Name: Linus Reynolds

partner's name: Logan King

Period: 3

Activity 1.1.1 Algorithmic Thinking and 1.1.2 Planning a Picture

LT #4: (Algorithms and Planning) Develops and analyzes algorithms and implements them using a programming language

We are going to dive right in with learning about an important programming term: ***algorithms***!

An algorithm is a series of steps to carry out a procedure or task. Think of it as a bullet point list or a numbered list to do something; think of it as a recipe.

Before we dive in, some logistics about submissions.

I. Put your name on all submissions to me in Canvas.

II. Collaboration is a huge aspect of this course. I would like you to work through theactivities with your table partner, but I will expect everyone to make their own submission. All of the responses pasted in your submissions should be of your own creation. This is very important for many reasons, but most of all they stand as the major medium for us to dialogue about your understanding of programming and the concepts of the course. They are a record of your understanding when the responses were written…it is absolutely fine and expected that the responses will often not be perfect; I just need to see how you are conceptualizing things so that I can provide helpful feedback to sharpen your understanding and skills.

III. The code you make is best done using paired programming, so if I ever ask for aprogram to be submitted or for snips from a program, these can ge of or from a shared program that you both wrote together. Just be sure to document your code with both of your names along with the month and year.

IV. You should look through all of the PLTW activities with go through and don't just skipto the parts for which I ask for responses. I use the responses to check up on the most important concepts, but there are many small details in the activities that are good to know or that provide some context that will help you with other parts of the activity.

V. Save all your work within appropriately labeled directories (folders). Have a folder forthis class called CSP. In that have a folder called Unit 1. In Unit 1 have a folder "1.1 Algorithms" and one called "1.2 Abstraction", etc. In the 1.1 Algorithms folder you will put copies of the several activities we do for that unit.

VI. **Write your responses in blue or red.** My comments are in green. PLTW instructions are generally in black.

VII. Keep the number formats the same as on the original doc. I will generally follow thenumbers as they are in the PLTW activities.

OK, that's enough for now. Let's get started.

1.1.1 Algorithmic Thinking

1. Instant Challenge:

a) Open a drawing software such as Microsoft Paint or Sketchbook, and turn youmonitor a bit so that you're partner can't see what you are making.

b) Draw a picture on your screen, using only simple shapes, such as straight lines,rectangles, and circles.

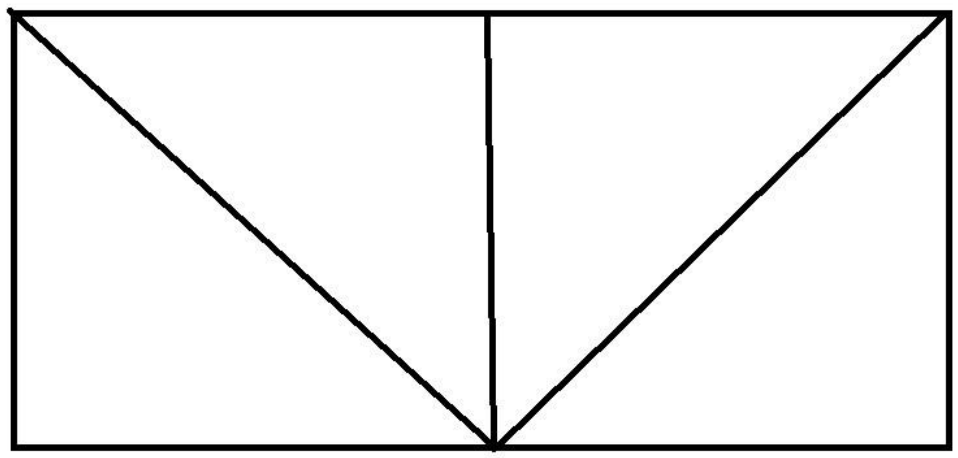
c) In the space below, write a numbered set of instructions for how to recreate yourdrawing:

1. Draw a medium-sized rectangle
2. Draw a straight line through the center of it, vertically
3. Do the same horizontally
4. Draw straight lines diagonally from one corner to the opposite, with both sets of corners

d) Copy and paste these instructions in an email to your partner and close the imageyou made. If you don't have a partner, get with another pair and make a 3-way swap of instructions.

e) Follow your partner's instructions exactly as they are written to produce your versionof the drawing. Do not add, skip or modify the steps.

f. Paste in a snip of the image you made.



g. Did your drawing look like the one that your partner made and for which your partnergave you the algorithm?

