Intro to Computational Thinking with Python

Course Description

This course introduces students to the basic ideas of computational thinking and its applications to problem solving in STEM fields. Students will use a Python IDE, through which they learn how to use abstraction, decomposition and pattern recognition to model problems and arrive at an algorithmic solution. Program code is presented with a dual purpose: as the main way to interact with a computer and as a proxy to organize ideas explicitly and communicate them to other people. Students taking Algebra I concurrently with this course will benefit the most, because many examples are drawn from Algebra I, so that students can visualize and manipulate the mathematical concepts in a more concrete form. This course is designed to help students achieve a CTE accreditation in Python.

Course Objectives

- Describe applications of Computational Thinking to solve Math, Science and Engineering problems.
- Model objects made of multiple parts, as well as their behaviors and interconnections, using variables and functions, and construct virtual artifacts that simulate them.
- Use the program development process to create, debug, and redesign computing artifacts.
- Implement creative projects in which Computational Thinking and code is used to create artistic or technical renderings of diagrams, illustrations or graphs.
- Demonstrate the use of code as a medium to communicate ideas and designs precisely.
- Demonstrate effective communication skills, through team working, oral presentations, and good written communication.

Assessing Performance

Formative assessment includes worksheets, practice activities for each lesson, and daily programming projects. Summative assessment includes quizzes and weekly as well as cumulative programming projects.

Equipment	Cost/Unit
Classroom set of computers; not Chromebooks.	$0 if you already have some, $500-600 per computer if you need to purchase

First Semester

Unit 1: Computing and Coding Basics	Computers hardware and software. Ethics of online communication. Coding environments. Syntax and semantics of Python. Pseudocode. Section 1
Unit 2: I/O Transformations and composition	Send and receive data to the user. Overlays and Translations. The algebra of graphical transformations. Combine parts to create complex objects in Python's turtle. Section 2.1 - 2.4
Unit 3: Symmetry, regularity and Virtual artifacts	Use variables to enable scaling of drawings. Compose turtle operations to create regular patterns (stars, regular polygons). Use variables to design scalable images to be revised in future programs for creating games (rock, paper, scissors and darts). Unit project to create an animation. Section 2.5 and beyond, each unit will come back and build on the programs.
Unit 4: Managing complexity through flowcharts and Boolean Logic	Problem decomposition. Hierarchical organization of code in the form of flowcharts. Decision structures utilizing if, else, and elif statements. Combine Boolen logic with turtle to make win/loss decisions in turtle games. Section 3

Second Semester

Unit 5: Patterns and repetition structures	Understand loops via recursion and iteration. Repeat segments of code to allow us to create functions such as GCD and Prime Factorization. Combine repetition structure with turtle to create complex graphs and examine linear and rotational motion. View Fractals as repetition structures. Section 4
Unit 6: Modeling with functions	Create functions to encapsulate code and enhance reusability. Convert previous programs to functions and allow them to be imported into new programs. Create functions with turtle to allow for transformations. Section 5
Unit 7: Data, calculations and file manipulation	Lists and tuples. Random numbers. Text processing. Calculations with integers and dollar amounts. Data science. Creating code to read large amounts of raw data and process it. Enhance turtle games by reading and writing user's scores. Section 6
Unit 8: Certification Preparation	Examine obscure topics that will be tested such as old formatting methods, advanced manipulation techniques and unit tests. Take multiple practice tests. Use Gmetrix practice platform to familiarize students with the layout of the certification exam. Sections 7 and 8

INTRODUCTION TO COMPUTATIONAL THINKING

1. Materials
   Internet access, 1-to-1 computer use daily, and access to LSU servers.
2. Required software, networking access, and access to LSU servers:
   - Students will need to sign up with online development and testing environments, including but not limited to codesandbox.io, jsfiddle.net, scratch.mit.edu and others.
   - Students will need access to YouTube instructional videos relevant to the course, as well as other educational video repositories.
   - Teachers will need to be able to access the LSU servers using several Internet protocols including but not limited to HTTPS and SSH.
   - Principals will need to communicate with the district's information technology department to ensure that there are no technological restrictions that block access to the LSU servers in the lsu.edu, college-readiness.lsu.edu or stempathways.lsu.edu domains on any port.
3. Required teacher collaborations
   Teachers will communicate with LSU instructors via emails, Google Drive,, Google Drive, and/or apps hosted on the LSU servers. Teachers will need to share sample student work with their designated LSU Pathway Point-of-Contact.
4. Required administration of course content, pre/post test, and research instruments
   All required materials and instruments will either be posted in the LSU servers, or their location announced via email.
5. Course Work
   Teachers must present the course material in sequence or as approved by collaboration with the LSU Pathway Point-of-Contact. Teachers are expected to deliver a minimum of 80% of the course material.
6. Other
   As this is a project-based learning class, we strongly suggest that each section of the course be limited to a *maximum* of 20 students. The course is dependent on the teacher providing feedback and reviewing student code. The course requires that teachers have adequate time to interact with each student.