AP Computer Science Principles Syllabus 2020-2021	
Textbooks and Online Resources (CR1)	Schneider, Michael G., and Judith Gersting. *Invitation to Computer Science*. National Geographic/Cengage Learning Code.org online curriculum Beauty and Joy of Computing online curriculum CMU online python sandbox Codingbat.com/python College Board AP Computer Science Principles Course and Exam description College Board AP Classroom
Opportunities to develop student understanding of the required content outlined in each of the Big Ideas. (CR2)	**Unit 1 - Digital Information (Code.org)** ***2 weeks*** Students explore how computers store complex information like numbers, text, images, and sound, and they debate the impacts of digitizing information (DAT). Alternating between lessons away from the computer ("unplugged"), and lessons that use digital tools called "widgets," this unit encourages an exploratory and collaborative approach to learning about digital information. To close out the unit, students debate the pros and cons of digitizing information and the impacts of digital information on society and culture at large **(CTP5) (IOC).** **Unit 2 - The Internet (Code.org)** ***2 weeks*** Students learn how the Internet works and discuss its impacts on politics, culture, and the economy **(CSN).** Throughout this unit, students use a digital tool called the Internet Simulator that simulates how different parts of the Internet work and forces students to grapple with and solve the problems each aspect of the Internet was designed to solve **(CTP1).** At the conclusion of the unit, students investigate an "Internet Dilemma," both from the standpoint of its technical background and its impacts on different groups of people **(CTP5) (IOC).** **Unit 3: Programming Basics in Snap!** ***6 weeks*** Unit 3 is a gentle introduction to programming using the block-based programming language, Snap! featured in the *Beauty and Joy of Computing* APCS Principles course. This introductory programming unit has a strong focus on planning, design and debugging as essential tools as students learn the syntax and mechanics of programming in Snap! **(CRD).** Students are being exposed to programming concepts in a very non-threatening way. More challenging concepts are "spiraled" and touched on many times before they are fully explained and incorporated into the student's programming "toolbox" at the appropriate point. Students are capable of exploring and discovering how to use the programming language as a problem-solving tool. **(AAP) (CTP2) (CTP3)** **Unit 4: List and Loops in Snap!** ***6 weeks***

Unit 4 is an introduction to the higher level programming concepts of data abstraction and iteration using the block-based programming language, Snap! featured in the *Beauty and Joy of Computing* APCS Principles course. (AAP) (CRD) This programming unit has a strong focus on planning, design and debugging as essential tools as students learn the syntax and mechanics of programming in Snap! There is a strong emphasis on creating projects, from design to implementation, designed to build computational thinking skills in students. There is also an emphasis on assessing work. Students need to be checked regularly at this point to make sure they are understanding and not just "doing". (CTP2) (CTP3)

Unit 5: Programming Basics in Python *3 weeks*
Students are introduced to text-based programming with Python 3 in Unit 5. (AAP) Students continue their learning of problem solving and computational thinking, using the Python programming language as a tool in the CMU Sandbox development environment. Creativity is key in this unit as students work with the graphics package inherent to CMU Sandbox to write algorithms and code to create 2-D works of art. (CTP2) Students will learn about variables, conditionals, iteration and functions as they tackle increasingly more difficult problems in Python.

Unit 6: Lists and Loops in Python *4 weeks*
Students continue their study of programming in Python as they study lists and iteration over lists. (AAP) (CTP3) Students solve increasingly challenging problems as they analyze, design and implement algorithms in the Python programming language. (CTP4)

Unit 7: Python Adventure Game Project *1 week*
Students work collaboratively to create a text-based adventure games as a Mock Create PT. (AAP) (CRD) (CTP5) Students are given starter code, which they will expand and personalize as they create their games. (CTP2) They will then practice writing responses in the style of the Create PT to get them ready to begin their performance task for submission to the College Board.

Unit 8 - Create PT Prep (Code.org) *1 week*
Students practice and complete the Create Performance Task (PT). The unit begins with a series of activities that ensure students understand the requirements of the Create PT, which they have practiced throughout the year.

Create Performance Task Completion *4 weeks*
Students are given 12 - 16 hours, in class, to complete their Create PT.

