AP Computer Science Principles Scope & Sequence

Days	Unit	Standard(s)/Outcome(s)	Essential/Guiding Questions
12	1 – Introduction to Programming	1. Students develop an interactive game they can install on their phones, generate conversation between animated characters, create abstract art, and explore storytelling animation through sprite interaction. 2. Learn to use pair programming and to create program documentation. 3. Investigate legal and ethical issues that arise in computing, especially with regard to data collection and privacy.	How can events be used in programming? In what ways can multiple programmers improve a program? How can society balance informational needs and privacy protections?
11	2 - Abstraction	1. Students implement an algorithm for a guessing game using local and global variables. 2. Use abstract data types and list traversal to build a quiz app. 3. Create predicates to filter lists in order to solve a crossword puzzle. 4. Use the modulus function and	How can variables be used to improve the function of a program? Why are abstract data types necessary in programming? In what ways are mathematical functions critical to

		a higher order function to code mathematical functions 5. Investigate the history, purpose, laws,evolution and enforcement of copyright.	programming? What impact has the computer had on copyright laws?
7	3 – Data Structures	1. Students explain complexity in a variety of contexts (maze navigation, fractal art, tic-tac-toe). 2. Use nested abstract data types and data I/O to develop a contact list app. 3. Evaluate the beneficial and harmful impacts of robots and AI.	How can recursion be used to simplify and improve programming? How does an abstract data type in conjunction with constructors manage complexity in programming? Do the benefits of advanced technologies outweigh their potential dangers?
5	Practice AP Create task	1. Students create a project of their own choosing as practice for the AP Create Task. 2. Select and use a development process 3. Plan and code their own program. 4. Test their program for errors. 5. Write about the development process.	How can you apply and showcase your knowledge of programming?

		6. Acknowledge any code developed by other people.	
10	4 – How the Internet works	1. Students learn about how the internet works. 2. The benefits and vulnerabilities of fault-tolerant systems. 3. Cybersecurity practices such as public key encryption and individual level practices and software to keep safe. 4. Digital data representation including binary representation. 5. Compression algorithms. 6. Consider the impact of the internet on human communication and the workplace.	How do networks operate? What elements make up Cybersecurity? How is information stored and used in a computer system? How does collaboration through the internet affect communities? What are the social implications of online interactions?
10	5 – Algorithms and Simulations	1. Students learn about program efficiency through exploration of the binary and linear search algorithms. 2. Learn about sequential, parallel, and distributed computing and determine the contexts in which each is most useful. 3. Consider the contexts in which simulation is useful and implement a simple solution.	How do computers organize and search through information? How do programs do multiple things at the same time? How do computers do problem solving? How do computers process data?

		4. Use Snap! Data tools to generate knowledge from data.	
9	AP Create Task	1. Students complete the AP Create task, 12 hours in class.	How can you apply and showcase your knowledge of programming?
4	6 - How Computers Work	1. Building on their understanding of abstraction and the way computers store data, students learn about the computer system abstraction hierarchy, with application software on top and transistors at the bottom.	
3	7 - Fractals and Recursion	1. Students expand their experience with recursion and functional programming through drawing projects that use recursive commands, mainly fractals	
5	8 - Recursive Functions	1. Students extend their understanding of abstraction and recursion through	

		exploration of recursive functions: sorting lists; both selection sort and partition sort; Pascal’s triangle; converting numbers to and from binary; finding the subset of a set; and building several higher order functions from scratch.	