Unit 1 Intro to Python with Tracy the Turtle

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **30 days**
Status: **Published**

Summary

This unit introduces students to the foundations of computer programming using Python within the CodeHS "Tracy the Turtle" environment, progressing from a student's very first program through loops, functions, and independent project design. Students begin by defining what programming is, writing their first sequential Tracy programs, and learning to distinguish low-level from high-level and compiled from interpreted languages. They then learn to navigate Tracy's coordinate grid using positioning commands (penup, pendown, goto, setposition, setx, sety) and turning commands (left, right, setheading/seth), and are introduced to strategies for checking and testing their own code since autograders cannot verify what appears on the canvas. Students next learn to use for loops to repeat code efficiently and combine loops with angle-based turning to build increasingly complex geometric designs. In the unit's final module, students learn to write comments and follow naming conventions that make code readable, define and call functions, add color, pensize, fill, and text to their graphics, and apply Top-Down Design to break a complex drawing into smaller, reusable functions. Throughout the unit, students build and extend a multi-part Etch A Sketch project (Parts 1–3), applying each new skill as it is introduced. The unit concludes with a five-day culminating project in which students independently plan and code an original graphic of their choosing, applying the full toolkit of commands and concepts developed across the unit.

Revision Date: July 2026

Essential Questions

- What is the difference between giving instructions to a computer and giving instructions to a person or an animal?
- Why do multiple programming languages exist, and what factors might lead a programmer to choose one language over another?
- How does a coordinate system allow a programmer to precisely control and predict what a program will do?
- How do loops and functions help programmers write more efficient, reusable, and organized code?
- What strategies help a programmer plan, test, troubleshoot, and clearly communicate their code to others?

Enduring Understandings

- A computer program is a precise, ordered sequence of instructions; the smallest change in wording, order, spelling, capitalization, or punctuation can change how a program runs or cause it to fail.
- Programming languages and coordinate systems are tools that programmers select and apply deliberately based on the needs of a given task, audience, or device.
- Breaking a complex problem into smaller, well-organized, and well-documented parts, using loops, functions, comments, and Top-Down Design, makes programs easier to write, test, reuse, and share with others.

Objectives

Students Will Know

- Programming/coding is giving explicit, sequential commands to a computer using a language it can interpret.
- Basic Tracy commands, including forward(), backward(), circle(), left(), right(), and setheading()/seth(), and how indentation, capitalization, and punctuation affect whether a program runs correctly.
- The difference between low-level and high-level programming languages, and between compiled and interpreted programming languages, with examples of each.
- The layout of Tracy's coordinate grid world and the purpose of the penup(), pendown(), goto(), setposition(), setx(), and sety() commands.
- The syntax and purpose of a for loop (for i in range(n):), and how loops reduce repeated code.
- How to turn Tracy using specific angle values, alone and in combination with loops, to create diagonal lines and complex shapes.
- The purpose and syntax of single- and multi-line comments, and best-practice guidelines for naming variables and functions.
- How to define and call a function, and why functions make programs more organized, readable, and reusable.
- The purpose of the color(), pensize(), begin_fill(), end_fill(), and extended circle() commands, and the write() command for adding text labels to a canvas.
- The process of Top-Down Design: breaking a large problem into smaller functions to avoid repeated code and improve program readability.

Students Will be Skilled At

- Writing, running, testing, and debugging Tracy programs of increasing complexity, using multiple strategies (autograder feedback, peer review, output/result-world comparison) to check their own work.
- Classifying a programming language as low-level or high-level and as compiled or interpreted.
- Locating and moving Tracy to specific coordinates and directions using x- and y-values, turning commands, and heading angles.
- Writing for loops to repeat code efficiently, including loops combined with angle-based turning.
- Using comments and clear naming conventions to make code easier to read, test, and debug.
- Defining and calling functions, and applying Top-Down Design to break a complex drawing into smaller, reusable functions.
- Using color, pensize, fill, and text commands to add creative detail to a graphic.
- Planning a multi-step graphic (e.g., sketching on graph paper) before coding it, and extending a saved project (Etch A Sketch) across multiple parts of the unit.
- Independently planning and coding an original, open-ended graphic design for a culminating project.

