PLTW COMPUTER SCIENCE

Activity 2.1.1

Transitioning from Blocks to Text

goals

- Compare and contrast lower-level programming languages with higher-level programming languages
- Learn basic rules related to programming with syntax
- Get started with Blockly
- Develop programs independently that uncover what is abstracted in block-based programming languages

description of task

Create programs in block-based languages and translate them to *Python* and other text-based programming languages

Essential Questions

1. What are some advantages to programming in a text-based language compared to a block-based programming language?
2. What are some of the ways concepts in blocks are represented in languages like *Python*® and JavaScript™? (Example: What do loops look like in a text language compared to a block?)
3. How realistic is it to expect coding professionals to be experts on all programming languages? What are they *really* experts at?

essential concepts

- Programming Language Abstraction
- Algorithms, Variables, Arguments, Procedures, Strings and Concatenation, Data Types, and Logic
- Arithmetic Operators, Relational Operators, and Logical Operators

- Conditionals and Event-driven Programming

Loops in Text-based Languages

While highly abstracted languages make coding simple, sometimes you will lose the ability to do exactly what you want to do in the program. Also, some aspects of coding, like managing mathematical operators or using loops, might be easier in a text-based language than being forced to use predefined blocks. The power of programming in a text-based language is that you have flexibility to manage certain aspects of coding outside of constrained, pre-built blocks.

In this activity you will identify what coding fundamentals look like in a text-based language. You'll begin to uncover what details are being abstracted or hidden by a programming language. The code blocks used in MIT App Inventor and text used in *Python* are not that dissimilar. The coding concepts remain the same, they just look a little different.

Blockly is the highly abstracted language that is the foundation for MIT App Inventor. With the "Try Blockly" editor, you can create simple programs with blocks in the integrated development environment (IDE), while displaying the same code in a text-based language. Later in this course, you will program in the text-based language *Python*. You are getting a preview in this activity to see how *Python* has the same essential computer science concepts.

1. Go to: https://developers.google.com/blockly/
2. Change the Language on the right to **Python**.
3. Select the play arrow play button to execute the code in the window.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

PLTW DEVELOPER'S JOURNAL In your journal describe what the sample program "Hello World" is doing in natural language or pseudocode. Identify some similarities and differences of the same code in block-based code versus text-based code.

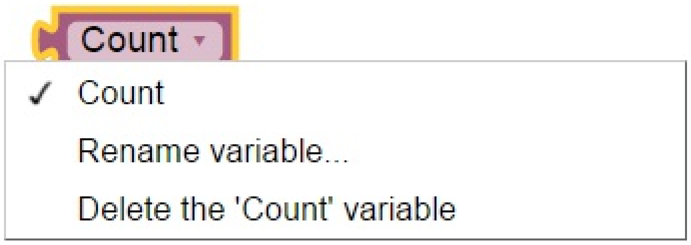
Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.	Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Now that you have seen how it works, you are going to use what you know to make some simple programs. The Blockly IDE does not let you save, so the programs you create will be simple and short.

Not all text-based languages have the same rules. Later in this course, you will program in the text-based language *Python*. As you explore what block code looks like as text, you will be introduced to new terms and some of the specific syntax rules specific to *Python*.

4. Drag the "Hello World" example into the trashcan to begin creating your own short programs.

   Important: Currently the variable *Count* remains in the text side of the IDE. Variables created in the IDE are retained in the *Variables* drawer on the left. To delete the *Count* variable, drag it out of the *Variables* drawer and use the drop-down list to select **Delete Block**. However, it will not create any problems if you choose to leave the *Count* variable defined.



Procedures, Functions, Parameters, and Arguments in Text-based Languages

The Countdown Loop

Have you noticed anything different about the block choices in this IDE compared to MIT App Inventor? What is different about the purple blocks?

