Python Programming Cover

Content Area: **Sample Content Area**
Course(s):
Time Period:
Length: **Semester**
Status: **Published**

Course Overview

In this course, students are introduced to programming and problem solving using the Python language. Algorithm development and basic problem solving techniques are introduced using a procedural approach. Topics covered include programming with numbers, strings, lists, tuples, sets, dictionaries, files, control structures, functions with parameter passing, scope and the Python libraries. Finding and fixing errors, using a debugger and an introduction to exception handling, are presented and implemented.

Course Name, Length, Date of Revision and Curriculum Writer

Python Programming

Elective

Sayreville War Memorial High School

2.5 Credits

Semester Course

Curriculum Writer: Atiyah Conry

Table of Contents

Table of Contents

Unit 01: Python I/O and Math

Content Area: **Sample Content Area**
Course(s):
Time Period: **1st Semester**
Length: **13 days**
Status: **Awaiting Review**

Summary of the Unit

This introductory Python programming unit will equip students with the fundamental building blocks of coding and computational thinking. Students will first explore how computers process information, from receiving input to producing output, gaining a deeper understanding of the technology they use every day. Through hands-on practice with the user-friendly Replit.com online environment, students will create their own Python programs, experiencing the power of code firsthand.

Building on these core concepts, students will master essential programming techniques. They will learn to display information on screen, gather input from the user, store data effectively, and perform basic calculations. This knowledge will enable them to create interactive programs that respond to user input and manipulate data, opening up a world of creative possibilities. By the end of this unit, students will not only be familiar with Python syntax, but they will also have developed problem-solving skills and a solid foundation for future programming endeavors.

Enduring Understandings

Python is a high-level programming language consisting of commands used to store, manipulate, and display data.

- Computers as Tools: Computers are powerful tools that can process information efficiently and accurately when given precise instructions through programming.
- Input/Output Interaction: Programs interact with users by taking input, processing it, and providing output. This interaction enables dynamic and personalized user experiences.
- Data Types and Operations: Different types of data require different handling, and specific operations can transform and manipulate that data to solve problems and generate insights.
- Mathematical Problem Solving: Programming allows for the automation of mathematical calculations, making it a valuable tool for solving complex problems and analyzing data.
- Syntax and Structure: Programming languages, like Python, have specific syntax rules and structures that must be followed for instructions to be understood and executed by the computer.

Essential Questions

- How do computers communicate with humans through input and output?
- How can we use Python to gather information from users and respond effectively?

- Why are different data types (e.g., strings, integers, floats) important in programming, and how do we convert between them?
- What are the fundamental mathematical operations in Python, and how can they be combined to solve problems?
- How do programming concepts like variables and functions help us organize and structure our code for clarity and efficiency?
- How can we use Python's math capabilities to create programs that analyze data, automate calculations, and make decisions?
- What are the real-world applications of using Python for input/output operations and mathematical computations?
- How does following the correct syntax and structure in Python ensure our code is interpreted and executed accurately by the computer?
- How can we apply our knowledge of Python I/O and math to create interactive and useful programs?

Summative Assessment and/or Summative Criteria

Students will create a Python program that demonstrates their understanding of the unit's concepts. The project should incorporate the following elements:

- Input/Output Interaction: The program should gather input from the user, process it, and provide meaningful output.
- Data Manipulation: The program should use variables to store different types of data (e.g., strings, integers, floats) and perform relevant operations on them.
- Mathematical Operations: The program should utilize at least two different mathematical operations to solve a problem or generate a result.
- Code Structure: The program should be well-organized, with comments explaining the purpose of different sections and clear variable names.

Project Examples:

- Calculator: A program that takes mathematical input from the user and performs the desired operation(s).
- Unit Converter: A program that converts between different units of measurement (e.g., Celsius to Fahrenheit, inches to centimeters).
- Interactive Story: A Mad Libs-style program that gathers words from the user and creates a story or poem.
- Simple Data Analysis: A program that analyzes a set of data (e.g.,

Students will complete written assessment covering topics from unit 1

Resources

- Replit online ID

- Project STEM curriculum Python curriculumE
- Python API (https://docs.python.org/3.12/)

Unit Plan

Topic/Selection Timeframe	General Objectives	Instructional Activities	Benchmarks/Assessments	Standards
Computer Programming and hig level programming languages (1 day)	SWBAT ○ Describe the different levels of programming languages ○ Identify high and low level languages ○ Define computer programming	○ Teacher will discuss computer programming and its definition ○ Teacher will discuss levels of programming language and the difference between the different levels ○ Students will research list of common and uncommon high level programming languages	completion of programming language research activity	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Basic output (1 day)	SWBAT ○ Create, run and save a python program	○ Teacher will review and discuss	Updated hello world project Project Stem Coding Activities Section 1.5	CS.9-12.AP

	○ Define the information processing cycle ○ Use print statements in Python	the information processing cycle ○ Teacher will introduce the replit IDE ○ Students will create hello world application ○ Teacher will discuss the different components of the Print statement		CS.9-12.DA CS.9-12.IC
Basic input (1 day)	SWBAT ○ Be able to define and utilize input and variables ○ Be able to describe the rules for good variable naming	○ Teacher will introduce topic ○ Teacher will discuss Print statements and variations in Python ○ Students will practice writing python statements	Project Stem Coding Activities Section 1.6	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Variables and their types (1 day)	SWBAT ○ Be able to define and utilize variables ○ Be able to distinguish	○ Teacher will introduce variable data structure	Project Stem Coding Activities Section 1.7	CS.9-12.AP CS.9-12.DA

	between the variable types of integer and string	○ TW discuss variable types and review primitive types in Python (int, float and String) ○ SW complete Project Stem coding activities		CS.9-12.IC
Lab Assignment: Silly Sentences (3 days)	SWBAT ○ Demonstrate knowledge of variables and print statements by completing Assignment	Students will write program that asks for user input and complete sentences base on the user input	Project Stem Assignment: Silly Sentences. Students will use input and output for	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Basic Mathematical equations (1 day)	SWBAT: ○ Define and utilize floats, operators and assignments. ○ List the symbols for the basic operators and exponents. ○ Perform the order of operations. ○ Be able to define and utilize modular division	○ Teacher will discuss math operators and statements in Python ○ Teacher will discuss the different types of division in Python ○ Students will complete Project Stem coding activities	Project Stem Coding Activities Section 2.2 and 2.3	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Math Functions (1 day)	SWBAT ○ Be able to define and utilize import, functions, and modules ○ Know how to calculate the square roots, absolute values, and power of numbers ○ Be able to generate random numbers with the random function	○ Teacher will will introduce Math functions and how they enhance arithmetic in Python programs ○ Teacher will discuss import statements and how they bring functionality into a program ○ Teacher will discuss random library and describe the psuedorandom values in programming languages ○ Students will complete project stem practice activities	Project Stem Coding Activities Section 2.4 and 2.5	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Lab Assignment: Room Area (3 days)	SWBAT ○ Demonstrate knowledge of input, output and math operations	○ Students will write program to calculate the area of given room with	Project Stem: Room Area assignment	CS.9-12.AP CS.9-12.DA

	in Python by completing a room area project	unusual dimensions		CS.9-12.IC
Unit 1 Exam (1 day)	Students will demonstrate knowledge of python input, output and mathematical operations by completing Unit 1 Exam	Students will take unit 1 Exam	Unit Exam	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Standards

