



JANUARY 11, 2017

LOOPS AND DECISION TOOLS

TOPIC 4

VINCENT A. DINOTO, JR.
JEFFERSON COMMUNITY AND TECHNICAL COLLEGE
Vince.dinoto@kctcs.edu

Contents

Loops and Decision Tools	3
If, Elif and Else	3
If/Else Code.....	4
If/Elif/Else Code.....	5
Assignment 4.1	9
Loops.....	9
While	10
For	11
Nesting	13
Assignment 4.2.....	14
Case Study	14
Index	15

Figures

Figure 1: Decision Symbol	3
Figure 2: Simple If - Else	4
Figure 3: Simple If Code	5
Figure 4: If/Elif/Else	7
Figure 5: If/Elif/Else Code	8
Figure 6: While Statement	10
Figure 7: While Code	10
Figure 8: While Results	11
Figure 9: For Statement	11
Figure 12: Nested For Statements	13
Figure 13: Partial Result List of Nested For Statements	14

Loops and Decision Tools

This module will discuss tools that allow for branching based upon the input parameters and to complete repetitive processes. Commands such as **If**, **While** and **For** will be discussed in this context. No geoprocessing commands will be needed, so therefore loading arcpy is not required in the scripts, but will not change the results. This module will be completed inside the IDE in programming mode.

These types of functions provide the real power of using a scripting language verses doing the processes in a typically manual fashion. Remember, the concept is to reduce the manual and repetitive processes through the use of Python scripts. The **If** statement was introduced in the second module and those concepts will be expanded on in this lesson.

If, Elif and Else

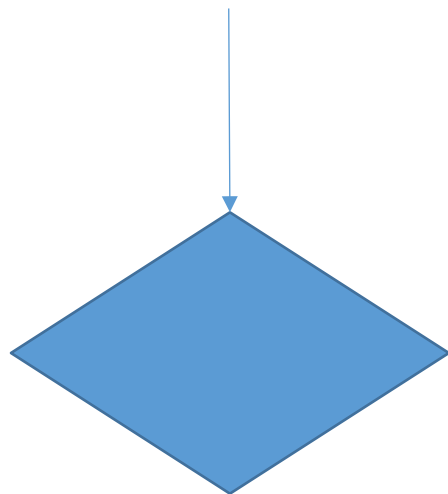


Figure 1: Decision Symbol

When a decision is required typically, a diamond shaped symbol is used and the input is shown coming into the top of the diamond and outputs are shown coming out of the sides and bottom. Care should be taken to make sure that lines in the diagram have arrows to show flow direction of the process. Typically, the symbol pathways are labeled as true and false.

The statement uses logic to determine how the branch will proceed. If the statement is true then it will branch in one direction and if false, a different pathway will be followed.

Contained within the diamond symbol is a function, which can be extremely simple or very complex. Logical operations such as equal, greater than, greater than or equal, less than and less than or equal are typical logic that might be used, which includes Boolean algebra. For example, simply comparing one variable to another can be the operation. The variables can be numeric or strings, but not a combination of the two.

The process will be to determine a logical decision in the blue diamond shaped symbol. If true, the pathway would lead to the green shaded box. Multiple functions can be located inside this box, when the operation is completed, the flow would be to the blue box with the curved corners, which would be outside of the loop. If the logic in the blue diamond shaped symbol is false then the flow would be to the tan colored box to the left, which would contain an **Else** statement.

