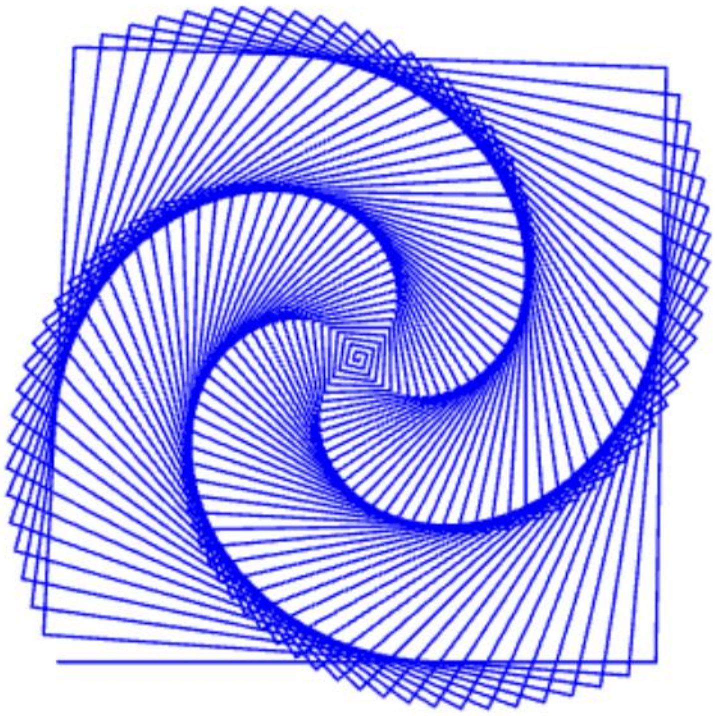
More Than Coding

TINA 1 PROGRAMMING COURSE

LESSON PLANS

FEBRUARY 28, 2016



OVERVIEW	2
EQUIPMENT AND ACCOUNTS REQUIRED; SUGGESTED AGE RANGE FOR STUDENTS AND PREREQUISITES	3
TIME REQUIRED AND SUGGESTIONS FOR COURSE DELIVERY	4
CROSS-CUTTING CONCEPTS: MATH AND ELA STANDARDS	6
NEXT GENERATION SCIENCE STANDARDS	9
VOCABULARY, LANGUAGE AND PROGRAM SUPPORTS	10
BACKGROUND BUILDING AND SUPPORT ACTIVITIES	11
DEPTH OF KNOWLEDGE (DOK) AND BLOOM'S TAXONOMY	13
ENRICHMENT, REMEDIATION AND PROGRESS MONITORING	13
ASSESSMENT	14
LESSONS: INTRODUCTION	15
SECTION 1: BASICS (INCLUING A NOTE ON CREATIVE SUITE)	16
SECTION 2: LOOPS I	31
SECTION 3: LOOPS II	40
SECTION 4: NESTED LOOPS	47
SECTION 5: VARIABLES	55

OVERVIEW

Tina 1 is a series of lessons designed to teach students computational thinking and computer programming as applied to graphics. The language itself is a simplified version of Python, which is used extensively in engineering, science and design work. The basic concepts are common to all programming languages. By the end of Tina 1, students will be able to write programs that draw complex designs using loops, nested loops, and variables.

Tina can be learned independently or under the guidance of a teacher. The course is divided into five sections, with eight levels in each section. Each level builds on the previous one and includes a tutorial and hints if students are stuck. There are links to YouTube videos that explain the steps. The online textbook, available from a drop down menu, describes the program and technical information in detail.

Although the program stands on its own, the value of the lessons is greatly enhanced by classroom discussion and solution sharing. At any stage of the course, students can freely create and test designs. By experimenting with line lengths, angles and repeated values, they will gain a deeper understanding of geometry. Students can save designs to their own NCLab folder, publish and share links to the designs, or submit them to NCLab for display on the Gallery page. Best of all, students can create an STF file to print out their 3D designs!

EQUIPMENT AND ACCOUNTS REQUIRED:

- **Personal computers or tablets, with Internet access:** one per student. Both PC and MAC platforms are supported. Tablets can also be used. Preferred browsers are Google Chrome or Firefox.
- **Projector or Smartboard** (optional but recommended) attached to a computer for demonstration or modeling
- **Accounts:** The Tina Course requires individual accounts for each student. Visit the FAQ page for information on free and paid accounts. https://nclab.com/faq/ Have names and passwords ready on Day 1 to make logging on a smooth process (small cards with this information can be passed out to each student)
- **Teacher Accounts:** It is helpful to have two teacher accounts: one for learning the program and maintaining progress as a teacher; and the other for demonstrating the program. This second account will look like what the students encounter and can be cleared as needed for each class that is taught.
- **Progress monitoring:** Students accounts associated with a teacher can be progress monitored from the teacher's NCLab desktop.
- **The teacher textbook** can be downloaded as a .pdf file from the Resources page https://nclab.com/resources/
- **The solution manual** can be accessed by following this link: https://docs.google.com/document/d/1iPAfL11RzLI353YDsDyeYGc-U0wR_iB81RrpJrdpn5M/edit
- **YouTube videos:** some schools block YouTube, so the demonstration videos may need to be unblocked by an administrator to make them available to students
- **Publishing:** Students should have a way to share a link to their games with others, such as a shared folder on a network drive; class or student wikis, web pages, blogs or email accounts; commercial networks such as Google Drives or Edmodo; or public social media network such as Facebook or Twitter.

  Publishing to the NCLab Gallery: submit games to https://nclab.com/turtle-gallery-submit/
  Student work can be viewed at: https://nclab.com/turtle-gallery/

SUGGESTED AGE RANGE FOR STUDENTS AND PREREQUISITES

Tina 1 is designed to teach students between ages 10-15. The younger students tend to progress more slowly but can still be successful. Students at the middle school or early high school level generally master the program with little difficulty. Students who have not been exposed to integers and geometry concepts such as ordered pairs, translation and rotation, may need some background preparation before starting some of the levels.

For students with no programming experience, NCLab recommended taking the Karel 1 and 2 courses before taking the Tina course. The Karel courses use a simplified version of Python which is easier to manage.

Students who are in 8th grade or higher will have been taught all the math prerequisites. Students in earlier grades may need math mini-lessons as the course progresses. The math standards and skills are addressed in each Section.

TIME REQUIRED AND SUGGESTIONS FOR COURSE DELIVERY

There are 40 levels or lessons in Tina 1, divided into 5 sections of 8 levels each. The following lessons are written for each section, with notes on the specific skills addressed at each level within the section. Students will naturally slow down as the coding becomes more complex. In general, the amount of time required for the course is about 10 to 15 hours of computer time, including one to two hours for each instructional level, and one hour for the art project level in each section. Here are some suggestions for lesson delivery:

- **As a camp or workshop** that allows long stretches of computer time. Generally, students will complete the Tina 1 course in four sessions of about 3 hours each, although younger or less experienced students will need more time, whereas students who learn easily could complete it in about half that time. Advanced students can experiment with designs, or continue on to Tina 2.

- **As a self-paced course for independent study**, for computer lab time, after school programs, programming clubs, gifted and talented programs or home study. Students are more likely to complete the course if they are encouraged and supported by adults, and if they have the opportunity to publish and print their designs.

- **As part of an elective computer programming class** at the middle school or early high school level. Tina 1 should be followed by Tina 2. Students who complete both courses will have been introduced to all the basic tools of programming graphic design.

  If taught as a class, time will be spent outside of the computer program to develop and enrich understanding of the underlying geometry and algebra concepts, demonstrate real world applications and share ideas. Students will also appreciate time to build designs on their own. Teachers may break away from programming to teach or review math skills in preparation for a particular Section. The following is a possible 5-day cycle of 50 minute classes for one Section:

  Day 1: Introduce concepts and vocabulary (7 minutes)
  Watch and discuss YouTube video (8 minutes)
  Complete levels 1 and 2 (20 minutes – about 10 minutes each)
  Review Levels 1 and 2 as a class (6 minutes)

Assist students with questions while others experiment with designs. (Remaining time)

Day 2: Complete levels 3, 4 and 5 (36 minutes total – about 12 minutes each)
Review Levels 3, 4 and 5 as a class (9 minutes)
Assist students with questions while others experiment with designs. (Remaining time)

Day 3: Complete levels 6 and 7 (30 minutes total – about 15 minutes each.)
Review Levels 6 and 7 as a class (9 minutes)
Assist students with questions while others experiment with designs. (Remaining time)

Day 4: Level 8: Art Project
Review concepts and vocabulary learned (7 minutes)
Introduction: expectations and parameters for project. Distribute checklists (3 minutes)
Plan and Design (15 minutes)
Write and test programs (20 minutes) – note: file can be submitted for course completion and then returned to for further design as a performance task.
Review progress and save files (5 minutes)

Day 5:
Assessment: quiz and/or journal entries. (10 minutes)
Complete Art Project as performance task. (30 minutes)
Share designs (walk around gallery) (10 minutes)

Encourage students who finish quickly to take notes, deepen their understanding of the concepts, and create designs. It is one thing to just complete a level; it is another to truly understand, remember, and apply.

- **As mini-lessons** of about 20-30 minutes each, addressing one to two levels at a time. This might be a good option for upper elementary and middle school where time is a premium. At this rate, the course would take most of a quarter (about 8 weeks) to complete. However, spreading out the lessons may be more successful at reaching students from a broader range of ability and background, because the course is chunked into smaller segments with teacher and peer support.

CROSS-CUTTING CONCEPTS: MATH AND ELA STANDARDS

Tina I teaches graphic design, and therefore requires a basic understanding of geometry. At the same time, Tina itself will greatly enhance the teaching of geometry because students can see how geometry affects design. By using variables, Tina also uses basic algebraic function concepts.

State Standards vary, but most fall along the lines of the Common Core Standards.

Geometry

Here is a link to the Common Core Geometry Standards progression from Kindergarten to 12th Grade:

http://www.corestandards.org/Math/Content/G/

It is helpful to refer to this progression when teaching Tina to a particular grade level. For example, 4th grade students are learning about lines, rays, and angles; 5th grade students are introduced to the Cartesian coordinate system; 6th grade students draw polygons in the coordinate system; 8th grade students study the effects of rotation, reflection, translation, and dilation. Students in 8th grade will have all the background knowledge they need to work independently. Students in earlier grades will need some geometry instruction to prepare them for Tina. An introduction will suffice for most students, as Tina is fairly simple to code. The specific background knowledge needed is referred to in the lesson plans for each section.

Section	Background knowledge required	Standards that apply
1	LEVEL 1.1: x,y coordination plane (Cartesian plane); ordered pairs; origin, x and y axes LEVEL 1.1: scale and interval on a coordinate plane LEVEL 1.2: line (line segments), rays and angles LEVEL 1.5 angles in regular polygons, supplementary angles, angles as divisions of 360 degrees. LEVEL 1.7: Integers, using ordered pairs in Quadrants II, III and IV of the coordinate plane	4.GA.1,2 and 3 (angles, attributes) 5.GA.1 (x,y coordinate plane) 5.GB.1,2 (attributes of regular polygons) 6.G.A.1 composing and decomposing shapes 6.G.A.3 shapes in the coordinate plane 7.G.A.1 scaled drawings 8.G.A.* translations and rotations, congruent and similar shapes
2	As above	
3	As above	
4	LEVEL 4.3: Decomposing angles	7.G.B.5 using angle relationships to solve problems.
5	LEVEL 5.1: Non-linear shapes and patterns created by the use of variables.	

Algebra

Tina makes use of algebraic relationships to write code that is simple and effective. Students apply the properties of operations, use variables as functions, generate patterns based on number relationships, and use order of operations.

The K-5 link for Operations and Algebraic Thinking is

http://www.corestandards.org/Math/Content/OA/ (Operations and Algebraic Thinking)

From 6th grade onward, the study of operations and algebraic thinking is subdivided into different areas of study.

In 6th and 7th grade, the standards focus on ratio and proportion:

http://www.corestandards.org/Math/Content/RP/

In 8th grade, functions:

http://www.corestandards.org/Math/Content/F/

6th through 8th grade, expressions and equations:

http://www.corestandards.org/Math/Content/EE/

As in geometry, students in grades earlier than 8th grade will need some background support in Algebra to understand what they are doing in Tina.

Section	Background knowledge required	Standards that apply
1	Use parentheses in numeric expressions. An brief introduction to rational/irrational numbers may help students understand when to use the goto command, and with the tangram project.	5.OA.A1 6.NS.C7 (rational numbers) 8.NS.A1, A2 (irrational numbers)
2	As above	
3	As above	
4	As above	
5	Patterns and shapes generated by a given rule (in this case, a command line). Write simple expressions with more than one operation Generate sequences of ordered pairs Understand integers Graphing points in all four quadrants of the coordinate plane	4.OA.C.5 Generate and analyze patterns. 5.OA.A1,2, B1 6.NS.C.6 6,NS.C.8

Students will also develop good math process skills as they learn to write code. In fact, all of the Common Core Standards for Mathematical Practices apply, so students may very well improve in any regular math studies as a result.

SMP 1: **Make sense of problems and persevere in solving them.** Each lesson is presented as a problem or puzzle to be solved. Students can test their programs instantly, as they go. This feedback encourages them to correct errors and continue until the task is completely solved.

SMP 2: **Reason abstractly and quantitatively.** Students learn how to write logical command sequences, including conditions and repeated routines (loops and nested loops)

SMP 3: **Construct viable arguments and critique the reasoning of others.** In the search for code that meets or beats the criteria, students naturally engage in discussions about the best way to solve a puzzle. They often help each other uncover errors. Class discussions and journals enhance this communication.

SMP 4: **Model with mathematics.** Coding, by its very nature, is translating actions, conditions and goals into defined terms and symbols, which in turn translates into tangible designs.

SMP 5: **Use appropriate tools strategically.** Students have to choose the most effective commands and sequences needed to solve the problem. Subroutines (loops), conditions, and commands are selected to create code that is efficient, robust, readable and flexible.

SMP 6: **Attend to precision.** Programs will not run correctly if there are any logical or syntax errors.

SMP 7: **Look for and make use of structure.** To solve a puzzle, students must break down a task into logical steps.

SMP 8: **Look for and express regularity in repeated reasoning.** Patterns are the key to writing repeated loops, nested loops and conditions.

English Language Arts W.x.10 Write routinely over extended time frames (time for research, reflection, and revision) and shorter time frames (a single sitting or a day or two) for a range of discipline-specific tasks, purposes, and audiences. In Tina, journaling and describing designs help cement learning and give students practice in informational writing.

English Language Arts SL.x. Discussion and collaboration are key ingredients in solving problems, applying learning and creating designs. In particular, students should be able to ask and answer specific questions.

NEXT GENERATION SCIENCE STANDARDS (NGSS)

ETS1, 2, 3: **Engineering Design**

From the NGSS website:

The core idea of engineering design includes three component ideas:

A. Defining and delimiting engineering problems involves stating the problem to be solved as clearly as possible in terms of criteria for success, and constraints or limits.

B. Designing solutions to engineering problems begins with generating a number of different possible solutions, then evaluating potential solutions to see which ones best meet the criteria and constraints of the problem.

C. Optimizing the design solution involves a process in which solutions are systematically tested and refined and the final design is improved by trading off less important features for those that are more important.

How Tina 1 fits this model:

- At the end of each Section, the final Level requires the design of an Art Project. This requires programming and mathematical skills, especially when students are learning how to control the outcome.
- The course sets **criteria and constraints** Students look for the **best solutions** with the simplest, most efficient and robust code.
- Students **test** their designs, learn from failures, and **optimize** their code, just as they would in real life engineering tasks.
- Once students develop some proficiency, they can **apply** their skills to an unlimited range of projects.

To view the Engineering Design in the NGSS document in detail:

http://www.nextgenscience.org/sites/ngss/files/Appendix%20I%20-%20Engineering%20Design%20in%20NGSS%20-%20FINAL_V2.pdf

VOCABULARY, LANGUAGE AND PROGRAM SUPPORTS

- **Instruction:** Text complexity (Lexile score) ranges from about 440 to 800L, suitable for 3rd to 4th grade upwards.
- **YouTube videos:** demonstrate steps learned in the lesson. Links are listed within the lessons.
- **Text size** can be adjusted for readability.
- **Design viewer:** Students can view their designs in the right hand block at any time, providing instant visual feedback.
- **Error message feedback:** If the program contains errors, the line and error type are flagged with comments.
- **Vocabulary:** words that are specific to programming, math terms, and any Tier II or III words that students may encounter are listed and described under each Section in these lesson plans.
- **Student Journal:** a journal is provided for concept and vocabulary review, and reflections on learning. It includes sketch pages to design programs while offline.

BACKGROUND-BUILDING AND SUPPORT ACTIVITIES

Art, Science, and Math are all rich in geometric patterns that can inspire design. Hands on activities in any of these areas help students make sense of computer graphics and abstract lines of code.

Art projects:

- Op art drawings, tessellations and origami develop skills with shapes, lines and angles.
- Pottery wheels, lathes and drills can be used to create rotational shapes and shells.
- Woven, knitted, crocheted, tie-dyed, quilted and beaded designs demonstrate repeated patterns in fabric arts.

Art appreciation: Geometry has been used throughout human history as a component of art.

- Mosaic patterns range from simple to complex, especially in later examples of Islamic art.
- Rules of composition, including the golden mean.
- Abstract art, especially art that is based on optical illusion.

Patterns in Science: Geometric patterns abound in nature: x-ray diffractions of atomic structures, seashells, the rotating arms of distant galaxies.

Patterns in Engineering: Cloverleaf-shaped highway interchanges, gears.

Geometry and Algebra: Instruction and review of geometry and algebra concepts (see lesson plans for specific skills).

Ruler and protractor work: physically drawing shapes, lines and angles. Specifically, learning the different ways to divide up 360 degrees into equal parts.

Geometric manipulatives: tangrams, two and three-dimensional shapes, linking rods and connectors such as K-nex, fraction block sets (circles, polygons, rods, tiles), all build physical and visual awareness of geometry. Tesselations are examples of nested loops.

Realia. Collect items that show patterns and repeated patterns. Students can bring items to make a tabletop or bulletin board display.

Examples

- Stamp sets (stamps have designs; designs can be stamped into a repeated pattern);
- Fabric items (braid, lace, knitted or woven patterns)
- Natural items such as seashells
- Manufactured items such as chains, bracelets, gears

- An oscilloscope or oscilloscope simulation app can be used to show wave patterns.

Student Journal Sharing. Journaling provides an opportunity to reflect on learning and deepen understanding of concepts and procedures. It is a place to imagine new designs and programs. All of this can be shared as partners, small groups or whole class.

- **Hour of Code** (https://code.org/learn): As a warm-up to Tina, students can benefit by exploring free Drag and Drop programming games found at Hour of Code.

DEPTH OF KNOWLEDGE

Most problems in the Tina 1 course have more than one possible solution given the parameters, although there are less solutions that fit the number of lines required. The object of the lessons is to write code that is clean and simple and to avoid redundancies that can lead to errors. As a result, the Depth of Knowledge (DOK) during the instructional phase is 1 to 3. Using the creativity suite, DOK 3 and 4 level problems can be created and solved.

BLOOM’S TAXONOMY

- Application and Analysis: Students must analyze the given parameters to come up with a solution. Students immediately apply what they are learning at each stage by writing a program.
- Synthesis and Creation: Students can create their own publishable designs, bringing together all the skills they have learned.

ENRICHMENT, REMEDIATION AND PROGRESS MONITORING

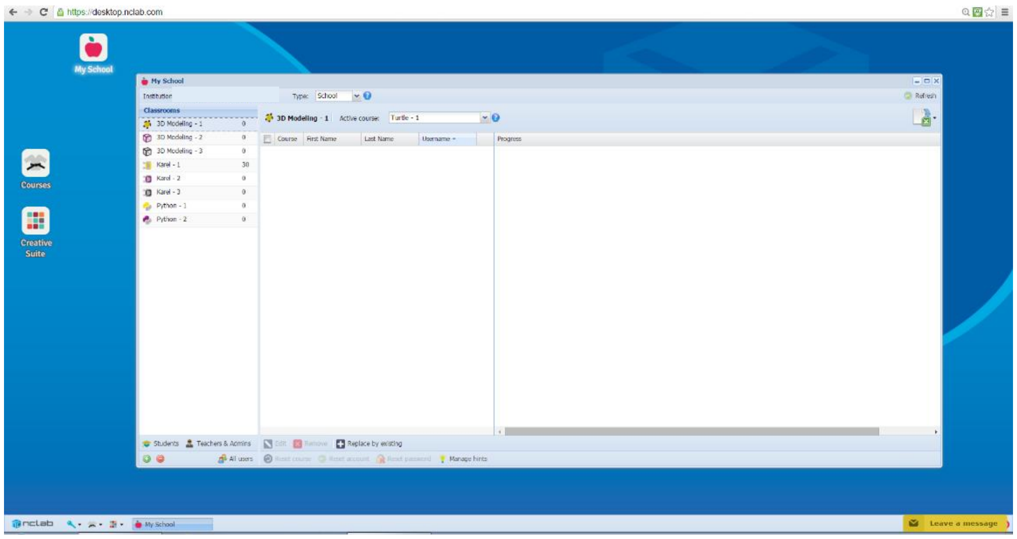
- Since the course is self-paced, students can move through the lessons based on their own rate of learning.
- Students must unlock the next levels, so it is not possible to race through or “cherry pick” the program without successfully completing each stage.
- Steps can be repeated at any time for review and reinforcement.
- Teachers should monitor and provide scaffolded support as needed. At some point, most students will hit their own personal threshold level in which they aren’t immediately successful. Point out the built-in hints and line prompts within the program. Follow up with discussions about what they learned from these hints.