Really close, his diagonal lines didn’t go all the way to the corners.

If you have extra time, try this one:

I made mine by using the Insert tab in Microsoft Word, but I will keep the algorithm generic so that you can use it using whatever sketching software you are using.

i. Make a circle about a half inch in diameter and fill it black.

ii. Copy this circle (or make another equivalent circle) and position it about 2 inches tothe right of the first circle (on centers).

iii. Make a rectangle that is about ¾" high and 3.5 inches wide. Fill it light blue andposition it so that the bottom edge overlaps the bottom circles by a few millimeters and the left edge of the rectangle is about a quarter inch to the left of the left edge of the left circle.

iv. Make another light blue rectangle about ¾" tall by 1" wide and align it on top of theother rectangle so that they are flush with each other on their right edges.

v. Make a red-filled triangle isosceles triangle where the long edge is horizontal, isplaced on top of the small blue rectangle, going past the left and right edges of the rectangle by a few millimeters. The vertex opposite the base of this rectangle should be centered over the midpoint of the base and about ½" away. (i.e. the height of this triangle is about ½".)

vi. Make a black right triangle where the right angle opens upward and to the left. Thetop vertex for this triangle should be halfway up the left edge of the wide blue rectangle. The vertex at the right angle should be a few millimeters above the bottoms of both circles. The bottom edge of the triangle should be the same length as the height.

Paste in a snip of the image you made.

Log in to Activity 1.1.1 in PLTW if you haven't done so already. Skip down to the section Algorithms.

6. a) Representing Algorithms. What are the four ways you can represent an algorithm? Be sure you understand what is meant by each way of representing the algorithm.

I. Natural Language: Ordered list of steps in a natural language, such as English or Spanish.

Ii. Flowcharts: A diagram to show a sequence of steps using shapes such as rectangles for actions, rhombuses for decisions, and arrows to show the flow of the algorithm.

Iii. Pseudocode: A written set of instructions. It looks a little like code, but it can’t be understood by a computer.

Iv. Programming language: A structured set of instructions with special keywords and formatting that can be understood and executed by a digital device.

b) What was the algorithm you wrote for #1 (describe what it is an algorithm for) and in which of the 4 ways did you represent the algorithm?

It was an algorithm to draw a specific shape. I represented the algorithm with natural language.

(7 through) 11. Python and the turtle module: Holy cow, the course is jumping in right away on the first day. I like it! (because I think you will like it more this way.) We did not talk about creating class objects and calling methods on objects until late in the first semester. I would like to take a moment to make sure you get the terminology down.
Python: a programming language

module: a pre-written program available for you to import and use in your own program. You usually have to import the module in your program in order to access it. In this activity we are using the `turtle` module.

class: Predefined code that is generally used in Python to define how an object (instance) of that class can be created, what attributes that object will have and the methods that carry out actions that can be done by or on the object. By convention the first letter of a class name is capitalized. Thus, you can tell that the `Turtle` class is a class defined in our `turtle` module.

a) name another class in the turtle module besides Turtle?Screen

method: a procedure defined in a class which an object of that class may call. The procedure generally defines an action done by or on the class object calling the method. `forward` is a method a `Turtle` object can call.

b) Name two other methods a Turtle can call?

right, left

attribute: Also known as a property. It is like a variable attached to the object whose data value describes some part of the state of the object. An object's state can be described by all of its attribute-value pairs. At the end of step 8, the Turtle object's `size` attribute has a value 3.

c) What is the value for the Turtle object's direction?

270

I'm totally shocked that PLTW has not introduced the concept of **variables**! A **variable** allows us to refer to objects or values. As a programmer you get to make up any name you want for a variable (however, I will teach you how to make strong variable names). To assign a value to a variable (or more technically in this case to create a variable that will refer to a certain object in memory) you put the name of the variable, then the "=", then the object you want it to point to (or the value you want to assign to the variable). The Turtle object in this lesson is referred to by the variable `painter`.

d) What other variable is used in the program and what type of object does it refer to?

