AP Computer Science Principles in Python Course Syllabus

Course Overview

AP Computer Science Principles introduces students to the breadth of the field of computer science. In this course, students will learn to design and evaluate solutions and to apply computer science to solve problems through the development of algorithms and programs. They will incorporate abstraction into programs and use data to discover new knowledge. Students will also explain how computing innovations and computing systems, including the Internet, work, explore their potential impacts, and contribute to a computing culture that is collaborative and ethical.

Prerequisites

This course is designed for students with no prior programming experience. However, a strong foundation in first-year high school algebra is recommended.

Required Resources

Online resources and labs provided by the instructor

- Online Text: JuiceMind.com
- Exam Registration / Project Submission: apclassroom.collegeboard.org

Course Goals

- Increase accessibility and participation in computer science.
- Develop confidence in computational problem-solving.
- Understand core computing principles and their societal impacts.
- Encourage the creation of computational artifacts as expressions of creativity.
- Foster collaborative problem-solving skills.
- Promote responsible and ethical use of technology.

Learning Environment

The course is structured as a blended learning experience, combining web-based resources with in-class activities. Students will engage in coding exercises, digital presentations, written reflections, and work collaboratively on projects. Lessons include interactive readings, quizzes, interactive exercises, and projects.

Programming Environment

Students will use a browser-based editor to write and execute Python programs, including creating graphical applications using the graphics library tkinter. This environment is designed to support the development of computational thinking skills from the beginning of the course.

The AP Create Performance Task

This end-of-course assessment is designed to gather evidence of student proficiency in each learning objective. The AP Create Performance Task (Create Task) is an in-class assessment, administered by the teacher, that allows students to exemplify their learning through an authentic, “real-world” creation. In the Create Task, students will design and implement a program to solve a problem, enable innovation, explore personal interest, or express creativity. Their development process should include exploration, investigation, reflection, design, implementation, and testing your program. Students will gain the experience necessary to complete the Create Task in class. This course involves several practice Create Tasks in which students will research topics in computing and create their own digital artifacts. Sufficient time is set aside in the course for students to prepare for and complete the Create Task.

The AP Exam

The AP Computer Science Principles end-of-course exam has consistent question types and weighting every year, so you and your students know what to expect on exam day.

Section I: End-of-Course Multiple-Choice Exam

70 multiple-choice questions | 120 minutes | 70% of score | 4 answer options

- 57 single-select multiple-choice
- 5 single-select with reading passage about a computing innovation
- 8 multiple-select multiple-choice: select 2 answers

Section II: Create Performance Task: Written Responses

Create Performance Task program code, video, and student-authored Personalized Project Reference | 9 hours in-class | 30% of score | 4 written response prompts | 60 minutes end-of-course exam

The second section of the AP Computer Science Principles Exam consists of a written response section where students demonstrate their understanding of their personal Create Performance Task by answering four written questions. Students will be given a copy of the Personalized Project Reference that they assembled during their submission of the Create Task to aid in their answering of these questions.

Assessment

Students are regularly assessed for knowledge and skills through completion of multiple-choice questions, programming exercises, and larger programming projects.

- **Formative Assessment:**
  - **Multiple-Choice Review:** All lessons are immediately followed by short multiple-choice review pages to assess essential knowledge. Students are given only one chance to answer each question.
  - **Exercises:** Lessons focused on programming skills are followed by one to three short programming exercises to assess mastery of essential skills. Students are given unlimited attempts to run and test their code against the exercise requirements, but only one chance to submit it for grading.
  - **AP Practice:** Lessons focused on programming skills end with short multiple-choice quizzes that present programming problems in the language and structure that they will expect on the AP Exam. Students are given only one chance to answer each question.
  - **Competitive Review Quizzes:** Lessons conclude with a live quiz in which students compete to answer review questions. These quizzes are scored for correctness and speed. After completing a quiz live, students may retake the quiz on their own to study.

- **Summative Assessment:**
  - **Unit Exams:** All units except those devoted to summative assessments contain a longer multiple choice exam that contains both regular and AP-style questions.
  - **Major Exams:** This course contains three major exams. Each is contained within its own unit.
    - **Midterm Exam (Unit 9):** After completing Unit 8, students will take a midterm exam. This exam is composed entirely of AP-style multiple choice questions.
    - **Practice AP Exam (Unit 17):** After completing the course but before sitting for the AP exam, students will take a practice exam to assess their readiness and guide further study.

        - **Final Exam (Unit 18):** After sitting for the AP exam, students will take a multiple-choice final exam. This exam covers all course content, but does not feature AP-style questions.

