CompuScholar, Inc.

Alignment to the Kentucky Computer Science Standards

9th - 12th Grade

Kentucky Standards Information:

CS Page	Kentucky Computer Science Education Page
Standards Link:	KY Academic Standards

CompuScholar Courses in this Grade Band:

Course Title:	**Digital Savvy**, ISBN 978-0-9887070-8-5 Course Description and Syllabus
Course Title:	**Web Design**, ISBN 978-0-9887070-3-0 Course Description and Syllabus
Course Title:	**Python Programming**, ISBN 978-1-946113-00-9 Course Description and Syllabus
Course Title:	**Java Programming (Abridged)**, ISBN 978-0-9887070-4-7 Course Description and Syllabus
Course Title:	**Java Programming (AP)**, ISBN 978-0-9887070-2-3 Course Description and Syllabus
Course Title:	**Windows Programming with C#**, ISBN 978-0-9887070-0-9 Course Description and Syllabus
Course Title:	**Unity Game Programming**, ISBN 978-0-9887070-7-8 Course Description and Syllabus

Kentucky's Computer Science standards are broken into "Core" and "Challenge" categories. The "Core" elements represent foundational skills expected to be taught in any introductory Computer Science course, while the "Challenge" skills (marked with *) may be deferred to more advanced courses.

Any CompuScholar "Programming" course can be used to meet either "Core" or "Challenge" requirements. Material from Digital Savvy and Web Design may be included as desired for relevant topics.

Kentucky Computer Science Standards (High School)

Network & The Internet	COMPUSCHOLAR ALIGNMENT
H-NI-01: Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, end devices, topology, and addressing.	Our courses contain relevant descriptions of major Internet components, including clients, servers, protocols, ISPs and addressing.
H-NI-02: Give examples to illustrate how sensitive data can be affected by viruses, malware and other attacks.	Our courses contain chapters or lessons on relevant security topics such as encryption (including SSL/TLS) and establishing defenses against mal-ware and viruses.

H-NI-03: Recommend security measures to address various scenarios based on factors such as usability, efficiency, feasibility, and ethical impacts.	Our courses contain chapters or lessons on security topics, including physical vs. electronic security mechanisms where relevant.
H-NI-04*: Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology). *	Network design considerations, including scalability, bandwidth, security and points of failure are addressed where relevant.
H-NI-05*: Compare ways software developers protect devices and information from unauthorized access. *	

Data Analysis	**COMPUSCHOLAR ALIGNMENT**
H-DA-01*: Evaluate the tradeoffs in how data elements are organized and where data is stored.*	Our courses cover representation of data in multiple formats, from simple encoding of data types up through data structures and object definitions with properties. Choices of data type, data structure, or object definitions have pros and cons and may impact the desirable characteristics of the program or the maintainability of the underlying code.
H-DA-02: Collect data using appropriate data collection tools and techniques to support a claim or to communicate information.	Our team projects and other labs give students opportunities to research topics, obtain data sets, and produce digital artifacts or apps to visualize and explain the aggregated information.
H-DA-03*: Understand and design database structures to optimize search and retrieval.*	Our Digital Savvy course describes a basic relational database model and teaches students to use SQL for search and retrieval.
H-DA-04: Explain the privacy concerns related to the collection and generation of data.	Our courses contain relevant guidance on protecting personal information online and the persistence of online footprints.
H-DA_05: Use data analysis tools (e.g. formulas and other software data / statistical tools) to process and transform the data to make it more useful and reliable.	Our team projects and other labs give students opportunities to visualize and explain data by spreadsheet, charts & graphs or programmatic display, where relevant.
H-DA-06: Use data analysis tools and techniques to identify patterns and analyze data represented in complex systems.	Our team projects and other labs give students opportunities to visualize and explain data by spreadsheet, charts & graphs or programmatic display, where relevant.
H-DA-07: Create computational models that represent the relationships among different elements of data.	Student-driven projects give opportunities for analysis and representation of real-world data.
H-DA-08: Create interactive data visualizations using software tools to help others better understand real-world phenomena.	Our courses contain opportunities to explore and represent real-world data in the form of equations, charts and graphs and similar tools.