Wn, and it refers to a turtle screen.

17. After you have played around with making images using turtle in 16, write a program that will recreate the image you drew based on your algorithm in 1c.

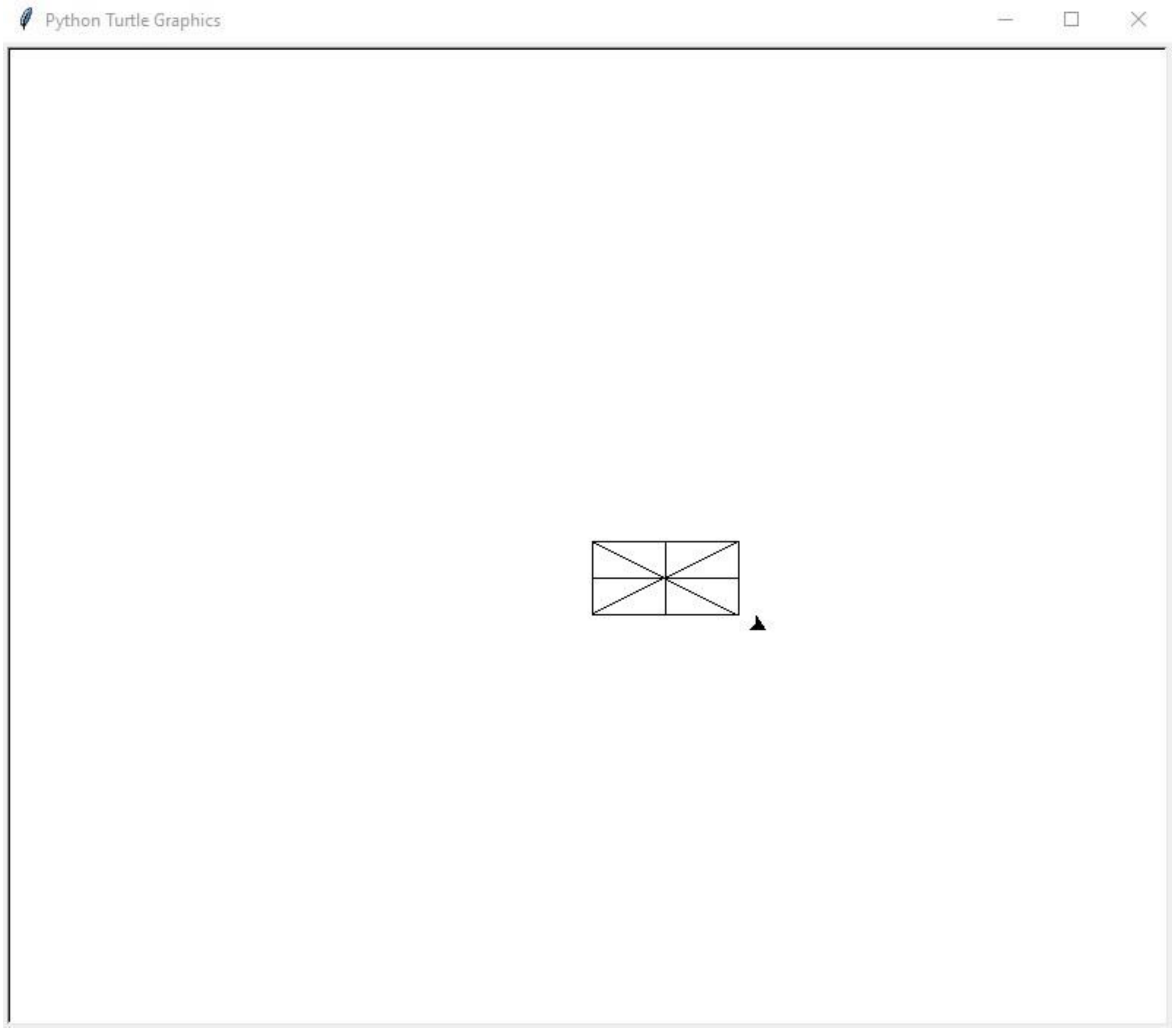
It's important to get in the habit of **document** your programs well. You document your program by including comments. **Comments** are not read by the compiler when the program is run, they are only for humans to read. You can make a line or part of a line a comment by putting a # in front of your comment. Everything after the # on the line will be "commented out", i.e. a comment for humans but unread by the compiler. To make multiline comments in Python, use 3 single quotes before and after the line you want to comment out. This is how we make program and function **docstrings**. The **program docstring** at the top of my program would look like this:

```
'''

D. Willson

July, 2020 This is a

demo for 1.1.1

'''
```