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

What was called a "procedure" in MIT App Inventor is called a "function" in Blockly. This is because the term "function" is slightly more descriptive for text-based languages.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

A function "block" in *Python* begins with the keyword "def" followed by the function name and parentheses. Inside the parentheses are special types of variables called "parameters" or "arguments".

```
def do_something():
```

Important: The interchangeable use of computer science terms

The terms “procedure” and “function” are often used by coding professionals interchangeably because they are so closely related. The terms “parameter” and “argument” also represent another example of interchangeable words used when communicating about code.

The interchangeable use of words in computer science can make it confusing sometimes and that’s okay. Feel free to use them interchangeably as well. In later lessons, you will discover the subtle differences between these concepts, but these differences don’t matter right now.

Functions, Variables, Loops, Strings, Print Statements, and Incrementing

You’re going to create a simple algorithm that will display a countdown from 10 to 0 each time the user selects the **OK** button on the embedded web page. At the end of the count down, a string will display to tell the user they have reached the end.

To do this, you will need a procedure or function, a loop that counts down, a variable, a string, a procedure or function call, and a print function.

5. Drag out a *Function* block and define a function called *CountDown*.

   Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

6. Add a Loop that will define the variable *i.*

   Important: The variable *i* will store the number that is being incremented down. If you read the blocks in natural language, you might say, “Count with variable i starting at 1 and ending at 10, decrementing by 1.”

   Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

PLTW DEVELOPER’S JOURNAL Previously, when you incremented, you typically gave the variable the name “Count”. Why do you think this programming environment provides the letter *i* as the default name for the index that is storing the variable that is incrementing?

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

7. Create an algorithm that will **count *down*** from 10 to 0. Each time the user clicks the play button in the IDE, it should count down by 1. At the end of the countdown, it should display a string telling the user they have reached the end as shown below:

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

8. Try running the program to see whether it behaves as you expect.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

In this example, you need to add the predefined *print()* function to return the outputs of the *CountDown ()* function to the embedded web page. In *Python*, this commonly used function looks like this:

```
print()
```

There are a lot of pre-built functions in the *Python* programming language for tasks you use a lot. Just like in App Inventor, when you create a procedure, you must also call the procedure. In this example, you have created a function; you now need to call the function and print the results to the web browser page. In other words, you are using a print function to return the values of the *CountDown* function.

```
print(CountDown())
```

9. Add a print function that calls the *CountDown()* function and returns the value to the embedded page.

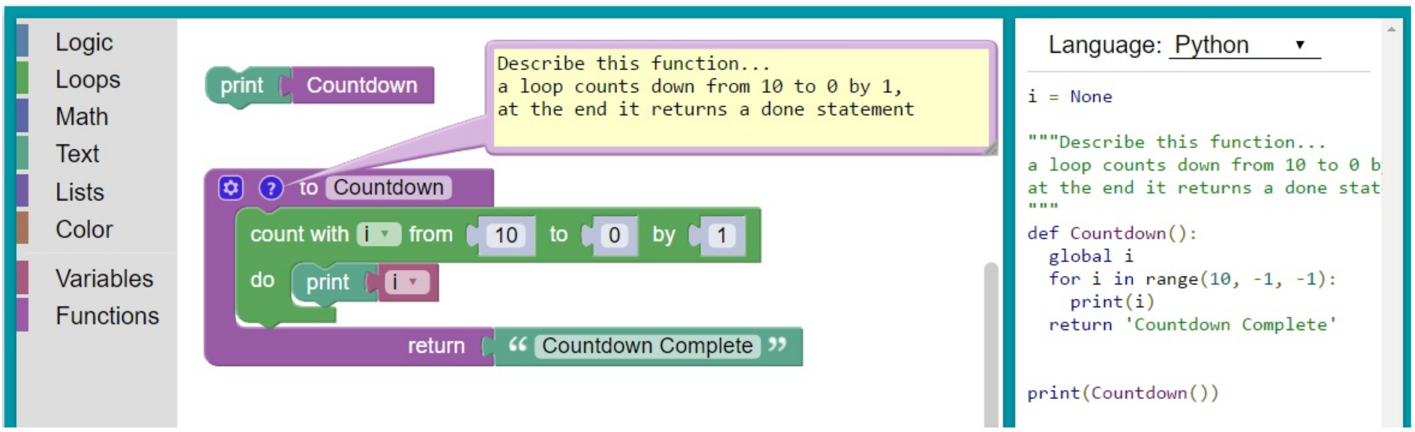
Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Commenting Code

Any text within triple quotation marks in *Python* is considered a comment and will not be executed by a program. This allows you to write messages in the code for others (and yourself) to understand what specific functions do in the future when you revisit your code.