- **Projects:** This course contains several major projects to assess key programming and computer science skills. Each is contained within its own unit.
    - **Tell a Story (Unit 6):** This project assesses students' ability to make design decisions and translate them into code.
    - **Wordle (Unit 10):** This project assesses students' ability to write and reason about a large, complex program in preparation for the Create Task.
    - **Loaded Dice (Unit 11):** This project assesses students' ability to write and reason about a large, complex program in preparation for the Create Task.
    - **Design Thinking (Unit 15):** This multi-step project serves to reinforce students' ability to iteratively design a large, complex program in preparation for the Create Task.

Recommended Grading Scheme

Term 1 (Pre-Midterm Exam):

Category	**Weighting**	**Method**
Review and AP Practice	20%	Completion
Exercises	25%	Correctness (graded automatically)
Unit Exams	25% (5 total, 5% each)	Correctness (graded automatically)
Project - Create Your Own Story	10%	Quality of Submission (graded by teacher)
Midterm Exam	20%	Score (graded automatically)

Term 2 (Pre-Midterm Exam):

Category	Weighting	Method
Review, AP Practice, and Exercises	30%	Completion
Unit Exams	10% (2 total, 5% each)	Correctness (graded automatically)
Project - Wordle	5%	Quality of Submission (graded by teacher)

Project - Loaded Dice	5%	Quality of Submission (graded by teacher)
Project - Software Development Process	10%	Quality of Submission (graded by teacher)
Create Performance Task	10%	Completion (50% of grade), Quality of Submission (graded by teacher, 50% of grade)
Practice AP Exam	10%	Completion (50% of grade), Score (graded automatically, 50% of grade)
Final Exam	20%	Score (graded automatically)

Notes:

- Most formative assessment is graded for completion so as to not punish students who take more time to learn. However, exercises are graded for correctness, because students have unlimited chances to verify that their solution works before submitting it.
- Exercises in term 2 are graded for completion because they're very challenging and there aren't many of them. The programming knowledge required to successfully complete term 2 exercises is beyond what a student would need in order to perform well on the exam.
- The Create Task and Practice AP Exam are graded half for completion and half for quality to balance reassuring students that they can succeed and encouraging them to take it seriously.
- Although the Create Task is graded, it is important to ensure that a teacher does not provide useful feedback (such as a grade) before students submit to the College Board. Students' work *must* entirely be their own.

Pacing

Lesson plans are structured around a 45-minute class period, and lesson folders should take 45 minutes to complete, unless otherwise stated. The instructional content of a lesson should never take more than this long to teach, but exercises and review may spill over into homework based on student ability.

In total, JuiceMind's AP Computer Science Principles in Python is designed to take 135 days to complete using only material included within this course. This is shorter than a typical school year to allow for additional review and supplemental content, and to accommodate

the Create Performance Task (April 30) and AP Exam (typically around May 15), which occur before the end of the school year.

Course Objectives

This course follows the AP Computer Science Principles Framework, focusing on key computational thinking practices and big ideas:

Computational Thinking Practices:

- Practice 1: Design and evaluate computational solutions.
- Practice 2: Develop and implement algorithms.
- Practice 3: Incorporate abstractions in program development.
- Practice 4: Evaluate and test algorithms and programs.
- Practice 5: Investigate computing innovations.
- Practice 6: Contribute to an inclusive and ethical computing culture.

Big Ideas:

- Creative Development: Focus on iterative design and experimentation.
- Data: Explore the role of data in computing innovations.
- Algorithms and Programming: Learn to integrate algorithms and abstractions.
- Computing Systems and Networks: Study data transfer and internet operations.
- Impact of Computing: Examine ethical, privacy, and security issues.

Course Breakdown

Unit 1: Introduction to Computer Science (3 periods - 2 hrs 15 mins)

Students are introduced to computer science and the ways it affects our lives. They will also learn about the main objectives of this course and the AP Exam.

- **Topics Covered:**
  - Computer Science Connections:
    - Art
    - Music
    - Agriculture
    - Social Media
  - The AP Exam
  - The Create Performance Task

Unit 2: Solving Problems in Computer Science (5 periods - 3 hrs 45 mins)

Students are introduced to the basics of computational problem-solving, including algorithmic thinking and flowchart design.

- **Topics Covered:**
  - Problem-Solving Strategies
  - The Growth Mindset
  - Algorithms
- Flowcharts
- Pseudocode
- **Featured activity:**
  - 2.2.4: Designing Flowcharts: [Algorithms and Programming (AAP), Computational Thinking Practice 2]: Students develop and implement algorithms using flowcharts for getting ready for school and for preparing a simple pasta dish.

