PLTW COMPUTER SCIENCE

Activity 3.1.4

Iteration and Counts: Course Registration

goals

- Learn to create, manipulate, and access lists in *Python*
- Develop a program using tracer rounds
- Pair Program to develop code that solves a problem and generates understanding

description of tasks

- Combo Menus Revisited: Modify your previously developed code to accept more than one order.
- Course Registration: Modify a course registration program to explore functions in *Python.*

Essential Questions

1. How does parameterization generalize a specific solution?
2. How is abstraction managing complexity in my program?
3. How is iteration managing program flow?

essential Concepts

- Python Programming Language and Cloud Computing
- Coding Fundamentals: Algorithms, Functions, Variables, Arguments, Procedures, Operators, Data Types, Logic, Strings and Concatenation
- Version Control, Iteration, and Debugging
- Tracer Bullet Development

Resources

Activity 3.1.4 Student Files

For Loops and Practicing Syntax

You have already seen iteration in MIT App Inventor and even a bit in *Python* in the previous activity. In *Python* you can use for loops and while loops to accomplish iteration. Anything that can be done with one type of loop, with some modification, can be done with the other.

One of the most common needs a *Python* programmer has is to do an action with every element in a list. In the last activity, you were provided the for loop code to do this in CSE_313_skeleton.py. For your reference, here is an example of doing an action for all items in a list:

```
for post in post_list:
    print post
```

The first line contains:

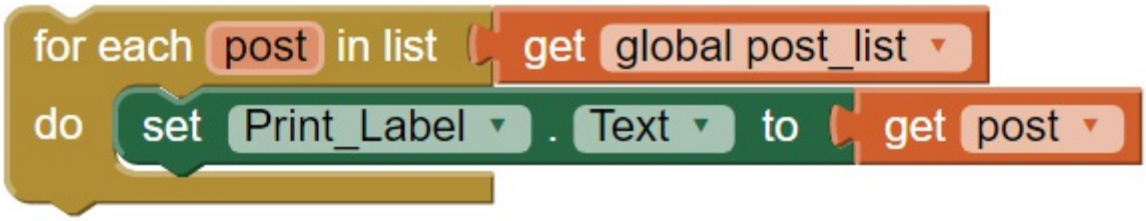
- the keyword “for”
- a variable identifier (in this case “post”) used to reference elements
- the keyword “in”
- a variable containing a list
- the : character showing that the rest is coming on the next indented line

In other words:

for	keyword
post	variable identifier of each individual element
in	keyword
post_list	a variable identifier for the list

Natural language: FOR each ELEMENT IN this LIST

The loop structure is used in different programming languages, but the syntax can vary and abstraction levels can vary. For example, here is the same loop structure in block-based programming:



The functionality of this structure is to take the first element in list and store it in the variable item, execute all statements in the indented block that follows—in this case, print the element—before moving on to the next element in post_list. The loop repeats this process until there are no more elements left in post_list.

1. Create a new python file to use as a work file for this activity.
2. Examine the following loop. Make a hypothesis, what do you expect the output to be?

```
my_list = [0,1,2]
for number in my_list:
    print number
```

3. Open a new terminal tab with one of the following methods:
   - Click the + sign to select to open a new terminal.
   - Shortcut keys you can also use: Alt + T.
4. In the new terminal window, identify the programming language you want to code in: type "python" and press enter.

```
python
```

5. If necessary, open a new bash tab. To test your hypothesis of what you expected the output to be, enter the statements into the *Python* interactive interpreter and please note:
   - After you type the line containing the for-loop header and press Enter, you will receive a different prompt containing an ellipsis (…) instead of the usual triple right-chevrons (>>>).
   - When this happens, you must include four spaces at the beginning of your next statement or you will receive this error:

```
IndentationError: expected an indented block
```

6. When you are done typing the statements in the body of your for loop, press **Enter** at the final ellipsis prompt to execute your loop.

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

7. Examine the following for loop. Before you run the code, guess what the output will be.

```
x = 0
my_list = [0,1,2]
for number in my_list:
    x = x + number

print x
```

Important: Make sure to leave a blank line to exit out of the loop before adding the print statement.

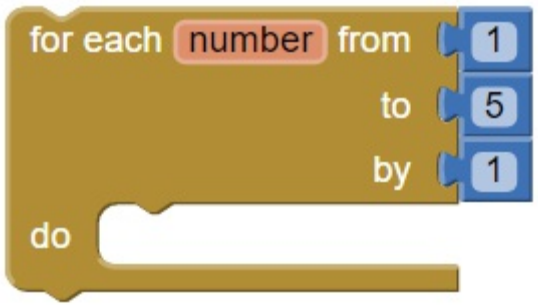
8. Test the code to see whether you are correct.