10. Click the **?** (Question Mark) and add a comment describing the *CountDown* function you created. Notice, you only get one option for commenting code right now. In a full text environment, you can use the # symbol to comment line by line if you wish to. You will learn more about commenting code in text-based languages later.
11. Take a moment to look at the *Python* code. Compare what you see with what you know.



```
Describe this function...
a loop counts down from 10 to 0 by 1,
at the end it returns a done statement
```

```
i = None

"""Describe this function...
a loop counts down from 10 to 0 b
at the end it returns a done stat
"""
def Countdown():
  global i
  for i in range(10, -1, -1):
    print(i)
  return 'Countdown Complete'


print(Countdown())
```

PLTW DEVELOPER’S JOURNAL

1. What type of variable is *i* in your blocks? What is being done to the variable?
2. What is the name of your procedure? How does this now look in the Python code?
3. Does it matter that procedures are now called “functions”?
4. What pre-defined function do you need to use to show a user information on the embedded web page?
5. How are strings identified as the message and not part of the code?
6. How are comments identified?
7. How are loops represented in *Python*?
8. When creating your own new function, what keyword do you start with?

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

Passing and Returning Values (Parameters)

A parameter is a special kind of variable, used in a subroutine to refer to one of the pieces of data provided as input to the subroutine. These pieces of data are called arguments.

Argument	The values that a program provides to a function or subroutine.
Parameter	A variable defined in the function to receive specific information.

Important: Sometimes coding professionals use the terms “argument” and “parameter” interchangeably. In this course, “argument” is a better choice because *Python* does not use the term “parameter”.

When you click the play button on the *CountDown()*:

- The *print()* function is called.
- The *print()* function calls the *CountDown()* function.
- The *CountDown()* function initializes the global variable to “10”.
- Each time the variable is incremented in the function or sub-routine, that parameter gets passed to the print function and is displayed on the embedded web page. Then the program waits for you to click the **OK** button.
- Each time the variable comes back to the *CountDown()* function, it checks to see whether it has reached the end of the range.
    - If no, it increments again and passes the argument to the print function as before.
    - If yes, the *CountDown()* function returns the string “countdown complete”.

You are now going to adjust your blocks to change the program so you can program the flow for how functions pass and return values.

12. What would happen if you remove the print statement in the loop? Discuss with your partner and justify what you think will happen before testing it.
13. After you have an educated guess, delete the print statement from the loop and run the program. Is it still counting down?

    Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

    - What answer did you get? Look at the loop and determine why you got the response of -1 .
    - How can you verify that you are right?

    Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

14. Clear the programming environment by dragging your blocks to the trash.

Sequencing in Text-based Languages

What's Your Name?

You will now create a program that asks for the user's name and then displays a message back to the user with the input block.

15. Create a variable "Name".
16. Set the variable Name to a string entered by a user. In this programming environment, you can do this with a pre-built function called *text_prompt()*. This function is designed to ask whatever string you type into the quotation marks.
17. Type "What is your name?", into the string in the *text_prompt()* block.
18. *Print()* the Name to the embedded page.
19. Run your program and note the outcome.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

When programming in blocks, you defined and initialized variables anywhere there was space in the IDE. Your procedures were also placed anywhere you wanted.

Important: In text-based languages, code is executed in the order, or *sequence*, of the functions that are called as the program moves from the first line to the last line of code. The need to pay attention to *sequencing* was abstracted in many places when working in block-based programming.

