00 Into to Python Pacing Guide

Content Area: **TEMPLATE**
Course(s):
Time Period: **Semester**
Length: **20 Weeks**
Status: **Published**

OBJECTIVES, ESSENTIAL QUESTIONS, ENDURING UNDERSTANDINGS

Objectives

- Students will be able to use basic python commands to draw shapes with different colors and sizes.
- Students will be able to represent a complex image using geometric shapes and colors.
- Students will be able to fill in a chart to explain their abstraction process.
- Students will be able to critique their own program for opportunities for improvement.
- Students will be able to organize a set of instructions into a function.
- Students will be able to use user interaction to alter components of a canvas.
- Students will be able to explain what a parameter does and why a function might need one?
- Students will be able to compare and contrast mouse interactions.
- Students will be able to determine if an algorithm will run inside conditions.
- Students will be able to write a helper function to simplify a long function.
- Students will be able to compare and contrast if, if-elif-else, and multiple if's.
- Students will be able to use the keyboard to interact with a program.
- Students will be able to compare and contrast nested conditionals and compound conditionals.
- Students will be able to explain the similarities and differences between && and ||.
- Students will be able to generate and test code to create complex shapes.
- Students will be able to group multiple shapes together to apply a method to each item in the group.
- Students will be able to compare and contrast how groups of shapes and individual shapes.
- Students will be able to generate and test code to apply methods to groups of shapes.
- Students will be able to explain onStep is used to create motion.
- Students will be able to generate and test code to simulate motion on the screen.
- Students will be able to display curved shapes on a canvas.
- Students will be able to display arrows on the canvas.
- Students will be able to determine the scope of a variable.
- Students will be able to compare and contrast helper functions and helper variables.
- Students will be able to compare and contrast for loop and onStep.
- Students will be able to compare and contrast /, //, and %.
- Students will be able to explain angle To and PointInDir.
- Students will be able to generate and test code that creates a random number between two values.
- Students will be able to generate and test code that uses nested loops to create a 2D grid.

- Students will be able to use the world properties to create and add objects to a list variable.
- Students will be able to generate an instruction to move one after the other using the same motion
- Students will be able to generate an instruction to move all objects in a list simultaneously
- Students will be able to generate and test code that incorporates a list search to randomly select an item in a list to complete a specific set of instructions.

Essential Questions

- Why are algorithms essential to programming?
- How can programs be used for creative expression?
- How can computing extend traditional forms of human expression and experience?
- How are systems of interacting modules necessary for the management of complex tasks?
- How do functions contribute to abstraction?
- How do programmers choose the correct data structure to manage a program's complexity?
- How do modules manage the complexity of a program?
- How does a programmer decide between readability, implementation, and performance?
- Why do programmers consider security, dependability, and accessibility when designing and evolving a program.
- Why are multiple approaches need when debugging a program?
- What are the advantages and disadvantages of using nested conditions instead of compound conditionals?
- How does a programmer use interaction from the user to control the flow of programming?
- How does a personal experience and strengths affect the development and testing of a program?
- Why are loops used in programming?
- How are nested loops used in daily life?
- Why do programs communicate responses to themselves?
- What are the advantages and disadvantages of using a for loop over a while loop?
- Why do we often purposely repeat tasks in our lives?
- How does the role of coordinate geometry assist in perform complicated tasks in Python?
- Why are mathematical and logical concepts fundamental to computer programming?
- How is the way data is organized and stored affect reliability, accessibility, privacy, and speed?
- Why are organization structures important to programming?
- How is a storyboard incorporated into creating a computer program and everyday tasks?
- How do computers use simulation and modeling to represent real-world phenomena?
- Why is randomness important but unachievable inside a computer?

- In what ways do simulation and modeling extend our knowledge and benefit society?

Enduring Understanding

- A variety of abstractions built upon binary sequences can be used to represent all digital data.
- String variables are different than numeric variables; there are times when you need to convert a string variable to a numeric variable so as to perform mathematical operations.
- Instructions, control structures, functions, and expressions are used to create a program.
- Methods can be world or class level.
- Parameters act as a placeholder to communicate values from one method to another.
- Parameters allow a method to be used in more than one situation.
- Programming breaks down complex tasks into multiple simple actions.
- Programming uses mathematical and logical concepts.
- There are trade-offs when representing information as digital data.
- Programming uses mathematical and logical concepts to determine which sequence of code is executed.
- Python utilizes both mouse and keyboard for user interaction with a program.
- Properties of objects are not only defined within Python but can be customized to create new unique properties
- Python using repetitive code and step functions to simulate motion on a canvas.
- Python uses the concept of stop motion to simulate movement on the screen.
- Iteration is a layer of abstraction that simplifies code and makes it more readable
- Variables are only accessible within their scope
- The life and scope of different types of variables are important for efficiency in a computer program.
- Simple mathematical operations can represent complex cases in a program.
- Foundations of coordinate geometry are used when comparing the locations of objects on a canvas.
- The organization of items eliminates time spent writing a program.
- Programmers can organize the data structures used in their daily lives called a list.
- Parameters are necessary when lists contain objects that are to perform an action.
- Multiple levels of abstraction are used to write programs
- Programs can be developed to solve problems
- Programs are developed, maintained, and used by people for different purposes.
- Computing enhances communication, interaction, and cognition.