Unit 9 - Data (Code.org) *2 weeks*
Students explore and visualize datasets from a wide variety of topics as they hunt for patterns and try to learn more about the world around them (DAT)**.** Students work with datasets in App Lab. They are asked to make use

	of a data visualizer tool that assists students in finding data patterns. Students learn how different types of visualizations can be used to better understand the patterns contained in data sets and investigate hypotheses. At the conclusion of the unit, students learn about the impacts of data analysis on the world around them, before completing a final project in which they must uncover and present a data investigation they've completed independently **(CTP5).** **Unit 10 - Cybersecurity and Global Impacts (Code.org)** ***2 weeks*** Students research and debate current events at the intersection of data, public policy, law, ethics, and societal impact **(IOC).** This unit is built around a simulated "future school" conference in which students must take on the persona of a stakeholder in a school setting and propose and debate technological innovations that could improve schools. Throughout the unit students learn about the privacy and security risks of many computing innovations, and learn about the ways some of these risks can be mitigated. **Students complete their Explore Curricular Requirement as part of this project as they investigate at least three computing innovations, then discuss and debate many other innovations with their classmates (CTP5, CTP6).** At the conclusion of the unit, the class holds a conference in which teams present their overall vision for a school of the future and the computing innovations that would power it. **Unit R: Test Review** ***1 week*** Students review the content from the entire year, using *Invitation to Computer Science*. previous assessments and the College Board's AP Classroom's Question Bank. Students will take a Mock AP Exam to practice answering the multiple choice questions in the time given.
Opportunities to develop student understanding of the big ideas. (CR3)	**1. Creative Development (CRD)** Throughout the course students are required to work collaboratively to design and develop programs. One example of this is the Bubble Simulation program in Unit 3: *In this 4-day lesson students learn about simulations of real-world phenomena and create a simulation of bubbles rising in liquid. They are guided through this process using the iterative and incremental development process. Students are then given practice writing about their programs and finally celebrating their work by sharing their unique simulations with the class.* **2. Data (DAT)** Students study data in increasing levels of abstraction, from bits to files to data visualizations. The implications of storing large quantities of data are explored and compression is considered as a more efficient way to store and transfer data. One example of this is the Compression lesson in Unit 2: *In this lesson, students use Code.org's Text Compression Widget to experiment with compressing songs and poems and try to find their 'personal best' compression. Students are also introduced to lossy*

	compression via the Lossy Text Compression widget. The lesson ends with a discussion of the situations where lossless compression is important and the situations where lossy compression is appropriate. **3. Algorithms and Programming (AAP)** Algorithms and Programming is the single most important concept in this course, as evidenced by the weighting on the AP exam and the attention given in the course. One example of this is the UPC Checker program from Unit 4: *Students will complete a large multi-day project, from design to testing. They will research UPC codes and learn about their significance. Students will then write a UPC checker program that validates a UPC code that they have stored in a list, with each digit of the UPC code a separate element of the list. This allows students to perform their calculations on the individual digits of the UPC code with relative ease.* **4. Computing Systems and Networks (CSN)** Unit 2 is devoted to the study of the Internet. Students use Code.org's Internet Simulator to gain an understanding of how the Internet works. One example of this is the Routing and Packets lesson in Unit 2: *In this lesson, students work in an updated Internet Simulator that lets students send messages with a dedicated To and From IP Address. Students start by connecting to a dedicated router and sending messages only to each other. Students continue to send messages and view the logs to notice that the messages are also taking different paths to reach the same destination. Students also learn that large messages sent over the Internet are actually divided into individual packets and explore the challenges this creates. At the end of the lesson students watch a video and learn about the User Datagram Protocol (UDP) and The Transmission Control Protocol (TCP), two different protocols for sending messages broken into packets.* **5. Impact of Computing (IOC)** Unit 10 is devoted to the study of Global Impact of Computing and Cybesecurity. Students engage in various activities as they study the impact that computing has had on society at large and how they can protect their privacy while engaged in computing. One example is the lesson on Security Risks in Unit 10, sourced from Code.org's curriculum: *Students investigate three different common security risks (phishing, keylogging, rogue access points) in a jigsaw activity. In groups, students create Public Service Announcement slides warning of the dangers of their assigned security risk. Then students are grouped with students who investigated other security risks and are instructed to share their slide and give a voice over. The activity ends with the class coming together to discuss the security risks as a whole.*
Opportunities for students to develop the skills related to	**Computational Thinking Practice:** **1- Computational Solution Design (CR4)**