At the top of each program should be a program docstring. Be sure to download your python program and save it in your CSP files for Unit 1.1.

Paste a snip of your program along with the result of the program's execution. You will likely have to make multiple snips…make sure the snips are big enough so that they can easily be read.

Result:



Code:

```
'''

Made by Linus Reynolds

On 9/15/2021

1.1.1 turtle project
```

```
'''



#Imports Turtle module

import turtle as trtl



#Makes a turtle object

painter = trtl.Turtle()



#Makes rectangle

for i in range(2):

    painter.forward(100)

    painter.right(90)

    painter.forward(50)

    painter.right(90)



#Makes the vertical line

painter.penup()

painter.forward(50)

painter.right(90)

painter.pendown()
painter.forward(50)

painter.penup()


#Makes the horizontal line

painter.right(90)

painter.forward(50)

painter.right(90)

painter.forward(25)

painter.right(90)

painter.pendown()

painter.forward(100)

painter.penup()


#Makes the diagonal lines

painter.left(90)

painter.forward(25)

painter.left(116)

painter.pendown()

painter.forward(111)

painter.penup()
painter.right(116)

painter.forward(49)

painter.right(117)

painter.pendown()

painter.forward(112)



#Moves the turtle out of the way

painter.penup()

painter.forward(20)



#Keeps the screen from closing right after the program finishes

wn = trtl.Screen()

wn.mainloop()
```

1.1.2 Planning a Picture

5. a) What does an input statement do?

An input statement asks the user for input, for example input(“What is your name?”) would ask the user what their name was, and they could type it in, and it would be saved to a variable.

b) What is user_name and what is it used for?user_name is a variable. It is used for storing data, in this case it is a string.