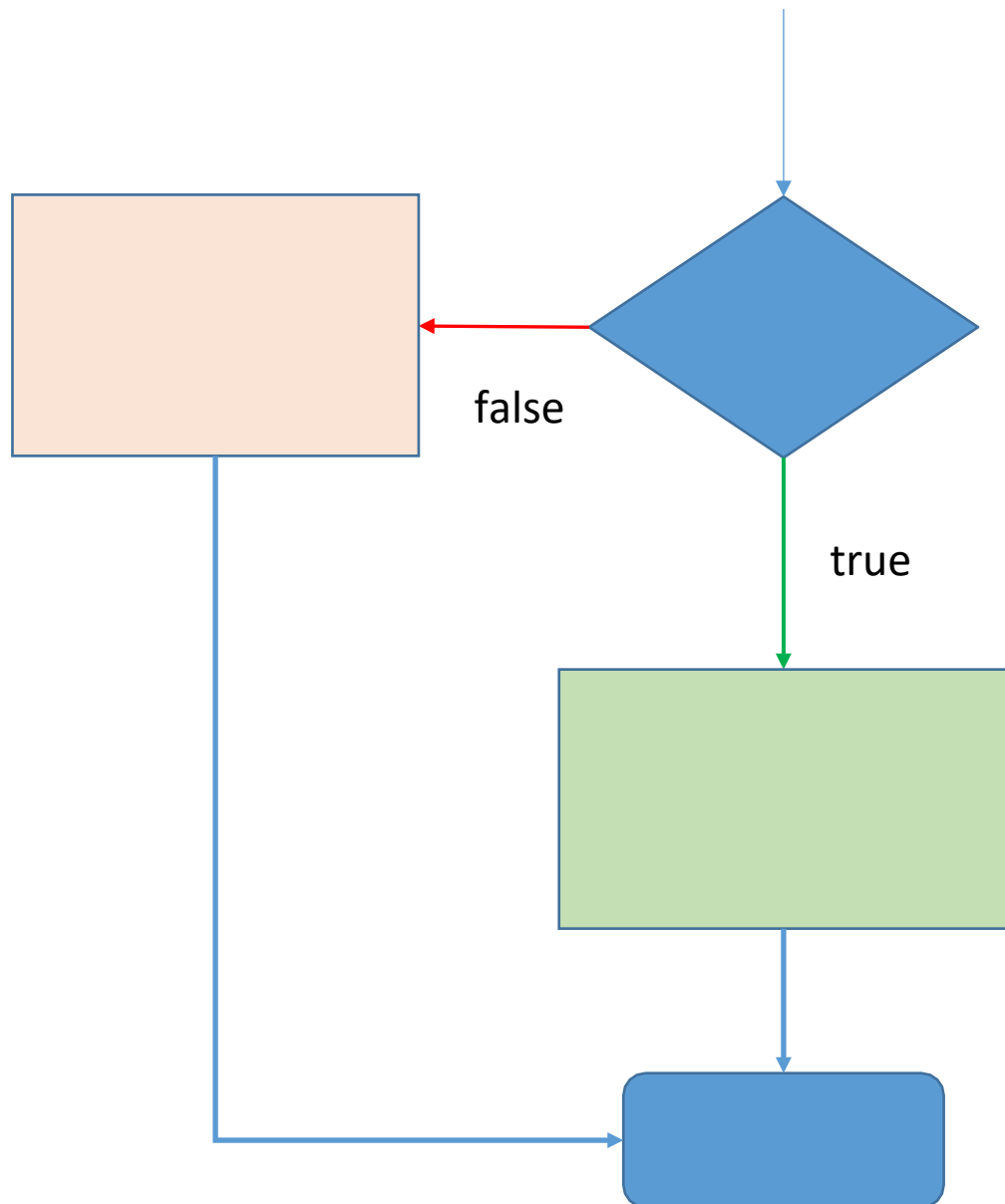


Figure 2: Simple If - Else

Multiple processes could be completed within this element just as was done with the true pathway. Once these operational parameters are completed, the flow would be to the blue box with the curved corners. Thus when the **if** statement is completed the program continues to the same logical location no matter which process pathway was followed.

If/Else Code

These concepts are applicable to any scripting or programming language, but the specific Python script completes the application. The code should be accomplished in programming mode and **not** in immediate mode. The code contains three unused statements, which include the import of `arcpy`, the import of the environment and the workspace pathways. These statements were part of the template used in previous lessons.

The **If** statement must have a specific format, which has a colon at the end of it, the dash at the beginning of the statement is not typed and was added by the IDE. In the IDE the indentation of the line is added, if an IDE is not used it would manual indentation. The indentions are critical and must always have the same number of spaces.

```
#Created by Vince DiNoto
#
#Example of using the if, |and else commands from python with arcpy
#
#load the ArcPy module
import arcpy
#load the environment
from arcpy import env
#if not imported could be used like arcpy.env.workspace
#since imported it is
env.workspace = r"C:\Users\vdinotojr0001\gis data\KY.gdb"
#if command
- if hours < 12:
    studentType="beginning freshman"
- else:
    studentType="student has at least 12 credit hours"
#outside of the if statement
print "done"
```