- In a camp or workshop setting, it is important to build in physical breaks. Students tend to stay longer than they should in front of the computer.
- In any setting, encourage opportunities to interact and discuss progress.
- Set design challenges for students, especially for those who go through the levels quickly.

ASSESSMENT

Assessment built into the program:

- Within each level, students get immediate feedback from the program. The design created by their code is displayed to the right, and any syntax errors are noted in the dialogue box below their code.
- Upon successful completion of a level, students will unlock the next level. Likewise, upon successful completion of each section, students will receive a certificate and unlock the next section.
- Teachers can monitor the progress of their students by clicking on the My School apple. This opens a new window. Select Turtle-1 to Active Course.


https://desktop.nclab.com

Type: School

3D Modeling - 1 Active course: Turtle - 1

Classrooms	
3D Modeling - 1	0
3D Modeling - 2	0
3D Modeling - 3	0
Karel - 1	30
Karel - 2	0
Karel - 3	0
Python - 1	0
Python - 2	0

Journals: A Student Journal is included in the course materials and can be used as a portfolio artifact.

Quizzes: Students can complete paper and pencil or online quizzes (in development).

Art projects as performance assessments: The final level in each section is an Art Project, in which students write a program that creates a geometric design based on the skills learned in that section. In the Lesson Plans, a printable assignment is included at the end of each section that can be used to score the designs.

Student Feedback: At the end of each level, students are asked to evaluate the level of difficulty by clicking on an EASY, MEDIUM or HARD button. Students can add specific comments. This gives the NCLab designers valuable feedback for improving the games.

Here is a possible grading scheme that includes several methods of assessment:

SUGGESTED GRADING SCHEME FOR TINA 1

Type of Assessment	Points	Number	Total Points
Section Completion	50	5	250
End of Section Art Project	50	5	250
Journal Responses	50	5	250
TOTAL POINTS			750

LESSONS

Note: The best way to prepare for these lessons is to do them as a user either ahead of time or alongside the students. Answer keys for all levels are available at

https://docs.google.com/document/d/1iPAfL11RzLI353YDsDyeYGc-U0wR_iB81RrpJrdpn5M/edit

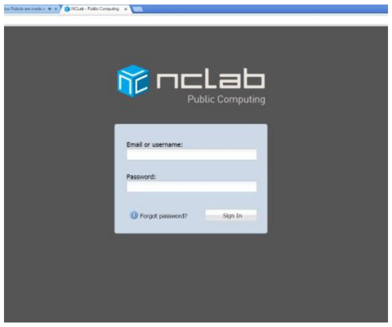
INTRODUCTION TO THE COURSE (ABOUT 20-30 MINUTES)

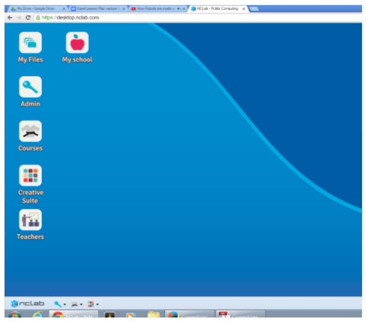
In the very first session, allow for time to log in the students and show them where the course is located on the desktop. Demonstrate the log in steps and first lesson on a computer (for larger classes, attached to a projector or Smartboard if available).

Modeling How to Access the Course:

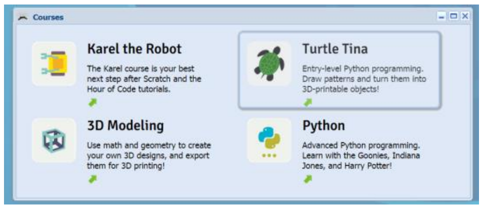
Model how to log in, select the course from the Desktop and navigate screens with initial level, using Smartboard or projector. Have the students complete each step as you go along.

Log in to account https://desktop.nclab.com/. Select "Courses" (doubleclick)





Select Tina Turtle and then Tina 1.

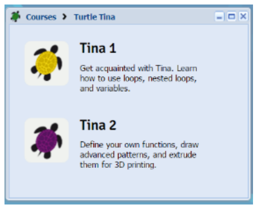


Karel the Robot
The Karel course is your best next step after Scratch and the Hour of Code tutorials.

Turtle Tina
Entry-level Python programming. Draw patterns and turn them into 3D-printable objects!

3D Modeling
Use math and geometry to create your own 3D designs, and export them for 3D printing!

Python
Advanced Python programming. Learn with the Goonies, Indiana Jones, and Harry Potter!

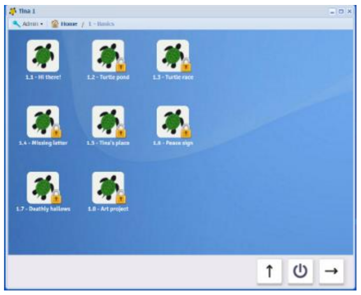


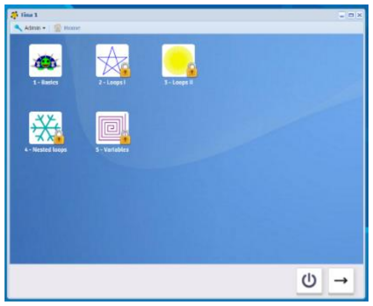
Tina 1
Get acquainted with Tina. Learn how to use loops, nested loops, and variables.

Tina 2
Define your own functions, draw advanced patterns, and extrude them for 3D printing.

Select the Section (1-Basics) and Level (1.1. -)

Note that only one section and level are available. The rest are locked until the section or level is successfully completed.





SECTION 1 BASICS: LEVELS 1.1-1.8

Objectives: Students will be able to write Python code that directs the turtle icon. In Section 1, students will learn the basic parameters, movement and turning commands to draw lines, angles and simple figures in an *x,y* plane.

Vocabulary:

Commands (these are on the printable cards), presented in order of appearance in the levels:

`tina.go(n),` where *n* = the number of steps.

This command moves the turtle forward, creating a line on the graph

It can also be written as

`tina.forward(n)`

`tina.fd(n)`

`tina.left(n),` where *n* = the number of degrees to turn.

`tina.right(n),` where *n* = the number of degrees to turn.

These commands will turn the turtle by the number of degrees indicated.

Left can also be written as `tina.lt(n)`, and right as `tina.rt(n)`

`tina.up();tina.down()`

Pen up (so that Tina moves forward without drawing a line) and pen down (to resume drawing with forward movement):

`tina.width(n)`, sets the line width, where *n* is the line width in units.

`tina.hide()` hides the turtle so that it doesn't show in the final drawing

`tina.back(d)`, where *d* is the distance Tina backs up. Tina will not draw when backing up.

`tina.extrude(n),` where *n* is the width to be extruded. Extrude sets the width of the shape in the z axis so that the design can be printed. Extrude also hides Tina, so the hide command does not need to be used if the extrude command is used.

`tina.goto(x,y).` Tina will draw a line to a specified coordinate pair. This is a useful command for angles that are not integers or decimals to the nearest tenth. It should only be used if necessary.

Time required:

Once students have learned how to log into the Desktop and select their course, most will complete Section 1 in about 90 minutes, <u>excluding the art project</u>.

Background knowledge/Introductory Set/Purpose:

- Using the following link, show students an overview of Tina 1 (which starts with Basics and ends with Variables) and Tina 2 (which starts with Functions and ends with Rotational Solids and Shells). Here is the link to the course overview: https://nclab.com/turtle-course-details/

- Share a couple of examples of design in art and nature, using the Wikipedia links suggested in Background or selecting from the many pictures and videos available on the Internet.

- Purpose: Section 1 (Levels 1.1-1.8) introduces basic commands and parameters used in Tina. This information is on the help cards for Section 1, which can be introduced now, reviewed as students work through the Levels, and summarized at the end of the Section.

- Math background: students may need extra instruction depending on their background in math. This can be explicitly taught, reviewed, or assigned as determined by the teacher. Math concepts used in Section 1 include:

  LEVEL 1.1: *x,y* coordination plane (Cartesian plane); ordered pairs; origin, *x* and *y* axes

  LEVEL 1.1: scale and interval on a graph

  LEVEL 1.2: line (line segments), rays and angles

  LEVEL 1.5 angles in regular polygons, supplementary angles, angles as divisions of 360 degrees.

  LEVEL 1.7: Integers, using ordered pairs in Quadrants II, III and IV of the coordinate plane

Direct instruction:

As previously recommended, it is helpful to have two teacher accounts: one for learning the course ahead of the students, and one for modeling steps to the students. This second account is useful for instruction as it will show the screens as they appear for the first time to a user and can be hard reset as needed.

Since this is a self-paced course, the amount of modeling should be kept to a minimum. It is helpful to introduce each Section, and also to pause the progress of the class if observation shows the need for a class-wide demonstration or lesson. Pause points are suggested in each level. YouTube videos are at the beginning of Levels 1.2 and 1.4.

Since this is the first Section, do Level 1.1 together as a class to give students a feel for how the program works.

Help cards: these can be printed, distributed and discussed.

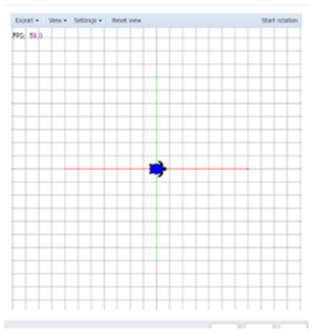
Tina's default positions and settings in an *x,y* coordinate plane (Cartesian plane)

Tina starts at the origin (0,0)

Tina facing east (positive x axis)

Each grid space is 10 units

Tina's default color is BLUE



The *x* axis is red the *y* axis is green, and the *z* axis is blue. In Tina 1, students work within the 2-dimensional *xy* plane, and only extrude into the *z* axis when creating a printable object using the `extrude` command.

Basic commands (Levels 1.1 to 1.4)

Moving forward: `tina.go(d)`, where `d` is the distance. Remember: The grid interval is 10 units, so to move one grid space, type `tina.go(10)`

Turning left or right: `tina.left(a)` or `tina.right(a)`, where `a` is the number of degrees to be turned.

Color: `tina.color(COLOR)`, where `COLOR` is one of the many colors stored in the program language. This can be a normal name like `RED`, or a defined name such as `WOOD`. Colors are all capital letters.

Basic commands (Levels 1.5 to 1.7)

Pen up (so that Tina moves forward without drawing a line) and pen down (to resume drawing with forward movement): `tina.up();tina.down()`

Line width: `tina.width(`*n*`)`, where *n* is the line width in units.

Hiding the turtle so that it doesn't show in the final drawing: `tina.hide()`

Backing up: `tina.back(`*d*`)`, where *d* is the distance Tina backs up. Tina will not draw when backing up.

Extruding the design to form a 3D printable shape: `tina.extrude(`*n*`)`, where *n* is the width to be extruded. Extrude also hides Tina, so the hide command does not need to be used if the extrude command is used.

Go to a coordinate pair: `tina.goto(`*x*`,`*y*`)`. Tina will draw a line to a specified coordinate pair. This is a useful command for angles that are not integers or decimals to the nearest tenth. It should only be used if necessary.

Saving and Sharing (Level 1.8)

To make a 3D printable file, remember to extrude the design using `tina.extrude(`*n*`)`, where *n* is the width to be extruded. -

To save, go to File -> Save STL to NCLab and save the file.

To publish, open the File Manage ("My Files" icon):

- Right-click on the STL file.
- Choose Publish to the web.
- Select the first option "Anyone can view".
- Press CTRL-C to copy the file to the clipboard.
- Visit http://nclab.com/turtle-gallery to share the design.
- Designs can also be published on Facebook, Twitter, G+ or a location designated by your teacher or school ________________________.

Individual/Group practice:

The program is designed to be used individually by students. Encourage peer support, sharing and discussion.

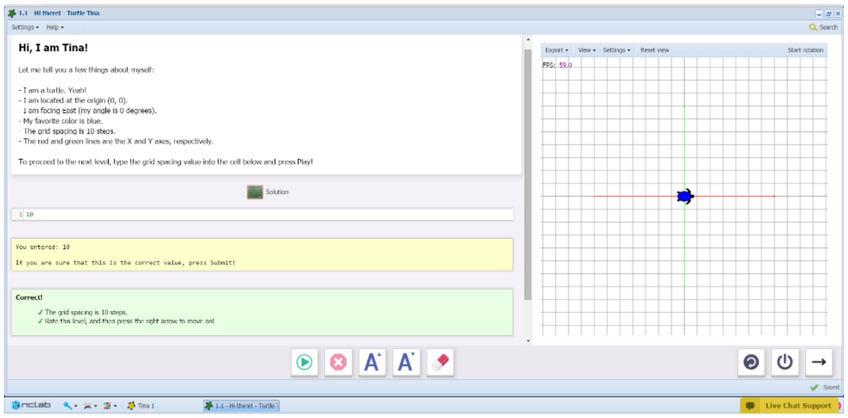
Levels 1.1-1.8

1.1 Hi There!

In this simple introduction, students answer the question by typing in an integer that represents the grid spacing.

A clue is given in the text: “The grid spacing is 10 steps”.

Once the integer is entered into the command line, the student is prompted to submit the completed response by pressing the green button on the lower right hand corner of the screen.



1.1 - Hi there! - Turtle Tina

Hi, I am Tina!

Let me tell you a few things about myself:

- I am a turtle. Yeah!
- I am located at the origin (0, 0).
- I am facing East (my angle is 0 degrees).
- My favorite color is blue.
- The grid spacing is 10 steps.
- The red and green lines are the X and Y axes, respectively.

To proceed to the next level, type the grid spacing value into the cell below and press Play!

```
10
```

```
You entered: 10

If you are sure that this is the correct value, press Submit!
```

Correct!

✓ The grid spacing is 10 steps.
✓ Rate this level, and then press the right arrow to move on!

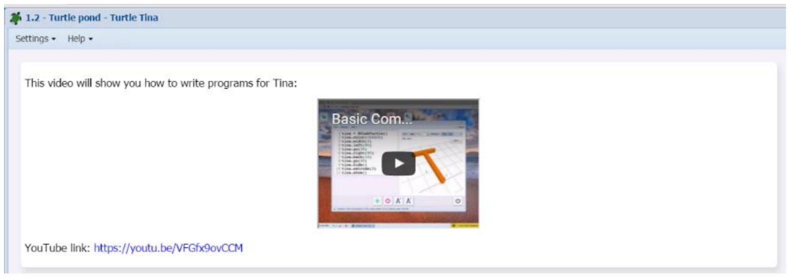
If correct, this screen will appear.

The green message box lists the successful tasks, and may contain additional pointers.

It will also ask students to rate the level as too easy, just right, or too hard and provide any comments. This feedback helps the program developers make improvements.

1.2 Turtle Pond

In Turtle Pond, the student is prompted to watch the embedded video, which teaches basic commands. The video can be viewed on YouTube at https://www.youtube.com/watch?v=VFGfx9ovCCM&feature=youtu.be. If students do not have access to You Tube, the video can be shown to the class.

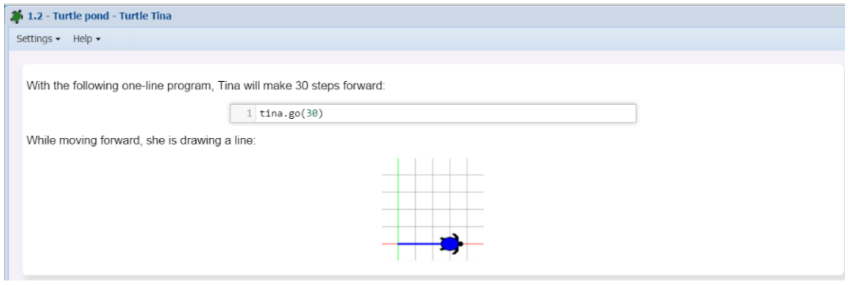


1.2 - Turtle pond - Turtle Tina

This video will show you how to write programs for Tina:

YouTube link: https://youtu.be/VFGfx9ovCCM

The first command to be learned is `tina.go(n)`, which moves the turtle in a straight line by the number of steps indicated in the parentheses. While moving forward, the turtle is drawing a line (reminder: each grid square = ten steps).



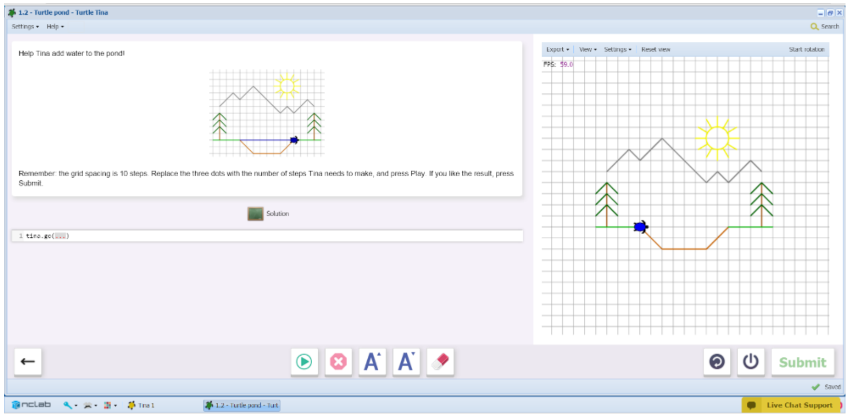
With the following one-line program, Tina will make 30 steps forward:

```
tina.go(30)
```

While moving forward, she is drawing a line:

Students practice this command by drawing a line that will fill the pond.

A model of what the design should look like is included in the upper left pane, along with any specific instructions such as the commands to be used and the number of lines in the program.



Help Tina add water to the pond!

Remember: the grid spacing is 10 steps. Replace the three dots with the number of steps Tina needs to make, and press Play. If you like the result, press Submit.

```
tina.go(...)
```

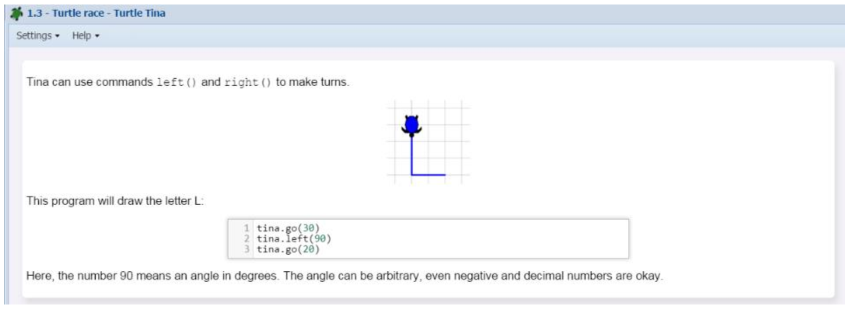
After filling in the command with the number of steps, students can preview the results by pressing the green play button at the bottom of the screen.

If the results are satisfactory, they can submit their program to check the solution and move on to the next level.

1.3 Turtle Race

In this level, students learn to use the left and right commands.

The level starts by teaching how to write the turning commands, including the need to specify the angle in degrees.



Tina can use commands `left()` and `right()` to make turns.