PLTW DEVELOPER'S JOURNAL Record the basic structure and syntax that defines a for loop.

Range Function

Another useful function in *Python* is the range function. The range function identifies a number to start at, a number to stop at, and how much to change the number by each time. It will loop through as many times as necessary to get to the end value.

As for the previous loop structure, this image shows block-based programming at a higher level of abstraction. The math and control blocks just fit together in the range control block and allow a developer to enter custom numbers. You did not need to think about anything more then plugging in numbers.



The *Python* coding language is a lower level of programming abstraction than block-based language, and must have the following syntax:

```
range(start, stop[, step])
```

If the syntax is not exact, the loop will not execute.

The range control block is a great way to start thinking about *Python* code. The first argument is the number to start at, and the second number is the number at which to stop. The third argument is optional; it is the increment value, or the number to count by.

The square brackets [ ] in the documentation indicate that it is optional to include the part inside the brackets. If you include the part, do not include the brackets. Do not confuse these brackets with the brackets indicating a list.

9. From the official *Python* documentation, try these examples in the interpreter:

```
range(10)
range(1,11)
range(0,30,5)
range(0,10,3)
range(0,-10,-1)
range(0)
range(1,0)
```

10. Record your expectations for the lists created by each of the following calls to range, along with a brief explanation of why you think it will produce the lists you predict.

```
range(5)
range(2,5)
range(5,1,-1)
range(1,5,-1)
```

PLTW DEVELOPER'S JOURNAL Test each of the above range functions in the interpreter and record notes for yourself. Especially note if any range() behaved differently than you expected.

Combining the For Loop and the Range Function

You have used the for loop to iterate over a list as in:

```
for post in post_list:
```

For loops can also be used to iterate a specific number of times. In fact, for loops and the range function are often used together in *Python*.

Here is an example of a for loop that iterates 10 times. Notice how it might appear to iterate 11, but remember, 11 is not included in the range function as written:

```
for number in range(1,11):
    print number
```

11. Given the example above, try writing loops using the range function that prints each of the following. Please note that "inclusive" means it should print all the numbers, including 1 and 19 in the first loop for example.
    - ☐ Numbers 1 to 19 inclusive

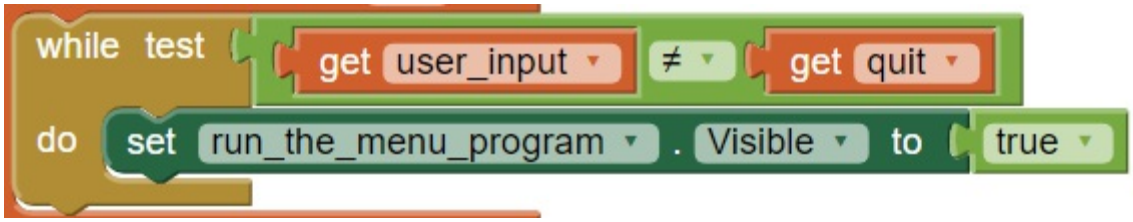
- Numbers 20 to 1 inclusive
- Even numbers from 2 to 10 inclusive
- Odd numbers from 9 to 1 inclusive

- 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 Record information about using the range function in loops.

While Loops

Another type of loop is a **while loop**. While loops check for a condition and continue to loop as long as the condition is true. While loops are used in block-based programming, as in this example:



The syntax of a while loop header is:

```
Header - keyword while followed by an expression to evaluate:
    Indented block of code to be executed.
```

Important: An expression in a header may itself contain multiple sub-expressions.

In **Activity 3.1.3**, you were provided skeleton code that included the following while loop:

```
while user_input != "quit":
    #The do part of the loop was the whole combo menu program
    #It continued to loop until the user typed "quit".
```

With while loops, the block of code is executed until the expression in the header evaluates to False. In the example of Activity 3.1.3, the header returned true (the user input does not equal "quit") and looped the program. If the user entered "quit", the evaluation would then return false, because the user input would be equal to quit. That is why a user of the CSE_313_post program from Activity 3.1.3 can continue entering commands until they enter the text "quit".

