REIMAGINED CLASSROOM

Painting Poetry with Python

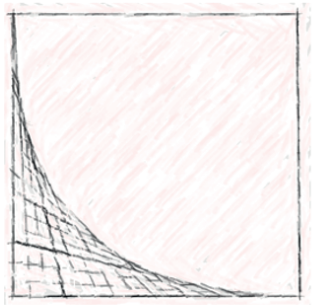
Lesson Plan

Grade Level: 9th–12th Grade

Time Required: 1–2 Class Periods (70 Minutes)

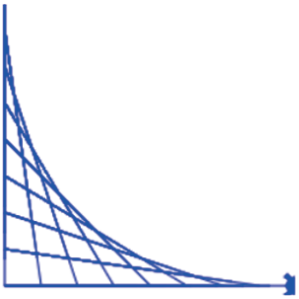
Programming Skill Level: Intermediate

Tension: *The state of being stretched tight*



```
import turtle
tina = turtle.Turtle()
tina.color("blue")
tina.shape("turtle")
tina.speed(10)
tina.pensize(1)
x=0
y=200

while x<225:
    tina.penup()
    tina.goto(0,y)
    tina.pendown()
    tina.goto(x,0)
    x=x+25
    y=y-25
```



DEFINE SKETCH CODE PLOT

OVERVIEW & PURPOSE

Students will leverage art, English, math, and computer science as they generate beautiful visual representations of abstract vocabulary words. The lesson begins with an unplugged activity where students must pencil-sketch icons that represent poetic vocabulary words. Next, students "plug-in" and use a free online coding platform to learn the basics of Python's "turtle" library, a set of commands that lets students draw pictures and animations with code. Finally, students customize their Python code to draw a vocabulary word chosen from a list and then their partner attempts to guess the chosen word.

EDUCATION STANDARDS

1. **Common Core Mathematics:** CCSS.MATH.CONTENT.HSA.CED.A.2: Create equations in two or more variables to represent relationships between quantities; graph equations on coordinate axes with labels and scales.
2. **Common Core ELA:** CCSS.ELA-LITERACY.RL.9-10.4: Determine the meaning of words and phrases as they are used in the text, including figurative and connotative meanings; analyze the cumulative impact of specific word choices on meaning and tone.
3. **CSTA Computer Science:** 3A-AP-18: Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
4. **ISTE: 1.6 C:** Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

OBJECTIVES

Students will be able to:

1. Draw visual representations of poetic vocabulary words using lines
2. Import and use the “turtle” library in Python
3. Use object-oriented programming, functions, and variables to navigate a Cartesian coordinate system
4. Develop custom Python code to draw line-based images of vocabulary words

MATERIALS NEEDED

1. Laptops or tablets with access to ***Tynker.com***
2. Blank paper
3. Pencils
4. Access to ***NearPod Lesson: “Painting Poetry with Python”***
    - Alternatively, you can teach this lesson with the included slide presentation and ***video***

PREPARATION

Before teaching this lesson:

1. Ensure your students can access the free development environment on ***Tynker.com***.
2. Review all presentation materials and the instructional video.
3. Consult with an ELA teacher about relevant vocabulary words that can be used for this lesson (there are words provided here, but feel free to replace them!).
4. Ensure students have a basic background knowledge of writing and debugging simple Python scripts and graphing points and lines in a (x,y) plane.

Note: This lesson can be taught "live" using NearPod or the included presentation materials. Alternatively, students can access the NearPod content on their own for asynchronous learning.

ACTIVITY

Warm-Up (5 minutes)

1. Present slide 3 to the class, showing an image created using only straight lines. Ask students to discuss the following question: *What does this drawing make you think or feel? Why?*
    a. The goal of this question is to demonstrate that simple line drawings can communicate emotions, feelings, and definitions.

2. Review the objectives on slide 5. Students will learn to draw visual representations of words using pencils and paper, then repeat the process using the Python programming language.

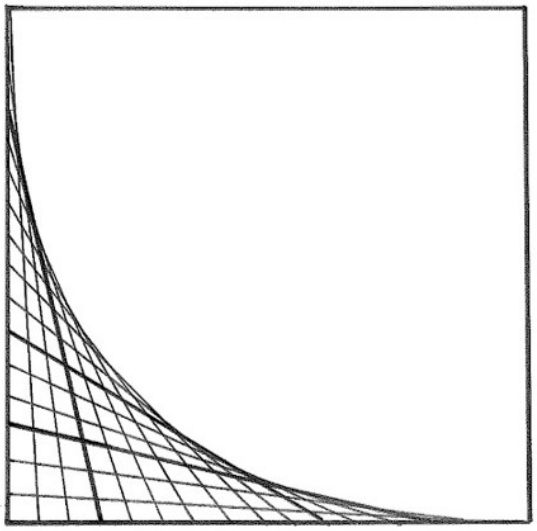
Unplugged Activity: Drawing Words (15 minutes)

3. Choose a student to read the information on slide 7: *Oftentimes, the main goal of artists, poets, and designers is to* ***communicate*** *something (information, an emotion, etc.). One of the simplest, most effective tools for visual communication is the* ***line****.*

4. Share the examples on slides 8–13 of how a word from a poem could be "drawn" or "visualized" using only straight lines.
    a. For example, the word *tension* could be represented by drawing the sketch below:

Mom was stressed,
Tired,
Spread thin.