This program will draw the letter L:

```
tina.go(30)
tina.left(90)
tina.go(20)
```

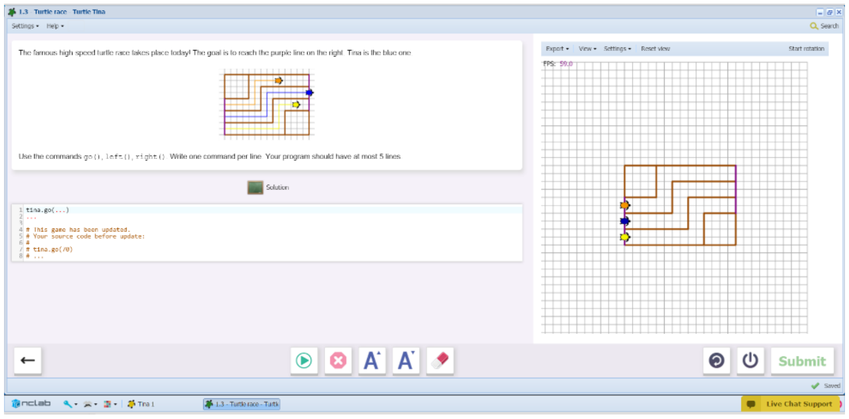
Here, the number 90 means an angle in degrees. The angle can be arbitrary, even negative and decimal numbers are okay.

Students practice the turns by writing a program to guide Tina through a racetrack, using the go, left and right commands.

Commands:
`go(),left(),right()`

Number of lines: 5

Remind students to run the program using the green arrow to see if it works before submitting.



The famous high speed turtle race takes place today! The goal is to reach the purple line on the right. Tina is the blue one.

Use the commands `go()`, `left()`, `right()`. Write one command per line. Your program should have at most 5 lines.

```
tina.go(...)
...

# This game has been updated.
# Your source code before update:
#
# tina.go(70)
# ...
```

1.4 Missing Letter

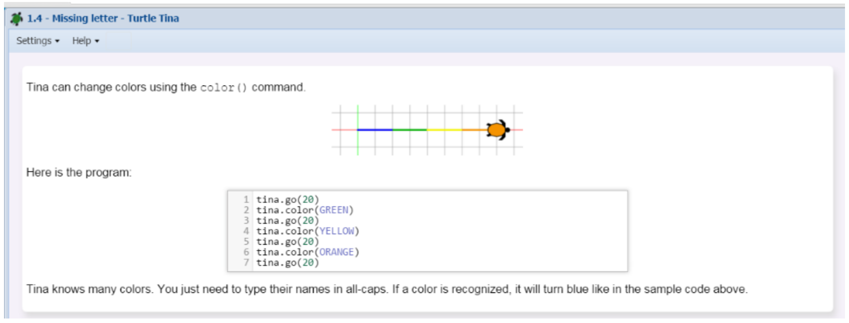
Before starting this level, students view a second video on basic commands. This video can be viewed at https://www.youtube.com/watch?v=2n5JfOyYfok&feature=youtu.be

The video includes an explanation of comment lines (text strings) that can be inserted anywhere in the program as a heading, explanation, or hint for that section of the program. It also explains how to use xy coordinate pairs, and how to reorient Tina to a specific angle.

Level 1.4 starts by teaching how to use the color command `tina.color(COLOR)`

The program recognizes many colors. Students will know if they have typed a valid color because the text will change to blue.

Color commands are typed with all capital letters.



Tina can change colors using the `color()` command.

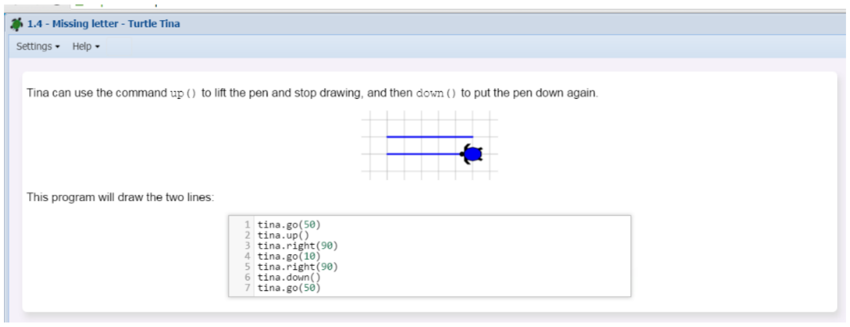
Here is the program:

```
tina.go(20)
tina.color(GREEN)
tina.go(20)
tina.color(YELLOW)
tina.go(20)
tina.color(ORANGE)
tina.go(20)
```

Tina knows many colors. You just need to type their names in all-caps. If a color is recognized, it will turn blue like in the sample code above.

Next, 1.4 teaches about the `tina.up()` pen up and `tina.down()` pen down commands.

Tina will move without drawing when the pen is up, and will draw as she moves when the pen is down. This enables her to create spaces, or to move to another location before she resumes drawing.



Tina can use the command `up()` to lift the pen and stop drawing, and then `down()` to put the pen down again.

This program will draw the two lines:

```
tina.go(50)
tina.up()
tina.right(90)
tina.go(10)
tina.right(90)
tina.down()
tina.go(50)
```

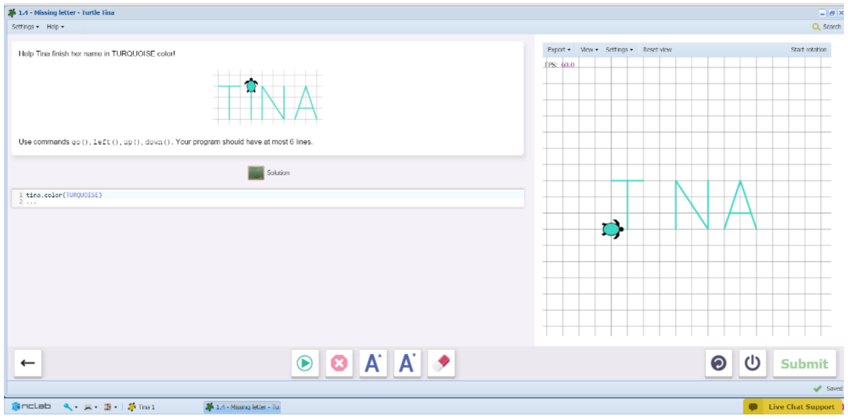
Students then practice using the color command to draw the letter I in TINA using the color TURQUOISE.

The other letters have already been created.

Commands:

`go(), left(), up(), down()`

Number of lines: 6



Help Tina finish her name in TURQUOISE color!

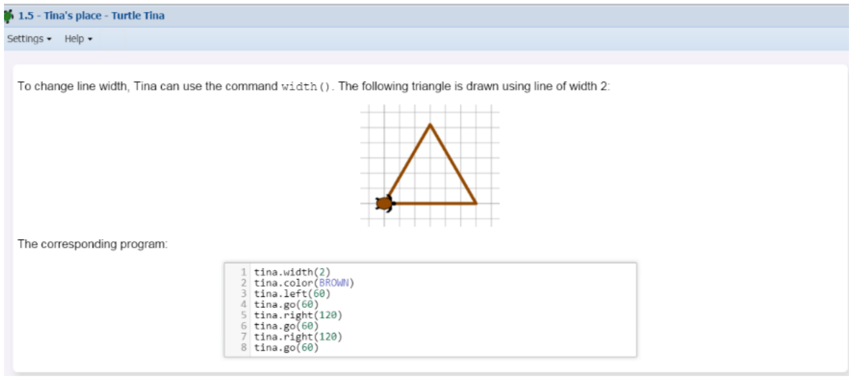
Use commands `go()`, `left()`, `up()`, `down()`. Your program should have at most 6 lines.

```
tina.color(TURQUOISE)
...
```

1.5 Tina's Place

In this level, students learn how to specify the line width using the `tina.width(X)` command.

The width X can be any number between 0.1 and 10.



To change line width, Tina can use the command `width()`. The following triangle is drawn using line of width 2:

The corresponding program:

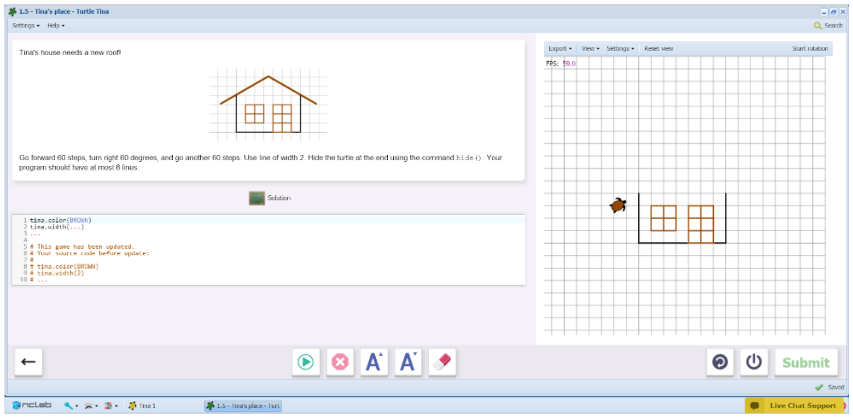
```
tina.width(2)
tina.color(BROWN)
tina.left(60)
tina.go(60)
tina.right(120)
tina.go(60)
tina.right(120)
tina.go(60)
```

Students practice the command by creating a brown roof with a width of 2 units for Tina's house.

Remember, the directions in the upper left box describe which steps need to be included in the program. At the end of this program, students practice the `tina.hide()` command to make the turtle disappear when the drawing is rendered.

Commands: `go()`, `right()`, `hide()`

Number of lines: 6



Tina's house needs a new roof!

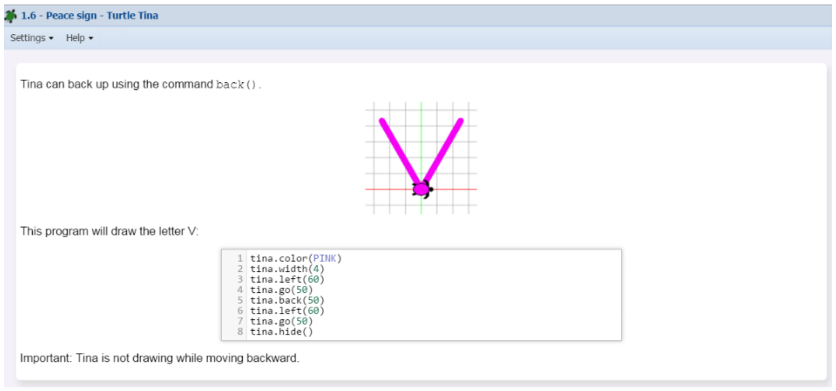
Go forward 60 steps, turn right 60 degrees, and go another 60 steps. Use line of width 2. Hide the turtle at the end using the command `hide()`. Your program should have at most 6 lines.

```
tina.color(BROWN)
tina.width(...)
...

# This game has been updated.
# Your source code before update:
#
# tina.color(BROWN)
# tina.width(2)
# ...
```

1.6 Peace Sign

1.6 begins by teaching students how to use the command `back()` to get Tina to move backward.

This is a useful command because Tina will not draw when moving backward, so it can be used to reposition her without using the pen up and pen down commands.



Tina can back up using the command `back()`.

This program will draw the letter V:

```
tina.color(PINK)
tina.width(4)
tina.left(60)
tina.go(50)
tina.back(50)
tina.left(60)
tina.go(50)
tina.hide()
```

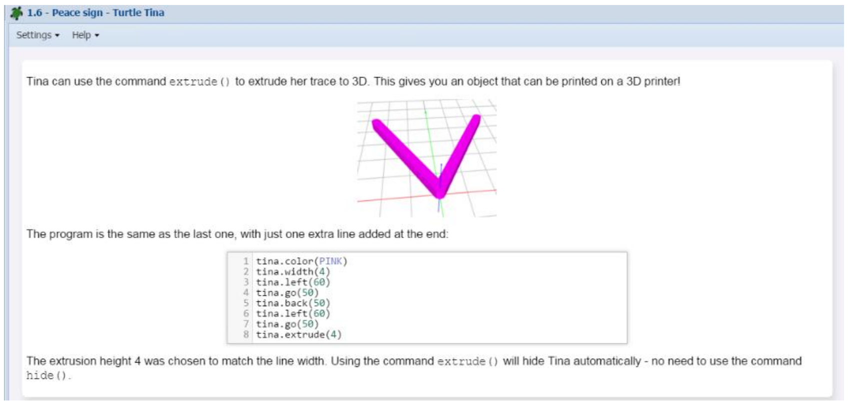
Important: Tina is not drawing while moving backward.

Students then learn the command `extrude(n)` which turns the object into a 3D printable object.

`n` is the thickness (3rd dimension, z-axis) of the extrusion.

It is recommended to keep the thickness close to the value of the width of the line.

The `extrude()` command automatically hides Tina, so there is no need to add a hide command line.



Tina can use the command `extrude()` to extrude her trace to 3D. This gives you an object that can be printed on a 3D printer!

The program is the same as the last one, with just one extra line added at the end:

```
tina.color(PINK)
tina.width(4)
tina.left(60)
tina.go(50)
tina.back(50)
tina.left(60)
tina.go(50)
tina.extrude(4)
```

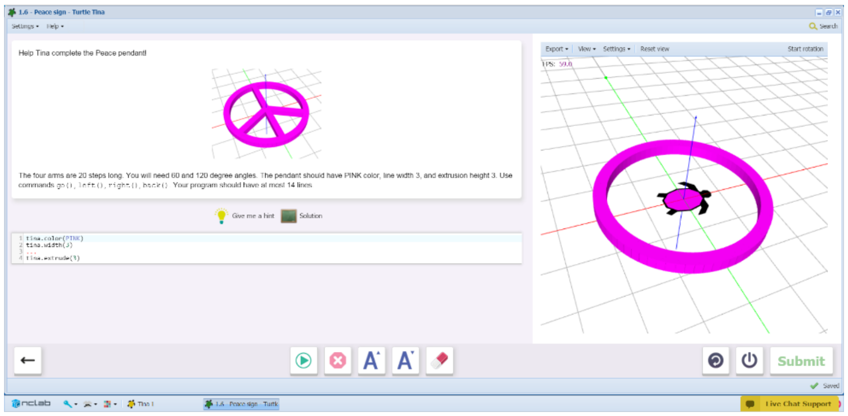
The extrusion height 4 was chosen to match the line width. Using the command `extrude()` will hide Tina automatically - no need to use the command `hide()`.

Students apply what they have learned to complete the peace sign.

Commands: `go()`, `left()`, `right()`, `back()`

Number of lines: 14

Students may approach the task differently in their choice of turns, forward and backward movements. Suggest that they compare their programs.



Help Tina complete the Peace pendant!

The four arms are 20 steps long. You will need 60 and 120 degree angles. The pendant should have PINK color, line width 3, and extrusion height 3. Use commands `go()`, `left()`, `right()`, `back()`. Your program should have at most 14 lines

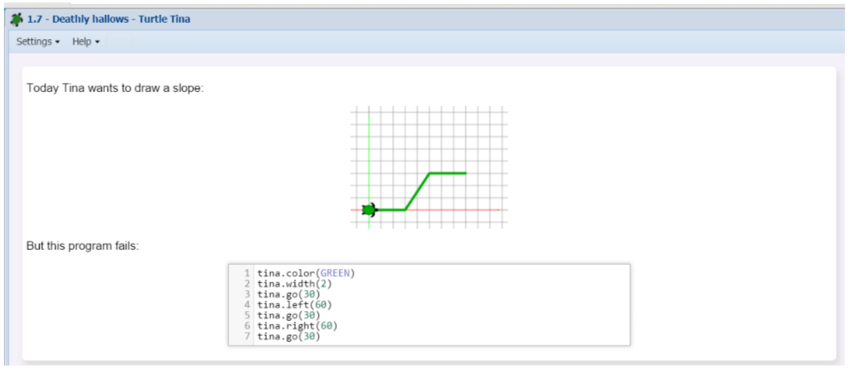
```
tina.color(PINK)
tina.width(3)
...
tina.extrude(3)
```

Once the peace sign is successfully completed, student receive a note explaining how to save their files as .STL printable files. See Level 1.8 for a copy of this note. Directions are also on the 4th help card.

1.7 Deathly Hallows

In this level, students learn how to construct slopes.

The first screen is an example of a failed program. If you are guiding students through the lessons, or they are journaling, ask them why this program fails.



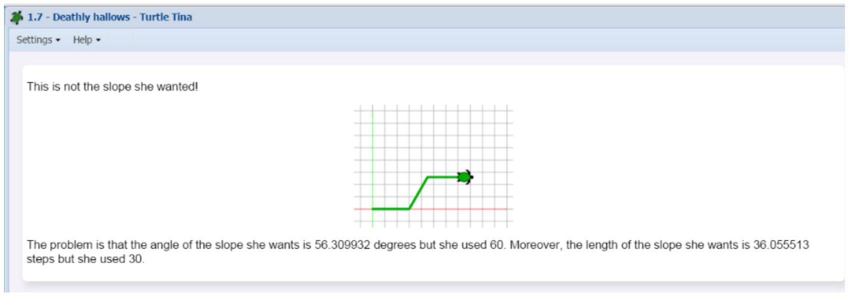
Today Tina wants to draw a slope:

But this program fails:

```
tina.color(GREEN)
tina.width(2)
tina.go(30)
tina.left(60)
tina.go(30)
tina.right(60)
tina.go(30)
```

The failure is explained on the second screen.

Both the angle and the length of the line would be very difficult to specify as a number. The commands used so far would produce inaccurate results.

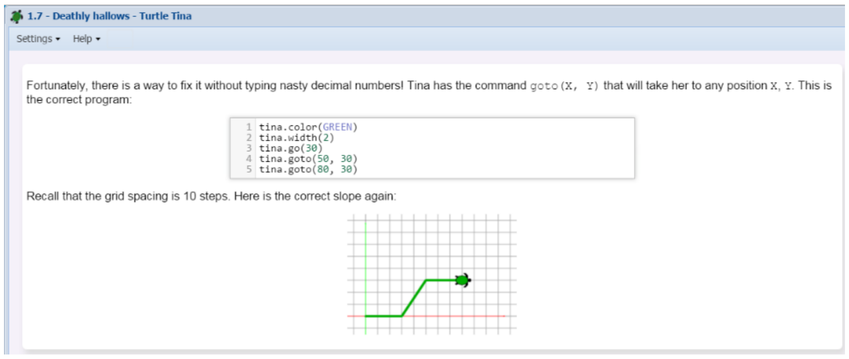


This is not the slope she wanted!

The problem is that the angle of the slope she wants is 56.309932 degrees but she used 60. Moreover, the length of the slope she wants is 36.055513 steps but she used 30.

The command `goto(X, Y)` will draw the line to the specified X, Y coordinate.

This avoids problems with line length and turning angle.



Fortunately, there is a way to fix it without typing nasty decimal numbers! Tina has the command `goto(X, Y)` that will take her to any position `X`, `Y`. This is the correct program:

```
tina.color(GREEN)
tina.width(2)
tina.go(30)
tina.goto(50, 30)
tina.goto(80, 30)
```

Recall that the grid spacing is 10 steps. Here is the correct slope again:

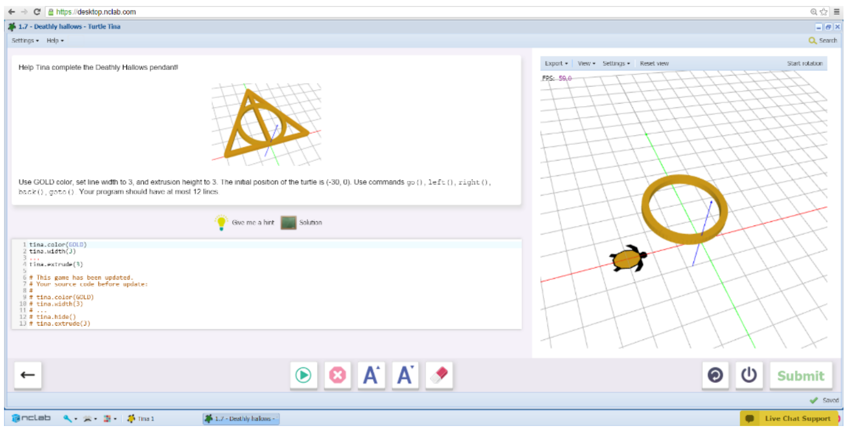
Students practice the new command by completing the Deathly Hallows pendant.

Both the triangle and the bisecting line must be drawn.

The instructions specify line width, extrusion height, and color; and Tina's starting position

Commands: `go()`, `left()`, `right()`, `back()`, `goto()`

Number of lines: 12



Help Tina complete the Deathly Hallows pendant!

Use GOLD color, set line width to 3, and extrusion height to 3. The initial position of the turtle is (-30, 0). Use commands go(), left(), right(), back(), goto(). Your program should have at most 12 lines.