Here is a while loop that prints the numbers from 0 to 10 inclusive:

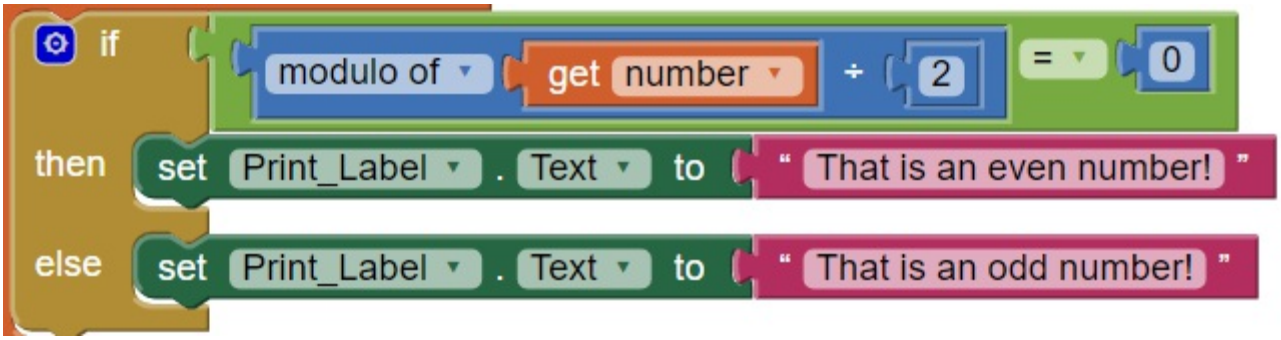
```
x = 0
while x < 11:
    print x
    x = x+1
```

PLTW DEVELOPER'S JOURNAL Record information about a while loop, such as what it is, what is the syntax, and when it might be used.

Modulo Operator in Loops

The modulo operator is something you saw in block-based programming. It does division of two numbers, but it focuses on the remainder of the division instead of how many times the number is divided.

12. Take a minute to review modulo in block-based programming. What would this code section do, and how do you know?



Just like the MIT App Inventor example, the modulo operator (%) in *Python* can be used to determine evenness or oddness by finding the result of mod 2. If the result is 0, the number is even; if the result is 1, the number is odd. For example, this is expression evaluates to True:

```
4 % 2 == 0
```

13. Create and test while loops that print each of the following:
    - ☐ Numbers from 1 up to 5 inclusive
    - ☐ Numbers from 10 down to 1 inclusive
    - ☐ Even numbers from 2 to 10 inclusive
    - ☐ Odd numbers from 9 down to 1 inclusive; use the mod operator and a conditional

Important: Having each of these while loops in the same file, with the same variable, will cause different results. Use different variable letters, comment out loops as you get the desired output, or use separate python files.

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

Combo Menu Iteration Using Loops

As you learned earlier, for loops are generally used to iterate over each element in a collection, like a list. They also are used to do something a specific number of times.

While loops are generally used when execution may need to end early, or at a specific time, such as by user input as with the CSE_313_post program.

14. Duplicate and rename your code from CSE_313_combo_menu_list, to "CSE_314_combo_list_loop".
15. Modify the Combo Menu code using the loop of your choice to make it so that the user has the option to keep entering more orders in the "CSE_314_combo_list_loop" file.
    - It is possible to highlight multiple lines and press the Tab key on the keyboard to increase the indent of those lines.
    - You can also "un-indent" by highlighting lines and pressing Shift+Tab.
    - You may need to change how the order is printed, but make sure it is printed inside the loop.

Understanding Functions and Methods

After updating your combo menu to include a loop, you are going to apply loops to skeleton code. The code includes a few different files that import information from each other. To understand these imports, it is important to understand functions and **methods**.

16. Download the **source files**.
    - CSE_314_skeleton_registration.py
    - CSE_314_skeleton_student.py
    - CSE_314_course.py
17. In your workspace, right-click in an open space around the files and select to create a new folder.
    - Name this folder CSE_314.
    - Upload the CSE_314_skeleton_registration.py, CSE_314_skeleton_student.py, and CSE_314_course.py files to this folder.
    - Open all three files.
18. Examine the following function that currently exists in the Student class:

    ```
    def get_first_name(self):
        return self.first_name
    ```

    At the end of this activity, you will need to be able to modify this function and copy the basic code in the registration file. It is important to read the function. Otherwise, you might struggle in the next part of the activity. In the files, add code and explanations to help you later.

A function definition header begins with the keyword “def” and is followed by an identifier for that function, in this case “get_first_name”. The identifier is followed by an open parenthesis, option arguments separated by commas, and a closed parenthesis. The line is terminated by the : character.

If the function belongs to a class and you would like it to be accessible as a method (from outside of the class to get information contained in specific instances of that class), the first argument in the function definition should usually be “self”, as it is in this function.

As with loops and conditionals, the code that is executed when the function is called is included in an indented block following the function definition header. In the case of the example function, this body is only one line of code.

The return statement in the body of this function is a special statement that feeds a result back to the program that called that function. In this case, in the Student class, the get_first_name function returns the first name of a particular student.

