# Advanced Python and AI Programming Syllabus

High School - Year Long Course (110 hours)

## Course Overview and Goals

Advanced Python and AI Programming builds on foundational programming skills to teach object-oriented design, common data structures and algorithms, and the fundamentals of artificial intelligence and machine learning, all using Python. Students write and test increasingly complex programs, working with third-party libraries and packages, custom classes, and structures such as stacks, queues, linked lists, trees, and graphs. Three hands-on build projects — a text adventure game, a music player, and a machine learning classifier — give students the opportunity to apply what they learn to original, personally meaningful programs. By the end of the course, students will be able to design and implement object-oriented programs, analyze and choose data structures and algorithms based on efficiency, and build, train, and evaluate basic machine learning models.

### Learning Environment

The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each module of the course is broken down into lessons. Lessons may be composed of interactive learning pages, explorations, simulations, free-response prompts, coding exercises, and quizzes. Each module ends with a comprehensive quiz that assesses students' mastery of that module's material.

### Development Environment

Students write and run Python programs in the browser using the CodeHS editor.

### Prerequisites

This course is intended for students who have already completed a foundational programming course and are comfortable with Python basics such as variables, functions, loops, and conditionals. A supplemental Python Programming Fundamentals module is available within the course for students who need a refresher before beginning Module 1.

### Technology Requirements

To complete all activities and exercises in this course, students must have access to the 3rd party sites and tools listed here: [insert course whitelist link]

### More Information

Browse the content of this course at https://codehs.com/course/advanced_python/explore

## Course Breakdown

Module 1: Welcome to Advanced Python and AI (2 weeks / 10 hours)

Students review and extend core Python skills, including program structure, documentation, formatted output, exception handling, and lambda functions, to prepare for the object-oriented and data-driven work ahead.

| | |
|---|---|
| Topics Covered | • Main function & program structure<br>• Comments and docstrings<br>• Print parameters & f-strings<br>• Exception handling<br>• Lambda functions<br>• Python keywords<br>• Debugging practice<br>• Unit quiz |
| Example Assignments | • **Your Name and Hobby**<br>  ○ Students build a short program using a helper function and a `main()` function to print a personalized introduction.<br>• **Playlist Printer**<br>  ○ Students complete a function that prints songs in order using `print` parameters and f-strings.<br>• **Choosing the Right Exception**<br>  ○ Students debug a program by identifying and correcting mismatched exception types in `try`/`except` blocks.<br>• **Square and Multiply Numbers**<br>  ○ Students write lambda functions with arithmetic expressions and print the results. |

## Module 2: Object-Oriented Programming (2 weeks / 10 hours)

Students learn to design and implement classes, using constructors, getters and setters, inheritance, polymorphism, and magic methods to model real-world entities in code.

| | |
|---|---|
| Topics Covered | • Classes & objects<br>• Constructors<br>• Getters and setters<br>• Inheritance<br>• Polymorphism<br>• Magic methods<br>• Composition<br>• Unit quiz |
| Example Assignments | • **Movie Class**<br>  ○ Students define a `Movie` class with a constructor and attributes to store information about movies.<br>• **Participant Inheritance**<br>  ○ Students build a class hierarchy modeling participants and leadership roles in a community organization.<br>• **Awards Ceremony**<br>  ○ Students use polymorphism so different member subclasses each count their own service hours correctly.<br>• **Magic Legit Events**<br>  ○ Students extend an existing class with magic methods and a new attribute in an open-ended challenge. |

## Module 3: Libraries and Packages (2 weeks / 10 hours)

Students learn to use standard and third-party Python libraries to work with web data, perform numerical computing, clean and visualize datasets, and test their own code.

| | |
|---|---|
| Topics Covered | • Standard vs. third-party libraries<br>• Math and random modules<br>• Requesting web data<br>• Numerical computing with NumPy<br>• Cleaning data with pandas<br>• Visualizing data<br>• Unit testing with pytest<br>• Unit quiz |
| Example Assignments | • **Shuttle Planner**<br>  ○ Students use the `math` and `random` modules to plan a community shuttle trip.<br>• **Create Your Own Post**<br>  ○ Students write a helper function that sends a POST request and handles the response or error.<br>• **Clean the Donation**<br>  ○ Students use pandas to clean and summarize a food drive's donation log, removing impossible values.<br>• **Create Tests**<br>  ○ Students write pytest tests for a function that tracks food servings for a community kitchen. |

## Module 4: Build Your Own Adventure Game (1 week / 5 hours)

Students plan and build an original text adventure game across a series of milestones, applying classes, packages, and program design skills from earlier modules to their own story.

| | |
|---|---|
| Topics Covered | • Project planning<br>• Character classes<br>• Items and player choices<br>• Game loop design<br>• Using packages in a project<br>• Peer review<br>• Project reflection |
| Example Assignments | • **Plan Your Adventure**<br>  ○ Students design the blueprint for their own adventure game, specifying characters, goals, and story.<br>• **Project Milestone: Your Characters**<br>  ○ Students build the character classes for their own game based on their plan.<br>• **Project Milestone: The Game Loop**<br>  ○ Students add items and build a full game loop that runs their story through real player choices.<br>• **Peer Review**<br>  ○ Students play a classmate's game and review their code, providing specific and constructive feedback. |