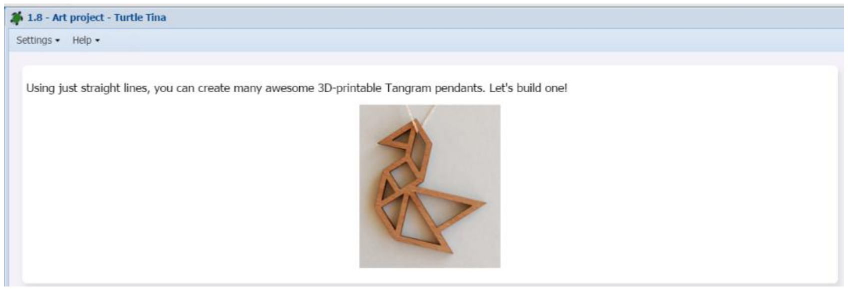
```
tina.color(GOLD)
tina.width(3)
...
tina.extrude(3)

# This game has been updated.
# Your source code before update:
#
# tina.color(GOLD)
# tina.width(3)
# ...
# tina.hide()
# tina.extrude(3)
```

Have students carefully compare their rendered pendant with the example. They can rotate it to check. Are the angles and line lengths the same?

Remind students to only use the `goto()` command when necessary. They should specify line lengths and angles whenever possible.

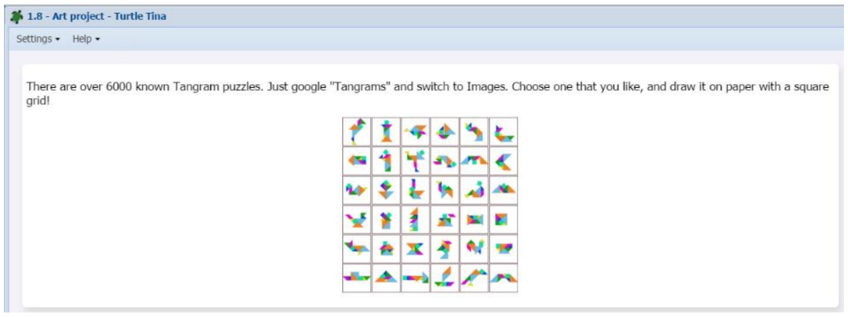
1.8 Art Project

In this level, students will design their own tangram pendant. The photo at the right shows an example of a completed pendant.



Using just straight lines, you can create many awesome 3D-printable Tangram pendants. Let's build one!

The second screen shows several tangrams and encourages students to look for images on line.

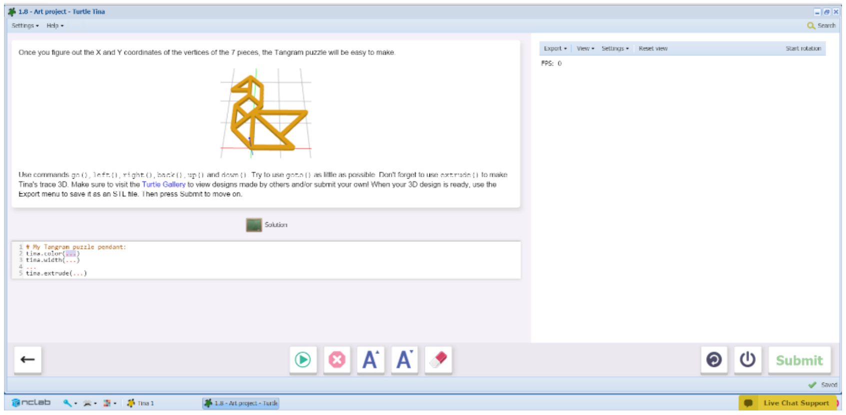


There are over 6000 known Tangram puzzles. Just google "Tangrams" and switch to Images. Choose one that you like, and draw it on paper with a square grid!

This project is the final level for Section 1 and can be used as an assessment.

Commands: `go(), left(), right(), back(), up(), down(), goto()`

Number of lines: will vary with design.

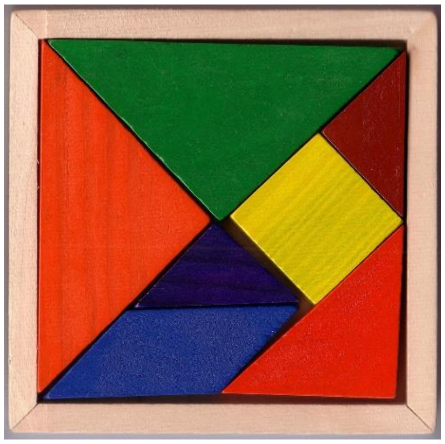


Once you figure out the X and Y coordinates of the vertices of the 7 pieces, the Tangram puzzle will be easy to make.

Use commands `go()`, `left()`, `right()`, `back()`, `up()` and `down()`. Try to use `goto()` as little as possible. Don't forget to use `extrude()` to make Tina's trace 3D. Make sure to visit the Turtle Gallery to view designs made by others and/or submit your own! When your 3D design is ready, use the Export menu to save it as an STL file. Then press Submit to move on.

```
# My Tangram puzzle pendant:
tina.color(...)
tina.width(...)
...
tina.extrude(...)
```

Since the lengths of the sides are not integers (see the note below), students may need to use the goto command in many cases.



Background on tangrams:

The tangram consists of seven pieces as described below:

Choosing a unit of measurement so that the seven pieces can be assembled to form a square of side one unit and having area one square unit, the seven pieces are:

- 2 large right triangles (hypotenuse 1, sides $\sqrt{2}/2$, area $1/4$)
- 1 medium right triangle (hypotenuse $\sqrt{2}/2$, sides $1/2$, area $1/8$)
- 2 small right triangles (hypotenuse $1/2$, sides $\sqrt{2}/4$, area $1/16$)
- 1 square (sides $\sqrt{2}/4$, area $1/8$)
- 1 parallelogram (sides of $1/2$ and $\sqrt{2}/4$, area $1/8$)

If students scale up these relationships so that the hypotenuse of the large triangle is 40 units, they should be able to place the shapes using the `go, left, right` and `goto` commands without much difficulty. This makes some of the sides integer lengths (lengths of the two equal sides of the medium triangle, the hypotenuse of the smallest triangles, the longest side of the parallelogram, the diagonal of the square) equal to 20 units. The interior angles are combinations of 45, 90 and 135 degrees.

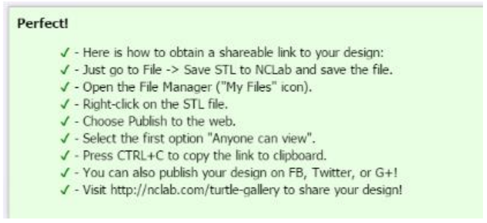
Students will benefit from playing with physical tangram sets, or virtual ones such as the free tangram puzzles on abcya.com, to get a feel for these shape relationships.

To simulate a true design or engineering project, have students draw and annotate their design on squared paper. This will help them determine the line lengths and turns, and which coordinates to use in the goto command, and will serve as a written record of their work.

You may want to modify the assignment for younger or less experienced students. The program will accept simpler designs than full tangrams as Art Projects.

Saving and sharing the design:

Once students have completed the design, they can export it as an STF file for printing, using the pull-down menu above the drawing on the right hand side. Here are the instructions as shown in the program. Students will see this window once they have submitted their design.



Perfect!

✓ - Here is how to obtain a shareable link to your design:
✓ - Just go to File -> Save STL to NCLab and save the file.
✓ - Open the File Manager ("My Files" icon).
✓ - Right-click on the STL file.
✓ - Choose Publish to the web.
✓ - Select the first option "Anyone can view".
✓ - Press CTRL+C to copy the link to clipboard.
✓ - You can also publish your design on FB, Twitter, or G+!
✓ - Visit http://nclab.com/turtle-gallery to share your design!

Project differentiation:

Since it is composed of several adjoining shapes, a tangram takes many lines of programming and thus an investment of time to complete. This is a chance for students to exercise all the skills they have learned so far. They will appreciate subsequent lessons more after trying to figure out how to build all these shapes on the coordinate plane.

Suggested questions for post-session discussion (students can use their journals to write down their ideas and responses) (10-20 minutes):

When you made your final design, did you use goto()? Why did you decide to use it instead of a combination of go(), left() and right()?

How do you decide what angle to use with the left() or right() commands?

How many lines of programming did your project need to create the shapes?

What would you create with Tina?

Assessment:

Within the program itself, students receive a White Belt of First Degree certificate upon successful completion of Section 1. The certificate can be printed, emailed or shared on social media.

Students Journal entries can be used as assessment.

The Art Project can be used as a performance task or portfolio artifact. Here is a possible 50-point assignment card:

Section 1 Assignment

END OF SECTION 1: ART PROJECT (50 POINTS)

Objective: Create and publish a design based on tangrams.

Planning (15 points): After you have decided on a design, draw out the shapes on grid paper or in your journal.

- Describe the shapes. What commands will be needed to draw them? Name each shape (for example Purple Triangle 1) and describe the color, starting position, line lengths, turning angles, and coordinates for the goto command if needed.
- Write out the order in which you will program the shapes. Where do the shapes attach to each other? When will the up, down, and back commands be useful?

Writing the code (25 points): Write the program.

- Before you write the code for each shape, write a comment line that names and describes the shape.
- Write the code and test as you go..
- Include examples of all the commands learned in Section 1:
  `go(), left(), right(), back(), up(), down()` and `goto()`

Saving the file, sharing and publishing (10 points)

- Publish the design to your folder. Inform someone else about the game by providing the link on ____, or by email.

Support:

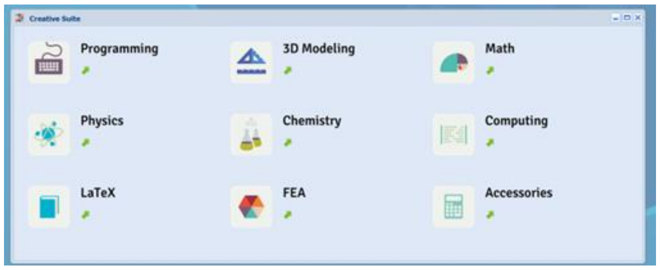
For students who need extra support:

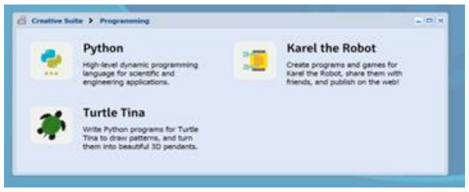
- Show them the next step needed.
- Use the printable help cards.
- Partner them with a more experienced student.
- Decrease the number of shapes required for the Art Project.

Enrichment: Creative Suite

For students who are ready to create their own designs, Turtle Tina is available in Creative Suite.

- Students can create their own files.
- Creative Suite can be accessed by doubleclicking the Creative Suite button on the left side of the NCLab desktop, or from the pullup menu at the bottom of the screen.
- On the Creative Suite menu, click on Programming, then Turtle Tina.
- The opening screen includes a YouTube video and sample code. Students can run the code and experiment with it.
- The video and a List of Commands are always available from the Help menu.
- To create a new file, select New from the File menu.
- The file is saved in the same way as the Art Project, to the NcLab Folder.





Python
High-level dynamic programming language for scientific and engineering applications.

Karel the Robot
Create programs and games for Karel the Robot, share them with friends, and publish on the web!

Turtle Tina
Write Python programs for Turtle Tina to draw patterns, and turn them into beautiful 3D pendants.

SECTION 2 LOOPS I: LEVELS 2.1-2.8

Objectives: In this section, students learn to use loops to draw regular polygons, circles, and stars.

Vocabulary:

> **Loop:** a set of repeated commands that is done a certain number of times. The first line states the number of times. The **body of the loop** is the set of commands that will be repeated.
>
> **For-Loop:** In Tina, a specific type of loop is called a For-Loop, because it begins with the phrase For i in range()
>
> **An example of how to write a loop is given at the beginning of Level 2.1.**

Prerequisite skills: Completion of Section 1, and an understanding of the basic commands learned in that section. Since Section 2 builds shapes based on repeated patterns, it helps to have an understanding of symmetry and regular polygons.

Time required: Time required will vary based on student ability and experience. Most students will complete this section in two or three hours, excepting the art project.

Background knowledge/Introductory Set/Purpose:

Use photographs such as those suggested in Background Activities to demonstrate repeated patterns in science, art, and nature.

Patterns that are repeated can be written as loops. This has many benefits:

> It is easy to change the number of times a pattern is to be repeated.
>
> It is easy to test and fix code within the loop. You are only editing one line of code rather than several of the same lines if the loop wasn't created.

Direct Instruction and Modeling:

2.1 is described in detail below. The video can be played to the class.

Instructional screens can be shown on a projector or Smartboard (remind students to refer to these for help).

Individual/Group practice:

The program is designed to be used individually by students. Encourage peer support, sharing and discussion.

Help cards: these can be printed, distributed and discussed.

Writing a For loop to repeat a set of commands:

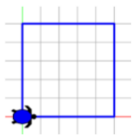
For i in range(4):	range(4) means that the following lines will be repeated four times.
	Note that the first line ends with a colon.

The next two lines form the body of the loop.

tina.go(50)	These are the commands that will be repeated 4 times.
tina.left(90)	The body commands must be indented 2 spaces from the first line.

The complete set of commands look like this:

```
For i in range(4):
  tina.go(50)
  tina.left(90)
```



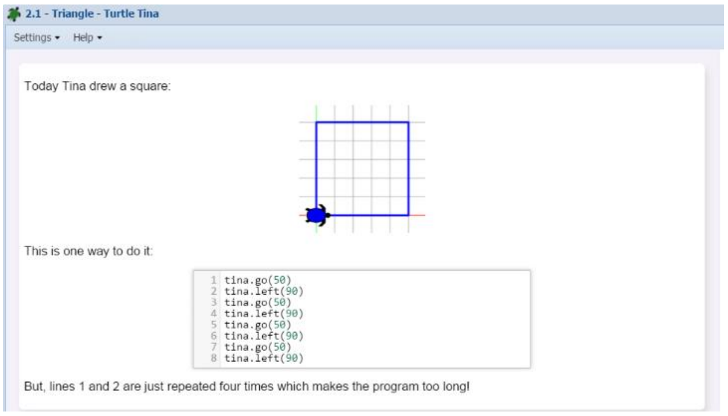
Self-paced Instruction: Levels 2.1-2.8

2.1 Triangle

Triangle starts with a You Tube video explaining how to use loops. The video can be viewed from within the program or by following this link:

http://youtu.be/sFiTiTVJR8E

The next screen describes a problem with writing repeated commands.



Today Tina drew a square:

This is one way to do it:

```
1 tina.go(50)
2 tina.left(90)
3 tina.go(50)
4 tina.left(90)
5 tina.go(50)
6 tina.left(90)
7 tina.go(50)
8 tina.left(90)
```

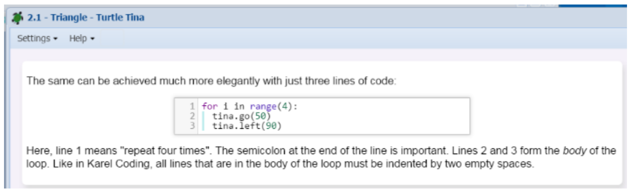
But, lines 1 and 2 are just repeated four times which makes the program too long!

In order to make a square, Tina repeats the same 2-step sequence of moving forward and turning 90 degrees four times.

We want our programs to be efficient. Long programs are more error prone and difficult to manage.

Instead, we can write a repeat loop that will tell Tina to do the sequence 4 times.

Students who have learned Tina will remember the Repeat command. In Tina, the command looks somewhat different.



The same can be achieved much more elegantly with just three lines of code:

```
1 for i in range(4):
2   tina.go(50)
3   tina.left(90)
```

Here, line 1 means "repeat four times". The semicolon at the end of the line is important. Lines 2 and 3 form the *body* of the loop. Like in Karel Coding, all lines that are in the body of the loop must be indented by two empty spaces.

The first command line is written as:

For i in range(4): range(4) means that the following lines will be repeated four times.

Note that the first line ends with a colon.

The next two lines form the body of the loop.

tina.go(50) These are the commands that will be repeated 4 times.

tina.left(90) The body commands must be indented 2 spaces from the first line.

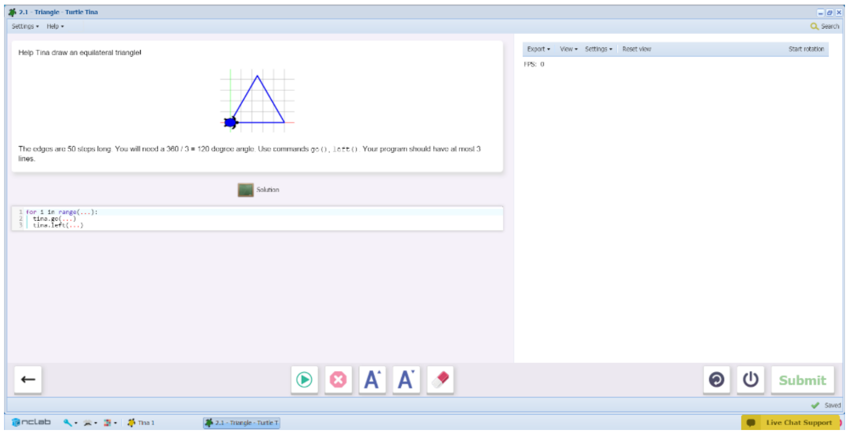
Stress the importance of syntax to the students. The program will not recognize the loop without the colon on the first line, and the indentations on the body text.

Students practice creating a loop to draw a triangle.

The geometry and steps are explained in the upper left hand box, along with what the drawing should look like when done.

Number of lines: 3

Commands: `go()`, `left()`

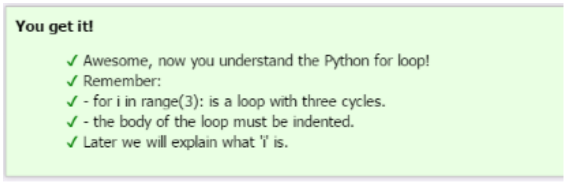


Help Tina draw an equilateral triangle!

The edges are 50 steps long. You will need a 360 / 3 = 120 degree angle. Use commands `go()`, `left()`. Your program should have at most 3 lines.

```
for i in range(...):
  tina.go(...)
  tina.left(...)
```

When done, students will see this comment that reviews the components of the loop.



You get it!

- ✓ Awesome, now you understand the Python for loop!
- ✓ Remember:
- ✓ - for i in range(3): is a loop with three cycles.
- ✓ - the body of the loop must be indented.
- ✓ Later we will explain what 'i' is.

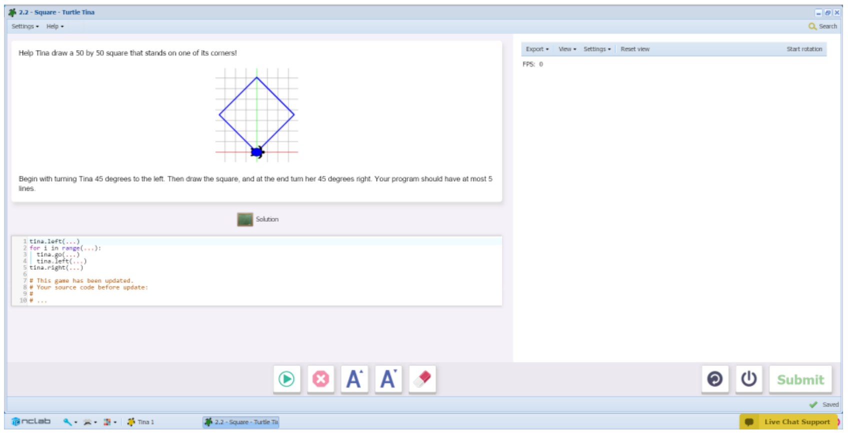
2.2 Square

Students practice writing loops, this time for a square turned 45 degrees.

Tina should end up in the origin position facing east.

Number of lines: 5

Commands: `go()`, `left()`, `right()`

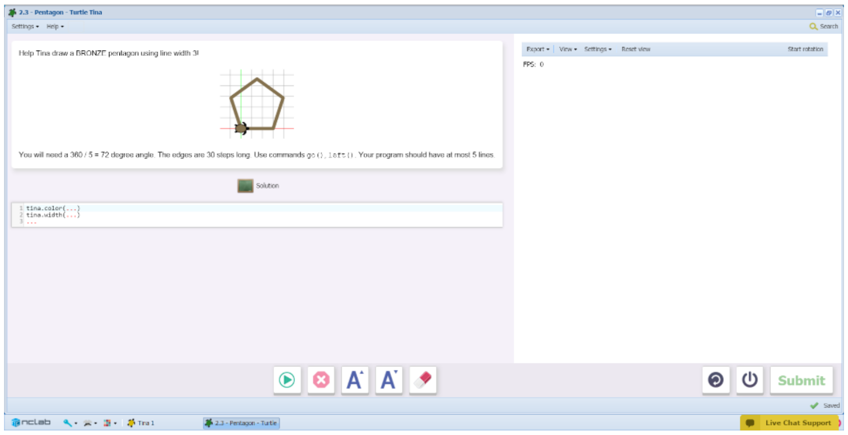


Help Tina draw a 50 by 50 square that stands on one of its corners!