Computational Thinking Practices: 1- **Computational Solution Design (CR4)** 2- **Algorithm and Program Development (CR5)** 3- **Abstraction in Program Development (CR6)** 4- **Code Analysis (CR7)** 5- **Computing Innovations (CR8)** 6- **Responsible Computing (CR9)**	*In Unit 4, during the Weather Reporting System lesson, student pairs will design and create a weekly weather reporting system that stores, analyzes and manipulates the values stored in a list. Each pair designs and codes a program that will:* • *Collect and store one week's worth of high temperatures that are entered by the user at runtime* • *Report the highest temperature of the week* • *Report the average temperature of the week* • *Report the number of days with a temperature above 75 degrees* • *Additional functionality at the discretion of the student pair* **2- Algorithm and Program Development (CR5)** *In Unit 5, students are developing algorithms and implementing them in Python. One example is the Coding a Flowchart lesson:* • *Students will take a flowchart that has been given to them showing the completion of an everyday task that involves making decisions, such as getting dressed in the morning, and write the code to represent the flowchart with print statements indicating the flow of the program.* • *Next, students will then create their own flowchart describing the completion of another everyday task that involves making decisions, and then write the code to represent the flowchart that they created with print statements.* **3- Abstraction in Program Development (CR6)** *In Unit 3's Changing List Contents lesson, students will work with lists in Snap! by learning to add, insert, replace and remove items from a list. After identifying commands used in both pseudocode and Snap, students will pseudocode and code a treasure list based on a story about an ever-changing list of treasures found in a grandmother's attic, as described in the story about Attic Treasure.* **4- Code Analysis (CR7)** *In Unit 3, during the Debugging and Problem Solving lesson, for several problems involving drawing regular polygons, students are presented with a description of what a code segment is intended to accomplish. They are also given code that does not quite accomplish the desired outcome. Students work in pairs to analyze the code, determine the error and update the code so that it performs the drawing task as described.* **5- Computing Innovations (CR8)** *In Unit 10's Innovation Simulation Project, students take on the roles of different stakeholders in school communities converging at a convention where they eventually will deliver a proposal on the best computing innovation for a Future School. Students explore what a computing innovation is, research three different computing innovations including the*

	purpose of the innovation, it's use of data, and potential benefits and harms of each innovation. Then each group presents their project to the class. **6- Responsible Computing (CR9)** *In Unit 4's Getting Creative with Libraries lesson, students work in pairs to make an entertaining creative story-telling project that incorporates the external libraries available in Snap! such as text-to-speech and world maps. After collaborating on their creative project for the better part of 2 days, students will celebrate their creativity as each pair shares their project with the class.*
Provides a minimum of three opportunities for students to investigate different computing innovations. (CR10)	Students complete the Explore Curricular Requirements in Unit 10: Cybersecurity and Global Impacts, during the Innovation Simulation project where they act as delegates proposing innovations to improve a "future school." **(sourced from Code.org's APCSP curriculum)** • Students research and discuss three different computing innovations and consider the data that is being collected along with the function and purpose of the innovations (**CI 1, Prompt B**). • Students choose one computing innovation to develop into a one-page proposal. Included in this proposal are the purpose and benefits of the computing innovation, its function, the data collected, and potential concerns along with how those concerns could be addressed (**CI 2, Prompts A, B, and C**). • Students work with their groups to create a presentation about their computing innovations with the goal of convincing others that their innovations together are the best to address the concerns of the "future school." Students discuss their innovations with their group to create a unifying theme for their presentation. Before the presentations are delivered, groups meet together to learn about each other's innovations and give feedback on presentations. At this point in the project, students will have explored and discussed all the innovations in their own group along with the top innovations from another group (**CI 3, Prompts A, B, and C**).
Students are provided at least 12 hours of dedicated class time to complete the AP Create Performance Task. (CR11)	**Create Performance Task Completion** ***4 weeks (after Unit 8)*** Students are given 12 - 16 hours, in class, to complete their Create PT.