Figure 3: Simple If Code

1. The **If** statement would be in the blue diamond in Figure 2 and the “beginning freshman” string would be in the green box in Figure 2. The *studentType* of “beginning freshman” would only occur under the condition that the **If** statement was true, which means the student had less than 12 credit hours.
2. The value of the variable the hours can be loaded in immediate mode prior to writing the script or could be a line added above the **If** statement defining the variable.
3. If the statement is false, i.e. the value is 12 or greater than the pathway would lead to the **Else** statement. The pathway would have followed the red arrow to the tan box in Figure 2. The tan box would contain the **Else** statement and anything below the **Else** statement that is indented. The value of the *studentType* variable would be “student has at least 12 credit hours”.
4. Then both pathways would lead to the **print** statement, which is represented with the blue curved corner box in figure 2. The value of *studentType* is not displayed, so the done statement could be modified instead to print the value of the variable.

If/Elif/Else Code

In the first example, there was only one decision and it was either true or false. In Python, scripting multiple decision statements can be grouped together by using the **Elif** statement, which means **Else If**. So when the **If** statement is false the process can move to an Elif statement. When the **Else If** statement when is true it provides a different solution and when false either

another **Elif** statement could be used or an **Else** statement is used to end the routine. Independent of the pathway, the script will then return to a common point, outside of the indentions in the script.

For example, if a college wants to determine the status level (in plain text and not as a number) of each student that has been geocoded automatically, the **if**, **elif**, **else** might be used. Therefore, using the number of credit hours earned, the student could be classified. The criteria might be like the following:

- Less than 12 credit hours they are listed as beginning freshmen.
- Between 12 and 29 hours, they are listed as freshmen.
- Between 30 and 44 credit hours, they are listed as a second year students.
- In addition, if 45 hours or more they are listed as nearing graduation.
- At the end of the process the status level of each student should be printed, it could also be placed into the data as a new field.

See Figure 4 to follow the flow chart, the flow chart starts in the upper right corner.

1. The value is compared to see if it is less than 12 and when true the process will drop straight down to the green box below the first blue diamond and the *studentType* will be set to 'beginning freshmen'.
2. If false than the value must be 12 or more, so the next comparison is with 30, which will give a true for all values between 12 and 29 and would drop to box below the second blue diamond and the *studentType* will be set to 'freshmen'.
3. If false, we know the value of credit hours must be at least 30 and the next decision will determine if it is less than 45. If true it uses the commands in the third green box from the right and the *studentType* will be set to 'second year'.
4. If false it uses the tan colored box, since the student must have at least 45 credit hours and the *studentType* will be set to 'nearing graduation'.
5. In all cases the student type is printed.