Her loving smile
Stretched like a face on a balloon:
***Tension** pulling all directions.*



5. Pass out pencils and paper to all students. They will be given five minutes to draw a visual representation of the word ***graceful***. Review the definition and instructions on slide 14, then allow students time to draw. (Note: You may choose to find a poem with the word *graceful* for students to read first!)

6. Repeat the process for a second word, ***aggressive***, on slide 15. Students will have 5 more minutes to draw this word. (Note: You may choose to find a poem with the word *aggressive* for students to read first!)

7. Present slide 16. Allow students to share both of their drawings with a partner. Students will discuss these questions:

a. Can your partner guess which drawing is *graceful* and which is *aggressive*?
b. Explain to students that we just represented words using art! We can go one step further and use coding as our artistic tool.

Guided Activity: Python's "Turtle" Library (25 minutes)

9. Students will now learn the fundamental coding and commands behind the "turtle" library. Turtle is a drawing program in Python that lets you program the movement of a virtual pen.

10. Present slide 19 to the class and allow students to make educated guesses about what each line of code does. Then, review slides 20–25 to detail the function of each command.

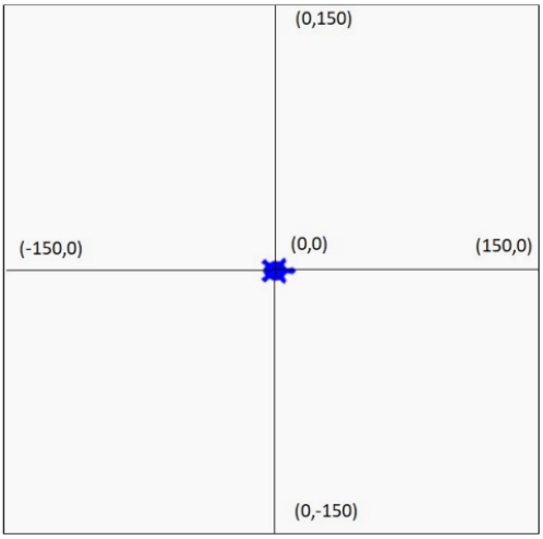
EDITOR
main.py

```
1 import turtle
2 sally = turtle.Turtle()
3 sally.color("blue")
4 sally.speed(10)
5 sally.pensize(1)
6
7 sally.forward(100)
8 sally.left(90)
9 sally.forward(100)
10
```

Command	What does it do?
import turtle	This line imports a library called “turtle.” A **library** is a special set of pre-written commands. The turtle library lets you code a virtual pen!
sally=turtle.Turtle()	Next, we can name our pen whatever we want! Let’s name her “sally.”
sally.color(“blue”)	This command controls the color of the pen. You can try other colors like “red” or “black”!
sally.speed(10)	This command controls the speed of the pen, 1 being the slowest, 10 being the fastest, and 0 being “instant.”
sally.pensize(1)	This command controls the thickness of the pen (1–10)
sally.forward(100)	This command tells the pen how many pixels to move forward
sally.left(90)	This command rotates the pen 90 degrees to the left. You can change the degree number and direction!

11. Allow students to review the previous commands, then discuss a set of new commands on slide 27. The definitions of these commands are on slide 28. It's best to picture the pen starting at (0,0) on an (x,y) axis. These commands navigate the pen around the plane:

a. **sally.goto(100,100)** would send sally to the point (100,100)
b. **sally.penup()** moves sally without drawing
c. **sally.pendown()** moves sally while drawing
d. **sally.write("hi there")** would print "hi there" to the screen!



12. Students will now watch the video on slide 30 (***link to video***). They should follow along on Tynker.com at their own pace, pausing the video and attempting each knowledge check. This video will give students a fundamental understanding of how to use the turtle library.