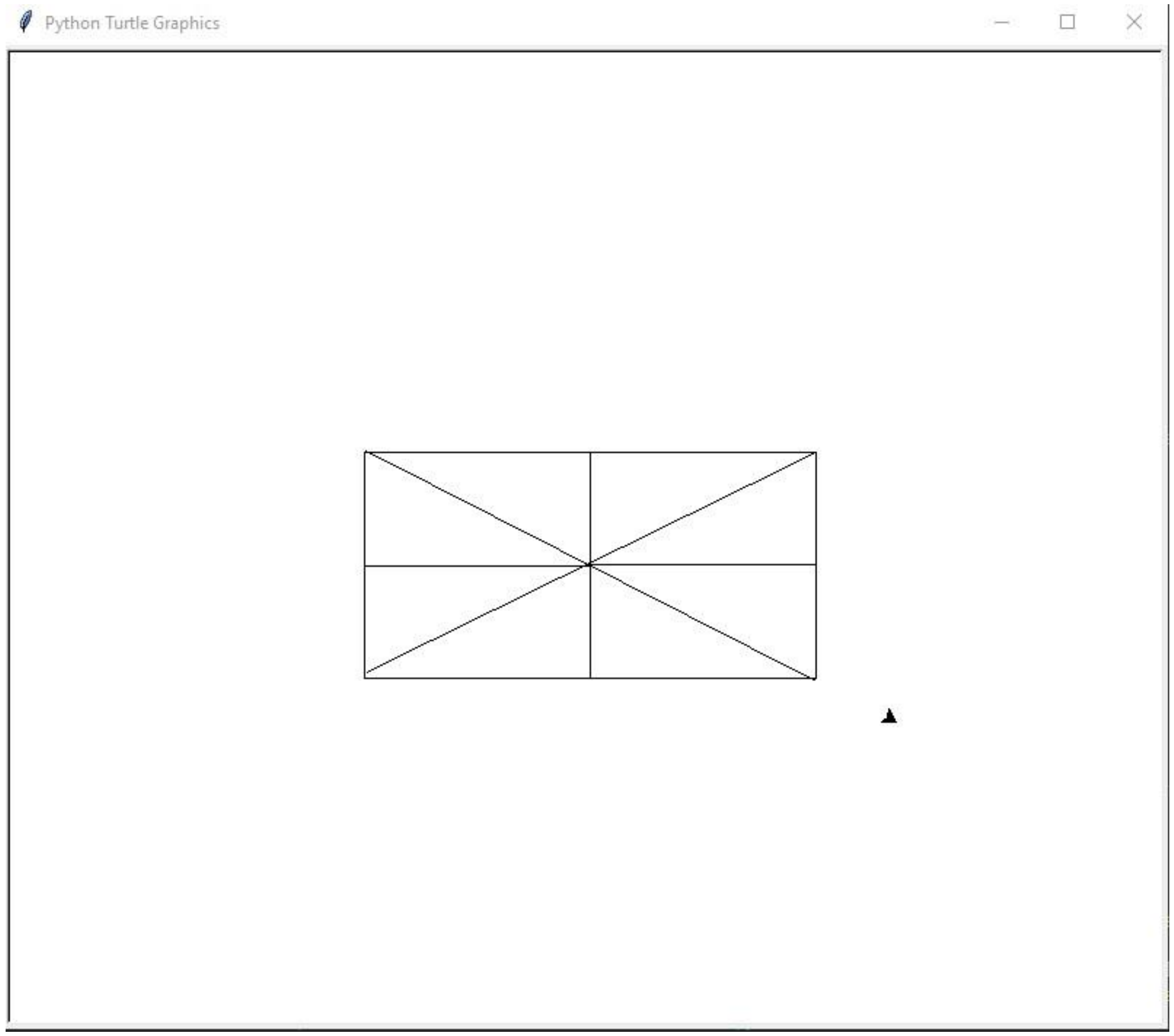
c) What does a print statement do?

Prints something to the console. For example, print(“Hello World!”) would display Hello World! to the console.

8. The built in function int(), is a casting function. When you **cast** a value, you are returning the value as a specified data type. int(var) will take the value of whatever is inside the ( ) – which is called the argument -- and return that value as an integer (if it can be turned into one). Thus, int('27') returns a 27; int(5.0) returns a 5; int(12/4) returns a 3; and int(3.14) returns an error, as does int('ten').

11. Be sure to ask me if you don't get any of the answers for these 3 questions.

21. a) Paste a snip of what it is that your program will make (i.e. the sketch you made for 20 or the output you have from a run of 21).



(scale 3.1)

b) What inputs does your program prompt for from the user?

Asks you what scale you want the picture to be. For example, if you say 3.1, as shown above, it will make the picture 3.1 times as big. If you say 0.5, it will be half the size.

c) Upload your Python file in this submission along with this completed activity doc. Itshould be named with this activity, your initials and the problem (step) number. For example, if I were to name my file for this question it would be 1-1-2_DW_21. Make sure that it is documented properly. (If I had a partner with the initials MJ and we wrote the program together, the name of the file should be 1-1-2_DW-MJ_21, and both of our names would be in the documentation...and we would both have a copy of the file.)

C1: What is the difference between the input and output of a program, and how can one affect the other? Give an example using *Python* code.

```
#import random module

import random as rndm

#Makes random number from 0 to 100

randomVar = rndm.randrange(100)

#Sees if it is below 50

if randomVar <= 50:

      #Asks user what they want to print

    inputVar = input("Type what you want this to print here ")

    #Prints it

      print(inputVar)

elif randomVar > 50:

      #Asks them what they want to print

    inputVar = input("Type some awesome text here")

      #Prints it, but inside of some other text

    print('The person who said this: "' + inputVar + '" is awesome.')
```

A way an input can affect an output is with print(). In this example, it prints the input, and print is an output. Therefore, the input is affecting the output. Also shown above is a way an output can affect an input. It generates a random number, and depending on the number, it will change what the input statement says.