Begin with turning Tina 45 degrees to the left. Then draw the square, and at the end turn her 45 degrees right. Your program should have at most 5 lines.

```
tina.left(...)
for i in range(...):
  tina.go(...)
  tina.left(...)
tina.right(...)

# This game has been updated.
# Your source code before update:
#
# ...
```

2.3 Pentagon



Help Tina draw a BRONZE pentagon using line width 3!

You will need a 360 / 5 = 72 degree angle. The edges are 30 steps long. Use commands `go()`, `left()`. Your program should have at most 5 lines.

```
tina.color(...)
tina.width(...)
...
```

Students practice making loops by drawing a pentagon. The width and color are specified.

This time, students write the entire loop themselves.

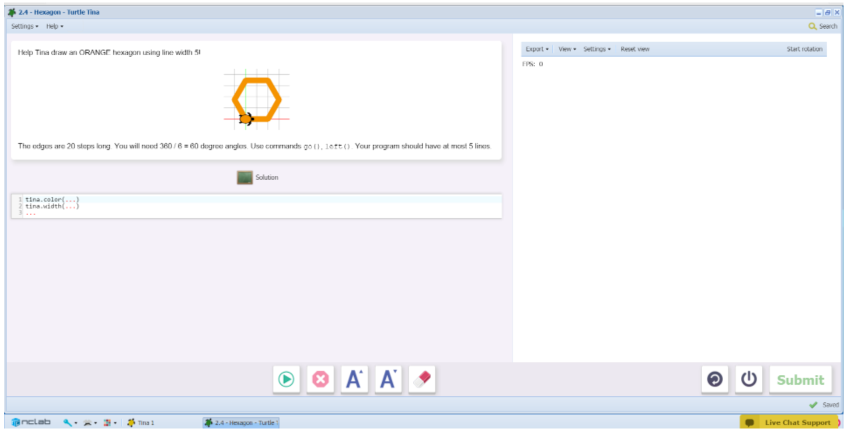
Number of lines: 5

Commands: `go()`, `left()`

If needed, remind students that the angles can be computed by dividing 360 degrees by the number of angles.

2.4 Hexagon

Students practice making loops by drawing a hexagon. The width and color are also specified.



Help Tina draw an ORANGE hexagon using line width 5!

The edges are 20 steps long. You will need 360 / 6 = 60 degree angles. Use commands `go()`, `left()`. Your program should have at most 5 lines.

```
tina.color(...)
tina.width(...)
...
```

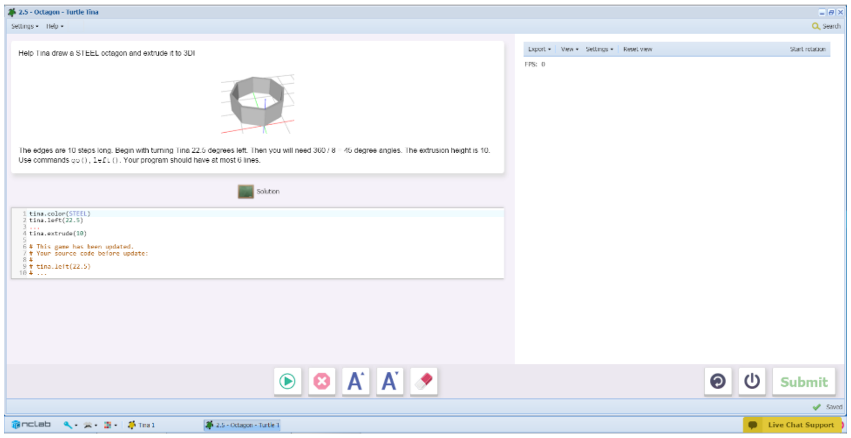
As in 2.3, students write the entire loop themselves.

Number of lines: 5

Commands: `go()`, `left()`

2.5 Octagon

Students practice making loops by drawing an octagon. The width and color are specified.



Help Tina draw a STEEL octagon and extrude it to 3D!

The edges are 10 steps long. Begin with turning Tina 22.5 degrees left. Then you will need 360 / 8 = 45 degree angles. The extrusion height is 10. Use commands `go()`, `left()`. Your program should have at most 6 lines.

```
tina.color(STEEL)
tina.left(22.5)
...
tina.extrude(10)

# This game has been updated.
# Your source code before update:
#
# tina.left(22.5)
# ...
```

As in 2.3, students write the entire loop themselves.

Number of lines: 6

Commands: `go()`, `left()`

2.6 Circle

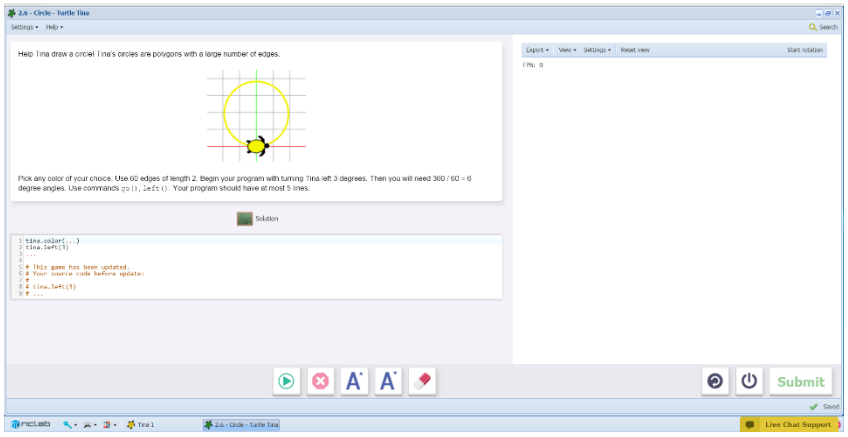
Students practice making loops by drawing a circle. The color is specified.

As in 2.3, students write the entire loop themselves.

Number of lines: 5

Commands: `go()`, `left()`

In this case, the "circle" is a sixty-sided polygon. This is how programs actually draw circles: the more sides, the smoother the circle.


2.6 - Circle - Turtle Tina
Settings ▾ Help ▾
Export ▾ View ▾ Settings ▾ Reset view Start rotation
FPS: 0
Solution
Submit
Saved
Live Chat Support

Help Tina draw a circle! Tina's circles are polygons with a large number of edges.

Pick any color of your choice. Use 60 edges of length 2. Begin your program with turning Tina left 3 degrees. Then you will need 360 / 60 = 6 degree angles. Use commands `go()`, `left()`. Your program should have at most 5 lines.

```
tina.color(...)
tina.left(3)
...

# This game has been updated.
# Your source code before update:
#
# tina.left(3)
# ...
```

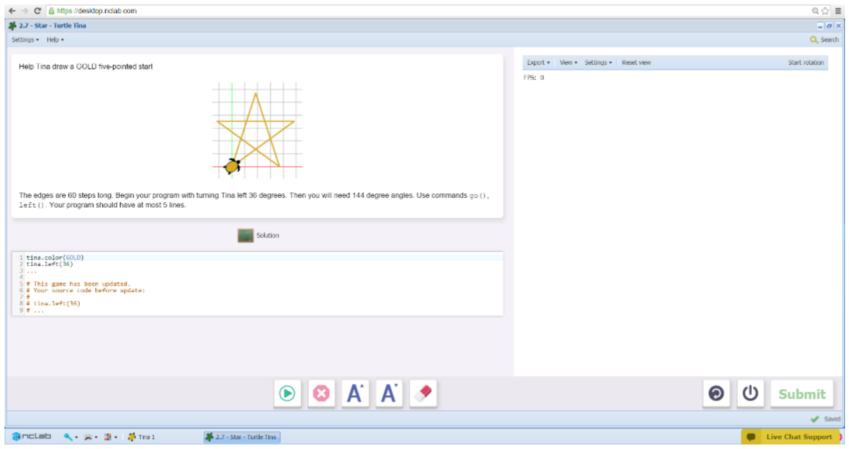
2.7 Star

Students practice making loops by drawing a star. The color is specified.

As in 2.3, students write the entire loop themselves.

Number of lines: 6

Commands: `go()`, `left()`



Help Tina draw a GOLD five-pointed star!

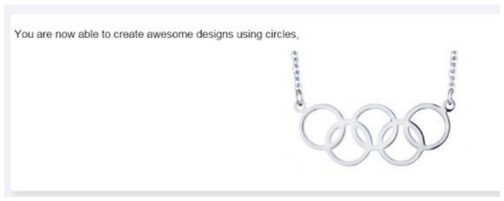
The edges are 60 steps long. Begin your program with turning Tina left 36 degrees. Then you will need 144 degree angles. Use commands `go()`, `left()`. Your program should have at most 5 lines.

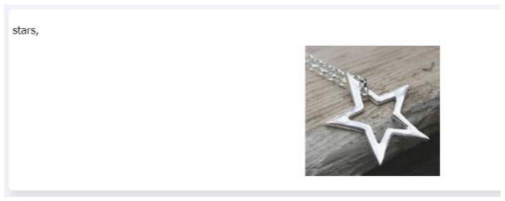
```
tina.color(GOLD)
tina.left(36)
...

# This game has been updated.
# Your source code before update:
#
# tina.left(36)
# ...
```

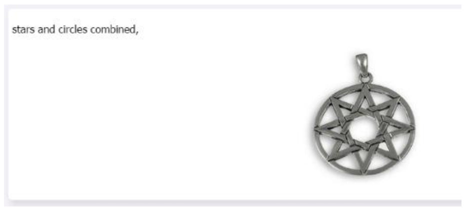
Note that turning angle is the supplementary angle (144 degrees) to the angle formed by the rays that make the star's points (36 degrees).

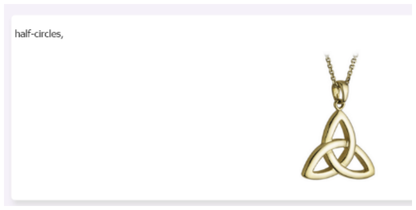
2.8 Art Project

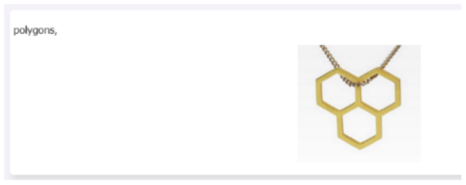
Students will create a pendant as their project for Section 2. Several screens show examples that can be created.

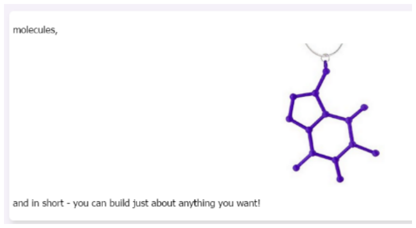

You are now able to create awesome designs using circles,


stars,


stars and circles combined,


half-circles,


polygons,

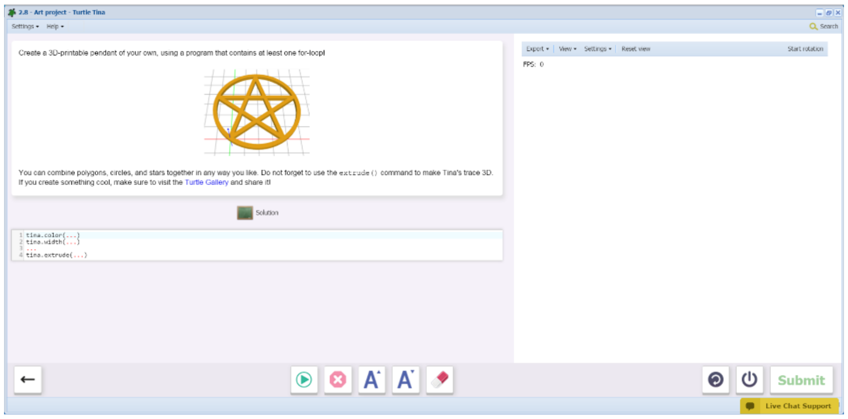

molecules,

and in short - you can build just about anything you want!

The project requires at least one loop**.**

The figure must also be extruded in order to print.

The number of lines and commands used will vary.



Create a 3D-printable pendant of your own, using a program that contains at least one for-loop!

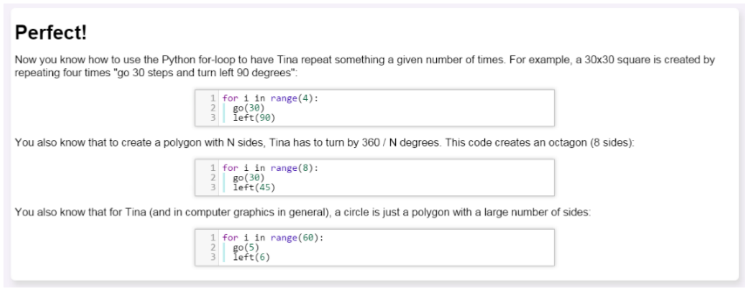
You can combine polygons, circles, and stars together in any way you like. Do not forget to use the `extrude()` command to make Tina's trace 3D. If you create something cool, make sure to visit the Turtle Gallery and share it!

```
tina.color(...)
tina.width(...)
...
tina.extrude(...)
```

Students will see this message:

Make sure to visit http://nclab.com/turtle-designs to share your design!

The next screen summarizes the main points which have been learned in Section 2.



Perfect!

Now you know how to use the Python for-loop to have Tina repeat something a given number of times. For example, a 30x30 square is created by repeating four times "go 30 steps and turn left 90 degrees":

```
for i in range(4):
  go(30)
  left(90)
```

You also know that to create a polygon with N sides, Tina has to turn by 360 / N degrees. This code creates an octagon (8 sides):

```
for i in range(8):
  go(30)
  left(45)
```

You also know that for Tina (and in computer graphics in general), a circle is just a polygon with a large number of sides:

```
for i in range(60):
  go(5)
  left(6)
```

After completing this section, students earn a Yellow Belt of First Degree which they can print or share on social media.

Possible questions for post-session discussion:

What commands are used for all of these figures? (go, left)

How do you decide what angle to turn? (Divide 360 by the number of turns)

What happens if you change the number of times i is repeated (the range)?

What happens if you change the degrees turned even slightly?

Assessment:

Students will receive a printable “Yellow Belt” certificate upon completion of Section 2. See Assessment section for journal and project ideas.

Students Journal entries can be used as assessment.

The Art Project can be used as a performance task or portfolio artifact. Here is a possible 50-point assignment card:

SECTION 2 ART PROJECT: CREATE A PENDANT (50 POINTS)

Objective: Create and publish a pendant design in Tina Turtle.

Planning (15 points): After you have decided on a design, draw out the shapes on grid paper or in your journal.

- Describe the shapes. What commands will be needed to draw them? Name each shape (for example, Gold Star) and describe the color, starting position, line lengths, turning angles, and coordinates for the goto command if needed.
- Write out the order in which you will program the shapes and lines in the pendant design. Where do these components attach to each other? When will the up, down, and back commands be useful?

Writing the code (25 points): Write the program.

- Before you write the code for each shape, write a comment line that names and describes the shape.
- Write the code and test as you go.
- Use at least For one loop in your design, including the beginning line and the body of the loop.
- Use some or all of the commands learned in Section 1: `go(), left(), right(), back(), up(), down()` and `goto()`

Saving the file, sharing and publishing (10 points)

- Publish the design to your folder. Inform someone else about the game by providing the link on ____, or by email.

SECTION 3 LOOPS II: LEVELS 3.1-3.8

Objectives: In this section, students use For loops to build more complex objects, including linear and rotated patterns.

Vocabulary: This is a good time to review some of the terms used in programming.

> **Algorithm:** a series of logical steps that leads to the solution of a task. Students may be familiar with algorithms used in operations such as subtraction and long division.
>
> **Logical error:** a mistake in an algorithm. Planning helps reduce the number of errors.
>
> **Computer Program:** An algorithm written using a programming language.
>
> **Syntax:** the way a command line is written.
>
> **Syntax error:** a mistake in spelling, operators, indentations, spaces

Time required:

Time required will vary based on student ability and experience. Most students will complete this section in two hours.

Prerequisite skills: Completion of Section 2.

Background knowledge/Introductory Set/Purpose:

Why do we make patterns? For real life examples, have students look for objects that are made up of repeated patterns (examples: a length of chain, a basket, a sweater). In programming, we look for repeated patterns when building complex objects, so that our code is simple, manageable and consistent.

Purpose: Section 3 (Levels 3.1-3.8) teaches how to build more complex shapes than the simple polygons of Section 2.

Direct Instruction and Modeling:

Show the videos in 3.1 on a projector. Discuss any questions that arise. http://youtu.be/sFiTiTVJR8E

Model or guide the first lesson (3.1), based on student needs. Otherwise, there are no new commands to be learned in this section. Additional instruction may be needed, and this is specifically addressed in each level.

Individual/Group practice:

The program is designed to be used individually by students. Encourage peer support, sharing and discussion.

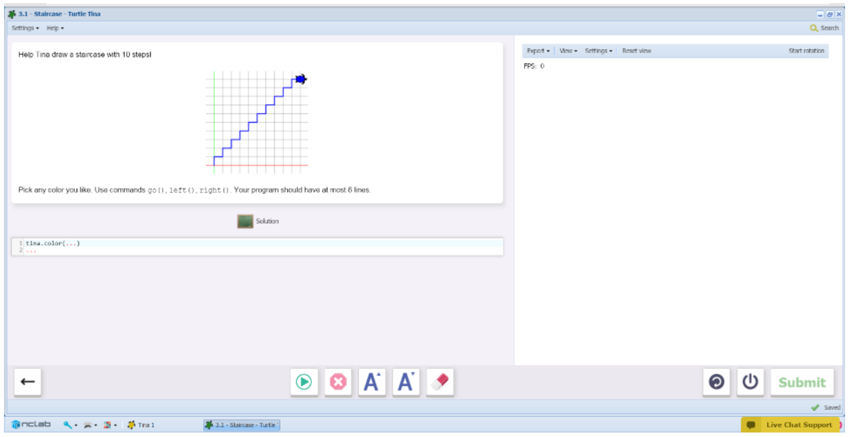
Self-paced Instruction: Levels 3.1-3.7

3.1 Staircase

Staircase begins by showing the For-Loops video again as a refresher. The video can be viewed at

http://youtu.be/sFiTiTVJR8E

After picking a color, students create a staircase using a For-Loop.



Help Tina draw a staircase with 10 steps!

Pick any color you like. Use commands go(), left(), right(). Your program should have at most 6 lines.

```
tina.color(...)
...
```

Reminder: colors are all-caps.

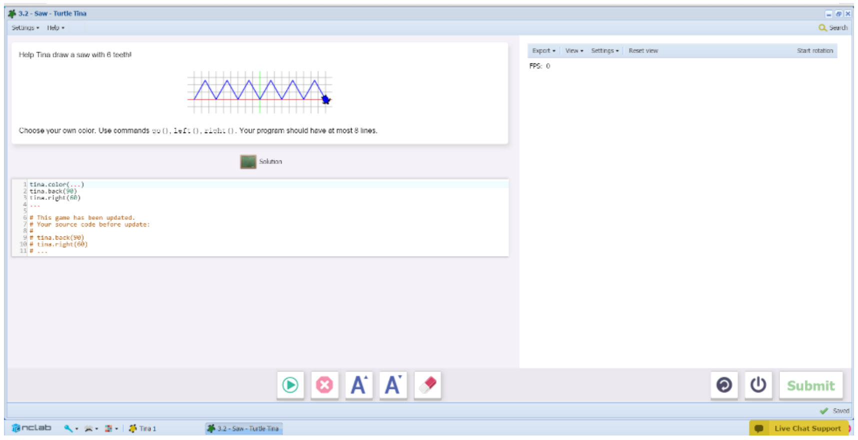
Number of lines: 6

Commands: `go(), left(), right()`

Reminder: For-Loop automatically indents the body text if the syntax is correctly written. The most common mistake is to forget the colon. If it is there, the indent will show up.

3.2 Saw

Students create a sawtooth pattern using a For-Loop.



Help Tina draw a saw with 6 teeth!

Choose your own color. Use commands go(), left(), right(). Your program should have at most 8 lines.

```
tina.color(...)
tina.back(90)
tina.right(60)
...

# This game has been updated.
# Your source code before update:
#
# tina.back(90)
# tina.right(60)
# ...
```

Students can pick any color.