12.9.3.IT-PRG	Programming & Software Development
CS.9-12.8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.CS.2	Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3	Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2	Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.3	Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4	Explain the relationship between binary numbers and the storage and use of data in a computing device.
CS.9-12.8.1.12.DA.5	Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.2	Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.2.12.ED.1	Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
CS.9-12.8.2.12.ED.5	Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic

	considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.ITH.3	Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.
CS.9-12.DA	Data & Analysis
	Successful troubleshooting of complex problems involves multiple approaches including research, analysis, reflection, interaction with peers, and drawing on past experiences.
	Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Suggested Modifications for Special Education, ELL and Gifted Students

Special Education Students:

- Visual Supports: Utilize visual aids like flowcharts and diagrams to illustrate the input/output process and order of operations.
- Chunked Instruction: Break down tasks into smaller, manageable steps with clear instructions and examples.
- Alternative Input Methods: Allow students to use voice-to-text or other assistive technologies for inputting code or responses.
- Reduced Complexity: Modify assignments to focus on core concepts, reducing the number of steps or complexity of calculations.
- Frequent Check-Ins: Provide regular feedback and guidance to ensure understanding and prevent frustration.
- Extra Practice: Offer additional practice opportunities with scaffolding and support to build confidence.

English Language Learners (ELLs):

- Visuals and Realia: Use visuals, diagrams, and real-world objects to reinforce vocabulary and concepts.
- Simplified Language: Provide instructions and explanations in clear, concise language, avoiding jargon and idioms.
- Bilingual Resources: Offer dictionaries or translation tools to support vocabulary acquisition.
- Peer Support: Pair ELLs with supportive peers to provide language scaffolding and collaboration opportunities.
- Multilingual Coding Examples: Use code examples with comments in multiple languages to aid understanding.
- Cultural Connections: Relate programming concepts to familiar cultural contexts or real-world scenarios.

Gifted Students:

- Open-Ended Challenges: Offer open-ended projects or problems that allow for creativity and exploration beyond the basic requirements.
- Advanced Concepts: Introduce more complex Python features like conditional statements or loops to extend learning.

- Independent Research: Encourage students to research and explore additional libraries or modules related to their interests.
- Mentoring Opportunities: Pair gifted students with mentors who can provide guidance and support in their advanced explorations.
- Real-WorldApplications: Connect programming concepts to real-world problems or challenges in fields like data science or artificial intelligence.
- Collaboration: Encourage collaboration with other gifted students to create more sophisticated or innovative projects.

Suggested Technological Innovations/Use

- Students should use online resources to research topics covered
- Students will use java IDE to test and execute code
- Students will use Java API online resource

Cross Curricular/Career Readiness, Life Literacies and Key Skills Practice

9.1 21st Century Life and Career Skills: All students will demonstrate the creative, critical thinking, collaboration, and problem-solving skills needed to function successfully as both global citizens and workers in diverse ethnic and organizational cultures.

9.3– Career and Technical Education Career Cluster: Information Technology (IT)
• 9.3.12.IT.11: Demonstrate knowledge of the hardware components associated with information systems.
• 9.3.12.IT-SUP.9: Employ technical writing and documentation skills in support of an information system.
Pathway: Programming & Software Development (IT-PRG)
• 9.3.12.IT-PRG.4: Demonstrate the effective use of software development tools to develop software applications.
• 9.3.12.IT-PRG.5: Apply an appropriate software development process to design a software application.
• 9.3.12.IT-PRG.6: Program a computer application using the appropriate programming language.
• 9.3.12.IT-PRG.7: Demonstrate software testing procedures to ensure quality products.

English Language Arts
• Journal writing • Close reading of industry-related content • Create a brochure for a specific industry • Keep a running word wall of industry vocabulary Social Studies • Research the history of a given industry/profession • Research prominent historical individuals in a given industry/profession • Use historical references to solve problems
World Language • Translate industry-content • Create a translated index of industry vocabulary • Generate a translated list of words and phrases related to information technology

Math • Compare and contrast use of equations and variables in algebra and programming. • Program graphics and use the properties of geometric shapes • Compare the computer graphic coordinate system with the Cartesian coordinate plane in math • Compare probability and the use of random numbers in computer programming. • Track and track various data, such as industry's impact on the GDP, career opportunities or

among of individuals currently occupying careers

Fine & Performing Arts • Create a poster recruiting young people to focus their studies on a career in Information Technology

Science • Research the environmental impact of a given career or industry • Research latest developments in Information technology • Investigate applicable-careers in STEM fields

Unit 02: Control Structures

Content Area: **Sample Content Area**
Course(s):
Time Period: **1st Semester**
Length: **19 days**
Status: **Awaiting Review**

Summary of the Unit

This unit builds on the foundational programming concepts learned in the introductory Python unit. Students will dive into control structures, the essential tools that allow programs to make decisions and repeat actions based on specific conditions. They will explore conditional statements (`if`, `else`, `elif`) to create branching paths within their code, enabling programs to respond differently depending on user input or other data. Additionally, they will learn about loops (`for`, `while`) which empower programs to automate repetitive tasks, making them more efficient and powerful.

Through a series of engaging exercises and projects, students will gain hands-on experience implementing control structures to create more dynamic and interactive programs. They will learn to design logical flows, test conditions, and control the execution of code blocks. By the end of the unit, students will have a solid grasp of control structures, a fundamental building block for creating sophisticated and intelligent software.

Enduring Understandings

- Program Flow: Programs don't just execute line by line; control structures allow for dynamic decision-making and repetition, altering the flow of execution based on conditions.
- Conditional Logic: Conditional statements (if, else, elif) enable programs to evaluate conditions and execute different code blocks accordingly, making them adaptable and responsive.
- Loops and Iteration: Loops (for, while) automate repetitive tasks, allowing programs to process data efficiently and perform actions a specified number of times or until a condition is met.
- Logical Thinking: Programming with control structures requires strong logical thinking skills to design algorithms that effectively address different scenarios and problem-solving approaches.
- Code Efficiency: Control structures can significantly improve the efficiency and readability of code by eliminating the need to write redundant instructions.
- Building Complexity: Control structures are the building blocks for creating complex and sophisticated programs that can adapt to varying inputs and situations.

Essential Questions

- How do control structures (if-else statements and loops) change the way a program executes?
- How can we use conditional statements to create programs that make decisions based on different inputs or conditions?
- What types of problems can be solved more efficiently using loops, and how do we choose the appropriate type of loop (for or while)?