Unit 3: Programming with Python (10 periods - 7 hrs 30 mins)
This unit delves into the fundamentals of Python, focusing on variables, user input, basic data types, and expressions.

- **Topics Covered:**
  - Code
  - Input/Output
  - Printing
  - Variables
  - Data Types and Type Casting
  - Operators and Expressions
  - Libraries and Prewritten Functions
  - Basic Graphics Programming
- **Featured activity:**
  - 3.9.1: Mad libs [Algorithms and Programming (AAP), Creative Development (CRD), Computational Thinking Practice 1, Computational Thinking Practice 2]: Students have the opportunity to work in groups to design and implement the Mad-libs game in Python.

Unit 4: Control Structures (14 periods - 10 hrs 30 mins)

Students learn to implement control structures in Python, including conditional logic and loops.

- **Topics Covered:**
  - Boolean Logic
  - Conditional Statements
  - For Loops and While Loops
  - Advanced Control Structures - Nested Loops, Compound Conditionals, Break Statements
- **Featured activity:**
  - 4.10.1: Guess your classmate: [Algorithms and Programming (AAP), Creative Development (CRD), Computational Thinking Practice 1, Computational Thinking Practice 2]: Students have the opportunity to work in groups in a collaborative and inclusive development platform in order to design a game that prompts questions, to see if the player can guess who the correct classmate the game is referring to.

Unit 5: Functions (10 periods - 7 hrs 30 mins)

This unit focuses on the creation and use of functions in Python, including parameter passing, return values, and scope.

- **Topics Covered:**
  - Defining and Calling Functions
  - Parameters and Arguments
  - Return Values
  - Scope and Lifetime of Variables
  - Randomness and Libraries
  - Bugs and Exception Handling
  - Key and Mouse Events
- **Featured activity:**
  - 5.7.3: Buggy Processing: [Algorithms and Programming (AAP), Computational Thinking Practice 4]: Students have the opportunity to evaluate and test a calculator program in order to identify possible syntax and logic errors that can be introduced by the user.

Unit 6: Practice Project: Tell a Story (5 periods - 3 hrs 45 mins)

In this unit-long project, students will develop their own interactive stories using Python. They'll get the chance to express themselves creatively, assess their own work, and give feedback to their peers.

- **Topics Covered:**
  - Brainstorming
  - Pseudocode as a Planning Tool
  - Abstraction and Organization through Functions
  - Self-Assessment & Peer Feedback
  -

Unit 7: Data Structures (7 periods - 5 hrs 15 mins)

Students explore Python's data structures, such as lists, tuples, and strings, and learn how to manipulate these structures to solve problems.

- **Topics Covered:**
  - Lists
  - Tuples
  - Strings as Data Structures
  - List Operations, and List Methods
  - Iterating Through a List
  - For-Each Loops and Enumeration
  - Models and Simulations
  - Algorithmic efficiency
- **Featured activity:**
  - 7.7.1: Grade calculator: [Algorithms and Programming (AAP), Computational Thinking Practice 3]: Students have the opportunity to build a grade calculator program that calculates various statistics and assigns letter grades based on an input. Students will implement abstractions in order to develop this program.

Unit 8: Digital Information (14 periods - 10 hrs 30 mins)

Students will learn about binary numbers and explore how various forms of digital information are encoded in binary. They will learn how data is represented, processed & secured.

- **Topics Covered:**
  - Number Systems - Binary, Decimal, and Hexadecimal
  - Bits, Bytes, and S.I. Prefixes
  - Text Encodings - ASCII
  - Digital Images, Pixels, and RGB Color
  - Data Compression - Lossless vs. Lossy
  - Simple Cryptographic Algorithms - Caesar and Vigenère Ciphers

    - Modern Cryptography - Symmetric and Public-Key Encryption
- **Featured activity:**
    - 8.8.3: Grade calculator: [Algorithms and Programming (AAP), Data (DAT), Computational Thinking Practice 1]: Students will have the opportunity to write a program to efficiently guess the secret code to unlock a treasure chest.

Unit 9: Midterm Exam (2 periods - 90 mins)

This unit contains a multiple-choice exam designed to assess students' mastery of key programming and computer science skills. Questions in this exam are written using the style, language, and programming syntax of the AP Exam.

- **Units Assessed: 2-8**

Unit 10: Practice Project - Wordle (6 periods - 4 hrs 30 mins)

Students will delve into game development by creating their own version of the word-guessing game Wordle. This hands-on project reinforces fundamental programming concepts and computational thinking strategies, providing an opportunity to apply and review programming skills while practicing for the Create Task portion of the AP Exam. After completing the program code, students will answer written responses in a similar style to the Create Task.