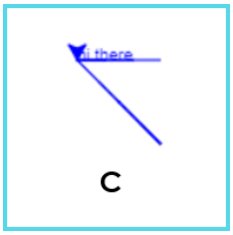
13. Once students have completed the tutorial video, ask the knowledge check question on slide 32: *Which image would the code below draw?*

EDITOR
main.py

```
1 import turtle
2 sally = turtle.Turtle()
3 sally.color("blue")
4 sally.speed(10)
5 sally.pensize(1)
6
7 sally.forward(50)
8 sally.right(90)
9 sally.penup()
10 sally.forward(50)
11 sally.pendown()
12 sally.goto(0,0)
13 sally.write("hi there")
```

A


B



C

D

Independent Activity: Custom Python Art (20 minutes)

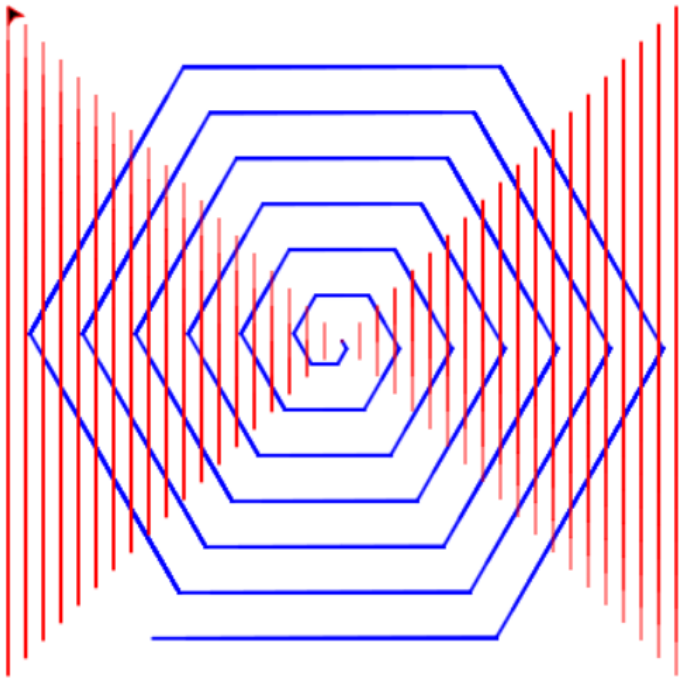
14. Students will now create their own custom art in Python to represent a word of their choice (or a word from the provided list). Review the activity instructions and requirements on slide 34:

a. Choose one word from the list and write a simple 3–5 line poem using the word.
b. Create a turtle program that draws a visual representation of the word
c. Your program must include:
   i. At least 8 different lines
   ii. At least 2 different line colors
   iii. At least 2 different line thicknesses
   iv. Use of the “goto” and “forward” commands
   v. Use of the “penup” and “pendown” commands

15. The sample code and drawing from the example on slide 35 is provided below for your convenience:

EDITOR
main.py

```
1  import turtle
2
3  sally = turtle.Turtle()
4  sally.pensize(2)
5  sally.pencolor("blue")
6  sally.speed(10)
7
8  for i in range(40):
9    sally.forward(i * 5)
10   sally.right(60)
11
12 sally.pensize(1)
13 sally.pencolor("red")
14
15 for i in range(20):
16     sally.penup()
17     sally.goto(10*i,10*i)
18     sally.pendown()
19     sally.goto(10*i,-10*i)
20     sally.penup()
21     sally.goto(-10*i,-10*i)
22     sally.pendown()
23     sally.goto(-10*i,10*i)
24
25 turtle.done()
```



16. When students have programmed their words, allow them to share their creations with another student and discuss the question on slide 36: *Can their partner guess which word they drew?*

17. Activity Submission: Students can submit their work in the following ways:

i. Method 1: Students screenshot their code/drawing and submit the screenshot online through an LMS

ii. Method 2: Students copy and paste their code to a text document and submit that online through an LMS

iii. Method 3: Students raise their hand and teacher spot checks completion

18. Extension: For students that finish early, present the challenge on slide 37. Students will attempt to use the following commands to create filled-in colored circles to represent the word *target*:

a. sally.fillcolor("red")

b. sally.begin_fill()

c. sally while drawing

d. sally.end_fill()

ASSESSMENT

19. (3 points) Pass out blank paper to students. Present the Exit Ticket question on slide 40: *Juan wants to draw a red square using the turtle library, but he's made three mistakes in his code. Identify and correct all three mistakes.*

a. Award one point per identified and corrected mistake:

```
1  import turtle
2  sally = turtle.Turtle()
3  juan.color("blue")
4  juan.speed(10)
5  juan.pensize(3)
6
7  juan.penup()
8  juan.forward(50)
9  juan.right(90)
10 juan.forward(50)
11 juan.right(90)
12 juan.forward(50)
13 juan.right(90)
14 juan.forward(50)
15 juan.right(90)
16
```

Rename turtle to "juan"

Change color to "red"

Change to "pendown()"

```
1  import turtle
2  juan = turtle.Turtle()
3  juan.color("red")
4  juan.speed(10)
5  juan.pensize(3)
6
7  juan.pendown()
8  juan.forward(50)
9  juan.right(90)
10 juan.forward(50)
11 juan.right(90)
12 juan.forward(50)
13 juan.right(90)
14 juan.forward(50)
15 juan.right(90)
16
```

ADDITIONAL RESOURCES

The book *Flatland* by Edwin Abbot (also a short film) is another fantastic connection between literature, art, math, and physics! For more information, ***check out Flatland here!***