Warning: In *Python*, functions cannot be called until after they are defined. For this reason, it is good practice to define any functions you will use at the top of your file. Otherwise, you risk getting a NameError: name 'function_name' is not defined message.

Course Registration Tracer Program

19. Upload the activity files to your Cloud9 work space. Do not change the names, as they are linked together that could impact the files ability to call functions from each other.

20. Look through the source files comments.

21. Complete the “TODO” comments in each of the skeleton files and then run and test your code.

    - Look at the existing code in the file to see the layout and specifics of how other parts are done to copy style and syntax basics.
    - When you copy and paste code as a guide, you may need to revise parts of what you pasted so the program addresses the part of the program you copied it into, instead of doing the part you copied it from multiple times.
    - You can use the tab key for “code completion”. While you type, the editor makes a best guess as to what you want to type, and pressing the Tab key will insert the best guess into your code.
    - The files depend on each other, so changing the name of a file means you need to update the first few lines of the other files you are working with that access that file.
    - Debugging tip: You may not see an output that you were not expecting, because it is coming from a different file, through the file you are running. Be sure to look at the output of both skeleton files when debugging.
    - Read the errors. They will tell you what file and what line to look at. Start with the first error, as later errors might adjust after you fix the first one in the program.
    - Capitalization matters! Double check that you are using consistent capitalization throughout all files with the same words.
    - If you get outputs that includes <bound method …> check whether you included the parentheses when you called a method.

- Some ways to format your output:

"\n" Creates a new line in a print string output

"\t"	Creates a tab in a print string output
" "	Creates a new line in a print string output. (The / indicates that the tag is both an open and close tag. The creates a single line break in this location.
"\"	At the end of a line while typing a string, type \ then hit enter to continue a long string on the next line in *Python*, but still have the entire string output print on the same line to a user.

22. When you have completed your code, let your teacher know.

Create Your Own Function

You are now going to define your own function identified by the name *assignment1* with no arguments. When called, the function should display the text “Assignment 1 Completed”. Use the other parts of this activity to complete each of the following steps.

23. In Cloud9, create a new python file, CSE_314_assignments.py.
24. Define a function named “assignment1”.
25. When the function is called, it should print “Assignment 1 Completed”.
26. Call that function. Run and test your code.
    - Look at the files you just modified to see how those functions are set up.
    - Review what the arguments are in previous functions, so you know what to leave blank. You will still need the punctuation around where an argument would normally be.
    - Calling an assignment is the same code pattern used to create the assignment, just without the def keyword and the “:” character.
27. In CSE_314_assignments.py, create a second function identified by “assignment2” with no arguments that returns the string "Assignment 2 Completed".
28. In the same file, call that function. Run and test your code.
29. What do you notice about how the *assignment1* function behaves compared with the *assignment2* function?

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

30. Modify your call to *assignment2* so that it stores the result of the function in a variable, like so: assignment2_result = assignment2()
31. Run and test your program. Does this change how it behaves?
32. Beneath the assignment statement that you just added, add a print statement to display the contents of *assignment2_result*.
33. Run and test your program, does the function *assignment2* display the text, or is something else happening?

34. Modify your *assignment2* function so that it displays "Assignment 2 Completed" in preparation for the next part of this activity.

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

Argument and Conditions

Assume you have bugs in both assignment1 and assignment2 functions. To track down the bugs, you are going to create a temporary function to control the flow of execution. In other words, you will write a function to control how and when you call assignment1() and assignment2(). In this case, the temporary function will be called execute_assignment. So, start debugging.

35. Create the function execute_assignment that has one argument with identifier assignment_number.
36. Within this function, use a conditional to execute assignment1 or assignment2 depending on the value of assignment_number.
37. Add user input to your program so that when the user enters “1” or “2”, that value is passed to the execute_assignment function.
38. Run and test your program to be sure that when the user enters 1, "Assignment 1 Completed" is displayed, and when they enter 2, "Assignment 2 Completed" is displayed.

PLTW DEVELOPER’S JOURNAL Record techniques you can use in the future whenever you program to make it easy to test small pieces of code individually.

39. Modify this program with a loop so that it allows the user the option to execute additional assignments until they type “quit”.

Modifying the Registration Program

You may have found it annoying that printing a list of all students also included all of their courses. It makes the output needlessly long when all you might want is their first and last names. To get only their names, you could write a custom function in the Student class.

40. Modify the registration program files so that it no longer prints all of the courses taken by a student when you list all students.

Conclusion

1. Describe “for loops” and “while loops”.
2. How can the use of functions make your code more readable?
3. How do loops change the flow of control in a *Python* program?
4. How do functions change the flow of control in a *Python* program?
5. How did you interpret and respond to the **essential questions**? Capture your thoughts for

future conversations.