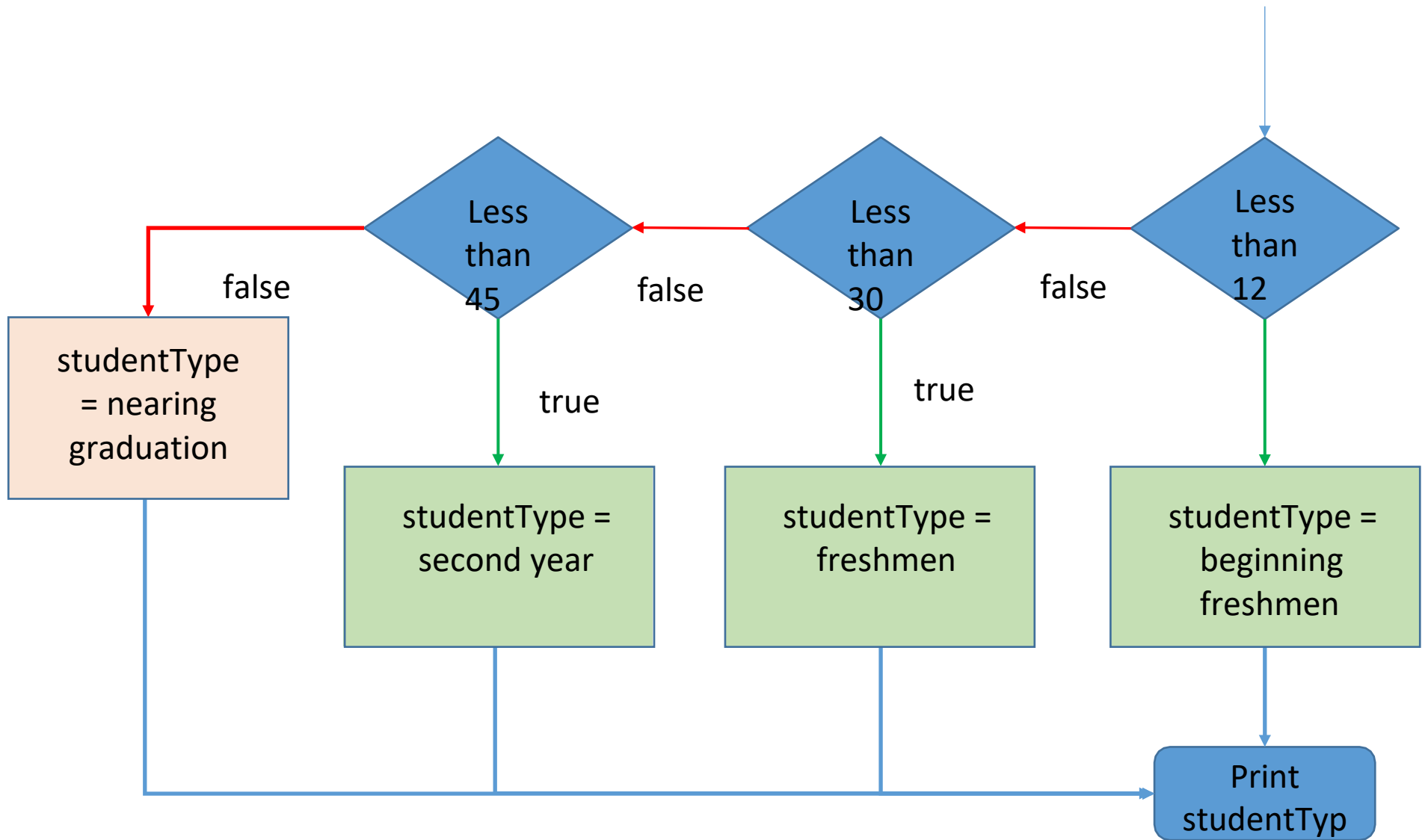


Figure 4: If/Elif/Else


```

#Created by Vince DiNoto
#
#Example of using the if, else if and else commands from python with an
#
#load the ArcPy module
import arcpy
#load the environment
from arcpy import env
#if not imported could be used like arcpy.env.workspace
#since imported it is
env.workspace = r"C:\Users\vdinotojr0001\gis data\KY.gdb"
#if command
- if hours < 12:
    studentType="beginning freshman"
- elif hours <30:
    studentType="freshman"
- elif hours < 45:
    studentType="second year"
- else:
    studentType="nearing graduation"
print studentType

```

Figure 5: If/Elif/Else Code

The code view for the flow chart of Figure 4.

1. As previously noted the first three commands are not used in this process.
2. A value for hours would need to be defined outside of this script.
3. The **if** statement is comparing hours to the number 12.
 - If the hours are less than 12 then it will complete items that are indented and move then to the print statement,
 - If false, it will move to the first **elif** statement.
4. The first elif statement is comparing to see if the value is less than 30.
 - If the hours are less than 30 then it completes the items indented and moves to the **print** statement,
 - If false the hours must be more than 29 and moves to the second **elif** statement.
5. The second elif state is comparing to see if the value is less than 45.
 - If the hours are less than 45 then it completes the items indented and moves to the **print** statement.
 - If false the hours are more than 44 and moves to the **else** statement.
6. The else statement provides a value for the *studentType* and moves to the print statement.
7. The key to all of the logic statements is the proper indention of the lines.

Assignment 4.1

No geoprocessing is done in this assignment and no external files are required. All data will be inputted into the code or through the immediate mode. Provide your instructor with the script and the results screen. To accomplish this assignment the learner will need to nest if statements inside other if statements.



Determine the weight and height of ten individuals, each individual characteristics can be inputted manually, i.e. inputting the weight and height. If the learner wishes to use an array and a **For** statement (next topic) extra points will be awarded. The following classifications will be used:

- If 72 inches are higher classify as tall and
 - If over 200 pounds classify as heavy
 - If under 150 pounds classify under-weight
 - All others classify as normal
- If less than 72 inches but greater than 65 inches classify as average and
 - If over 180 pounds classify as heavy
 - If under 135 pounds classify under-weight
 - All others classify as normal
- If 65 inches or less classify as short and
 - If over 170 pounds classify as heavy
 - If under 120 pounds classify under-weight
 - All others classify as normal

Loops

When the same operation needs completion multiple times, a loop statement is required. For example, if you have a list variable and want to do the same operation on each member of the list. There are two types of loop statements the **While** statement which requires that a counter be incremented and the **For** statement which will look at every member of a list variable.