- How do boolean expressions (True/False) play a crucial role in controlling the flow of programs with conditional statements and loops?
- What are the potential pitfalls or errors that can arise when using control structures, and how can we debug our code effectively?
- How can we combine multiple control structures to create more complex programs with nested conditions or multiple loops?
- How do control structures contribute to the overall logic and structure of a well-designed program?
- What are some real-world examples of programs that rely heavily on control structures to achieve their functionality?

Summative Assessment and/or Summative Criteria

Option 1: Project-Based Assessment

Students will create a Python program that demonstrates their mastery of control structures. The project should incorporate the following elements:

- Conditional Statements: The program should use `if`, `else`, and/or `elif` statements to make decisions based on user input or other conditions.
- Loops: The program should use `for` or `while` loops to automate repetitive tasks.
- Logical Flow: The program should have a clear and logical flow of execution, with appropriate use of indentation and control structures.
- Error Handling: The program should include basic error handling mechanisms to prevent crashes and provide helpful feedback to the user.

Project Examples:

- Number Guessing Game: The program generates a random number, and the user has to guess it within a certain number of attempts. The program provides feedback on whether the guess is too high or too low.
- Quiz Game: The program asks the user a series of questions and keeps track of the score. The program provides feedback on correct and incorrect answers.
- Text-Based Adventure: The program presents the user with a series of choices that affect the outcome of the story. The program uses conditional statements to create branching paths.
- Grade Calculator: The program takes a list of grades from the user and calculates the average. The program uses loops to iterate over the list of grades.

Resources

- Replit online ID
- Project STEM curriculum Python curriculumE
- Python API (https://docs.python.org/3.12/)

Unit Plan

Topic/Selection Timeframe	General Objectives	Instructional Activities	Benchmarks/Assessments	Standards
Boolean expressions and logical operators (1 day)	SWBAT • Be able to list the symbols for different relational operators • Understand and be able to use logical operators • Use logical operators to create boolean expressions	○ Students will complete Project Stem coding Activities	Project Stem Coding Activities Section 3.3	CS.9-12.AP CS.9-12.DA CS.9-12.IC
If Else Statements (2 days)	SWBAT ○ Be able to define and code if-else statements ○ Be able to define and code nested elif (else-if) statements ○ Be able to determine when it would be appropriate to use an if statement, if-else statement,	○ Teacher will review If Else statements and how they enhance the enhance the computer program ○ Students will work with partners to write code that will ○ Students will complete Project Stem	Project Stem Coding Activities Section 3.1 and 3.2	CS.9-12.AP CS.9-12.DA CS.9-12.IC

	or else-if statement	coding Activities		
Lab Assignment: Chat Bot (3 days)	Students will Demonstrate knowledge of if statements by creating a simple chat bot	○ Students will complete chat bot assignment	Project Stem: Chat Bot Assignment	CS.9-12.AP CS.9-12.DA CS.9-12.IC
While Loops (2 days)	SWBAT ○ Use a while loop and a do while loop in a Python program ○ Be able to define and code while loops and loop control variables ○ Describe the purpose of a loop in a computer program.	○ Teacher will introduce the idea of the loop in programming ○ Students will review the structure of the while loop ○ Students will complete Project Stem coding Activities	Project Stem Coding Activities Section 4.1	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Counting vs. user controlled loops (2 days)	SWBAT ○ Be able to end loops using count variables and user input ○ Be able to state when to use them	○ Teacher will discuss loops and how the two loop structures ○ Students will will work to build examples of both ○ Class will discuss scenarios when one should be	. Project Stem Coding Activities Section 4.2 and 4.3	CS.9-12.AP CS.9-12.DA CS.9-12.IC

		used over the other ○ Students will complete Project Stem coding Activities		
Lab: guessing game (3 days)	SWBAT demonstrate knowledge of the while loop	○ Students will complete guessing game assignment	Guessing game: User will be asked to enter numbers attempting to guess secret random number until they quit or get the answer correct. Enhanced: students will give feedback about input (hot or cold) based on guess relationship to secret value.	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Range function and for loops (2 days)	SWBAT ○ Define and utilize the range function using 1, 2, or 3 parameters ○ Be able to define and code for loops ○ Learn when to use a for loop rather than a while loop	○ Students will complete Project Stem coding Activities	Project Stem Coding Activities Section 4.6, 4.7 and range practice worksheet	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Loop Algorithms (1 day)	SWBAT ○ Use a for loop with a sum variable to	○ Students will complete Project Stem coding Activities	Project Stem Coding Activities Section 4.9	CS.9-12.AP CS.9-12.DA CS.9-12.IC

	sum a set of variables ○ use loops for counting and summing			
Lab (2 days)	SWBAT ○ Demonstrate knowledge of do-while loops and conditional statements by completing the crack the code assignment.	○ Students will work on the "crack the code" programming assignment. ○ Extension: Students will be asked to add phase of the assignment that users would have to pass to "crack the code"	In this assignment, you will create a program requiring a secret code to “unlock.” The program should first welcome the user and ask the user to input his/her name. Then the program will greet the user using the entered name. To “crack the code,” the user must input three integer numbers which satisfy the following conditions: The first number must be the number 3. The second number can be the number 1 or between 33 and 100, inclusive. The third number must be a positive number that is either evenly divisible by 3 or evenly divisible by 7	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Unit 2 Test (1 day)	SWBAT • Demonstrate knowledge of loops and conditional statements by completing written	Students will will take unit 2 exam	Unit 2 Test	CS.9-12.AP CS.9-12.DA CS.9-12.IC

	assessemem ts			

Standards

12.9.3.IT-PRG Programming & Software Development
CS.9-12.8.1.12.AP.1 Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.5 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.CS.2 Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3 Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2 Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.3 Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4 Explain the relationship between binary numbers and the storage and use of data in a computing device.
CS.9-12.8.1.12.DA.5 Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.2 Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.2.12.ED.1 Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
CS.9-12.8.2.12.ED.5 Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.ITH.3 Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.
CS.9-12.DA Data & Analysis
Successful troubleshooting of complex problems involves multiple approaches including research, analysis, reflection, interaction with peers, and drawing on past experiences.
Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Suggested Modifications for Special Education, ELL and Gifted Students

Special Education Students:

- Visual Aids: Utilize flowcharts and diagrams to illustrate the flow of logic in control structures.
- Simplified Examples: Break down complex code into smaller, more manageable chunks with clear explanations and visuals.
- Scaffolded Activities: Provide guided practice with gradually increasing complexity, starting with fill-in-the-blank exercises and progressing to independent problem-solving.
- Reduced Cognitive Load: Limit the number of new concepts introduced at once, focusing on mastery of individual control structures before combining them.
- Alternative Assessments: Offer options for demonstrating understanding beyond traditional tests, such as oral explanations or visual representations of code logic.
- Assistive Technologies: Allow the use of text-to-speech or speech-to-text tools to support students with reading or writing difficulties.

