage Learning. (for econometric applications in practicals)
7. For introduction to dynamic programming : https://web.mit.edu/15.053/www/AMP-Chapter-11.pdf

Module-wise chapters from the core readings:
1. Module 1: PPE, Lectures 1–8; EDTC, Chapter 1
2. Module 2: EDTC, Chapter 2; QE, Linear Algebra lecture
3. Module 3: EDTC, Chapters 3–4; QE, LLN and CLT lecture
4. Module 4: EDTC, Chapters 5–6; QE, Finite Markov Chains lecture, MIT lecture link
5. Module 5: No prescribed readings; students draw on all modules and select simple application-specific references

Course prepared by: Dr. Priyanka Arora

Student responsibilities: Attendance, feedback, discipline: as per university rules.

Course reviewers:
1.Dr. Nishika Bhatia, Associate Professor & Assistant Dean, Jindal School of Banking and Finance, Jindal Global University.
2. Dr. Mukta Jain, Assistant Professor, Jindal Global Law School, Jindal Global University.