Intro to Python Course Syllabus

Course Overview

Intro to Python introduces students to many ideas across the field of computer science. In this course, students will learn to design and evaluate solutions and to apply computer science to solve problems through the development of algorithms and programs. They will incorporate abstraction into programs and use data to discover new knowledge. Students will exit this course as confident Python programmers with a strong grasp of the fundamentals of computer science.

Prerequisites

This course is designed for students with no prior programming experience. However, a strong foundation in first-year high school algebra is recommended.

Required Resources

Online resources and labs provided by the instructor

- Online Text: JuiceMind.com

Course Goals

- Increase accessibility and participation in computer science.
- Develop confidence in computational problem-solving.
- Understand core computing principles.
- Encourage the creation of computational artifacts as expressions of creativity.
- Foster collaborative problem-solving skills.
- Promote responsible and ethical use of technology.

Learning Environment

The course is structured as a blended learning experience, combining web-based resources with in-class activities. Students will engage in coding exercises, digital presentations, written reflections, and work collaboratively on projects. Lessons include interactive readings, quizzes, interactive exercises, and projects.

Programming Environment

Students will use a browser-based editor to write and execute Python programs, including creating graphical applications using the graphics library tkinter. This environment is designed to support the development of computational thinking skills from the beginning of the course.

Assessment

Students are regularly assessed for knowledge and skills through completion of multiple-choice questions, programming exercises, and larger programming projects.

- **Formative Assessment:**
  - **Review Quizzes:** All lessons are immediately followed by short multiple-choice quizzes to assess essential knowledge. Students are given only one chance to answer each question.
  - **Exercises:** Lessons focused on programming skills are followed by one to three short programming exercises to assess mastery of essential skills. Students are given unlimited attempts to run and test their code against the exercise requirements, but only one chance to submit it for grading.

- **Summative Assessment:**
  - **Unit Exams:** Most units end with a single, longer multiple choice exam.
  - **Final Project (Unit 7):** This project assesses students' ability to make design decisions and translate them into code.

Recommended Grading Scheme

Category	Weighting	Method
Review	20%	Completion
Exercises	45%	Correctness (graded automatically)
Unit Exams	25% (5 total, 5% each)	Correctness (graded automatically)
Project - Tell a Story	10%	Quality of Submission (graded by teacher)

Pacing

Lesson plans are structured around a 45-minute class period, and lesson folders should take 45 minutes to complete, unless otherwise stated. The instructional content of a lesson

should never take more than this long to teach, but exercises and review may spill over into homework based on student ability.
In total, JuiceMind's AP Cybersecurity is designed to take 110 days to complete using only material included within this course. This is shorter than a typical school year to allow for additional review and supplemental content, and to accommodate the AP Exam (typically around May 15), which occur before the end of the school year.

Course Breakdown

Unit 1: Introduction to Security (~10 class periods)

Students explore the foundations of cybersecurity and learn how attackers exploit human behavior and technological vulnerabilities.

Topics Covered:

- 1.1 Understanding Social Engineering
- 1.2 Suspicious Website Logins
- 1.3 Best Practices for Public Networks
- 1.4 AI-Based Cybersecurity Attacks
- 1.5 Leveraging AI in Cyber Defense

Students examine common cyber threats and learn how individuals and organizations can reduce risk through safer digital practices.

Unit 2: Securing Spaces (~21 class periods)

Students learn how cybersecurity extends beyond digital systems into physical environments. They explore how physical access and environmental vulnerabilities can compromise computer systems.

Topics Covered:

- 2.1 Cyber Foundations
- 2.2 Physical Vulnerabilities and Attacks

2.3 Protecting Physical Spaces
2.4 Detecting Physical Attacks

Students analyze real-world examples of physical breaches and design strategies for securing facilities and equipment.

Unit 3: Securing Networks (~26 class periods)

This unit focuses on protecting computer networks from unauthorized access and cyber attacks.

Topics Covered:

3.1 Network Vulnerabilities and Attacks
3.2 Protecting Networks: Managerial Controls and Wireless Security
3.3 Protecting Networks: Segmentation
3.4 Protecting Networks: Firewalls
3.5 Detecting Network Attacks

Students learn how network defenses work and explore techniques used to detect malicious activity.

Unit 4: Securing Devices (~23 class periods)

Students examine vulnerabilities in computing devices and the techniques used to protect them.

Topics Covered:

4.1 Device Vulnerabilities and Attacks
4.2 Authentication
4.3 Protecting Devices
4.4 Detecting Attacks on Devices

Students investigate authentication systems, device hardening, and strategies used to identify compromised devices.

Unit 5: Securing Applications and Data (~30 class periods)

Students learn how software and stored data are protected against cyber threats. They explore cryptography, access controls, and application security.

Topics Covered:

- 5.1 Application and Data Vulnerabilities and Attacks
- 5.2 Protecting Applications and Data: Managerial Controls and Access Controls
- 5.3 Protecting Stored Data with Cryptography
- 5.4 Asymmetric Cryptography
- 5.5 Protecting Applications
- 5.6 Detecting Attacks on Data and Applications

Students analyze how encryption and security practices help protect sensitive information in modern digital systems.