English Language Learners (ELLs):

- Visuals and Realia: Use visual aids like flowcharts, diagrams, and real-world examples to reinforce vocabulary and concepts related to control structures.
- Simplified Language: Provide instructions and explanations in clear, concise language, avoiding jargon and idioms.
- Bilingual Dictionaries: Encourage the use of bilingual dictionaries or online translation tools to support vocabulary acquisition.
- Peer Support: Pair ELLs with supportive peers to provide language scaffolding and collaboration opportunities.
- Code Examples with Comments: Provide code examples with comments in both English and the student's native language to aid comprehension.
- Scaffolding of Writing: Break down written assignments into smaller steps and provide sentence frames or templates to support language production.

Gifted Students:

- Challenge Problems: Offer advanced problems that require the application of multiple control structures in creative and complex ways.
- Open-Ended Projects: Allow for more autonomy and creativity in project design, encouraging exploration of advanced control flow patterns or algorithms.
- Independent Research: Encourage independent research into topics like optimization techniques or advanced control structures in other programming languages.
- Mentoring Opportunities: Pair gifted students with mentors in the field or provide opportunities to participate in coding competitions or hackathons.
- Advanced Concepts: Introduce concepts like nested loops, recursion, or exception handling to extend learning beyond the basic curriculum.
- Collaborative Projects: Encourage collaboration with other gifted students to tackle complex problems or create innovative software solutions.

Suggested Technological Innovations/Use

- Students should use online resources to research topics covered
- Students will use java IDE to test and execute code

- Students will use Java API online resource

Cross Curricular/Career Readiness, Life Literacies and Key Skills Practice

9.1 21st Century Life and Career Skills: All students will demonstrate the creative, critical thinking, collaboration, and problem-solving skills needed to function successfully as both global citizens and workers in diverse ethnic and organizational cultures.

9.3– Career and Technical Education Career Cluster: Information Technology (IT)
• 9.3.12.IT.11: Demonstrate knowledge of the hardware components associated with information systems.
• 9.3.12.IT-SUP.9: Employ technical writing and documentation skills in support of an information system.
Pathway: Programming & Software Development (IT-PRG)
• 9.3.12.IT-PRG.4: Demonstrate the effective use of software development tools to develop software applications.
• 9.3.12.IT-PRG.5: Apply an appropriate software development process to design a software application.
• 9.3.12.IT-PRG.6: Program a computer application using the appropriate programming language.
• 9.3.12.IT-PRG.7: Demonstrate software testing procedures to ensure quality products.

English Language Arts
• Journal writing • Close reading of industry-related content • Create a brochure for a specific industry • Keep a running word wall of industry vocabulary Social Studies • Research the history of a given industry/profession • Research prominent historical individuals in a given industry/profession • Use historical references to solve problems
World Language • Translate industry-content • Create a translated index of industry vocabulary • Generate a translated list of words and phrases related to information technology

Math • Compare and contrast use of equations and variables in algebra and programming. • Program graphics and use the properties of geometric shapes • Compare the computer graphic coordinate system with the Cartesian coordinate plane in math • Compare probability and the use of random numbers in computer programming. • Track and track various data, such as industry's impact on the GDP, career opportunities or among of individuals currently occupying careers

Fine & Performing Arts • Create a poster recruiting young people to focus their studies on a career in Information Technology

Science • Research the environmental impact of a given career or industry • Research latest developments in Information technology • Investigate applicable-careers in STEM fields

Unit 03: Functions

Content Area: **Sample Content Area**
Course(s):
Time Period: **1st Semester**
Length: **17 days**
Status: **Not Published**

Summary of the Unit

This unit delves into the power of functions, essential building blocks for organizing and modularizing code. Students will learn how to define their own functions, which are reusable blocks of code that perform specific tasks. They will discover the benefits of functions, such as improving code readability, promoting reusability, and simplifying complex programs. Students will explore different types of functions, including those that return values, those that operate on parameters, and those with default arguments. Through hands-on practice, students will apply their knowledge to design and implement functions that encapsulate specific functionality within their programs. By the end of this unit, students will have honed their ability to write clean, efficient, and reusable code, a crucial skill for tackling larger and more intricate programming projects.

Enduring Understandings

- Code Modularity: Functions enable the breakdown of complex tasks into smaller, manageable units, making code easier to read, understand, and maintain.
- Reusability: Functions can be reused throughout a program or across multiple programs, saving time and effort by avoiding redundant code writing.
- Abstraction: Functions allow programmers to focus on the higher-level purpose of a piece of code without getting bogged down in the specific details of its implementation.
- Parameters and Arguments: Functions can accept input (parameters) and produce output (return values), enabling them to be customized and used in different scenarios.
- Scope and Encapsulation: Variables defined within a function have local scope, which helps prevent unintended side effects and improves code organization.

Essential Questions

- When and why are functions necessary when creating a computer program?
- What are the different components to creating a Python function?
- How do we manage variable and data structure parameters?
- What are local and global variables?

Summative Assessment and/or Summative Criteria

Option 1: Project-Based Assessment

Students will create a Python program that demonstrates their understanding of functions. The project should incorporate the following elements:

- Multiple Functions: The program should include at least three custom-defined functions, each with a clear purpose and well-defined inputs and outputs.
- Parameter Passing: Functions should demonstrate the use of parameters to receive input and tailor their behavior accordingly.
- Return Values: At least one function should return a value that is then used by the main program or another function.
- Code Organization: The code should be well-organized, with functions clearly defined and appropriately named.

Project Examples:

- Text Analyzer: A program that uses functions to analyze text input, calculating statistics like word count, average word length, and frequency of specific words.
- Data Conversion: A program that uses functions to convert data between different units of measurement (e.g., temperature, currency) or formats (e.g., decimal to binary).
- Game with Functions: A simple game (like a quiz or number guessing game) that utilizes functions to handle tasks like displaying the game menu, generating questions or numbers, and evaluating user input.
- Calculator with Functions: A calculator program that uses functions to perform different mathematical operations (addition, subtraction, multiplication, division, etc.).

Students will complete test#3 reviewing the topics covered in the unit.

Resources

- Replit online ID
- Project STEM curriculum Python curriculumE
- Python API (https://docs.python.org/3.12/)

Unit Plan

Topic/Selection Timeframe	General Objectives	Instructional Activities	Benchmarks/Assessments	Standards