The same operation is done to each member in the indented section of code; variables are different for each iteration. The process is repeated when the indented content gets to the end unless the process has reached the specified number of iterations, such as using all the elements in a list variable. Loop statements can be nested together, which means that one loop statement can have another loop inside it, which requires additional care on indentation.

While

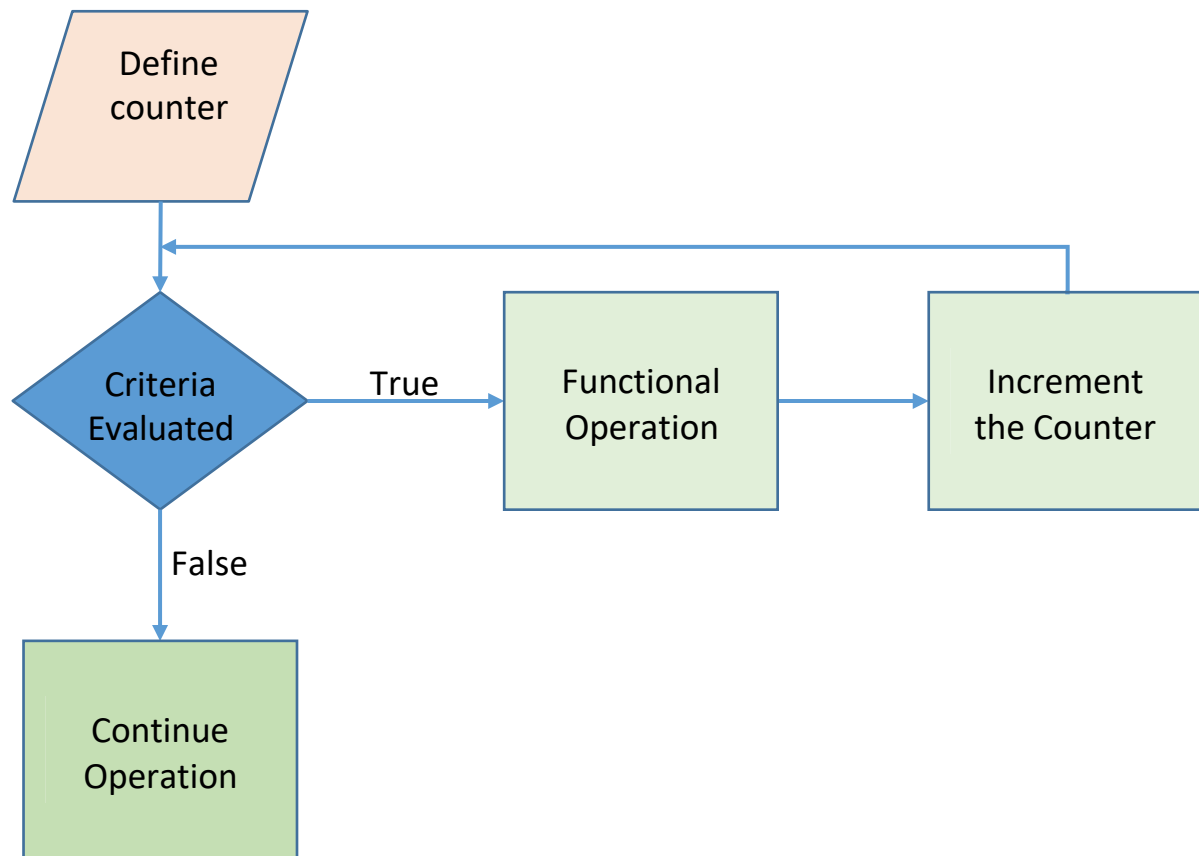


Figure 6: While Statement

In Figure 6 the flow chart is represented and the code is visible in Figure 7.

1. First, a counter is set equal to 1 outside of the loop. This counter will be incremented each time through the **While** statement until the criteria is no longer true in which case it will drop out of the loop and go to the first line without an indentation.
2. If the counter is less than or equal to 8 it is true it will produce the **print** statement that is indented. Note: the str is used with the counter since the counter is a number and the items in quotation marks are a string.
3. The next indented line will increment the counter by 1, notice the format.

```
#Created by Vince DiNoto
#GeoTech Center
#
#While Function
#
counter=1

while counter <= 8:
    print "counter = " + str(counter)
    #increment counter
    counter += 1
    print "the loop has successfully run"
```

Figure 7: While Code

4. A comparison is done again to see if the value has exceeded 8 if not it will go to the **print** statement and then increment the counter. When the counter is greater than 8 the routine will end and it will go to the first line without an indentation which would be to print the statement that the loop has run successfully.