- **Topics Covered:**
    - Program design, function, and purpose
    - Algorithm development
    - Errors and testing
    - Data and procedural abstraction
    - Reflecting and giving feedback

Unit 11: Practice Project - Loaded Dice (6 periods - 4 hrs 30 mins)

Students will explore randomness and simulation by programming a simple game that uses loaded dice. Through this project, they will apply Python programming skills to understand probability theory in a real-world context. This provides an opportunity to apply and review the coding skills while practicing for the Create Task portion of the AP Exam. After completing the program code, students will answer written responses in a similar style to the Create Task.

- **Topics Covered:**
    - Program design, function, and purpose
    - Algorithm development
    - Errors and testing
    - Data and procedural abstraction
    - Reflecting and giving feedback

Unit 12: The Internet (11 periods - 8 hrs 15 mins)

This unit focuses on the Internet and its global impact, exploring topics like the digital divide, global collaboration, and the ethical implications of technology.

- **Topics Covered:**
    - Networks, Packets, and Routing
    - The Internet and the World Wide Web
    - Addressing - IP, URLs, and the DNS
    - Protocols - TCP vs. UDP
    - Sequential, Parallel, and Distributed Computing
    - The Digital Divide
    - Ethical Issues in Technology Use and Copyright
- **Featured activity:**
    - 12.8.3: Sequential vs Parallel activity [Computing Systems and Networks (CSN), Computational Thinking Practice 1] Students have the opportunity to work in groups to search for a number in a data set using different strategies that allow them to reflect on the differences between sequential, parallel, and distributed computing.

Unit 13: Cybersecurity (5 periods - 3 hrs 45 mins)

Students will be introduced to fundamental aspects of cybersecurity, including common risks like phishing, keylogging, and malware. They will learn to evaluate online behaviors, analyze real-world threats, and develop basic cybersecurity strategies. This unit concludes with a short creative project in which students will make a PSA poster about cybersecurity risks.

- **Topics Covered:**
    - Cybersecurity and Passwords
    - Information Security - PII, Cookies, and Digital Footprints
    - Malware - Phishing, Keylogging, and Rogue Access Points

- **Featured activity:**
    - 13.4.5: Analyzing for information: [Computing Innovation 2, Prompt C, Impact of Computing (IOC), Data (DAT), Computational Thinking Practice 5]: Students have the opportunity to reflect on how exposing personal information and data online on various social media platforms can be used adversarially.

Unit 14: Data (5 periods - 3 hrs 45 mins)

This unit examines the role of data in modern society, including data collection, analysis, and visualization. Students will compare different methods of data visualization for accuracy and clarity and learn to be aware of potentially misleading visualizations.

- **Topics Covered:**
    - Uses of Data
    - Data Collection Methods
    - Data Analysis and Visualizations
- **Featured activity:**
    - 14.3.4: Data Collection - Self-driving cars: [Computing Innovation 2, Prompt B, Impact of Computing (IOC), Data (DAT), Computational Thinking Practice 5]: Students have the opportunity to reflect on what data self-driving cars collect and how this data is used towards improving both reliability and performance of self-driving.
    - 14.4.3: Investigating a Computing innovation: [Computing Innovation 3, Prompt A, Impact of Computing (IOC), Computational Thinking Practice 5]: Students have the opportunity to investigate a computing innovation involving either Mobile banking apps, virtual reality, GPS trackers, Smart home devices, or Cloud Computing and explain the impacts that it has on society, economy, and culture.

Unit 15: Create Performance Task (20 periods - 15 hrs)

This unit begins with a series of increasingly complex programming projects designed to prepare students to complete the Create Performance Task. Once they're ready, students will complete the Create Performance Task, dedicating 9 hours of class time to develop and finalize their project, which includes a program, a video, and a reference.

- **Topics Covered:**
    - Program design, function, and purpose
    - Algorithm development
    - Errors and testing

    - Data and procedural abstraction
    - Reflecting and giving feedback

Unit 16: AP Exam Review (9 periods - 7 hrs)

This unit serves as essential practice and preparation for the AP Computer Science Principles Exam. Students will complete a full practice exam, including both multiple choice and short answer questions.

- **Materials Included:**
    - Full multiple-choice practice exam
    - Written response prompts related to Create Performance Task code

Unit 17: Final Exam (3 periods - 2 hrs)

This unit contains a multiple-choice exam designed to assess students' understanding of this course's content. Unlike the AP Exam review questions, programming questions in this exam are written in Python.

- **Units Assessed: 2-14**