

Course: Programming 101 – Introduction to Python

CIP Course Title / Code: Computer Programming / 110201

Duration: Part one of a two-semester series

Grade Levels: 9-12

Requirements: Algebra I

Alternatives: Programming 100 – Computing for all, a slower-paced course taught primarily using visual tools.

Standards: Unfortunately, no state or national standards exist for Computer Science at this time so alignment is difficult. I will focus on SCANs.

This course will also address all three parts of EALR 1 of the Educational Technology standard: “students use technology within all content areas to collaborate, communicate, generate innovative ideas, investigate and solve problems.”

Big Ideas	Enduring Understanding	Essential Questions	SCANs
Communication	Requirements and limitations must be discussed.	How is programming a collaborative activity?	Basic: A, C; Writing: A, C; Listening: A, D; Speaking: A, B
Planning	Solutions to problems must be deliberately designed.	What activities need to occur before programming?	Self-Management: B, E
Exploration	Long-lasting learning occurs through experimentation.	How do we explore in an informed way?	Knowing How to Learn: A, B, C
Modeling	Real-world situations can be represented abstractly.	How do we approach a real-world problem to model?	Seeing Things in the Mind’s Eye: A, B; Mathematics: B
Patterns	Finding patterns allows for generalized problem solutions.	How do we look for patterns?	Reasoning: A, B, C; Mathematics: A, C
Decomposition	Complex tasks can be broken down into discrete steps.	What is the right number of steps?	Problem Solving
Abstraction	Concepts or ideas can be separate from their specific instances.	How does abstraction help us solve problems?	Problem Solving
Errors	Errors are inevitable in complex systems and must be anticipated to reduce harm.	What are the different kinds of errors? How do we protect our users?	Decision Making
Syntax	Programming languages help us express problem solutions to computers.	How do we avoid syntax errors? How are problem solving and syntax sometimes separate?	Reasoning: A, C

Desired Results: Programming 101 – Introduction to Python

Course Understandings	Course Essential Questions	Course Skills
<p>Students will understand that...</p> <ul style="list-style-type: none"> • Programming is a critically important skill in a broad range of industries including healthcare, transportation and finance • Computer scientists and programmers come from all walks of life • Programming is not a solitary activity • Programming language syntax is independent from problem solutions • Problem solving requires extensive planning • Variables allow us to name and re-use values stored in memory • Iteration is a core programming construct used to repeat tasks • Conditional execution lets our programs execute differently based on conditions • Data structures allow us to store and access related data efficiently and logically • Users are an unpredictable source of input 	<ul style="list-style-type: none"> • What is computer science? • What is programming? • When is Python a good tool to use? • Why should we care about computer science? • What does it take to be a good programmer? • How do we break down a problem into steps? 	<p>The student will develop basic programming skills including the ability to...</p> <ul style="list-style-type: none"> • Read and analyze specification documents • Ask for clarification on specifications • Analyze sample input and output • Draw memory state • Write program documentation <p>Specific Python skills will include the ability to:</p> <ul style="list-style-type: none"> • Create and manipulate variables • Branch from different conditions • Create and manipulate arrays • Iterate over data • Ask for user input • Read files • Print out relevant data to troubleshoot program errors

Acceptable Evidence: Programming 101 – Introduction to Python

Performance Tasks	Other Evidence	Self-Assessments
<ul style="list-style-type: none">• Program: draw image patterns, scale based on parameter• Program: play a guessing game with the user• Program: from a set of images, display a collage• Program: game of Breakout• Program: natural language modeling• Requirements: talk to someone about a program they would like to have and write a requirements sheet from it	<ul style="list-style-type: none">• Prompt: what is computer science and how does it affect your life?• Skill Check: bi-weekly puzzle challenges• Quiz: variables, iteration, conditionals• Quiz: arrays• Final: tracing code, devising algorithms	<ul style="list-style-type: none">• Self-assessment: was your guessing game well structured? What are some features you could have added?• Reflection: what skills have you learned that you can transfer to other aspects of your life?• Self-assessment: how thorough are your requirements? Do you think a developer could create the desired software with only your document?• Self-assessment: how good is your natural language modeler? How can you tell its text apart from human text? What would it take to make it more human?

Topic List: Programming 101 – Introduction to Python

<p align="center">Introduction</p> <p>Computer Scientists Your impressions Survey</p>	<p align="center">What are Computers?</p> <p>Parts of a computer Industries that use computers</p>	<p align="center">What are Computers?</p> <p>Embedded computers Ubiquitous Digital</p>	<p align="center">What is Computer Science?</p> <p>Programming Research New applications</p>	<p align="center">What is Programming?</p> <p>Creating computer apps Solving problems</p>
<p align="center">Introducing Python</p> <p>Scripting language Interactive Your first program</p>	<p align="center">Expressions and variables</p> <p>Types, type() Type casting Drawing pictures</p>	<p align="center">Strings</p> <p>Concatenation Printing</p>	<p align="center">Functions</p> <p>Avoiding redundancy Grouping subroutines</p>	<p align="center">Functions</p> <p>Parameters</p>
<p align="center">Requirements</p> <p>Reading assignment write-ups</p>	<p align="center">Comments</p> <p>Where to put them How many to use</p>	<p align="center">Programming style</p> <p>Usability Collaboration</p>	<p align="center">Scope</p> <p>Scope errors Troubleshooting</p>	<p align="center">Conditionals</p> <p>If/else Elif</p>
<p align="center">Iteration</p> <p>for x in range(n)</p>	<p align="center">Iteration</p> <p>while</p>	<p align="center">Media</p> <p>Using images</p>	<p align="center">Media</p> <p>Drawing with Python</p>	<p align="center">User input</p> <p>.readLine() Verifying input</p>
<p align="center">Interactive programming</p> <p>Separating into functions</p>	<p align="center">Loops techniques</p> <p>Sentinel loops Fencepost loops</p>	<p align="center">Lists</p> <p>Initializing Adding to Drawing</p>	<p align="center">Iterating over lists</p> <p>for x in list Reading elements Changing elements</p>	<p align="center">Nested Loops</p> <p>Outer vs inner loops</p>

Topic List: Programming 101 – Introduction to Python

<p align="center">Nested Loops</p> <p>Drawing complex figures</p>	<p align="center">Functions that return</p> <p>Using fruitful functions Not losing the output</p>	<p align="center">Functions that return</p> <p>Writing fruitful functions Writing clean code Descriptive main</p>	<p align="center">Random numbers</p> <p>What is meant by “random” Using good ranges</p>	<p align="center">Files</p> <p>Reading in files Manipulating data</p>
<p align="center">Files</p> <p>Line-based processing</p>	<p align="center">Files</p> <p>Writing out files</p>	<p align="center">Dictionaries</p> <p>Mapping keys to values</p>	<p align="center">Dictionaries</p> <p>Applications</p>	<p align="center">Natural Language Processing</p> <p>Tuples</p>
<p align="center">Errors</p> <p>Types of errors Error recovery</p>	<p align="center">Requirements</p> <p>Gathering requirements Interacting with users</p>	<p align="center">Requirements</p> <p>Making them exhaustive Program Managers</p>	<p align="center">Assertions</p> <p>Understanding code Thinking like a computer</p>	<p align="center">Limitations</p> <p>Problems computers can’t solve</p>