The pattern's center point is the origin, so the program starts by backing up Tina 90 spaces.

To start the pattern, Tina is turned 60 degrees to the right.

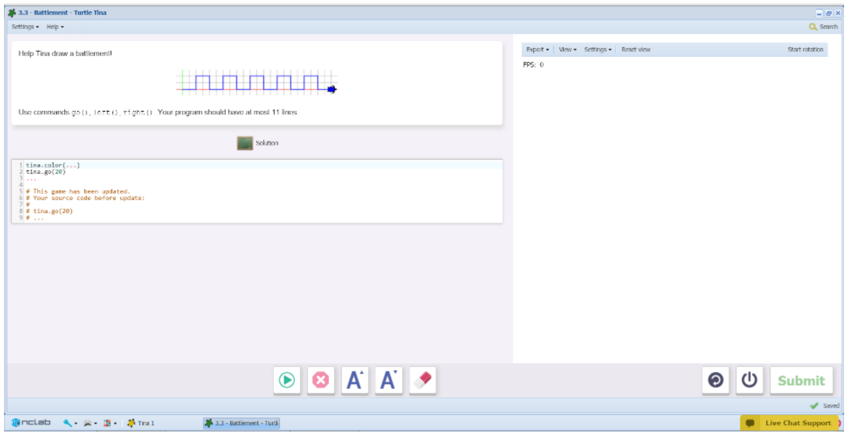
Number of lines: 8

Commands: `go(), left(), right()`

There are a number of ways to turn Tina as she is drawing the sawtooth. Have students compare solutions. Did they end up with Tina in the same orientation as the example shown in the upper left? Which solution is the most elegant (simple, effective and easy to understand)?

3.3 Battlement

Students practice creating a For-Loop to build a battlement. The loop contains several commands.



Help Tina draw a battlement!

Use commands go(), left(), right(). Your program should have at most 11 lines

```
tina.color(...)
tina.go(20)
...

# This game has been updated.
# Your source code before update:
#
# tina.go(20)
# ...
```

Encourage students to try different color names: for example, fruits or metals. The Python library is quite extensive!

Ask them: why do you think we started with Tina going 20 steps before beginning the loop? (We want her to end the pattern in a particular position)

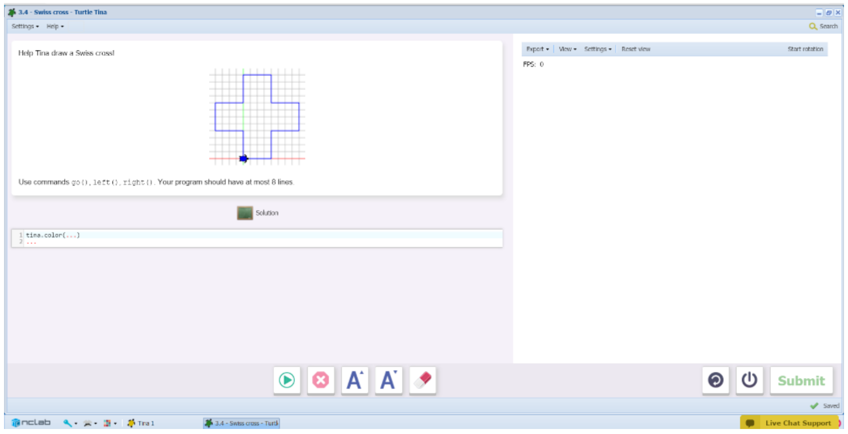
Number of lines: 11

Commands: `go(), left(), right()`

If you have access to an oscilloscope, it might be interesting to show students both the sawtooth and square wave patterns that they have just created. Ask them if they can hear the shapes.

3.4 Swiss Cross

Students practice writing a multi-step For-Loop.



Help Tina draw a Swiss cross!

Use commands go(), left(), right(). Your program should have at most 8 lines.

```
tina.color(...)
...
```

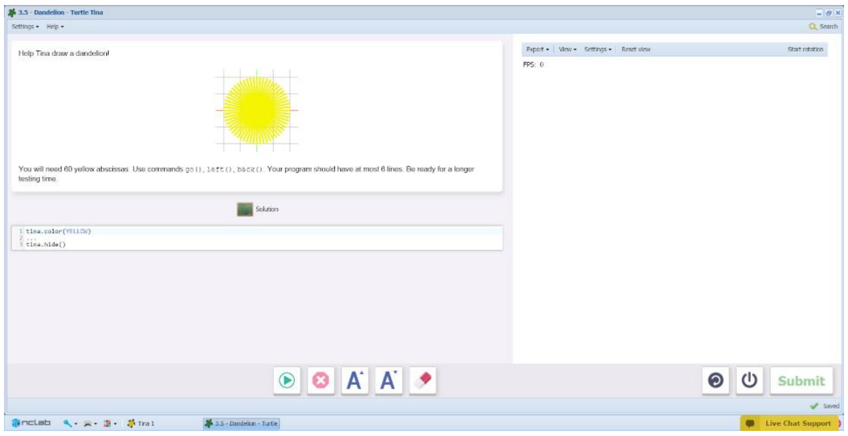
It is fun to see the results of wrong programing: the shapes can be quite interesting! For students who visualize the geometry and solve the problem quickly, suggest that they play with the code to see what happens. Mistakes lead to creativity!

Number of lines: 8

Commands: `go(), left(), right()`

3.5 Dandelion

Students practice the For Loop with a large number of repetitions (60).



Help Tina draw a dandelion!

You will need 60 yellow abscissas. Use commands go(), left(), back(). Your program should have at most 6 lines. Be ready for a longer testing time.

Solution

```
tina.color(YELLOW)
...
tina.hide()
```

For students who do not know the term, here is the definition of abscissa:

Abscissa: the perpendicular distance of a point from the vertical axis. Usually this is the horizontal coordinate of a point in a two-dimensional rectangular Cartesian coordinate system.

Students can also think of each petal as a ray starting from the origin.

The program itself is quite simple, but it has a lot of iterations and will take longer to run.

This time, there are fewer hints. Students count the interval themselves (20). They can also figure out the degrees to turn by dividing 360 by 60.

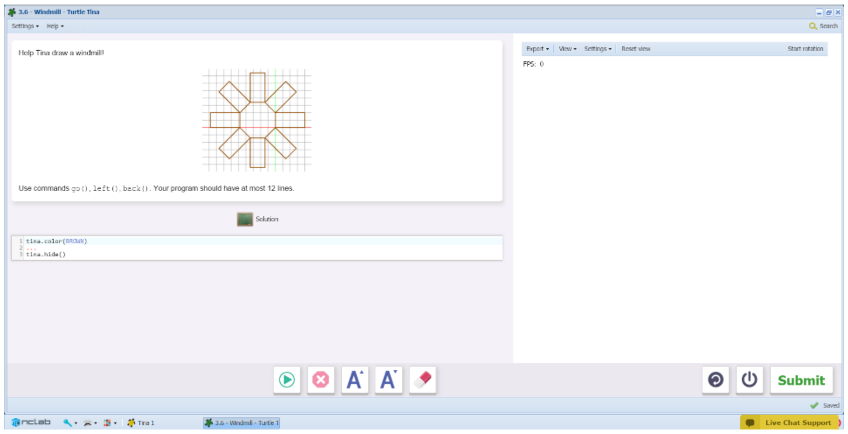
Tina needs to back up before drawing the next abscissa.

Number of lines: 6

Commands: `go(), left(), back()`

3.6 Windmill

Students practice the For Loop by drawing a windmill.



Help Tina draw a windmill!

Use commands go(), left(), back(). Your program should have at most 12 lines.

Solution

```
tina.color(BROWN)
...
tina.hide()
```

Encourage students to analyze the shape and explain their thinking. This can be done as a discussion (pair-share, small group, whole class) and/or as a written explanation. For example:

"The windmill is made up of eight rectangles. Tina needs to draw a rectangle, back up and turn to get ready to draw the next rectangle. There are eight rectangles in all, so she needs to turn 360/8 or 135 degrees."

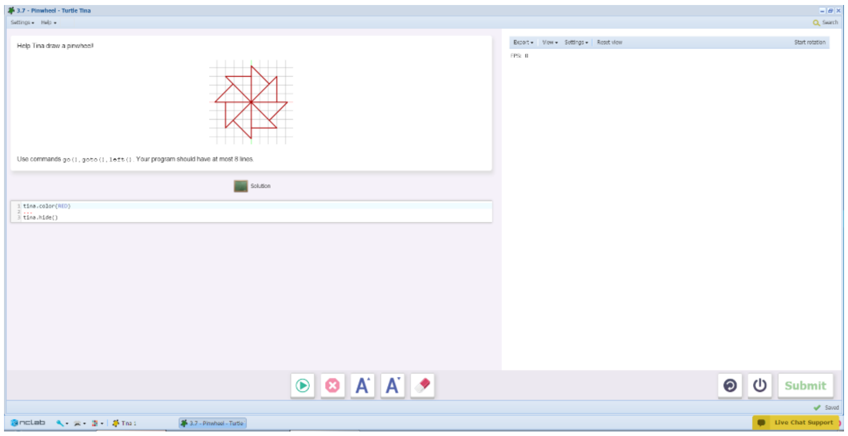
Some students might notice that there are repeated patterns within the loop. Nested loops are learned in later sections.

Number of lines: 12

Commands: `go(), left(), back()`

3.7 Pinwheel

Students practice the For Loop by drawing a pinwheel.



Help Tina draw a pinwheel!

Use commands `go()`, `goto()`, `left()`. Your program should have at most 8 lines.

```
tina.color(RED)
...
tina.hide()
```

Questions for students:

What type of triangle is being repeated?

- Isoceles (2 equal sides)
- Right angle (1 right angle)

How many degrees are in the other two angles?

- They each turn 45 degrees.
  Students can practice writing and solving this as an algebraic equation. 2(a) = 180 – 90 → a =(180-90)/2 → a=90/2 → a=45

Why are we using the `goto` command instead of specifying the distance?

- The actual distance is not an integer or simple decimal (in this case, it is 42.4264068712).
- The goto command has the additional benefit of returning Tina to the origin each time.

How do we figure out the angle to turn Tina to make the next triangle?

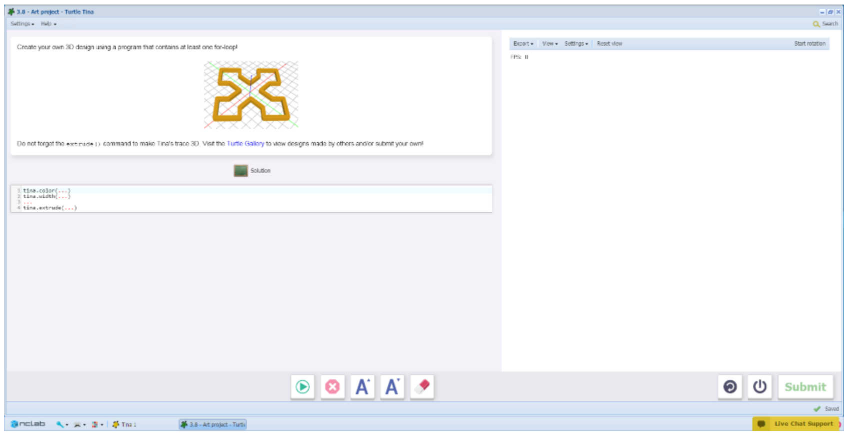
- By now, students should recognize that the turn angle to the next triangle is based on 360/n, where n = the number of objects, in this case 8. If turning to the right, the angle will be 360/8 = 45 degrees. If turning to the left, the angle will be the supplement, or 135 degrees.

Number of lines: 8

Commands: `go(), goto(), left()`

3.8 Art Project

Students create a 3D shape based on a repeated shape, as they have learned in Section 3. The shape should be extruded so that it is printable. Encourage students to visit the Turtle Gallery and submit their own drawings for publication.



Create your own 3D design using a program that contains at least one for-loop!

Do not forget the `extrude()` command to make Tina's trace 3D. Visit the Turtle Gallery to view designs made by others and/or submit your own!

```
tina.color(...)
tina.width(...)
...
tina.extrude(...)
```

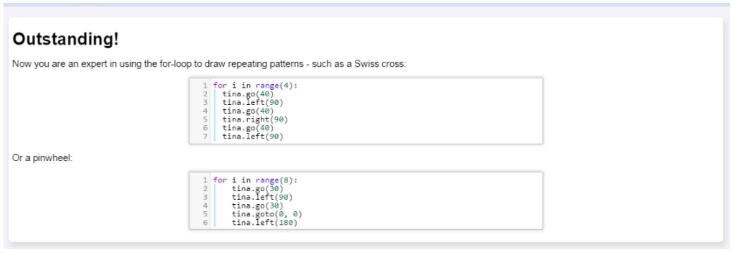
The color, width and extrude lines are already set up.

Students should draw some designs on grid paper first.

Once the object is created and saved to a file, encourage students to play with the attributes (color, width, extrude, angles, line lengths) and see how the shape changes.

After submitting the design, students will see a review of the new ideas they have learned on the next screen.

They will also earn a yellow belt of second degree, which can be printed, saved, emailed or shared.



Outstanding!

Now you are an expert in using the for-loop to draw repeating patterns - such as a Swiss cross:

```
for i in range(4):
  tina.go(40)
  tina.left(90)
  tina.go(40)
  tina.right(90)
  tina.go(40)
  tina.left(90)
```

Or a pinwheel:

```
for i in range(8):
    tina.go(30)
    tina.left(90)
    tina.go(30)
    tina.goto(0, 0)
    tina.left(180)
```

Questions for post-session discussion:

What are the benefits of writing loops into programs?

Review: why do we use the `back()` and `goto()` commands?

Assessment:

Assessment is built into the program. Students must complete a level successfully in order to unlock the next level. Students will receive a printable "Yellow Belt of Second Degree" certificate upon completion of Section 3. See Assessment section for journal and project ideas.

Students Journal entries can be used as assessment.

The Art Project can be used as a performance task or portfolio artifact. Here is a possible 50-point assignment card:

END OF SECTION 3: ART PROJECT (50 POINTS)

Objective: Create and publish a design made of repeated shapes and patterns in Tina Turtle.

Planning (15 points): After you have decided on a design, draw out the shapes on grid paper or in your journal.

- Describe the design. What commands will be needed? Name the object or pattern that you are designing.
- Looking at the design, decide which commands will be used. Will you need the `back()` command to start the next part of the pattern? Where will you use `goto()` instead of `go()`?
- If your design is composed of more than one repeated shape or pattern, write out the order in which you will place them in the program. The `color`, `width` and `extrude` program lines are already set up. Include the values that you use for each one in your design notes.

Writing the code (25 points): Write the program.

- Remember to include comment lines that describe what your program is doing.
- Write the code and test as you go.
- Use at least one For loop in your design, including the beginning line and the body of the loop.
- Use some or all of the commands learned in Section 1 and 2: `go(), left(), right(), back(), up(), down()` and `goto()`

Saving the file, sharing and publishing (10 points)

- Publish the design to your folder. Inform someone else about the game by providing the link on ______________________________________, or by email.

SECTION 4 NESTED LOOPS: LEVELS 4.1-4.8

Objectives: In this section, students learn to create nested For loops, or repeated patterns within repeated patterns.

Vocabulary:

Nested loops: a nested loop is a repeated pattern or loop, that is part of another repeated pattern, or loop. For example: if I want to make a row of 10 triangles, I can write a loop for to make a triangle, and nest it inside a loop that will draw that triangle ten times in a row.

Time required: Time required will vary based on student ability and experience. Most students will complete this section in about 2 hours, with another hour for the Art Project.

Prerequisite skills:

Completion of Section 3. Review geometry and algebra concepts as needed.

Background knowledge/Introductory Set/Purpose:

Look at examples of patterns within patterns (such as chains, braids, tessellations, knitted patterns, shells, etc.). Each of these can be thought of as nested loops. A set of instructions creates a pattern, and then this pattern is used to build another patter.

Purpose: Section 4 (Levels 4.1-4.7) introduces writing nested For loops.

Direct Instruction and Modeling:

Show the video in 4.1 that explains how to write a nested loop.

http://youtu.be/2QcdvLgXmbI

Model how to type the nested loop in the Level 4.1 lesson, with special attention to the indentations.

Additional suggestions for instruction and activities are included in several levels.

Individual/Group practice:

The program is designed to be used individually by students. Encourage peer support, sharing and discussion.

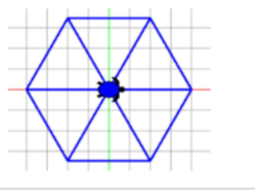
Help card for Section 4: Nested Loops

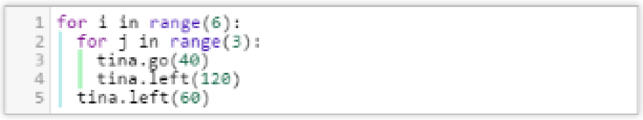
Writing Nested For Loops (Example from 4.1)

for i in range(6):	repeat 6 times, one for each triangle
for j in range(3):	repeat 3 times to make the triangle. The j loop must be indented 2 spaces because it is part of the for i loop

tina.go(40)
tina.left(120)
	body of the j loop
	indent two more spaces (4 total) to show that these lines are part of the j loop

tina.left(60)	This left turn is part of the i loop, so it is only indented two spaces. It will repeat the triangle created by j around the origin every 60 degrees.





```
for i in range(6):
  for j in range(3):
    tina.go(40)
    tina.left(120)
  tina.left(60)
```

Self-paced Instruction: Levels 4.1-4.7 (All previous commands may be needed but not necessarily listed under each lesson. Defined objects are listed)

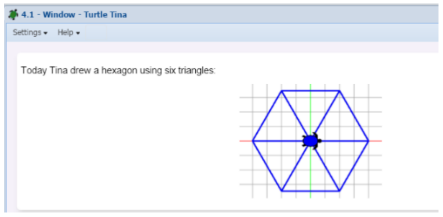
4.1 Window

4.1 starts with a YouTube video on nested loops, which can be watched from within the program or by following this link:

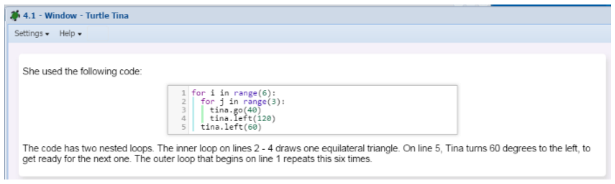
http://youtu.be/2QcdvLgXmbI

This video should whet the students' appetites for playing with the patterns to generate more complex ones. In this Section, students will make all of the shapes shown in the video.

A demonstration on the next two screens shows Tina creating a hexagon made of triangles rotated about the origin. The repeat loop for created the triangle is nested inside the repeat loop that reproduces the triangle. Each loop is identified by a different letter: the inner loop is *j*, the outer loop is *i*.



Today Tina drew a hexagon using six triangles:



She used the following code:

```
1 for i in range(6):
2   for j in range(3):
3     tina.go(40)
4     tina.left(120)
5   tina.left(60)
```

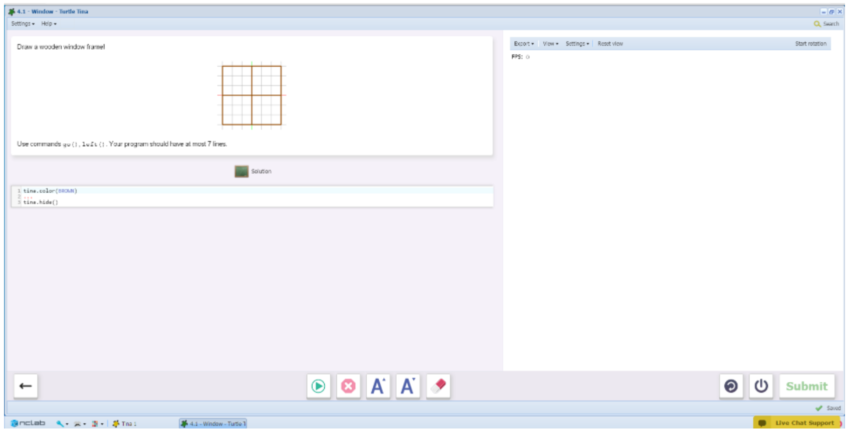
The code has two nested loops. The inner loop on lines 2 - 4 draws one equilateral triangle. On line 5, Tina turns 60 degrees to the left, to get ready for the next one. The outer loop that begins on line 1 repeats this six times.

Students then practice writing nested loops by creating a window made of four squares.

Students write the code for the two loops without prompts.