Functions (2 days)	SWBAT • Be able to explain the purpose of a function in programmin g • Be able to define your subprograms and call them in a program	○ The teacher will introduce the concept of a method. The teacher will discuss its parts and role in designing and developing a complex Java program. ○ The teacher will demonstrate a basic void method. ○ Students will complete Project Stem coding activities	Project Stem Coding Activities Section 7.2	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Parameters (3 days)	SWBAT ○ Be able to define a parameter and create a call for your subprograms with parameters. ○ Be able to define a subprogram that uses optional parameters	○ The teacher discusses the concept of parameters. ○ The teacher will demo how parameters are used to send information from the calling method to the called method. ○ Students will complete Project Stem coding activities	Project Stem Coding Activities Section 7.3	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Return Methods (3 days)	SWBAT ○ Be able to code functions that return values and use the returned values in your code.	○ The teacher discusses the concept of return methods ○ The teacher demo how return values allow users to ○ Students will alter the currency conversion program to use return values to communicate to the calling method. ○ Students will complete Project Stem coding activities	Project Stem Coding Activities Section 7.4	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Lab: calender conversion (3 days)	SWBAT ○ Demonstrate the ability to design and implement methods of varying complexity	○ The teacher will discuss project descriptions ○ Students will work to complete Methods Sampler Assignment ○ Students will be encouraged to create plans before starting projects and to add full documentation to code.	Project Stem: Calender Conversion Lab: In this assignment, you will create a calendar program that allows the user to enter a day, month, and year in three separate variables, as shown below.	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Lab: Escape room project (5 days)	Students will demonstrate mastery of the creation and use of methods	Students will ○ work on program that will use functions to simulate a escape room-themed game ○ Extension: Students will be asked to create a method to "crack" a message when no key is provided.	Escape room project: SW work to create an escape room project using functions (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Unit Exam (1 day)	Students will ○ Demonstrate knowledge of loops and condtitional statements by completing written assessments	○ Teacher will administer written unit exam covering topics from Unit 4	Unit Exam	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Standards

12.9.3.IT-PRG Programming & Software Development
CS.9-12.8.1.12.AP.1 Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.5 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.CS.2 Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3 Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2 Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.3 Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4 Explain the relationship between binary numbers and the storage and use of data in a computing device.
CS.9-12.8.1.12.DA.5 Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.2 Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.2.12.ED.1 Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
CS.9-12.8.2.12.ED.5 Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.ITH.3 Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.
CS.9-12.DA Data & Analysis
Successful troubleshooting of complex problems involves multiple approaches including research, analysis, reflection, interaction with peers, and drawing on past experiences.
Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Suggested Modifications for Special Education, ELL and Gifted Students

Special Education Students:

- Visual Representations: Use flowcharts or diagrams to illustrate the flow of data in and out of functions, highlighting the concept of input parameters and return values.
- Hands-on Activities: Incorporate hands-on activities like building function "machines" with physical objects to manipulate and visualize the input/output process.
- Chunking: Break down complex functions into smaller, more manageable parts, focusing on one concept at a time.

- Simplified Syntax: Start with simpler function examples using fewer parameters and more straightforward operations.
- Repetition and Practice: Provide ample opportunities for repetition and practice with guided exercises and immediate feedback.
- Assistive Technologies: Utilize text-to-speech or speech-to-text tools to assist with reading or writing code.

English Language Learners (ELLs):

- Visual Aids: Use visual aids like diagrams, infographics, and real-world examples to reinforce vocabulary and concepts related to functions.
- Simplified Language: Explain concepts in simple, clear language, avoiding technical jargon and idioms.
- Multilingual Resources: Provide bilingual dictionaries or translation tools to support vocabulary acquisition.
- Peer Support: Pair ELLs with supportive peers for collaborative learning and language scaffolding.
- Code Examples with Comments: Provide code examples with comments in both English and the student's native language to aid comprehension.
- Scaffolding of Writing: Break down written assignments into smaller steps and provide sentence frames or templates to support language production.

Gifted Students:

- Challenging Problems: Offer complex problems that require the design of multiple interconnected functions with varying inputs and outputs.
- Advanced Concepts: Introduce concepts like recursive functions, lambda functions, or higher-order functions to extend learning beyond the basics.
- Independent Research: Encourage independent research into topics like functional programming paradigms or how functions are used in different programming languages.
- Mentorship Opportunities: Connect gifted students with mentors who can provide guidance and support in their advanced explorations.
- Creative Projects: Allow for more open-ended projects that encourage creativity and experimentation with different ways of using functions.
- Collaboration: Encourage collaboration with other gifted students to tackle more ambitious projects that require teamwork and the integration of diverse functions.

Suggested Technological Innovations/Use

- Students should use online resources to research topics covered
- Students will use java IDE to test and execute code
- Students will use Java API online resource

Cross Curricular/Career Readiness, Life Literacies and Key Skills Practice

9.1 21st Century Life and Career Skills: All students will demonstrate the creative, critical thinking, collaboration, and problem-solving skills needed to function successfully as both global citizens and workers in diverse ethnic and organizational cultures.

9.3– Career and Technical Education Career Cluster: Information Technology (IT)
• 9.3.12.IT.11: Demonstrate knowledge of the hardware components associated with information systems.
• 9.3.12.IT-SUP.9: Employ technical writing and documentation skills in support of an information system.
Pathway: Programming & Software Development (IT-PRG)
• 9.3.12.IT-PRG.4: Demonstrate the effective use of software development tools to develop software applications.
• 9.3.12.IT-PRG.5: Apply an appropriate software development process to design a software application.
• 9.3.12.IT-PRG.6: Program a computer application using the appropriate programming language.
• 9.3.12.IT-PRG.7: Demonstrate software testing procedures to ensure quality products.

English Language Arts
• Journal writing • Close reading of industry-related content • Create a brochure for a specific industry • Keep a running word wall of industry vocabulary Social Studies • Research the history of a given industry/profession • Research prominent historical individuals in a given industry/profession • Use historical references to solve problems
World Language • Translate industry-content • Create a translated index of industry vocabulary • Generate a translated list of words and phrases related to information technology

Math • Compare and contrast use of equations and variables in algebra and programming. • Program graphics and use the properties of geometric shapes • Compare the computer graphic coordinate system with the Cartesian coordinate plane in math • Compare probability and the use of random numbers in computer programming. • Track and track various data, such as industry's impact on the GDP, career opportunities or among of individuals currently occupying careers

Fine & Performing Arts • Create a poster recruiting young people to focus their studies on a career in Information Technology

Science • Research the environmental impact of a given career or industry • Research latest developments in Information technology • Investigate applicable-careers in STEM fields

Unit 04: Data Structures

Content Area: **Sample Content Area**
Course(s):
Time Period: **1st Marking Period**
Length: **22 days**
Status: **Published**

Summary of the Unit

This unit delves into the world of data structures, the fundamental tools for organizing and storing data efficiently in Python. Students will explore a variety of built-in data structures like lists, tuples, dictionaries, and sets. They will learn how to create, manipulate, and access elements within these structures, understanding the unique characteristics and use cases for each. Through hands-on exercises and projects, students will gain proficiency in working with data structures to solve real-world problems. They will learn to choose the appropriate data structure for a given task, manipulate data efficiently, and leverage the power of Python's built-in functions and methods to streamline their code. By the end of the unit, students will have a solid foundation in data structures, enabling them to build more complex and data-driven applications in their future programming endeavors.

Enduring Understandings

- Organized Data: Data structures provide efficient and organized ways to store and manipulate collections of related data in a program.
- Choice of Structure: Different data structures (lists, tuples, dictionaries, sets) are suited for different types of data and use cases, impacting how the data can be accessed and modified.
- Data Manipulation: Understanding the methods and operations specific to each data structure is crucial for effectively manipulating and retrieving data within a program.
- Data Relationships: Data structures enable the representation of relationships between data elements, such as ordering, key-value pairs, or membership.
- Real-World Applications: Data structures are fundamental to various real-world applications, from simple lists and dictionaries to complex databases and algorithms.