The result of the script from Figure 7 is shown in Figure 8.

```
counter = 1
counter = 2
counter = 3
counter = 4
counter = 5
counter = 6
counter = 7
counter = 8
the loop has successfully run
```

Figure 8: While Results

For

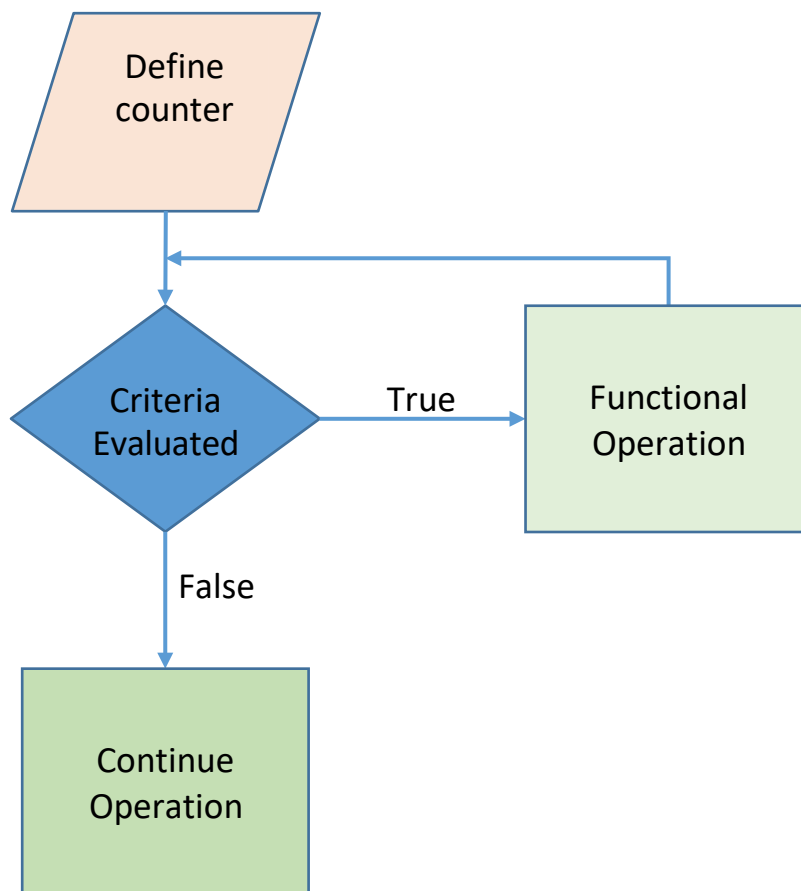


Figure 9: For Statement

The **For** statement is similar to the **While** statement but the counter will not be a numerical counter but instead will be a list variable. In Figure 10 the **For** statement code is displayed and corresponds to the diagram on Figure 9. The code contains only four functional lines. The usage of the colon and the indentation of the text are critical.

The *nameKCTCS* list variable contains 16 members but only a part of the list is displayed in Figure 10. While the last **print** statement is not required, it is used as a troubleshooting method to make sure that the entire process is being completed. It should be removed after the script is fully tested.

Code in Figure 10 based on the flow chart in Figure 9

1. In the first line of the code, the list variable is defined.
2. The **For** statement uses a variable college which contains a value from the list variable *nameKCTCS* variable, this variable will change each time through the loop. For example, the first time through the loop, college will have the value of Jefferson, the second time through college will have the value of Elizabethtown.
3. The next line of code will print the name of the college. Additional parameters could be added as required.
4. When all sixteen colleges have been printed, then all members of the list variable will have been used and the For statement will be completed and the print line outside of the indention will be executed.

```
#Created by Vince DiNoto
#GeoTech Center
#
#For Function
#
nameKCTCS=["Jefferson", "Elizabethtown", "Maysville", "Bluegrass",
for college in nameKCTCS:
    print college
print "done with list"
```

