Ohio CTE

Programming (2019)

IT FundamentalsLearners apply fundamental principles of IT, including the history of IT and its impact on society, common industry terms, systems theory, information storage and retrieval, database management, and computer hardware, software, and peripheral device configuration and installation. This base of knowledge and skills may be applied across the career field. 2

2.3 Data EncodingExplain and describe data encoding basics. 2.3

- 2.3.1 Identify and explain coding information and representation of characters (e.g., American Standard Code for Information Interchange [ASCII], Extended Binary Coded Decimal Interchange Code [EBCDIC], Unicode). 2.3.1
- 2.3.2 Convert between numbering systems (e.g., binary, hexadecimal, decimal). 2.3.2

2.9 Project Concept ProposalDevelop a project concept proposal. 2.9

- 2.9.1 Identify and incorporate branding strategies. 2.9.1
- 2.9.2 Determine the scope and purpose of the project. 2.9.2
- 2.9.3 Determine the target audience, client needs, expected outcomes, objectives, and budget. 2.9.3
- 2.9.4 Develop a conceptual model and design brief for the project. 2.9.4
- 2.9.5 Develop a timeline, a communication plan, a task breakdown, costs (e.g., equipment, labor), deliverables, and responsibilities for completion. 2.9.5
- 2.9.6 Develop and present a comprehensive proposal to stakeholders. 2.9.6

2.11 TroubleshootingSelect and apply troubleshooting methodologies for problem solving. 2.11

- 2.11.1 Identify the problem. 2.11.1
- 2.11.2 Select troubleshooting methodology (e.g., top down, bottom up, follow the path, spot the differences). 2.11.2
- 2.11.3 Investigate symptoms based on the selected methodology. 2.11.3
- 2.11.4 Gather and analyze data about the problem. 2.11.4
- 2.11.5 Design a solution. 2.11.5
- 2.11.6 Test a solution. 2.11.6
- 2.11.7 Implement a solution. 2.11.7
- 2.11.8 Document the problem and the verified solution. 2.11.8

2.12 Performance Tests and Acceptance PlansDevelop performance tests and acceptance plans. 2.12

- 2.12.1 Create a written procedure agreed by the stakeholders and project team for determining the acceptability of the project deliverables. 2.12.1

Skip to main content

2.12.2 Develop a test system that accurately mimics external interfaces. 2.12.2

2.12.3 Develop test cases that are realistic, compare with expected performance, and include targeted platforms and device types. 2.12.3

2.12.4 Develop, perform, and document usability and testing integration. 2.12.4

2.12.5 Make corrections indicated by test results. 2.12.5

2.12.6 Seek stakeholder acceptance upon successful completion of the test plan. 2.12.6

2.13 Rollout and HandoffPlan rollout and facilitate handoff to customer. 2.13

2.13.1 Include overall project goals and timelines in the rollout plan. 2.13.1

2.13.2 Communicate rollout plans to key stakeholders in a timely manner. 2.13.2

2.13.3 Conduct final review and approvals according to company standards. 2.13.3

2.13.4 Identify support staff, training needs, and contingency plans in the rollout plan. 2.13.4

2.13.5 Test delivered application to assure that it is fully functional for the customer or user and meets all requirements. 2.13.5

2.13.6 Deliver support and training materials. 2.13.6

Programming and Software SystemsLearners apply principles of computer programming and software development to develop code; build, test, and debug programs; create finished products; and plan, analyze, design, develop, implement, and support software applications. 5

5.1 Programming ConceptsDescribe programming concepts. 5.1

5.1.1 Describe how computer programs and scripts can be used to solve problems (e.g., desktop, mobile, enterprise). 5.1.1

5.1.2 Explain how algorithms and data structures are used in information processing. 5.1.2

5.1.3 Model the solution using both graphic tools (e.g., flowcharts) and pseudocode techniques. 5.1.3

5.1.4 Describe, compare, and contrast the basics of procedural, structured, object-oriented (OO), and event driven programming. 5.1.4