Essential Questions

- Why do we need different data structures to organize and store information in Python?
- What are the key differences between lists, tuples, dictionaries, and sets, and when should we use each one?
- How can we create, access, modify, and delete elements within different data structures in Python?
- How do data structures help us model relationships between data elements, such as ordering, key-value pairs, or membership?
- What are the common operations and methods we can perform on different data structures, and how do they affect the data?

- How can we use data structures to solve real-world problems involving organizing, searching, sorting, or filtering data?
- What are some examples of how data structures are used in various applications, such as web development, data analysis, or game development?
- How can we choose the most appropriate data structure for a given task to optimize our code for efficiency and readability?

Summative Assessment and/or Summative Criteria

Option 1: Project-Based Assessment

Students will create a Python program that demonstrates their understanding of data structures. The project should incorporate the following elements:

- Multiple Data Structures: The program should use at least two different data structures (lists, tuples, dictionaries, sets) in a meaningful way.
- Data Manipulation: The program should demonstrate the ability to create, access, modify, and/or delete elements within the data structures.
- Problem Solving: The program should solve a real-world problem or simulate a scenario where data structures are essential for organizing and manipulating information.
- Code Organization: The code should be well-organized, with clear comments explaining the purpose of each section and appropriate variable names.

Project Examples:

- Inventory Management System: A program that tracks inventory items, their quantities, and prices using lists or dictionaries.
- Contact Book: A program that stores contact information (name, phone number, email) using dictionaries or lists of dictionaries.
- Flashcard App: A program that uses dictionaries to store vocabulary words and their definitions, allowing users to practice and test themselves.
- Data Analysis Tool: A program that reads data from a file (e.g., CSV) and uses lists or dictionaries to store and analyze the data.

Resources

- Replit online ID
- Project STEM curriculum Python curriculumE
- Python API (https://docs.python.org/3.12/)

Unit Plan

Topic/Selection Timeframe	General Objectives	Instructional Activities	Benchmarks/Assessments	Standards
Lists (2 days)	SWBAT ○ Be able to define lists and explain their usefulness in programming ○ Be able to declare and add data to lists in Python ○ Be able to define the terms “element”, “index”, and "initializer list" as they relate to lists ○ Work out which index relates to which item in a list ○ Write code to access elements at given indices in a list	○ Teacher will introduce data structures and how they are used programming structures ○ TW introduce lists as an ordered data structure ○ TW introduce the concept of index numbers and square brackets []. ○ SW work through list practice activities	Project Stem Coding Activities Section 8.2 and 8.3	CS.9-12.AP CS.9-12.DA CS.9-12.IC

List functions (2 days)	SWBAT ○ Be able to define a parameter and create a call your own subprograms with parameters ○ Be able to define a subprogram that uses optional parameters	○ Teacher will introduce methods built into the list object ○ Students will be able to use the built list methods to compute common list operations ○ Students will complete Project Stem Activities	Project Stem Coding Activities Section 8.5	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Lab: Data Analyzer (2 days)	SWBAT ○ demonstrate knowledge of list and their functions by completing assignment	○ Students will complete data analyzer assignment	Data Analyzer: Students will write a program that will analyze of list of random numbers (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Tuples (2 days)	SWBAT ○ Demonstate the ability to design and implement methods of varying complexity	○ Teacher will introduce second data structure: Tuple ○ Teacher will explain Tuple as an ordered list the is immutable ○ SW complete tuple practice assignments (resources)	Tuple practice assignment (resourcess)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Lab: Secret code generator (3 days)	SWBAT ○ Demonstrate knowledge of tuples by creating a secret code generator assignment	○ Students will complete secret code generator assignment	Students will complete assignment that will use a tuple to change a string input into a numeric sequence (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Dictionary (2 day)	SWBAT ○ Create and manipulate the dictionary data structrue ○ define key, value pairs in a dictionary ○ Use built in methods to add, remove and alter data stored in a dictionary	○ Teacher will introduce dictionaries as unordered list ○ Teacher will discuss the key value pair and how its used access elements of the	Project Stem Coding Activities Section 12.1 and 12.2	CS.9-12.AP CS.9-12.DA CS.9-12.IC

		dictionary ○ Teacher will discuss dictionary methods ○ Students will complete project stem activities		
Sets (1 day)	SWBAT ○ Create and use sets in a Python project ○ Differentiate between all 4 data structures ○ Use the appropriate structure based on needs of the Python	○ Teacher will introduce a set as a immutable unindexed data structure ○ Teacher will discuss how to create a set ○ Teacher will discuss methods to manipulate sets ○ Students will complete set practice assignment	Set practice assignment (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Using for loops with data structures (3 days)	SWBAT ○ Create for loops to access elements of the different	• Teacher will discuss for loops and demonstrate how	Project Stem Coding Activities Section 12.3 and 8.4	CS.9-12.AP CS.9-12.DA

	data structure elements	the work with the 3 different data structures • Students will complete project stem activities		CS.9-12.IC
Lab: Trivia Game (4 days)	Students will • Demonstrate knowledge of data structures by completing the trivia game	Students will • complete trivia game assignment using lists, tuples and dictionaries	Trivia Game: Develop a trivia game of a chosen theme using the diffeen data structures to represent answer key and reward choices(resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Unit Exam (1 day)	SWBAT • Demonstrate knowledge of loops and condtitional statements by completing written assessememts	• Teacher will administer written unit examcovering topics from Unit 4	Unit Exam	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Standards

12.9.3.IT-PRG Programming & Software Development

CS.9-12.8.1.12.AP.1 Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.5 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.CS.2 Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3 Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2 Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.3 Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4 Explain the relationship between binary numbers and the storage and use of data in a computing device.
CS.9-12.8.1.12.DA.5 Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.2 Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.2.12.ED.1 Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
CS.9-12.8.2.12.ED.5 Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.ITH.3 Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.
CS.9-12.DA Data & Analysis
Successful troubleshooting of complex problems involves multiple approaches including research, analysis, reflection, interaction with peers, and drawing on past experiences.
Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Suggested Modifications for Special Education, ELL and Gifted Students

*Consistent with individual plans, when appropriate.

Suggested Technological Innovations/Use

- Students should use online resources to research topics covered
- Students will use java IDE to test and execute code
- Students will use Java API online resource

Cross Curricular/Career Readiness, Life Literacies and Key Skills Practice

9.1 21st Century Life and Career Skills: All students will demonstrate the creative, critical thinking, collaboration, and problem-solving skills needed to function successfully as both global citizens and workers in diverse ethnic and organizational cultures.