General Overview, Course Description or Course Philosophy

The course is designed to give students an introduction to the programming language python and developing abstraction skills. Students will have the opportunity to foster their ability to develop and apply computational problem-solving skills. This course utilizes graphical

problems that allow for multiple solutions and provides visual cues while debugging. Students will design creative solutions to address real-world problems. Students will engage not only independently but will work collaboratively to navigate the dynamic digital landscape.

CONTENT AREA STANDARDS

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.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.
CS.9-12.8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
CS.9-12.8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
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.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
CS.9-12.8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
CS.9-12.AP	Algorithms & Programming
	Individuals evaluate and select algorithms based on performance, reusability, and ease of implementation.
	Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. Modules allow for better management of complex tasks.
	Complex programs are developed, tested, and analyzed by teams drawing on the members' diverse strengths using a variety of resources, libraries, and tools.
	Trade-offs related to implementation, readability, and program performance are considered when selecting and combining control structures.

RELATED STANDARDS (Technology, 21st Century Life & Careers, ELA Companion Standards are Required)

LA.K-12.NJSLSA.L1	Demonstrate command of the conventions of standard English grammar and usage when writing or speaking.
LA.K-12.NJSLSA.L6	Acquire and use accurately a range of general academic and domain-specific words and phrases sufficient for reading, writing, speaking, and listening at the college and career readiness level; demonstrate independence in gathering vocabulary knowledge when encountering an unknown term important to comprehension or expression.
LA.K-12.NJSLSA.SL1	Prepare for and participate effectively in a range of conversations and collaborations with diverse partners, building on others' ideas and expressing their own clearly and

	persuasively.
LA.K-12.NJSLSA.SL2	Integrate and evaluate information presented in diverse media and formats, including visually, quantitatively, and orally.
MA.K-12.1	Make sense of problems and persevere in solving them.
MA.K-12.2	Reason abstractly and quantitatively.
MA.K-12.3	Construct viable arguments and critique the reasoning of others.
MA.K-12.4	Model with mathematics.
MA.K-12.5	Use appropriate tools strategically.
MA.K-12.6	Attend to precision.
MA.K-12.7	Look for and make use of structure.
MA.K-12.8	Look for and express regularity in repeated reasoning.
PFL.9.1.K12.P.1	Act as a responsible and contributing community members and employee.
PFL.9.1.K12.P.4	Demonstrate creativity and innovation.
PFL.9.1.K12.P.5	Utilize critical thinking to make sense of problems and persevere in solving them.
PFL.9.1.K12.P.6	Model integrity, ethical leadership and effective management.
PFL.9.1.K12.P.8	Use technology to enhance productivity increase collaboration and communicate effectively.
PFL.9.1.K12.P.9	Work productively in teams while using cultural/global competence.
TECH.9.4.12.CI.1	Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
TECH.9.4.12.CT.1	Identify problem-solving strategies used in the development of an innovative product or practice (e.g., 1.1.12acc.C1b, 2.2.12.PF.3).
TECH.9.4.12.CT.2	Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).
TECH.9.4.12.DC.1	Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content (e.g., 6.1.12.CivicsPR.16.a).

EVIDENCE OF LEARNING

Formative Assessments

- Checkpoints in each section
- Guided Practice
- Independent Practice
- Checklists
- Class Discussion
- Exit Tickets
- Rubrics
- Teacher Observation
- Exit/Entrance Tickets

Summative Assessments

- Quizzes
- Unit Tests
- Unit Creative Tasks
- Mid Course Project
- End of Course Project

RESOURCES (Instructional, Supplemental, Intervention Materials)

Example Creative Tasks

INTERDISCIPLINARY CONNECTIONS

Interdisciplinary connections are frequently addressed through examples and practice problems whereby creating solutions that draw from cultures around the world, athletics, mathematics and geography. Examples can be found in topic specific examples, practice exercises, guided projects and digital resources.

ACCOMMODATIONS & MODIFICATIONS FOR SUBGROUPS

See link to Accommodations & Modifications document in course folder.