20. Place the *print(Name)* function before the *text_prompt(msg)* function and run the program.

PLTW DEVELOPER'S JOURNAL Visit with your elbow partner and explain the output.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

21. Modify the code to concatenate a string you define in the code with the name the user enters when the program prompts them.

22. Here is what the text version of concatenation would look like:

```
print(str('You are awesome ') + str(Name))
```

PLTW DEVELOPER'S JOURNAL Discuss the following questions with your partner and record any information that you were unsure of.

1. How are the symbols used to identify the strings different from the quote marks used to identify comments?
2. In the "prompt for name with message", select the drop-down and change text to **number**. Note what you see in the text version of the code.
3. Run the code and enter a name even though "number" is selected in the drop-down list. What do you think "NaN" stands for? Why won't the program display your name?
4. Why do you think the programmer chose floats as the data type that can be entered for numbers and not integers?

Lists in Text-based Languages

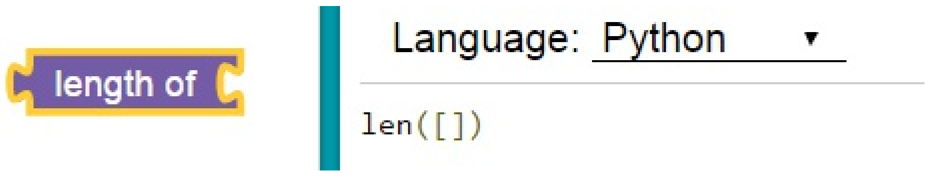
How Long is My Name?

In this section you will see how lists are accessed in a text-based language. *len()* is a new function to you that finds the length of a list. The function counts how many items are in the list.

You will create a basic program that:

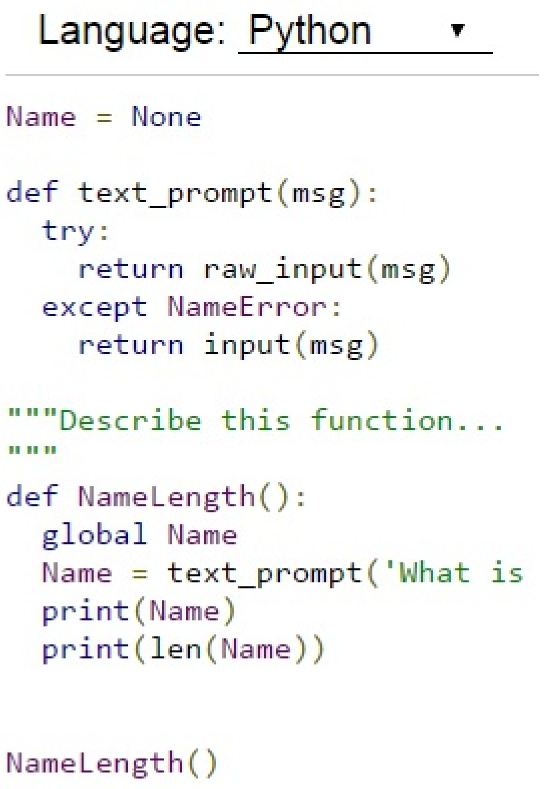
- Prompts a user for their name using the *text_prompt(msg)*
- Stores their name to a variable "Name" using a *function()* you create
- Prints the user's name using a *print(Name)* function
- Prints how many letters are in their name using the *len(Name)* function

23. Drag a *length of* block into the programming space. Notice the text representation of it in *Python*.



```
len([])
```

24. If functions are represented with parentheses (), how are lists represented in a text-based *Python*?
25. Based on the following text-based program, create the code using blocks to ask "What is your name?"



```
Name = None

def text_prompt(msg):
  try:
    return raw_input(msg)
  except NameError:
    return input(msg)

"""Describe this function...
"""
def NameLength():
  global Name
  Name = text_prompt('What is
  print(Name)
  print(len(Name))


NameLength()
```

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

26. Concatenate the output into one output that makes sense. Example:

    Hello Milo. Your name has 4 letters in it.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