9.3– Career and Technical Education Career Cluster: Information Technology (IT)
• 9.3.12.IT.11: Demonstrate knowledge of the hardware components associated with information systems.
• 9.3.12.IT-SUP.9: Employ technical writing and documentation skills in support of an information system.
Pathway: Programming & Software Development (IT-PRG)
• 9.3.12.IT-PRG.4: Demonstrate the effective use of software development tools to develop software applications.
• 9.3.12.IT-PRG.5: Apply an appropriate software development process to design a software application.
• 9.3.12.IT-PRG.6: Program a computer application using the appropriate programming language.
• 9.3.12.IT-PRG.7: Demonstrate software testing procedures to ensure quality products.

English Language Arts
• Journal writing • Close reading of industry-related content • Create a brochure for a specific industry • Keep a running word wall of industry vocabulary Social Studies • Research the history of a given industry/profession • Research prominent historical individuals in a given industry/profession • Use historical references to solve problems
World Language • Translate industry-content • Create a translated index of industry vocabulary • Generate a translated list of words and phrases related to information technology

Math • Compare and contrast use of equations and variables in algebra and programming. • Program graphics and use the properties of geometric shapes • Compare the computer graphic coordinate system with the Cartesian coordinate plane in math • Compare probability and the use of random numbers in computer programming. • Track and track various data, such as industry's impact on the GDP, career opportunities or among of individuals currently occupying careers

Fine & Performing Arts • Create a poster recruiting young people to focus their studies on a career in Information Technology

Science • Research the environmental impact of a given career or industry • Research latest developments in Information technology • Investigate applicable-careers in STEM fields

Unit 05: File I/O

Content Area: **Sample Content Area**
Course(s):
Time Period: **1st Marking Period**
Length: **19 days**
Status: **Published**

Summary of the Unit

This unit introduces the essential concept of file input/output (I/O) in Python, empowering students to interact with external data stored in files. Students will learn how to open, read, write, and close files in different modes, enabling them to both load data into their programs and save results for later use. They will explore techniques for working with various file formats, such as text files (`.txt`) and comma-separated value files (`.csv`), and learn how to handle errors that may arise during file operations. Through practical exercises and projects, students will gain hands-on experience in reading data from files to perform analysis, manipulating file contents, and writing data to files for persistent storage. By the end of the unit, students will have mastered the fundamentals of file I/O, equipping them with a valuable skill for working with real-world data and building more robust and data-driven applications.

Enduring Understandings

- Persistent Data: Files provide a way to store data persistently, allowing programs to access and manipulate information beyond the duration of a single execution.
- Data Interaction: Programs can interact with data stored in files, enabling them to read input, process it, and write output in a structured manner.
- File Formats: Different file formats (e.g., .txt, .csv) dictate how data is organized and require specific techniques for reading and writing.
- Error Handling: File operations can fail due to various reasons (e.g., file not found, permission errors), and robust programs must include error handling mechanisms to gracefully manage such situations.
- Data-Driven Applications: File I/O is essential for building data-driven applications that analyze, process, and transform information from external sources.

Essential Questions

- How can we use Python to read data from external files and store it in variables for further processing?
- What are the different modes for opening files (e.g., read, write, append), and when should we use each mode?
- How do we handle different file formats (e.g., .txt, .csv) in Python, and what are the best practices for reading and writing structured data?
- Why is error handling important in file I/O operations, and how can we use `try-except` blocks to gracefully handle exceptions?

- What are some common file-related errors (e.g., FileNotFoundError, PermissionError), and how can we address them in our code?
- How can we leverage file I/O to build applications that interact with external data sources, such as analyzing data from log files or generating reports?
- What are the ethical considerations of working with potentially sensitive or private data stored in files?

Summative Assessment and/or Summative Criteria

Students will complete project to generate ticketing system using all concepts learned in the course

Resources

Online Tutorials and Guides:

- Python File Handling (Programiz): A comprehensive guide covering file opening, reading, writing, closing, and error handling.
  - URL: https://www.programiz.com/python-programming/file-operation
- Python File I/O (W3Schools): A beginner-friendly tutorial with examples and exercises on file operations.
  - URL: https://www.w3schools.com/python/python_file_handling.asp
- Working with CSV Files in Python (Real Python): A detailed tutorial on reading and writing CSV files using Python's `csv` module.
  - URL: https://realpython.com/python-csv/
- Python Examples (Programiz): A collection of Python examples, including file I/O operations, that students can practice with.
  - URL: https://www.programiz.com/python-programming/examples

Books:

- "Automate the Boring Stuff with Python" by Al Sweigart: A popular book that covers file I/O and automation tasks in a practical way.
- "Python Crash Course" by Eric Matthes: A beginner-friendly book with a chapter on file I/O and projects that use file data.

Videos and Courses:

- Python Tutorial for Beginners (Codecademy): An interactive online course that covers file I/O basics.
  - URL: https://www.codecademy.com/learn/learn-python-3
- Python File Handling Tutorial (YouTube - freeCodeCamp): A video tutorial explaining file operations in Python.

- URL: Search "Python File Handling Tutorial freeCodeCamp" on YouTube.

Additional Resources:

- Python Documentation on File I/O: The official Python documentation provides in-depth reference on file operations and the `io` module.
  - URL: https://docs.python.org/3/library/io.html

Project Ideas and Datasets:

- Analyze Log Files: Provide log files and have students extract specific information or identify patterns.
- Process CSV Data: Give students CSV files containing weather data, stock market data, or other relevant datasets for analysis.
- Create a Simple Database: Students can create a program to store and retrieve data from a text file, mimicking a basic database.

Unit Plan

Topic/Selection Timeframe	General Objectives	Instructional Activities	Benchmarks/Assessments	Standards
What is file i/o (1 day)	SWBAT ○ Define and explain file I/O. ○ Differentiate file types (e.g., text, binary). ○ Identify input sources and output destinations in computing. ○ Understand the purpose and significance of file input. ○ Understand the purpose and significance of file output. ○ Discuss real-world applications of file I/O.	○ Teacher will discuss console i/o and how much of daa fed to programs do not come from console but from files. ○ Class will discuss examples of where this is evident	Dataset discussion and search results (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

		○ Students will research data sets in the real world and how they can be used to create feed programs that can solve real issues		