H-DA-09*: Evaluate the ability of models and simulations to test and support the refinement of hypotheses.*	Our courses contain opportunities for student to build models of real-world phenomena and simulate or predict results. Students are encouraged to alter or modify input data to observe the impact on resulting output and verify initial assumptions.

Algorithms & Programming	**COMPUSCHOLAR ALIGNMENT**
Algorithms	
H-AP-01: Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.	Our courses explain and use relevant, language-specific libraries such as the Java class library, the .NET framework or Python modules, with licensing considerations discussed as needed.
H-AP-02: Use a development process in creating a computational artifact that leads to a minimum viable product followed by reflection, analysis, and iteration.	Our courses contain team projects that allow students to define, design, build and test a unique project using standard SDLC stages.
H-AP-03: Use functions, data structures or objects to simplify solutions, generalizing computational problems instead of repeated use of simple variables.	Our programming courses cover simple data structures such as arrays and lists. Students will use each data structure in hands-on projects.
H-AP-04: Design and iteratively develop event-driven computational artifacts for practical intent, personal expression, or to address a societal issue.	Our courses contain opportunities for students to participate in simple SDLC stages to iteratively design and develop digital or computational artifacts of interest.
H-AP-05: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	Our courses cover Object-Oriented Programming (OOP), modular programming with functions, and functional decomposition of complex tasks down to manageable logical blocks.
H-AP-06: Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance and explain the benefits and drawbacks of choices made.	Our courses cover traditional flow control structures (conditionals, loops, functions) and the trade-offs in design, including selecting between appropriate flow control logic.
H-AP-07: Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.	Our courses describe how to use flowcharts to design algorithms to solve specific problems. Common sorting and searching algorithms or game-specific AI
H-AP-08: Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.	Our courses include opportunities for students to build their own functions/methods and objects.
H-AP-09: Evaluate and refine computational artifacts to make them more usable and accessible using systematic testing and debugging.	Our courses contain team projects that include a testing phase using a written test plan. Students will receive feedback from peers and incorporate that feedback into the final project.

H-AP-10: Systematically design and develop programs for broad audiences by incorporating feedback from users.	Our courses contain team projects that allow students to define, design, build and test a unique project using standard SDLC stages and traditional requirements, design and test documentation. Peer feedback is used to iteratively improve the results.
H-AP-11*: Design and develop computational artifacts working in team roles using collaborative tools.*	Our courses contain team projects in which students will experience standard software engineering roles and development life-cycles. Students will write project plans, requirements and design documents, and follow systematic test plans.
H-AP-12*: Describe how artificial intelligence drives many software and physical systems.*	Our courses contain context-appropriate descriptions of AI algorithms such as game AI and applications to real-world problems (e.g. self-driving cars).
H-AP-13: Use and adapt classic algorithms to solve computational problems.*	Our courses contain appropriate analysis and comparison of algorithms, including multiple sorting and searching approaches.
H-AP-14*: Evaluate algorithms in terms of their efficiency, correctness, and clarity.*	Our courses contain appropriate analysis and comparison of algorithms, including trade-offs in performance, coding complexity, and accuracy of
H-AP-15*: Compare and contrast fundamental data structures and their uses.*	Our programming courses cover lists, stacks and queues and the trade-offs involved with each representation. Students will use each data structure in hands-on projects.
H-AP-16*: Illustrate the flow of execution of a recursive algorithm.*	Our coding courses describe and demonstrate recursive algorithms and execution flow, where relevant.
H-AP-17*: Construct solutions to problems using student-created components, such as procedures, modules and/or objects.*	Our courses contain multiple opportunities for students to create their own functions and objects.
H-AP-18*: Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.*	Our courses compare and contrast standard algorithms for large scale sorting and searching. They additionally cover relevant AI algorithms to implement specific tasks or features.
H-AP-19*: Select and employ an appropriate component or library to facilitate programming solutions.*	Our coding courses use a variety of libraries and APIs appropriate for the language (e.g. Java Class Library, .NET Framework, Unity SDK).
H-AP-20*: Develop programs for multiple computing platforms.*	Our courses teach multiple languages, and those with easy cross-platform support (e.g. Java or HTML or Unity) are clearly defined as such, so student projects can be run on computers with different operating systems.
H-AP-21*: Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.*	Students will use IDEs and relevant collaborative tools and practices to develop numerous projects throughout each course. Team projects are included for group work.