Learning Plan

This unit is organized into three instructional modules taught in sequence, followed by a five-day culminating project, for an estimated total of 30 class periods (to be confirmed against MP1 pacing). Instruction follows a consistent structure throughout: students open each class by responding individually or in pairs to brief discussion questions, which also serve as a lesson recap and reflection tool at the end of class. Direct instruction is delivered through short CodeHS videos and slide decks, followed immediately by a comprehension quiz, a guided example walkthrough, and independent or paired coding exercises with increasing complexity. The Peer Code Review Template is available throughout the unit to structure paired feedback on any coding exercise.

Module 1: Programming Foundations & Tracy's Grid World (Lessons 1.1–1.4, approx. 4–5 days)

In Lesson 1.1, the teacher orients students to the CodeHS platform, then leads a discussion contrasting a computer's obligation to follow commands with a living thing's ability to choose whether to comply. Students watch the Intro to Tracy video, complete a comprehension quiz, walk through a worked example, and write their own first programs (Slinky, Stretched Slinky), deliberately experimenting with command order, capitalization, and indentation to see how each change affects the output, and complete a sequencing exercise using trace tables.

In Lesson 1.2, instruction shifts to understanding the landscape of programming languages: after the introductory video and quiz, students complete a Programming Language Hierarchy activity, work through free-response exercises on the popularity and history of programming languages, and build a student-created

timeline, connecting compiled vs. interpreted languages to the analogy of a recorded video versus a live guide.

In Lesson 1.3 (the longest lesson in the module), students learn to describe Tracy's location using x- and y-coordinates on a 400 x 400 pixel plane centered at the origin. After the video, quiz, and coordinates-practice exercises, students plan and code the Shorter Dashed Line and Caterpillar exercises using the penup(), pendown(), goto(), setposition(), setx(), and sety() commands, then begin Part 1 of the Etch A Sketch project — sketched on graph paper first and saved to a Sandbox program so it can be extended in later modules.

Lesson 1.4 closes the module with a brief warm-up review followed by the independently completed 10-question Tracy's World Quiz covering all of Module 1's vocabulary and commands.

Module 2: Turning, Loops & Angles (Lessons 2.1–2.5, approx. 5–10 days)

Lesson 2.1 is a short lesson (often combined with 2.2) that introduces students to methods for checking their own work, since CodeHS autograders cannot verify what actually appears on the canvas. After the video, students review the Structured Peer Review resource, complete the 4 Horizontal Circles exercise, and respond to a reflection free-response, discussing that there is often more than one command (goto, setposition, setx, sety) that can achieve the same result.

In Lesson 2.2, students learn the left(), right(), and setheading()/seth() commands to turn Tracy to face any direction. After the video, quiz, and example walkthrough, students complete a sequence of exercises of increasing difficulty — Square, X and Y Axes, Rectangle, 4 Columns, and Tunneling — building toward more complex multi-shape graphics.

In Lesson 2.3, students are introduced to for loops (for i in range(n):) as a way to repeat code without rewriting it, along with the bgcolor() command to change the canvas color. After the video, quiz, and example walkthrough, students rebuild earlier shapes more efficiently using loops (Square Using Loops, Dotted Line, Row of Circles, Color Changing Staircase, 4 Columns 2.0) and may extend their learning with the optional Looping Badge.

In Lesson 2.4, students combine turning at specific angles with for loops to draw diagonal lines and more advanced shapes. After the video, quiz, and example walkthrough, students complete the Asterisk, Four Circles, Hexagon, X Marks the Spot, and Circle Pyramid exercises, then complete Part 2 of the Etch A Sketch project, extending their saved Part 1 program using loops and angle-based turning.

Lesson 2.5 closes the module with an independently completed 10-question end-of-unit quiz covering for loops, left/right, setheading/seth, speed, and bgcolor.

