Outline of Queueing Theory:

**Week 1: Introduction to Queueing Theory**

* **Lecture:** Introduce the basic concepts of queueing systems, their components (arrival process, service mechanism, queue discipline), and key performance metrics.
* **Python Exercise:** Develop a plotting the probability density function for the exponential distribution and arrival process over time using Python.
* **Assignment:** Read the designated chapters and summarize the core concepts of queueing theory.

**Week 2: Stochastic Processes and the Poisson Process**

* **Lecture:** Cover the fundamentals of stochastic processes with a focus on the Poisson process and its applications in modeling customer arrivals.
* **Python Exercise:** Write a program to generate a Poisson arrival sequence and integrate it into a queueing simulation.
* **Assignment:** Analyze the characteristics of the Poisson process under different arrival rates and prepare a short report comparing simulation results with theoretical predictions.

**Week 3: Renewal Processes and Regenerative Theory**

* **Lecture:** Discuss renewal processes and the concept of regenerative points, and explain their role in analyzing queueing systems.
* **Python Exercise:** Simulate a queueing model (e.g., an M/G/1 queue) that incorporates a regenerative structure to observe the impact of regenerative points on performance.
* **Assignment:** Compute regenerative-related statistics from your simulation and compare these with theoretical values.

**Week 4: Mathematical Analysis of Basic Queueing Models**

* **Lecture:** Dive into the mathematical analysis of basic models such as M/M/1 and M/M/c, covering balance equations, transition probabilities, and performance measures.
* **Python Exercise:** Implement simulations for these models and calculate key metrics such as average waiting time and system length.
* **Assignment:** Compare the performance differences between single-server and multi-server queues under varying loads and write an analytical report.

**Week 5: Queueing Systems with General Service Time Distributions**

* **Lecture:** Examine systems where service times follow a general (non-exponential) distribution, focusing on models like G/M/1 and G/G/1.
* **Python Exercise:** Simulate a queueing system with non-exponential service times and analyze how different service distributions affect performance.
* **Assignment:** Design experiments by varying the service time distribution and investigate its impact on waiting times and system stability, then document your findings.

**Week 6: Priority Queueing Systems**

* **Lecture:** Introduce priority queueing strategies, discussing both non-preemptive and preemptive priority schemes and their mathematical formulations.
* **Python Exercise:** Develop a simulation of a multi-priority queueing system to observe how different priority rules affect performance metrics.
* **Assignment:** Create and simulate a two-tier (or multi-tier) priority queueing system, and write a report discussing the advantages and drawbacks of various priority strategies.

**Week 7: Queueing Networks**

* **Lecture:** Explore the concepts behind multi-node queueing systems (e.g., Jackson networks) and methods for analyzing them.
* **Python Exercise:** Build a simple queueing network simulation using Python to model customer flow between multiple service nodes.
* **Assignment:** Design a network of several service nodes, analyze how interactions between nodes affect overall system performance, and submit your analysis.

**Week 8: Stability Analysis of Queueing Systems**

* **Lecture:** Discuss the conditions for system stability, the concept of critical load, and describe the behavior of systems under different loading conditions.
* **Python Exercise:** Simulate queueing systems under varying loads to examine the transition from stability to instability.
* **Assignment:** Modify system parameters to document the transition from stable to unstable behavior and write a detailed experimental report.

**Week 9: Numerical Methods for Queueing Systems**

* **Lecture:** Introduce numerical methods (e.g., Markov chains and solving systems of linear equations) for computing performance metrics in queueing systems.
* **Python Exercise:** Use numerical techniques to determine the steady-state distribution of a queueing model and compare it with simulation results.
* **Assignment:** Select one model, apply both numerical and simulation methods, and write a comparative analysis of the results.

**Week 10: Approximate Solutions for Queueing Systems**

* **Lecture:** Discuss common approximation methods (such as heavy traffic approximations and fluid limits) used for complex or large-scale queueing systems.
* **Python Exercise:** Implement an approximation algorithm in Python and compare its performance with exact solutions under various scenarios.
* **Assignment:** Evaluate the accuracy of the approximation method by comparing it to the precise solution, discussing its range of applicability and potential sources of error.