Number of lines: 7

Commands: `go()`, `left()`



Draw a wooden window frame!

Use commands `go()`, `left()`. Your program should have at most 7 lines.

```
tina.color(BROWN)
...
tina.hide()
```

4.2 Mosaic I

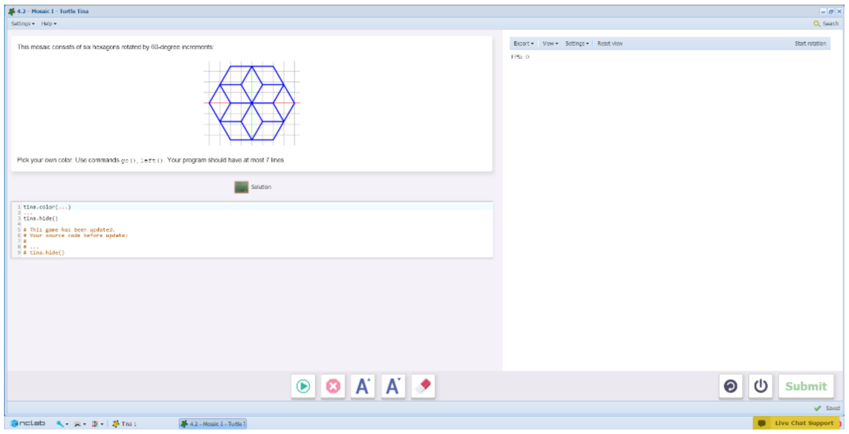
Students practice writing nested loops, this time with a hexagon repeated six times.

Number of lines: 7

Commands: `go(), left()`

Here is a hands-on activity that could go along with nested loops:

- Use one type of pattern block tiles and create a repeated pattern.
- Measure angles on the shapes and relate these angles to the ones they are writing in the program.



This mosaic consists of six hexagons rotated by 60-degree increments:

Pick your own color. Use commands `go()`, `left()`. Your program should have at most 7 lines.

```
tina.color(...)
...
tina.hide()

# This game has been updated.
# Your source code before update:
#
# ...
# tina.hide()
```

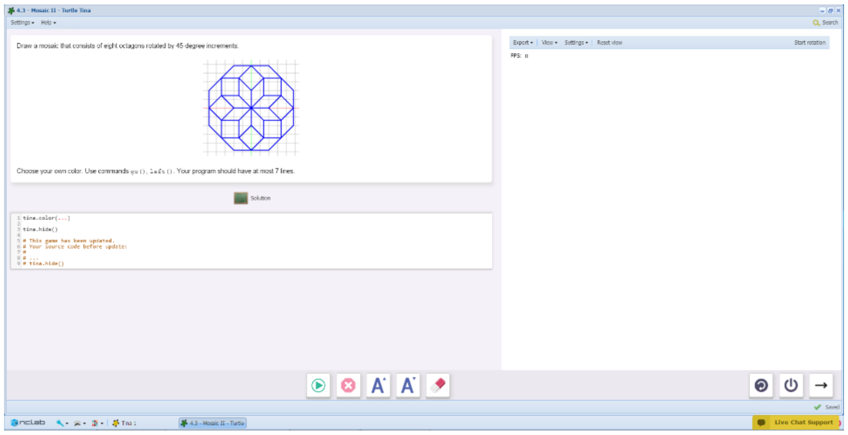
4.3 Mosaic II

Students practice writing nested loops, this time with an octagon repeated eight times.

Number of lines: 7

Commands: `go(), left()`

As in 4.2, this shape can be compared to one built with pattern blocks.



Draw a mosaic that consists of eight octagons rotated by 45-degree increments.

Choose your own color. Use commands `go()`, `left()`. Your program should have at most 7 lines.

```
tina.color(...)

tina.hide()

# This game has been updated.
# Your source code before update:
#
# ...
# tina.hide()
```

4.4 Mosaic III

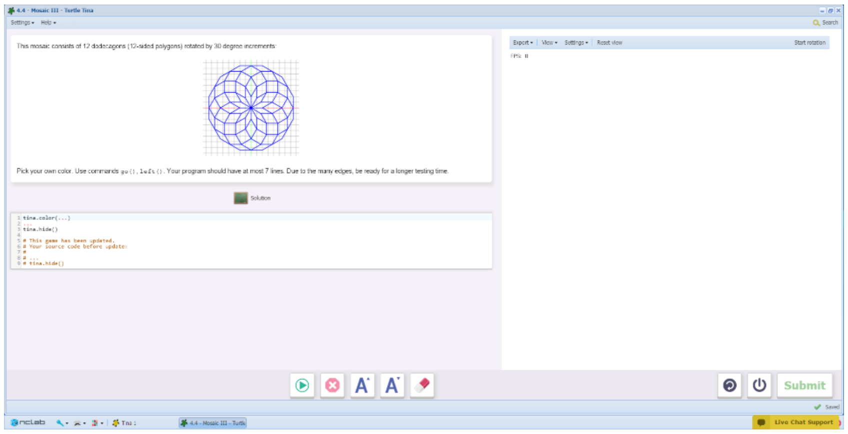
Students practice writing nested loops, this time with a dodecahedron repeated twelve times.

Number of lines: 7

Commands: `go()`, `left()`

As in 4.2, this shape can be compared to one built with pattern blocks.

Due to the number of computations, this program takes a little longer to run.



This mosaic consists of 12 dodecagons (12-sided polygons) rotated by 30 degree increments:

Pick your own color. Use commands `go()`, `left()`. Your program should have at most 7 lines. Due to the many edges, be ready for a longer testing time.

```
tina.color(...)
...
tina.hide()

# This game has been updated.
# Your source code before update:
#
# ...
# tina.hide()
```

4.5 Sunflower

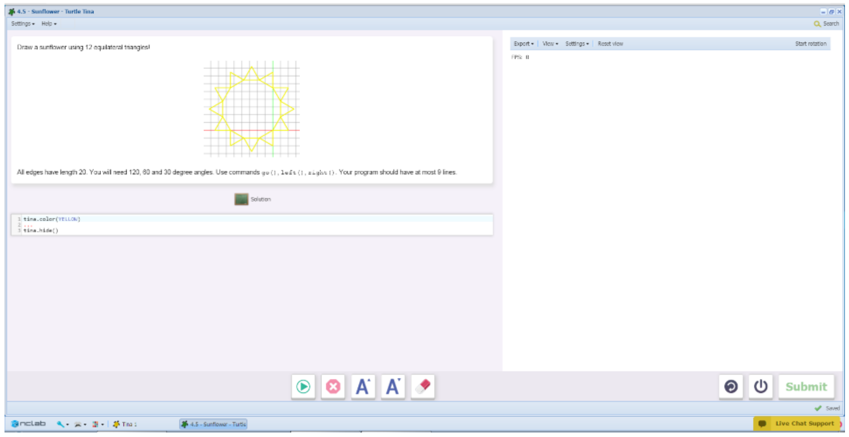
Students practice writing nested loops by drawing a sunflower figure composed of triangles.

Number of lines: 9

Commands: `go()`, `left()`, `right()`

The edge length (20) and angles (30, 60, 120) are specified in the instructions.

Hint: think in a counterclockwise direction. Where would the next triangle start?



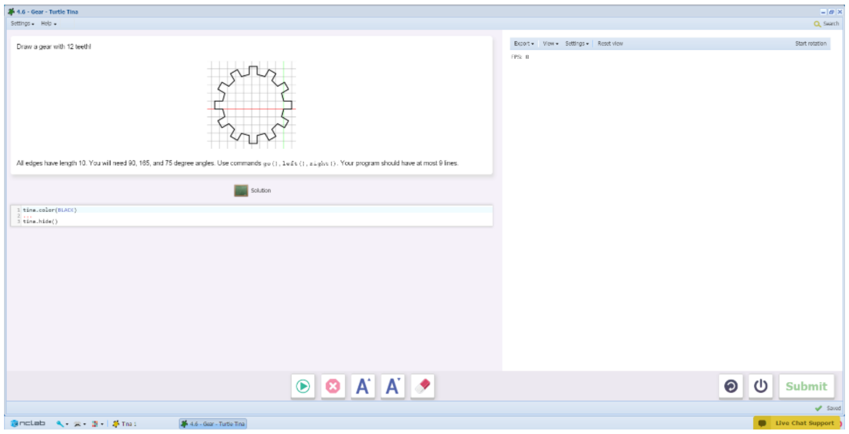
Draw a sunflower using 12 equilateral triangles!

All edges have length 20. You will need 120, 60 and 30 degree angles. Use commands `go()`, `left()`, `right()`. Your program should have at most 9 lines.

```
tina.color(YELLOW)
...
tina.hide()
```

4.6 Gear

Students practice writing nested loops by drawing a gear figure.



Draw a gear with 12 teeth!

All edges have length 10. You will need 90, 165, and 75 degree angles. Use commands `go()`, `left()`, `right()`. Your program should have at most 9 lines.

```
tina.color(BLACK)
...
tina.hide()
```

Number of lines: 9

Commands: `go()`, `left()`, `right()`

Edge length: 10

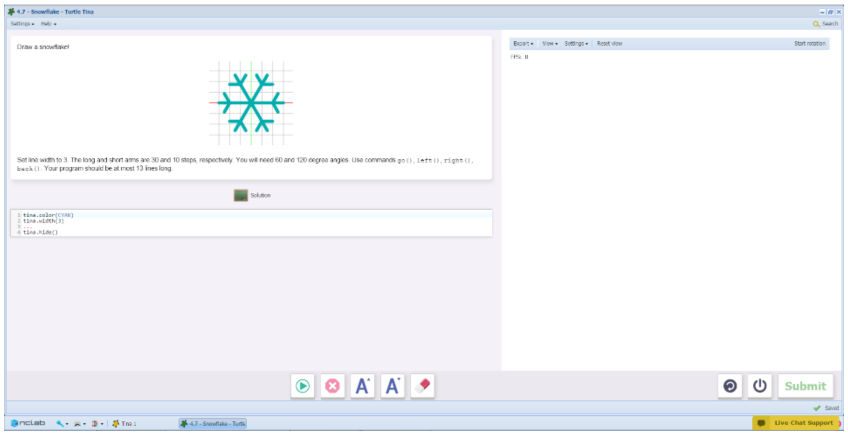
Angles used: 90, 165, 75

Math Challenge: how are the angles 165 and 75 derived?

- There are 12 partial squares in the figure.
- 360 degrees/12=30 degrees.
- Each partial square is rotated 30 degrees from the previous one, but the starting point is different.
- We split the rotation into two segments, because Tina has to move an interval of 10 along the first direction, and then rotate to start the next partial square.
    - 180-165=15 degrees to the right
    - 90-75=15 degrees to the right
    - Tina needs to rotate 15+15=30 degrees to start the next partial square.

4.7 Snowflake

Students practice writing nested loops by drawing a snowflake.



Draw a snowflake!

Set line width to 3. The long and short arms are 30 and 10 steps, respectively. You will need 60 and 120 degree angles. Use commands `go()`, `left()`, `right()`, `back()`. Your program should be at most 13 lines long.

```
tina.color(CYAN)
tina.width(3)
...
tina.hide()
```

Number of lines: 13

Commands: `go()`, `left()`, `right()`, `back()`

Edge length:

- 30 for the long arms
- 10 for the short arms

Line width: 3

Angles used: 60, 120

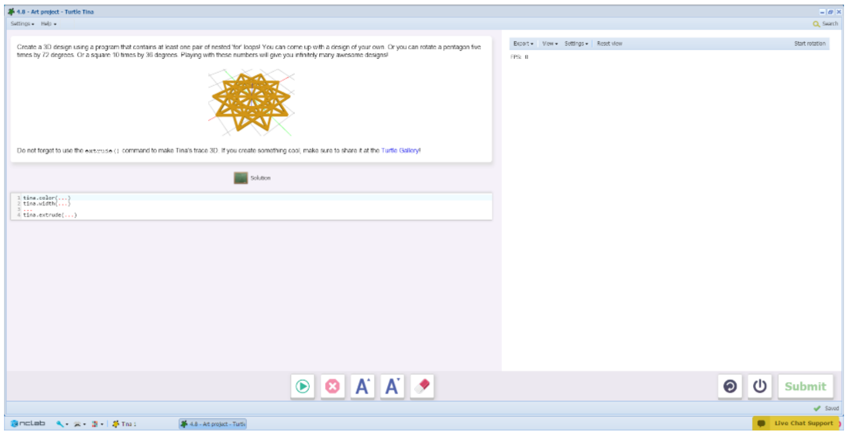
Hint: The length 30 is made up of 20 + 10

4.8 Art Project

Students create their own rotated geometric pattern using nested loops.

Reminder: patterns must be extruded in order to print as a 3D object.

Files should be saved, then shared. Files can be submitted to the Tina Turtle Gallery.



Create a 3D design using a program that contains at least one pair of nested 'for' loops! You can come up with a design of your own. Or you can rotate a pentagon five times by 72 degrees. Or a square 10 times by 36 degrees. Playing with these numbers will give you infinitely many awesome designs!

Do not forget to use the `extrude()` command to make Tina's trace 3D. If you create something cool, make sure to share it at the Turtle Gallery!

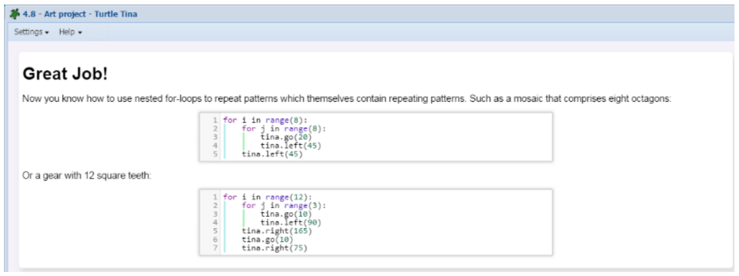
```
tina.color(...)
tina.width(...)
...
tina.extrude(...)
```

Possible questions for post-session discussion:

Review concepts learned in Section 4.

Students are reminded of which patterns they have learned on the screen that follows Art Project.

What is the main difference between the code for the mosaic and the code for the gear?



Great Job!

Now you know how to use nested for-loops to repeat patterns which themselves contain repeating patterns. Such as a mosaic that comprises eight octagons:

```
for i in range(8):
    for j in range(8):
        tina.go(20)
        tina.left(45)
    tina.left(45)
```

Or a gear with 12 square teeth:

```
for i in range(12):
    for j in range(3):
        tina.go(10)
        tina.left(90)
    tina.right(165)
    tina.go(10)
    tina.right(75)
```

- The mosaic is just rotated around the origin by the number of times specified in the outer loop.
- The starting position for each partial square must be moved in the outer loop before starting the next partial square.

Decompose angles to make different patterns (as in Gear), or lengths of a ray (as in Snowflake).

Assessment:

Assessment is built into the program. Students must complete a level successfully in order to unlock the next level. Students will receive a printable "Yellow Belt of Third Degree" certificate upon completion of Section 4. See Assessment section for journal and project ideas.

Students Journal entries can be used as assessment.

The Art Project can be used as a performance task or portfolio artifact. Here is a possible 50-point assignment card:

END OF SECTION 4: ART PROJECT (50 POINTS)

Objective: Create and publish a design created by at least one pair of nested loops, such as a rotated shape or pattern, in Tina Turtle.

Planning (15 points): After you have decided on a design, draw out the shapes on grid paper or in your journal.

- Describe the design. What commands will be needed? Name the object or pattern that you are designing. If the drawing is complex, just draw the first few repetitions of the pattern.
- Looking at the design, decide which commands will be used. Will you need the `back()` command to start the next part of the pattern? Where will you use `goto()` instead of `go()`?
- If your design is composed of more than one repeated shape or pattern, write out the order in which you will place them in the program. The `color`, `width` and `extrude` program lines are already set up. Include the values that you choose for each one in your design notes.

Writing the code (25 points): Write the program.

- Remember to include comment lines that describe what your program is doing.
- Write the code and test as you go.
- Use at least one nested For loop in your design.
- Use some or all of the commands learned in Section 1 and 2:

  `go(), left(), right(), back(), up(), down()` and `goto()`

Saving the file, sharing and publishing (10 points)

- Publish the design to your folder. Inform someone else about the game by providing the link on ____, or by email.

SECTION 5 VARIABLES: LEVELS 5.1-5.8

Objectives: In this section, students learn to use variables in the For loop. Instead of i being defined as one value, a fixed number of equal steps, i can now represent a range of values. These values can change the outcome of each repetition of the loop. Any operation (addition, subtraction, multiplication, division) can be used to include i in the commands, creating interesting linear and non-linear designs.

Vocabulary:

Variable: in terms of programming, variable is the name and value of something that will be recorded in memory. In the For loop, the i is an **index or counting variable**. If we set i to a range of values, then i will change each time the loop starts the body of the program. If we then use i as part of a command, that command will output a different value each time.

Range: the range is the lower and upper limit of the variable i. Note that if the range is set with only one value, then the lower limit of the range is assumed to be 0, with the number in the parentheses being the upper limit. Important: the final value will be the difference between the upper and lower limit. For example, in range (1,11), the final value used in the program is 10, or 11-1.

Angle: `angle()` resets Tina to a specific angle. `tina.angle(0)` will reset the turtle to face east.

Time required:

Time required will vary based on student ability and experience. Most students will complete this section in about 3 hours. The Art Project will take about one hour.

Prerequisite skills:

Completion of Section 4. The level of difficulty in both concept and skill is increasing and you may find a divergence in rate of success among your students, especially in the younger grades. An understanding of patterns in operations (non-linear patterns in multiplication and division), properties of operations (associative, distributive) and the order of operations is helpful, since the variable is being used as part of the command expression. Negative numbers are used, so students will need some background in integers. Experience with function input/output tables will also help students understand the impact of increasing the variable each time the loop is run. Give students time to play with the values of lengths and angles, similar to the demonstration in the video, to get a feel for how the variable changes the program.

Background knowledge/Introductory Set/Purpose:

Explain the concepts of writing a program when you do not know the number of steps required in advance. This is when we need the **while** statement. In these lessons, while is used as "**while not home**"; in other words, the robot will repeat the actions until he reaches the home space.

How to use variables: show video (follow link on 5.1 or here) http://youtu.be/tJXRgo4M0qg

This video really is a case of "a picture is worth a thousand words". The results created by using the variable are impressive and inspiring.

Purpose: Section 5 (Levels 5.1-5.7) introduces the use of variables in For loops, and practices all previous concepts and commands.

Direct Instruction and Modeling:

Show and discuss the steps on the instructional screens in 5.1 to the students, described in detail in Level 5.1 below.

Individual/Group practice:

The program is designed to be used individually by students. Encourage peer support, sharing and discussion.

Help card for Section 5: Variables and Ranges

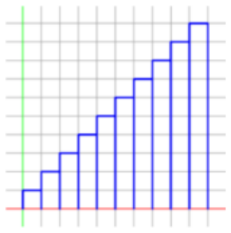
Writing Loops That Contain Variables (Example from 5.1)

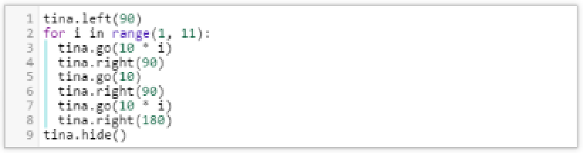
for i in range(1,11): The first value of i will be **1**, the last value will be **10**, or (11-1). The way this range is written, i will only increase by 1 each time.

i is then called within the body of the loop. For example:

tina.go(10*i) The first time, Tina will go 10***1**, or 10.
The second time, Tina will go 10***2**, or 20.
And so forth, until
The last time, Tina will go 10***10**, or 100.

Here is the example from 5.1, in which Tina builds a staircase.





```
tina.left(90)
for i in range(1, 11):
  tina.go(10 * i)
  tina.right(90)
  tina.go(10)
  tina.right(90)
  tina.go(10 * i)
  tina.right(180)
tina.hide()
```

New command in Section 5

`tina.angle(n)`, where n is the number of degrees. 0 degrees faces Tina east.

Self-paced Instruction: Levels 5.1-5.7 (All previous commands may be needed but not necessarily listed under each lesson. Defined objects are listed)

5.1 Pan Flute

Pan Flute begins with a YouTube video that explains the use of variables. The video can be viewed by following this link:

http://youtu.be/tJXRgo4M0qg

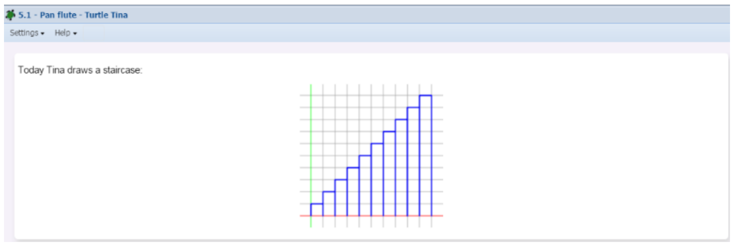
The video demonstrates how to use the range variable as a counting variable.