H-AP-22*: Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., introducing errors).*	Our courses contain multiple projects where starting code is provided and students will expand or complete the initial project. Regression testing is described in chapters of debugging and testing.
H-AP-23*: Evaluate key qualities (including correctness, usability, readability, and efficiency) of a program.*	Each coding course contains a debugging chapter that includes code review as a methodical process for evaluating a program.
H-AP-24*: Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.*	Each coding course discusses common programming languages as well as specialty languages intended for specific problems.

Impacts of Computing	**COMPUSCHOLAR ALIGNMENT**
Culture	
H-IC-01: Reduce bias and equity deficits through the design of accessible computational artifacts.	Students are presented with examples and projects that are free from bias and inequity, and are similarly encouraged to follow ethical practices in their own development.
H-IC-02: Evaluate and assess how computing impacts personal, ethical, social, economic, and cultural practices.	Our courses contain relevant lessons on the global impact of computing, ethical computing concepts, intellectual property and licensing, careers in computing, etc.
H-IC-03: Research how computational innovations that have revolutionized aspects of our culture might have evolved from a need to solve a problem.	Each course contains lessons on new or evolving aspects of relevant technology as well as impacts of technology on society.
H-IC-04: Explain the beneficial and harmful effects that laws governing data (intellectual property, privacy etc.) can have on innovation.	Our courses cover intellectual property laws, copyright considerations and various types of software licensing.
H-IC-05: Evaluate and design computational artifacts to maximize their benefit to society.*	Our courses contain relevant lessons on the global impact of computing. Students learn about their personal digital footprint and understand that digital identities and online actions have long-term or permanent consequences.
H-IC-06: Evaluate the impact of the digital divide (i.e. inequity of computing access, education and influence) on the development of local communities and society.	Our courses contain lessons on the global impact of computing and digital accessibility.
H-IC-07*: Demonstrate ways computational design (i.e. algorithms, abstractions and analysis) can apply to problems across disciplines.*	Our courses cover a variety of algorithms that can be applied to real-world situations in mathematics, physics, economics, etc.
H-IC-08: Debate laws and regulations that impact the development and use of software and the protection of privacy.	Our courses cover intellectual property laws, copyright considerations and various types of software licensing.

Computing Systems	COMPUSCHOLAR ALIGNMENT
H-CS-01: Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.	Our courses contain lessons that describe how various hardware components (e.g. CPU, RAM, disk drives) encapsulate computing tasks such as processing, short-term storage and long-term
H-CS-02: Compare levels of abstraction and interactions between application software, system software and hardware layers.	Our courses describe the relationships between hardware, operating systems, device drivers, and a variety of end-user applications.
H-CS-03: Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.	Our courses contain dedicated troubleshooting and debugging information for relevant technology. The programming courses describe how to use a variety of debugging approaches, including code analysis, tracing (logging) and setting breakpoints in a debugger. Best practices and common troubleshooting tips are provided as needed.
H-CS-04: Categorize the roles of operating system software.	Our courses discuss relevant operating system features and tasks including basic file and application management.
H-CS-05*: Illustrate ways computing systems implement logic, input, and output through hardware components.*	Our courses contain appropriate discussions of the major hardware components of a computing system.