27. Examine the text version of this program. You might notice that the print function containing the concatenation looks a little strange in *Python*. What do you think the *.join* did compared to how you have seen concatenation done before?

```
print(''.join([str(x) for x in ['Hello ', Name, ' your name has ', len
```

Importing Libraries in Text-based Languages

Guessing Game Revisited

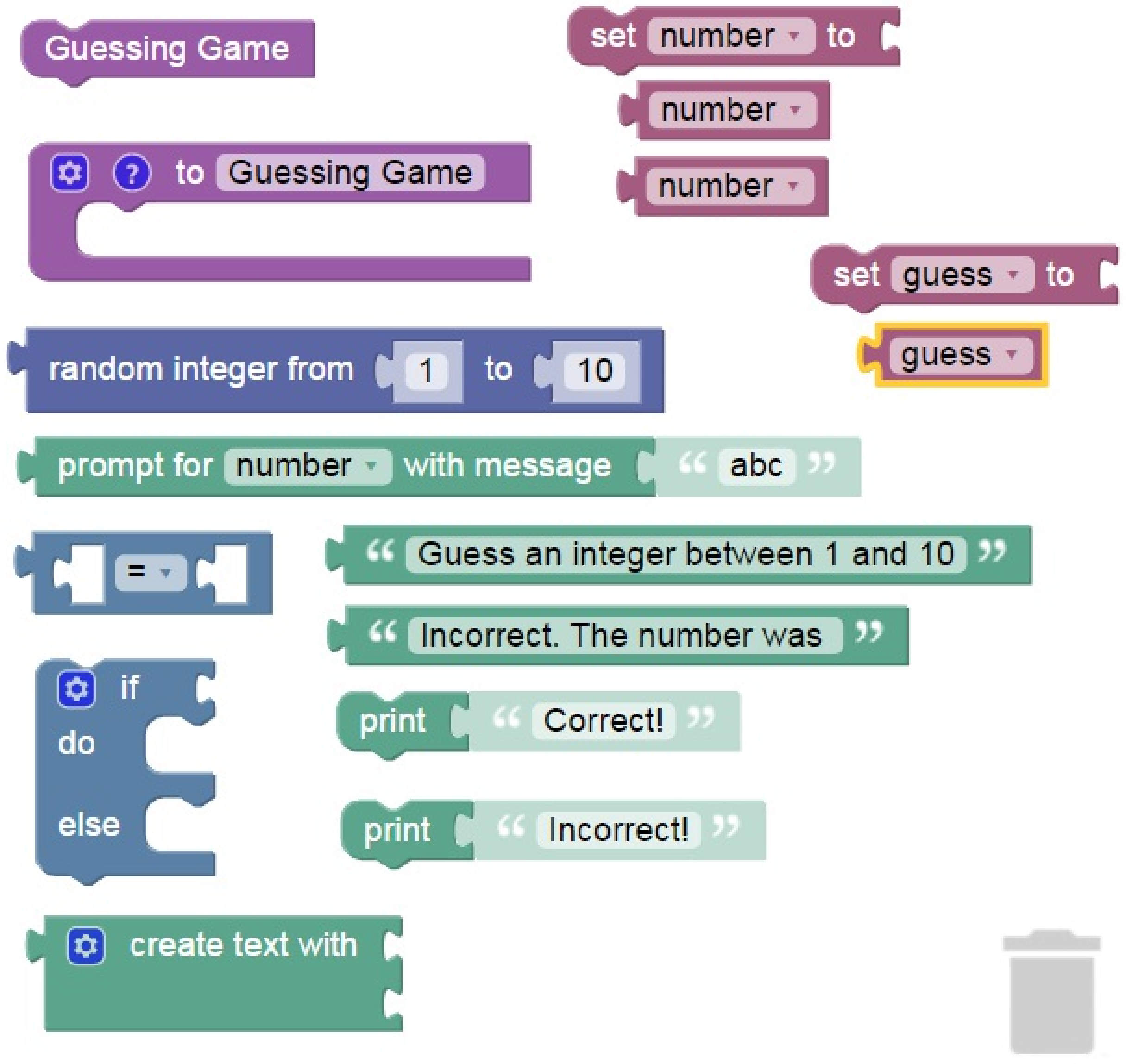
In this section you will explore text representations of **conditionals**, **variables**, ***Python* imports**, **functions**, and **operators**.