Module 3: Designing & Communicating Solutions (Lessons 3.1–3.7, approx. 8–12 days)

In Lesson 3.1, students learn to use single- and multi-line comments to document their code, and revisit two previously completed programs (Four Circles, Circle Pyramid) to add comments explaining what each section does and why comments are useful for themselves and other readers of their code.

Lesson 3.2 is a short lesson (often combined with 3.1 or 3.3) introducing rules and guidelines for naming variables and functions so that programs remain readable and function correctly.

In Lesson 3.3, students are introduced to functions: what they are, why they are used, how to define and call them, and how to rewrite an earlier program using functions. After the video, quiz, and example walkthrough, students complete X and Y Axes with Hash Marks, Beaded Bracelet, and Shape Stack, with the optional Functions Badge available as an extension.

In Lesson 3.4, students add creative control to their graphics using the extended circle() command along with color(), pensize(), begin_fill(), and end_fill(). After the video, quiz, and example walkthrough, students complete Rainbow Octagon, Circle Square Triangle, Four Colored Triangles, Colorful Bracelet, and Kid's Shapes Toy.

In Lesson 3.5, students learn to use the write() command, along with additional attributes to control font style and placement, to add text labels to the canvas. After the video, quiz, and example walkthrough, students complete Square with Labeled Coordinates, Kid's Shapes Toy with Labels, and Baseball Diagram.

In Lesson 3.6, students are introduced to Top-Down Design — breaking a large program into smaller functions to avoid repeated code and improve readability. After the video, quiz, and example walkthrough, students complete Bubble Wrap, Bubble Wrap 2.0, and Sidewalk, then complete Part 3 of the Etch A Sketch project, extending their saved program using functions and Top-Down Design.

Lesson 3.7 closes the module with the independently completed Designing and Communicating Solutions Quiz, a 10-question assessment covering commenting, naming rules, defining and calling functions, Top-Down Design, color, pensize, begin_fill/end_fill, circle with extended parameters, and write.

Culminating Project: Open-Ended Turtle Graphics Design (5 days)

Following Module 3, students spend five class days independently planning and coding an original graphic of their own choosing, applying the full range of commands and concepts developed across the unit — coordinate positioning, turning and angles, loops, functions, comments, naming conventions, color/fill, and text. Students should sketch their design on graph paper and outline it using Top-Down Design before coding.

It is recommended that the teacher provide a rubric in advance specifying which unit concepts must be demonstrated (e.g., at least one loop, one function, one use of color or fill, appropriate comments) and close the project with a peer or class gallery walk in which students share and discuss their finished designs.

Assessment

Formative Assessment Comprehension quizzes following each instructional video (e.g., 1.1.2, 1.2.2, 1.3.2, 2.2.2, 2.3.2, 2.4.2, 3.1.2, 3.3.2, 3.4.2, 3.5.2, 3.6.2), guided example walkthroughs, scaffolded coding exercises with solution references and problem guides for teacher feedback, the Structured Peer Review resource, the 4 Horizontal Circles: Reflection free response, beginning- and end-of-class discussion questions/exit tickets, and Peer Code Review Template exchanges during coding exercises.

Summative Assessment Lesson 1.4.1, Tracy's World Quiz (compiled vs. interpreted and high- vs. low-level languages; forward, circle, backward, penup/pendown, setposition, goto, setx/sety); Lesson 2.5, End-of-Unit Quiz (for loops, left/right, setheading/seth, speed, bgcolor); and Lesson 3.7.1, Designing and Communicating Solutions Quiz (commenting, naming rules, functions, Top-Down Design, color, pensize, begin_fill/end_fill, extended circle, write) each a 10- question, independently completed multiple-choice assessment. The three-part Etch A Sketch project (Parts 1–3) also serves as a graded summative measure: students extend a single saved Sandbox program across all three modules, applying coordinate positioning in Part 1, loops and angle-based turning in Part 2, and functions and Top-Down Design in Part 3: with each part scored against the Etch A Sketch grading criteria for correct command use and code organization.