Reading a character from a file (2 days)	SWBAT ○ Understand the importance of reading files and the role it plays in file handling tasks. ○ Use the open() function to open a file in read mode and the close() method to close the file properly. ○ Read the entire contents of a file using the read() method and store them in a variable ○ Handle common file exceptions, such as FileNotFoundError, when attempting to read a file. ○ Understand the purpose and usage of the readline() method in Python for reading lines from a file. ○ Demonstrate how to open a file and use readline() to read a single line at a time.	○ Teacher will introduce the concept of getting data from files ○ Teacher will introoduce the readlines method and how txt files can be accessed from a python program ○ Teacher will demonstrate file input ○ Students will complete assignments	File Practice assignment 1 (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Reading all lines from a file (3 day)	SWBAT: ○ Understand the purpose and usage of the readlines() method in Python. ○ Read and retrieve multiple lines from a file using readlines( ). ○ Iterate through the lines obtained from readlines() and perform operations on each line. ○ Apply different manipulations and processing techniques to the lines read from a file. ○ Recognize the advantages and use cases of using readlines( ) in file handling scenarios.	○ Teacher will demonstrate how Python handles white space ○ Teacher will demo how to use loops to read the complete file based on the content ○ Students will work on practice assignments	File Practice Assignment 2 (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Writing to a file(2 days)	SWBAT ○ Understand the purpose and importance of writing to files in programming. ○ Use the "w" mode to overwrite the contents of a file. ○ Use the "a" mode to append new data to the end of a file. ○ Demonstrate the ability to write text and data to a file using appropriate file handling techniques. ○ Be able to generate random numbers with the random function	○ Teacher will demo writing files and the different specifications that come with creating the link to the output file. ○ SW complete practice assignments	File practice assignment 3 (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
File Practice Project (3 days)	SWBAT ○ Demonstrate knowledge of File i/o by updating previous project	○ Students will update past project to read and write to file instead of previous console i/o	File i/o: update past project to include file i/o elements (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC
Unit 5 Exam (1 day)	Students will demonstrate knowledge of python input, output and mathematical operations by	Students will take unit 1 Exam	Unit Exam	CS.9-12.AP CS.9-12.DA

	completing Unit 1 Exam			CS.9-12.IC
Final Project (7 days)	Students will demonstrate knowledge of python by completing final project based on description	Students will complete project	Final Project (resources)	CS.9-12.AP CS.9-12.DA CS.9-12.IC

Standards

12.9.3.IT-PRG Programming & Software Development
CS.9-12.8.1.12.AP.1 Design algorithms to solve computational problems using a combination of original and existing algorithms.
CS.9-12.8.1.12.AP.5 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
CS.9-12.8.1.12.AP.6 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
CS.9-12.8.1.12.CS.2 Model interactions between application software, system software, and hardware.
CS.9-12.8.1.12.CS.3 Compare the functions of application software, system software, and hardware.
CS.9-12.8.1.12.CS.4 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
CS.9-12.8.1.12.DA.2 Describe the trade-offs in how and where data is organized and stored.
CS.9-12.8.1.12.DA.3 Translate between decimal numbers and binary numbers.
CS.9-12.8.1.12.DA.4 Explain the relationship between binary numbers and the storage and use of data in a computing device.
CS.9-12.8.1.12.DA.5 Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena.
CS.9-12.8.1.12.IC.2 Test and refine computational artifacts to reduce bias and equity deficits.
CS.9-12.8.2.12.ED.1 Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
CS.9-12.8.2.12.ED.5 Evaluate the effectiveness of a product or system based on factors that are related to its requirements, specifications, and constraints (e.g., safety, reliability, economic considerations, quality control, environmental concerns, manufacturability, maintenance and repair, ergonomics).
CS.9-12.8.2.12.ITH.3 Analyze the impact that globalization, social media, and access to open source technologies has had on innovation and on a society's economy, politics, and culture.
CS.9-12.DA Data & Analysis

Successful troubleshooting of complex problems involves multiple approaches including research, analysis, reflection, interaction with peers, and drawing on past experiences.
Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

Suggested Modifications for Special Education, ELL and Gifted Students

Special Education Students:

- Visual Supports: Use diagrams and flowcharts to illustrate the process of file I/O, including opening, reading, writing, and closing files.
- Simplified Instructions: Break down tasks into smaller, manageable steps with clear and concise instructions.
- Hands-on Activities: Provide opportunities for hands-on practice, such as creating and manipulating files with different types of data.
- Alternative Assessment: Allow students to demonstrate understanding through alternative formats like oral presentations, visual representations, or simplified coding tasks.
- Assistive Technologies: Utilize text-to-speech or speech-to-text tools to support students with reading or writing difficulties.
- Differentiated Content: Modify or reduce the complexity of file formats or tasks based on individual student needs.

English Language Learners (ELLs):

- Visual Aids: Incorporate visual aids like pictures, diagrams, and real-world examples to reinforce vocabulary and concepts related to file I/O.
- Simplified Language: Provide instructions and explanations in clear, concise language, avoiding jargon and technical terms.
- Multilingual Resources: Offer bilingual dictionaries or translation tools to support vocabulary acquisition.
- Scaffolding of Writing: Provide sentence frames or templates to guide students in writing code or explanations.
- Peer Support: Pair ELLs with native English speakers for collaborative learning and language support.
- Code Examples with Comments: Provide code examples with comments in both English and the student's native language.

Gifted Students:

- Challenging Tasks: Offer more complex file formats (e.g., JSON, XML) or data manipulation tasks.
- Advanced Topics: Introduce advanced topics like file compression, encryption, or working with binary files.
- Independent Research: Encourage independent research on file I/O concepts or related topics like data parsing and analysis.
- Real-World Applications: Explore real-world applications of file I/O, such as data analysis, automation, or web scraping.
- Mentoring Opportunities: Connect gifted students with mentors who can provide guidance and support in their advanced explorations.

- Open-Ended Projects: Allow for more creativity and exploration in project design, encouraging students to apply file I/O to their own areas of interest.

Suggested Technological Innovations/Use

- Students should use online resources to research topics covered
- Students will use java IDE to test and execute code
- Students will use Java API online resource

Cross Curricular/Career Readiness, Life Literacies and Key Skills Practice

9.1 21st Century Life and Career Skills: All students will demonstrate the creative, critical thinking, collaboration, and problem-solving skills needed to function successfully as both global citizens and workers in diverse ethnic and organizational cultures.

9.3– Career and Technical Education Career Cluster: Information Technology (IT)
• 9.3.12.IT.11: Demonstrate knowledge of the hardware components associated with information systems.
• 9.3.12.IT-SUP.9: Employ technical writing and documentation skills in support of an information system.
Pathway: Programming & Software Development (IT-PRG)
• 9.3.12.IT-PRG.4: Demonstrate the effective use of software development tools to develop software applications.
• 9.3.12.IT-PRG.5: Apply an appropriate software development process to design a software application.
• 9.3.12.IT-PRG.6: Program a computer application using the appropriate programming language.
• 9.3.12.IT-PRG.7: Demonstrate software testing procedures to ensure quality products.

English Language Arts
• Journal writing • Close reading of industry-related content • Create a brochure for a specific industry • Keep a running word wall of industry vocabulary Social Studies • Research the history of a given industry/profession • Research prominent historical individuals in a given industry/profession • Use historical references to solve problems
World Language • Translate industry-content • Create a translated index of industry vocabulary • Generate a translated list of words and phrases related to information technology

Math • Compare and contrast use of equations and variables in algebra and programming. • Program graphics and use the properties of geometric shapes • Compare the computer graphic coordinate system with the Cartesian coordinate plane in math • Compare probability and the use of random numbers in computer programming. • Track and track various data, such as industry's impact on the GDP, career opportunities or among of individuals currently occupying careers

Fine & Performing Arts • Create a poster recruiting young people to focus their studies on a career in Information Technology

Science • Research the environmental impact of a given career or industry • Research latest developments in Information technology • Investigate applicable-careers in STEM fields