In Tina, as in most programming languages, counting starts at 0 instead of 1. The video demonstrates this by printing out a list of the range values.

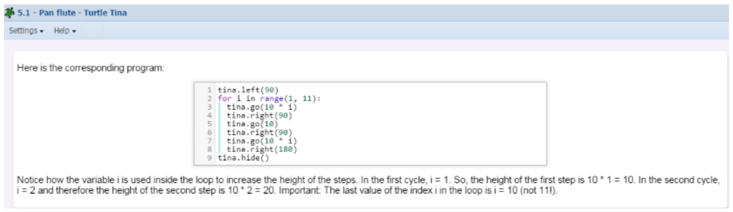
The range variable can also be written as a list of starting and ending values (For i in range(1,101):).

Next, the video shows an example of how to use the counting variable inside the loop to draw a pattern. Tina turns left 90 degrees, but each time she travels one unit further before turning. Students will be impressed by the pattern that results (you may want to have them predict it first). The final two examples show what happens to the pattern when the turn angle is changed by only one or two degrees, which is an excellent opportunity to show why precision is important.

In the next two screens, the counting variable is used to draw a staircase.



Today Tina draws a staircase:



Here is the corresponding program:

```
tina.left(90)
for i in range(1, 11):
  tina.go(10 * i)
  tina.right(90)
  tina.go(10)
  tina.right(90)
  tina.go(10 * i)
  tina.right(180)
tina.hide()
```

Notice how the variable i is used inside the loop to increase the height of the steps. In the first cycle, i = 1. So, the height of the first step is 10 * 1 = 10. In the second cycle, i = 2 and therefore the height of the second step is 10 * 2 = 20. Important: The last value of the index i in the loop is i = 10 (not 11!).

First the range is described by the starting and ending values.

For I in range(1,11)

Next, Tina's go commands are written to go 10 spaces + i.

i will increase by 1 each time until the end value is reached.

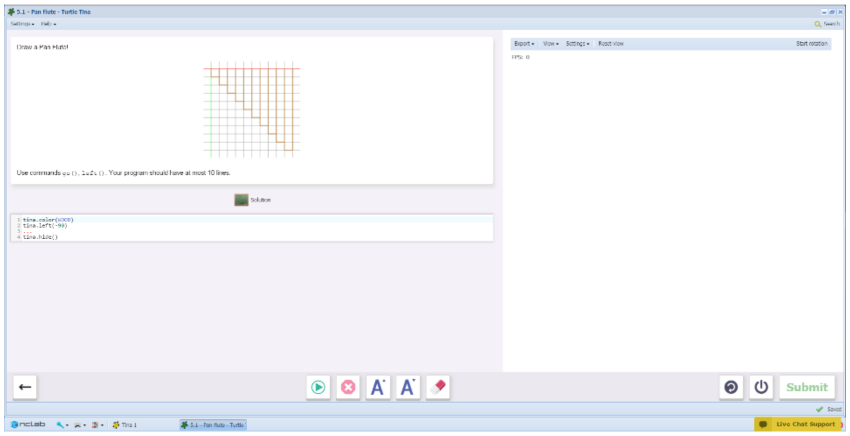
In this example, starting i at 0 makes sense: adding i will not change the first length.

In Pan Flute, students draw the same staircase, but it is upside down. This program makes use of negative numbers (tina.left(-90)), which may need to be explained to younger students who have not yet learned integers. Turning left -90 is the same as turning right 90 degrees. Also, younger students may not be familiar with the symbols used for multiplication and division on a computer (*, /). Apart from the initial turn, the program is identical to the staircase example.

Number of lines: 10

Commands: `go(), left()`

Line length intervals: 10

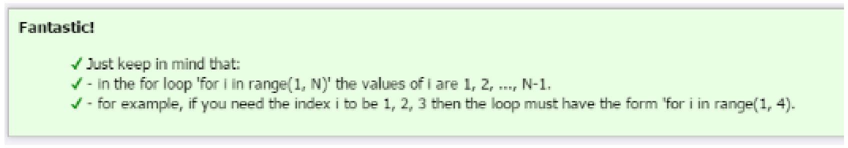


Draw a Pan Flute!

Use commands `go()`, `left()`. Your program should have at most 10 lines.

```
tina.color(WOOD)
tina.left(-90)
...
tina.hide()
```

After successfully submitting the program, students will see this reminder about range values.



Fantastic!

✓ Just keep in mind that:
✓ - in the for loop 'for i in range(1, N)' the values of i are 1, 2, ..., N-1.
✓ - for example, if you need the index i to be 1, 2, 3 then the loop must have the form 'for i in range(1, 4).

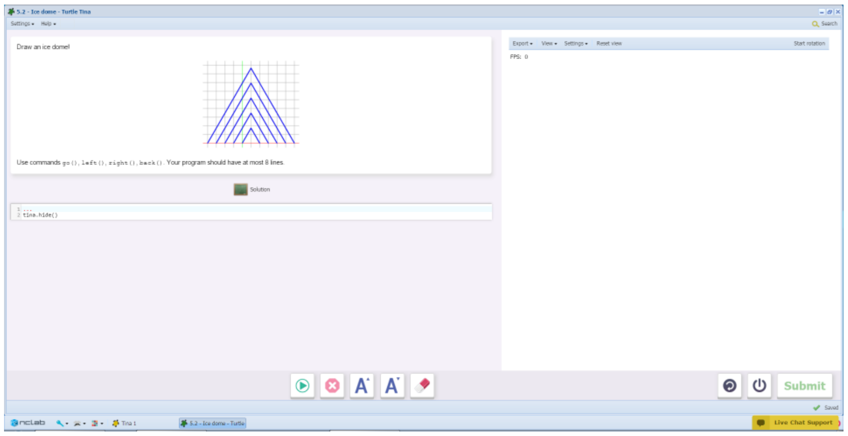
5.2 Ice Dome

In Ice Dome, students practice using the range variable to increase the drawing starting point and size for each iteration.

Number of lines: 8

Commands: `go(), left(), right(), back()`

Interval: 10



Draw an ice dome!

Use commands `go()`, `left()`, `right()`, `back()`. Your program should have at most 8 lines.

```
...
tina.hide()
```

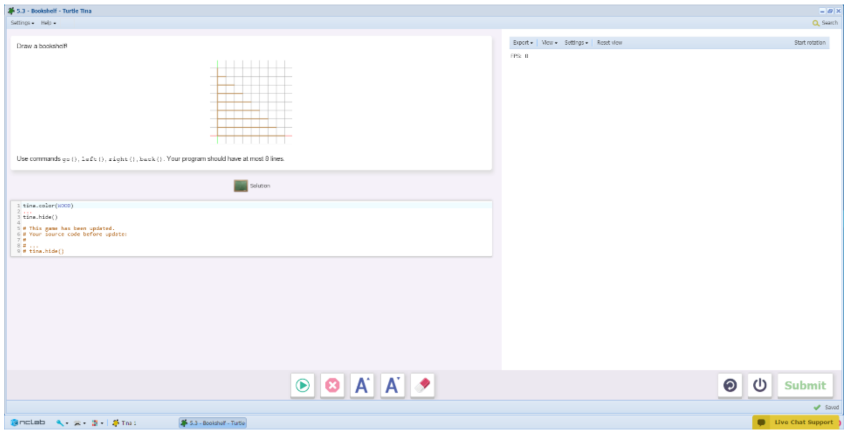
Encourage students to count the intervals carefully. The back command includes two operations. Students must multiply i by 20, then add another 10 to get the correct starting position for the next V in the dome.

5.3 Bookshelf

In Bookshelf, students continue practicing use of the range variables by drawing a bookshelf.

Number of lines: 8

Commands: `go()`, `left()`, `right()`, `back()`



Draw a bookshelf!

Use commands go(), left(), right(), back(). Your program should have at most 8 lines.

```
tina.color(WOOD)
...
tina.hide()

# This game has been updated.
# Your source code before update:
#
# ...
# tina.hide()
```

The line length for the go and back commands uses two operations, as in Ice Dome. This time, the counting variable x the interval (10*i) is subtracted from the starting value (90).

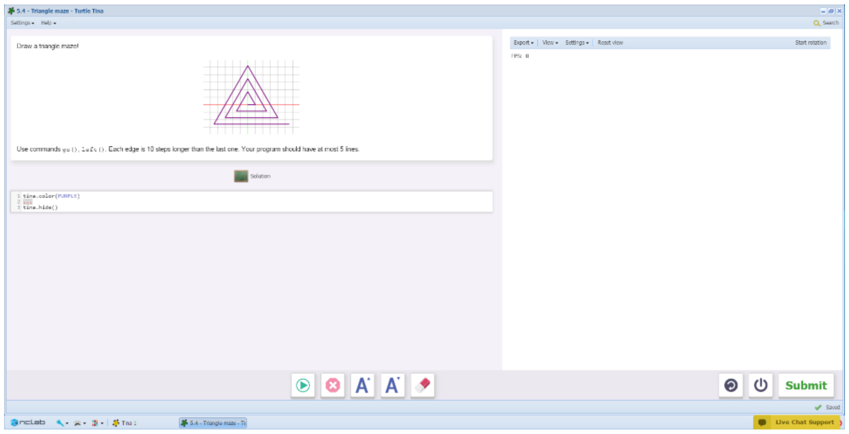
5.4 Triangle Maze

In Triangle Maze, students use the range counting variable to draw a triangle maze.

Number of lines: 3

Commands: `go()`, `left()`

Basically the program just does two things: go, and turn left. It is similar to the example used in the YouTube video in 5.1



Draw a triangle maze!

Use commands go(), left(). Each edge is 10 steps longer than the last one. Your program should have at most 5 lines.

```
tina.color(PURPLE)
...
tina.hide()
```

Students might want to play with the turning angle and the number of iterations after they have found the solution.

5.5 Square Maze

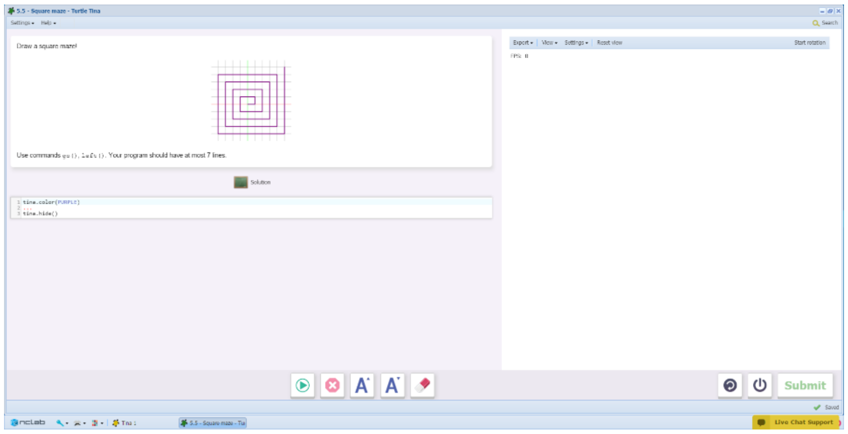
Students use the range counting variable to draw a square maze.

Number of lines: 7

Commands: `go()`, `left()`

Again, this is similar to the square maze shown in the video in 5.1

Students should pay attention to line length in this maze.



Draw a square maze!

Use commands `go()`, `left()`. Your program should have at most 7 lines.

```
tina.color(PURPLE)
...
tina.hide()
```

5.6 Black Hole

Students create a pattern using the range variable.

Students often draw these patterns as an art project on grid paper, and it may be a good idea to have them do this manually before attempting the code.

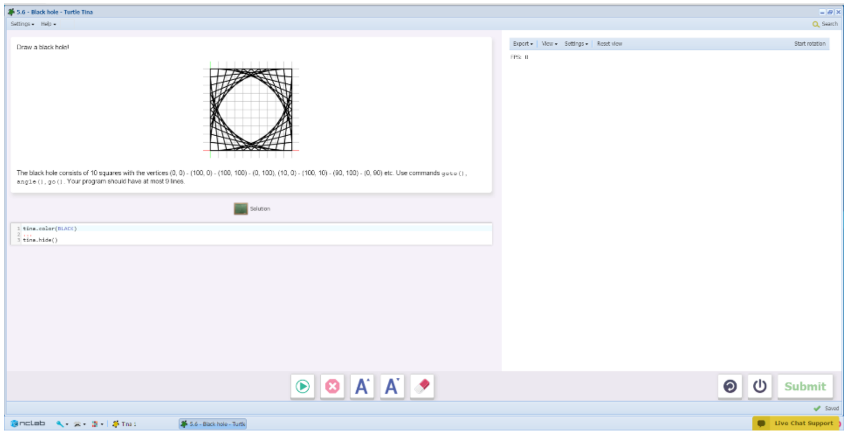
If students have this figure drawn on grid paper with the coordinate pairs of each end point labeled, they should be able to see how these coordinates change in order to draw the next line.

Number of lines: 9

Commands: `goto()`, `angle()`, `go()`

Vertices:

- (0,0)
- (0,100)
- (100,0)
- (100,100)



Draw a black hole!

The black hole consists of 10 squares with the vertices (0, 0) - (100, 0) - (100, 100) - (0, 100), (10, 0) - (100, 10) - (90, 100) - (0, 90) etc. Use commands `goto()`, `angle()`, `go()`. Your program should have at most 9 lines.

```
tina.color(BLACK)
...
tina.hide()
```

The suggested solution uses the `goto` command and coordinate pairs to draw the pattern.

Basically, the code goes around the figure 10 times with a set of 4 lines each time, advancing 10 to start the next cycle. The variable i is used to change the start and end points of each line, based on distance from the vertices' coordinates. Angle resets Tina to 0 before starting the next cycle.

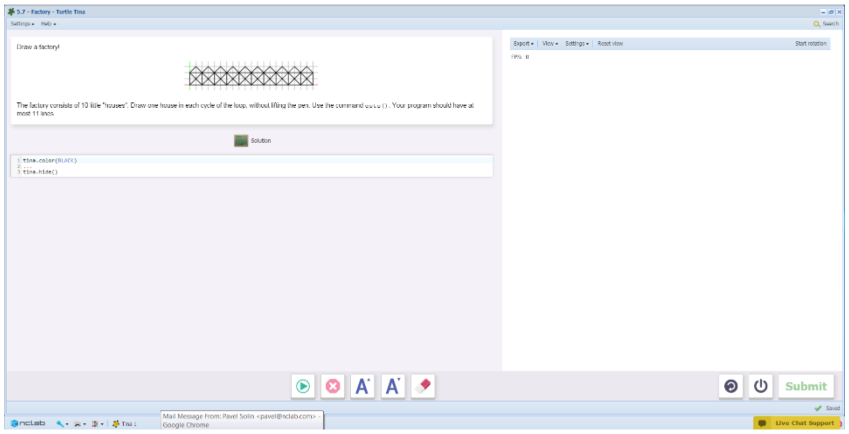
5.7 Factory

Students will create a factory using the range variable by drawing the figure of a "house" and repeating it ten times. The challenge is to do this without lifting the pen, so no `up`, `down` or `back` commands.

Students should try this on paper before attempting the code. This is a classic puzzle: draw all the lines without retracing any of them. The key is to go around the outside of the house first, then draw the "bowtie" inside, ending up at the starting position of the next house. Each line will require a line of code using the grid coordinates.

Number of lines: 11

Command: `goto()`



Draw a factory!

The factory consists of 10 little "houses". Draw one house in each cycle of the loop, without lifting the pen. Use the command goto(). Your program should have at most 11 lines.

```
tina.color(BLACK)
...
tina.hide()
```

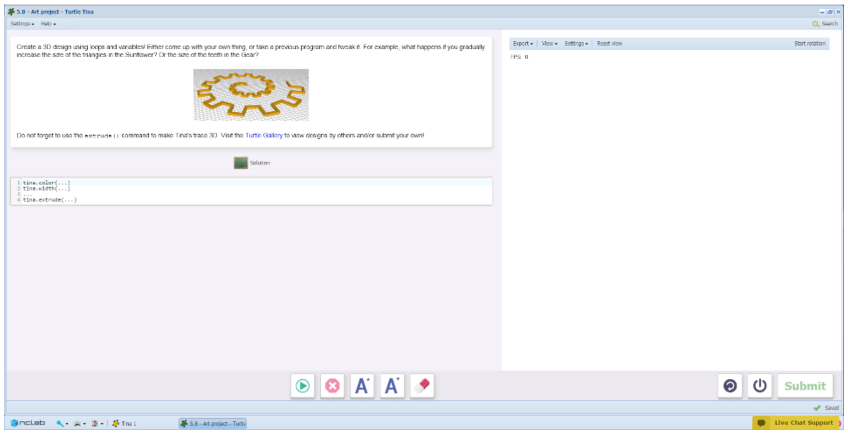
Hints for students:

- Do the coordinates change in the x direction? (Yes)
- By how much do they change for each house? (20, so we need to multiply i by 20 for each x component)
- Do the coordinates change in the y direction? (No. Even though the houses are in different places, their y components are all the same)
- Do we need the variable to describe the y component? (No, it can just be a number)

5.8 Art Project

Students create a printable design that incorporates range values into the loops.

Students can create a design from scratch, or use a previous design and experiment with it using the variable.

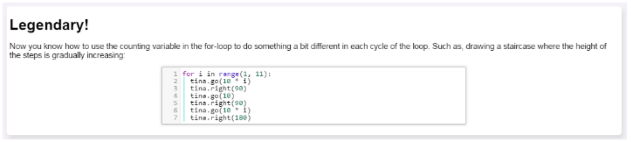


Create a 3D design using loops and variables! Either come up with your own thing, or take a previous program and tweak it. For example, what happens if you gradually increase the size of the triangles in the Sunflower? Or the size of the teeth in the Gear?

Do not forget to use the `extrude()` command to make Tina's trace 3D. Visit the Turtle Gallery to view designs by others and/or submit your own!

```
tina.color(...)
tina.width(...)
...
tina.extrude(...)
```

Once the design is submitted, students will see this screen.



Legendary!

Now you know how to use the counting variable in the for-loop to do something a bit different in each cycle of the loop. Such as, drawing a staircase where the height of the steps is gradually increasing:

```
for i in range(1, 11):
  tina.go(10 * i)
  tina.right(90)
  tina.go(10)
  tina.right(90)
  tina.go(10 * i)
  tina.right(180)
```

The following screen grants them their Yellow Belt in the Fourth Degree certificate.

Questions for post-session discussion:

This level required more thought and understanding of math principles. What were the greatest challenges?

What projects would you like to create using Creative Suite?

Assessment:

Assessment is built into the program. Students must complete a level successfully in order to unlock the next level. Students will receive a printable “Yellow Belt of Fourth Degree” certificate upon completion of Section 5. See Assessment section for journal and project ideas.

After completing Tina 1, students will be ready to start Tina 2 and learn more advanced programming skills.

Students Journal entries can be used as assessment.

The Art Project can be used as a performance task or portfolio artifact. Here is a possible 50-point assignment card:

END OF SECTION 5: ART PROJECT (50 POINTS)

Objective: Create and publish a design in Tina Turtle using a variable as part of a loop.

Planning (15 points): After you have decided on a design, draw out the shapes on grid paper or in your journal. **If your shape has a lot of repetitions (a large range in the variable), just draw the first few steps. If you intend to 3D print the design, keep the shape simple and sturdy.**

- Describe the design. What commands will be needed? Name the object or pattern that you are designing.
- What range limits are you going to use?
- Will you use the different quadrants in the coordinate plane, and ordered pairs?
- Looking at the design, decide which commands will be used. Will you need the `back()` command to start the next part of the pattern? Where will you use `goto()` instead of `go()`?
- If your design is composed of more than one repeated shape or pattern, write out the order in which you will place them in the program. The `color`, `width` and `extrude` program lines are already set up. Include the values that you choose for each one in your design notes.

Writing the code (25 points): Write the program.

- Remember to include comment lines that describe what your program is doing.
- Write the code and test as you go.
- Use a variable in the For loop and define the range with a beginning and ending value.
- Use the variable in the body commands.
- Use some or all of the commands learned in Section 1 and 2:

  `go(), left(), right(), back(), up(), down()` and `goto()`

Saving the file, sharing and publishing (10 points)

- Publish the design to your folder. Inform someone else about the game by providing the link on ____, or by email.

NOTES