Alternative/Performance Assessment: The five-day Culminating Project, an open-ended performance assessment in which students independently design and code an original graphic, evaluated against a teacher-provided rubric covering the unit's required commands and concepts. The optional Looping Badge and Functions Badge exercises provide additional performance-based extensions for students who finish early. As an alternative to the multiple-choice quiz format, students who need it may instead complete a Code Walkthrough/Portfolio Defense: a short one-on-one or small-group conference in which the student presents two or three completed programs (e.g., Etch A Sketch, the Culminating Project) and verbally explains how their code uses coordinates, loops, functions, and Top-Down Design, allowing the teacher to assess the same content knowledge through demonstration and discussion rather than a written test. District-Level/Benchmark Assessment Unit 1

Benchmark Assessment, administered at the end of the unit following the Culminating Project. Part A is a 15–20 question multiple-choice/short-answer section drawn from all three modules, covering language classification (low-level/high-level, compiled/interpreted), coordinate and positioning commands, for loops and angle-based turning, and comments, functions, naming conventions, and Top-Down Design. Part B is a short hands-on Tracy coding task in which students code a specified multi-shape design (e.g., using goto/setposition, at least one for loop, and at least one user-defined function) within a class period, scored against a rubric for correct syntax, accurate use of coordinates/loops/functions, and one creative-detail

command (color, fill, or text).

Materials

Core instructional materials

- CodeHS platform – Unit 1: Foundations of Programming with Tracy the Turtle (Lessons 1.1–1.4, 2.1–2.5, 3.1–3.7, plus the culminating project)

Supplemental materials

- CodeHS lesson slide decks and instructional videos for each lesson's introductory activity (e.g., 1.1.1, 1.2.1, 1.3.1, 2.2.1, 2.3.1, 2.4.1, 3.1.1, 3.3.1, 3.4.1, 3.5.1, 3.6.1)
- Printed handouts: Circles; Sequencing with Trace Tables; Programming Language Hierarchy; Tracy's Grid World Examples Exploration; Coordinates; Structured Peer Review (teacher and student versions where available)
- Lesson Notes Templates, teacher and student versions, Lessons 1.1–1.3
- Peer Code Review Template and Class Discussion Notes Template
- Graph paper for planning the Etch A Sketch project and the culminating project
- Teacher-created rubric for the culminating open-ended project
- Student laptops/Chromebooks with internet access and individual CodeHS accounts

Standards

8.1 Computer Science (2020 NJSLS Computer Science & Design Thinking, Grades 6–8)

8.1.8.AP.1 – Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.

8.1.8.AP.2 – Create clearly named variables that represent different data types and perform operations on their values.

8.1.8.AP.3 – Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

8.1.8.AP.4 – Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.

8.1.8.AP.5 – Create procedures with parameters to organize code and make it easier to reuse.

8.1.8.AP.8 – Systematically test and refine programs using a range of test cases and users.

8.1.8.AP.9 – Document programs in order to make them easier to follow, test, and debug.

8.1.8.CS.1 – Recommend improvements to computing devices in order to improve the ways users interact with the devices.

9.4 Career Readiness, Life Literacies & Key Skills (2020 NJSLS)

9.4.8.CT.1 – Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem and use critical thinking skills to predict which one(s) are likely to be effective.

9.4.8.CT.2 – Develop multiple solutions to a problem and evaluate short- and long-term effects to determine the most plausible option (applied to Top-Down Design and the culminating project).

9.4.8.TL.3 – Select appropriate tools to organize and present information digitally.

Interdisciplinary: English Language Arts (NJSLS-ELA, Grade 8)

RI.8.4 – Determine the meaning of words and phrases as they are used in a text, including figurative, connotative, and technical meanings (applied to programming and computer science vocabulary introduced in this unit).

SL.8.1 – Engage effectively in a range of collaborative discussions with diverse partners on grade 8 topics, texts, and issues (applied to paired coding, peer code review, and class discussion questions).

W.8.4 – Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience (applied to writing clear code comments and naming conventions).

Integrated Accommodations and Modifications

Suggested Strategies for Modification for Computer Science