## Module 5: Data Structures (4 weeks / 20 hours)

Students learn to implement and apply core data structures, including sets, stacks, queues, linked lists, hash tables, trees, heaps, and graphs, to solve realistic problems.

| | |
|---|---|
| Topics Covered | • Lists, tuples, and dictionaries<br>• Sets<br>• Stacks (LIFO)<br>• Queues (FIFO)<br>• Linked lists<br>• Hash tables<br>• Trees and heaps<br>• Graphs<br>• Unit quiz |
| Example Assignments | • **Word Count**<br>  ○ Students combine lists, tuples, and dictionaries to count and rank word frequency in a piece of text.<br>• **Browser History Tracker**<br>  ○ Students implement a stack-based class to manage browser history with the most recent page retrieved first.<br>• **Password Manager**<br>  ○ Students use a hash table to securely check passwords without storing them in plain text.<br>• **Maps as Graphs**<br>  ○ Students build a directed graph modeling the locations and connections of a fictional town. |

## Module 6: Algorithms (4 weeks / 20 hours)

Students analyze algorithm efficiency using Big O notation and implement classic search, sort, and recursive algorithms to understand their tradeoffs.

| | |
|---|---|
| Topics Covered | • Algorithm analysis<br>• Big O notation<br>• Linear search<br>• Binary search<br>• Sorting algorithms<br>• Recursion<br>• Merge sort<br>• Quick sort<br>• Unit quiz |
| Example Assignments | • **Testing Efficiency**<br>  ○ Students measure and compare the runtime of an improved search algorithm against the original version.<br>• **Draft Day Search**<br>  ○ Students build a fantasy-football draft tool that lets a user search and select from a list of available players.<br>• **Temperature Lookup**<br>  ○ Students implement two binary search variants to find the first and last |

| | |
|---|---|
| | position of a value in a sorted list.<br>• **Recursive Path Finding**<br>  ○ Students implement a recursive function that navigates a maze in an open-ended challenge. |

## Module 7: Build a Music Player (1 week / 5 hours)

Students design and build a working music player across a series of milestones, applying custom data structures and search and sort algorithms to their own project.

| | |
|---|---|
| Topics Covered | • Project planning<br>• Dictionaries and sets in practice<br>• Custom stacks, queues, and linked lists<br>• Search and sort implementation<br>• Program integration<br>• Efficiency analysis<br>• Peer review |
| Example Assignments | • **Data Modeling Plan**<br>  ○ Students decide which data structures and algorithms best fit each feature of their music player before building.<br>• **Playlist, Up Next, and Replays**<br>  ○ Students build stack, queue, and linked list structures from scratch and wire them into their player.<br>• **Search and Sort**<br>  ○ Students write and integrate a search feature and a sort feature into their music player.<br>• **Big O Analysis**<br>  ○ Students write an analysis of their finished player, naming the tool behind each feature and defending its cost against an alternative. |

## Module 8: Algorithms That Think (4 weeks / 20 hours)

Students explore foundational artificial intelligence concepts, including graph search algorithms, heuristic and greedy search, and core machine learning models such as linear regression, k-nearest neighbors, decision trees, neural networks, and clustering.

| | |
|---|---|
| Topics Covered | • Intro to AI<br>• Graph search (BFS/DFS)<br>• Weighted & heuristic search<br>• Linear regression<br>• K-nearest neighbors<br>• Overfitting & evaluation<br>• Decision trees<br>• Neural networks & perceptrons<br>• Clustering<br>• Unit quiz |
| Example Assignments | • **Shortest Route to the Festival**<br>  ○ Students use breadth-first search to find the shortest walking route |

| | |
|---|---|
| | through a street festival.<br>● **Lemonade Stand**<br>○ Students train a linear regression model that predicts lemonade sales based on temperature.<br>● **Book Sorter**<br>○ Students train a k-nearest neighbors classifier that sorts donated books by category.<br>● **Recycling Sorter**<br>○ Students train a perceptron that sorts containers into glass or plastic based on hand-sorted examples.<br>● |

## Module 9: Build a Classifier (2 weeks / 10 hours)

Students design, build, train, and evaluate an original machine learning classifier across a series of milestones, culminating in a documented and presented final project.

| | |
|---|---|
| Topics Covered | ● Project & dataset planning<br>● Building a labeled dataset<br>● Training a KNN model<br>● Evaluating on unseen data<br>● Comparing & tuning models<br>● Model documentation<br>● Showcase & presentation |
| Example Assignments | ● **Dataset Design and Planning**<br>○ Students commit to a design for their own labeled dataset before writing any code.<br>● **Build Your Dataset**<br>○ Students write their dataset to a CSV file, build a loader, and plan a training/testing split.<br>● **Training with KNN**<br>○ Students train a K-Nearest Neighbors model on their own dataset and classify new items.<br>● **Compare and Tune Your Model**<br>○ Students add a challenger model, tune both models, and assemble a finished program that reports results. |

## Module 10: Final Exam (1 hour)

Students complete a cumulative, multiple-choice exam that assesses their understanding of concepts from across the entire course, from Python fundamentals through object-oriented programming, data structures, algorithms, and AI/machine learning.