You will create code that generates a random number from 1 to 10 and allows the user to try to guess the number. You will need a conditional to compare whether the number is correct, then provide a message to the user if they guessed correctly or incorrectly.

- You can rename or delete previous variables by clicking the dropdown of a variable.
- You can create new variables in the *Variables* drawer.
- You will need to use the mutator block to modify the conditional you get from the logic drawer.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

28. Here are the blocks you will need, followed by the program written in *Python*. Drag out and arrange the blocks in the programming environment to create the program.



```
import random

guess = None
number = None

def text_prompt(msg):
    try:
        return raw_input(msg)
    except NameError:
        return input(msg)
"""Describe this function...
"""

def Guessing_Game():
    global guess, number
    number = random.randint(1, 10)
    guess = float(text_prompt('Guess an integer between 1 and 10'))
    if guess == number:
        print('Correct!')
    else:
        print(str('Incorrect. The number was ') + str(number))
Guessing_Game()
```

At the top is a new statement “import random”. *Python* has built-in libraries of common functions, such as picking a random number. Using search engines, you can find all sorts of built-in functions that you may call and import without having to make a whole function of your own.

Important: The import statements allow you to see further into the language abstraction than you have before. As you move into text-based coding, there are still procedural abstractions to take advantage of with the import statements.

29. Identify where the import is occurring in the code. Why does the code put it at the top?
30. Identify where the random function is actually being called. What do the values, also known as arguments, in the ( ) represent? *Python* has the same structure of the blocks, just without the words you see on the blocks.

Operators and Modulo in Text-based Languages

Wheel Circumference to Linear Distance

In future activities you'll learn about dead reckoning navigation and programming a self-driving vehicle based on the radius of its wheels and the total distance the SDV travels to complete a specified path on a grid mat.

For this program to help you calculate these details, assume you know the radius of the vehicle’s tires is 1.5 inches and the total time traveled is 333 seconds.

You are going to create a program that finds the circumference of a tire and also converts the total time traveled (333 seconds) to display as minutes and seconds (using modulo).

An example of a final program would be one that actually asks for inputs of things like the circumference of a tire. For now, making this variable a constant will help you debug your program with less variables to trouble shoot.

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

To create this code, you may want to use this other version of Blockly.

- Circumference formula is 2πr
- π can be represented mathematically with 3.14
- The radius of the tire is 1.5 inches
- Timer showed 333 seconds

31. Based on the *Python* code presented, create a program with blocks that gives you the correct outputs with the 1.5 and 333 values hardcoded in. Once that works, you can modify the code based on what you have learned to allow inputs for the radius and time:

```
import math

circumference = None
minutes = None
radius = None
remainder = None
seconds = None

"""Describe this function... This function finds the
circumference of a wheel based on the radius and
convert second into minutes and seconds using modulo.
"""

def FindCircumference():
    global circumference, minutes, radius, remainder, seconds
    circumference = (2 * 3.14) * radius
    minutes = seconds / 60
    print(round(minutes))
    remainder = seconds % 60
    print(remainder)
    return circumference

radius = 1.5
seconds = 333
print(FindCircumference())
```

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

32. Once you have the code behaving as you like, modify it to allow inputs for the radius and time, making your program interactive.
33. Once it is dynamic, capture a screenshot of your completed code in both the block view and in the *Python* view. You will submit these to your teacher, and save them to review later.
34. Complete the review:

Refer to your downloadable resources for this material. Interactive content may not be available in the PDF edition of this course.

PLTW DEVELOPER'S JOURNAL What terms will you be using interchangeably from time to time until you dig deeper into text-based coding?

Conclusion

1. How did you interpret and respond to the essential questions? Capture your thoughts for future conversations.