5.1.5 Describe the concepts of data management through programming languages. 5.1.5

5.1.6 Analyze the strengths and weaknesses of different languages for solving a specific problem. 5.1.6

5.1.7 Compare the functions and operations of compilers and interpreters. 5.1.7

5.1.8 Describe version control and the relevance of documentation. 5.1.8

5.2 Computational and String OperationsDevelop code that performs computational and string operations. 5.2

5.2.1 Compare primitive types of numeric and nonnumeric data (e.g., integers, floats, Boolean, strings). 5.2.1

Skip to main content

5.2.2 Identify the scope of data (e.g., global versus local, variables, constants, arrays). 5.2.2

5.2.3 Write code that uses arithmetic operations. 5.2.3

5.2.4 Write code that uses subtotals and final totals. 5.2.4

5.2.5 Write code that applies string operations (e.g., concatenation, pattern matching, substring). 5.2.5

5.3 Logical Operations and Control StructuresDevelop code that uses logical operations and control structures. 5.3

5.3.1 Explain Boolean logic. 5.3.1

5.3.2 Solve a truth table. 5.3.2

5.3.3 Write code that uses logical operators (e.g., and, or, not). 5.3.3

5.3.4 Write code that uses relational operators and compound conditions. 5.3.4

5.3.5 Write code that uses conditional control structures (e.g., if, if-then-else). 5.3.5

5.3.6 Write code that uses repetition control structures (e.g., while, for). 5.3.6

5.3.7 Write code that uses selection control structures (e.g., case, switch). 5.3.7

5.3.8 Write code that uses nested structures and recursion. 5.3.8

5.3.9 Write code that creates and calls functions. 5.3.9

5.3.10 Code error handling techniques. 5.3.10

5.3.11 Write code to access data repositories. 5.3.11

5.3.12 Write code to create classes, objects, and methods. 5.3.12

5.4 Integrated Development EnvironmentBuild and test a program using an integrated development environment (IDE). 5.4

5.4.1 Configure options, preferences, and tools. 5.4.1

5.4.2 Write and edit code in the integrated development environment (IDE). 5.4.2

5.4.3 Compile or interpret a working program. 5.4.3

5.4.4 Define test cases. 5.4.4

5.4.5 Test the program using defined test cases. 5.4.5

5.4.6 Correct syntax and runtime errors. 5.4.6

5.4.7 Debug logic errors. 5.4.7

5.5 Programming ConventionsDevelop programs using applications security practices. 5.5

5.5.1 Develop programs using data validation techniques. 5.5.1

5.5.2 Develop programs that use reuse libraries. 5.5.2

5.5.3 Develop programs using operating system calls. 5.5.3

Skip to main content

5.5.4 Develop programs that call other programs. 5.5.4

5.5.5 Use appropriate naming conventions and apply comments. 5.5.5

5.5.6 Format output (e.g., desktop, mobile, enterprise, reports, data files). 5.5.6

5.6 Software Development LifecycleApply the software development lifecycle (SDLC). 5.6

5.6.1 Determine requirements specification documentation. 5.6.1

5.6.2 Identify constraints and system processing requirements. 5.6.2

5.6.3 Develop and adhere to timelines. 5.6.3

5.6.4 Identify a programming language, framework, and an integrated development environment (IDE). 5.6.4

5.6.5 Identify input and output (I/O) requirements. 5.6.5

5.6.6 Design system inputs, outputs, and processes. 5.6.6

5.6.7 Document a design using the appropriate tools (e.g., program flowchart, dataflow diagrams, Unified Modeling Language [UML]). 5.6.7

5.6.8 Create documentation (e.g., implementation plan, contingency plan, data dictionary, user help). 5.6.8

5.6.9 Review the design (e.g., peer walkthrough). 5.6.9

5.6.10 Present the system design to stakeholders. 5.6.10

5.6.11 Develop the application. 5.6.11

5.6.12 Compare software methodologies (e.g., agile, waterfall). 5.6.12

5.6.13 Perform code reviews (e.g., peer walkthrough, static analysis). 5.6.13

5.6.14 Ensure code quality by testing and debugging the application (e.g., system testing, user acceptance testing). 5.6.14

5.6.15 Train stakeholders. 5.6.15

5.6.16 Deploy the application. 5.6.16

5.6.17 Collect application feedback and maintain the application. 5.6.17

5.7 Configuration/Change ManagementDescribe configuration management activities. 5.7

5.7.1 Explain version management and interface control. 5.7.1

5.7.2 Explain baseline and software lifecycle phases. 5.7.2

5.7.3 Analyze the impact of changes. 5.7.3

Web DevelopmentLearners apply principles of design and technology, inc[illegible] sta[illegible] to[illegible]

6.3 ScriptingIntegrate scripting into a web page. 6.3

6.3.1 Select and apply scripting languages used in web development. 6.3.1

Skip to main content

maintain web pages and websites with text, graphics, multimedia, scripting, linking, and data integration in a structure that is easy to navigate and accessible for all users via a variety of hardware and software platforms. [6]