**Week 11: Simulation Techniques and Tools for Queueing Systems**

* **Lecture:** Cover the advantages and limitations of discrete-event simulation in queueing analysis and introduce popular simulation libraries (e.g., SimPy, queueing-tool).
* **Python Exercise:** Construct a simulation framework for a queueing system using one of these libraries, and validate the simulation results.
* **Assignment:** Design a simulation experiment to study a particular queueing model and write a report detailing your methodology, results, and conclusions.

**Week 12: Advanced Simulation Applications in Python**

* **Lecture:** Delve deeper into discrete-event simulation (DES) principles and demonstrate their application in complex queueing systems, including event scheduling and state updates.
* **Python Exercise:** Develop a comprehensive simulation environment using SimPy (or a similar library) to model multiple interacting events within a queueing system.
* **Assignment:** Create a detailed simulation experiment based on a specified queueing model, complete with performance analysis and a written report.

**Week 13: Heavy Traffic Approximations and Limit Theorems**

* **Lecture:** Introduce heavy traffic approximations, fluid limits, and diffusion approximations to predict system behavior under high-load conditions.
* **Python Exercise:** Simulate a queueing system operating under heavy load and compare the simulation results with predictions from heavy traffic theory.
* **Assignment:** Design experiments that vary system load and assess the accuracy of the heavy traffic approximations, then prepare an analysis report.

**Week 14: Sensitivity Analysis and Queueing System Optimization**

* **Lecture:** Discuss techniques for sensitivity analysis of system parameters and introduce optimization strategies (e.g., adjusting the number of service channels or reallocating resources) to improve performance.
* **Python Exercise:** Develop a simulation script that systematically varies key parameters to perform sensitivity analysis and identify optimal configurations.
* **Assignment:** Choose a queueing model, determine the optimal set of parameters through numerical experimentation, and submit an optimization report detailing your findings.

**Week 15: Case Studies and Practical Applications**

* **Lecture:** Present real-world queueing system case studies (e.g., call centers, emergency rooms, telecommunication networks) and discuss design considerations and model limitations.
* **Python Exercise:** Select a real-world case and simulate it using theoretical models to evaluate performance and potential improvements.
* **Assignment:** Conduct an in-depth analysis of a chosen real-world queueing system, including simulation results and proposed enhancements, and compile a case study report.

**Week 16: Capstone Project and Final Presentations**

* **Lecture:** Review the key topics covered throughout the course, including various queueing models, simulation techniques, numerical and approximate methods, and optimization strategies. Engage in a Q&A and discussion session.
* **Python Exercise:** Students select a queueing model or real-world case to develop a comprehensive simulation project, integrating all techniques learned during the course.
* **Assignment:** Complete a final project report and presentation that synthesizes course material to solve a practical problem, demonstrating both theoretical understanding and programming proficiency.

**References**

* Leonard Kleinrock, *Queueing Systems, Volume 1: Theory*, Wiley, 1975.
* [SimPy Documentation](https://simpy.readthedocs.io/) – for simulation techniques in Python.

This course design aims to balance theory and practice, reinforcing each week’s lecture with a Python programming exercise and an assignment to solidify the concepts learned.

Refer to NYCU OpenCourseWare (陽明交大開放課程):

* Chapter 17 Queueing Theory (1/5): <https://ocw.nycu.edu.tw/?post_type=course_page&p=91439>
* Chapter 17 Queueing Theory (2/5): <https://ocw.nycu.edu.tw/?post_type=course_page&p=91445>
* Chapter 17 Queueing Theory (3/5): <https://ocw.nycu.edu.tw/?post_type=course_page&p=91451>
* Chapter 17 Queueing Theory (4/5): <https://ocw.nycu.edu.tw/?post_type=course_page&p=91461>
* Chapter 17 Queueing Theory (5/5): <https://ocw.nycu.edu.tw/?post_type=course_page&p=91467>
* Queueing Theory - Chapter 1 Introduction [https://hackmd.io/@kaeteyaruyo/B14oAPqJ9](https://hackmd.io/%40kaeteyaruyo/B14oAPqJ9)