Figure 10: For Statement Code

In Figure 11 displays the output of the code from Figure 10.

```
Jefferson
Elizabethtown
Maysville
Bluegrass
Hopkinsville
Henderson
Madisonville
Owensboro
Paducah
Big Sandy
Gateway
Ashland
Southeast
South Central
Hazard
Somerset
done with list
```

Figure 11: For Statement Results

Nesting

Nesting **For** statements occurs when one **For** statement is executed and a second statement is contained within the first. This is accomplished by using two different amounts of indentation. The inner statement would be indented a greater amount than the outer. The inner loop is executed until it has met the criteria and then returned to the outer loop. The next step of the outer loop is executed and then the inner loop is executed again. This process continues until the outer loop's criteria is met.

The code for the nested **For** Statement is shown in Figure 10.

```
#Created by Vince DiNoto
#GeoTech Center
#
#For Function
#
collegeSemester = ["fall", "spring", "summer"]
nameKCTCS=["Jefferson", "Elizabethtown", "Maysville", "Blueg
for college in nameKCTCS:
    print college
    for semester in collegeSemester:
        print "    semester = " +semester
print "done with list"
```

Figure 10: Nested For Statements

1. The first two statements define two string list variables, the first variables are the traditional college semesters and second variable is the colleges within KCTCS, the first list variable is composed of three members and the second is composed of 16 members (Note, not all members are visible in the screen capture). The variable definitions are outside of the **For** statements.
2. The **For** statement is using a variable 'college' which will have the value of one member of the list variable *nameKCTCS* each time through. The first value in this list is Jefferson. The next line of code will print the value of the variable *college* and the first time through that will be Jefferson.
3. The second **For** statement is nested inside the first (note the indentation), the *semester* variable is used with the *collegeSemester* list variable. The first time the variable *semester* would contain the value fall. The action to be taken in this **For** statement is printing the semester name. When the **print** command is executed, it will return to the inner **For** statement and determine if any additional members are in the *collegeSemester* list, the next value for the *semester* variable is spring. This is repeated until all members of the *collegeSemester* list have been used.
4. Once the inner For statement is complete the script returns to the first For statement increments the variable *college* to be the next member which is Elizabethtown, prints its value and then does the inner **For** statement three time.

5. This process is repeated until all colleges have been displayed along with the semesters.
6. Since there are 16 colleges and 3 semesters, the inner loop is completed a total of 48 times. A partial list of the results are shown in Figure 11.

Assignment 4.2

In this assignment a nesting of **For** and **While** statements will be created.

- The variable for the **For** loop should contain 13 unique values, what types of variables and values are left to the discretion of the learner. The **For** loop should print the value of one element of the variable before executing the **While** loop.
- A **While** statement should be an inner loop inside of the **For** loop and the counter should cycle through 15 times, the output from the **While** loop should be the value of the counter indented five spaces.
- Provide script and screen image.



```
Ashland
    semester = fall
    semester = spring
    semester = summer
Southeast
    semester = fall
    semester = spring
    semester = summer
South Central
    semester = fall
    semester = spring
    semester = summer
Hazard
    semester = fall
    semester = spring
    semester = summer
Somerset
    semester = fall
    semester = spring
    semester = summer
done with list
```

Figure 11: Partial Result List of Nested For Statements

Case Study Topic 4

In the Case Study the learner should utilize any components that have been learned to this point in the course, but not all components need to be used. The learner should see this as a real world case. The whole process must be done in a single script.

You have been asked to develop five regional geographies each containing at least four counties and they must be contiguous.

For each regional geography you will produce:

1. a census tract feature file and map
2. a road feature file and a map displaying the file
3. a stream/river feature file and map
4. a county boundary feature file and map

Your starting point will be individual county boundary files, individual county census tract files, roads of the entire state, and rivers of the entire state. The process must contain at least one **If**, **While** or **For** statement more than one can be used. Provide your instructor all files and maps generated. The information should include a discussion of the process, a flow chart and a list of variables with their values.



Index



JCTC_CIT 299_Module 2_Topic 4_Loops_And_Decision_Tools by Vincent A. DiNoto, Jr. is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

This workforce product was funded by a grant awarded by the U.S. Department of Labor's Employment and Training Administration. The product was created by the grantee and does not necessarily reflect the official position of the U.S. Department of Labor. The U.S. Department of Labor makes no guarantees, warranties, or assurances of any kind, express or implied, with respect to such information, including any information on linked sites and including, but not limited to, accuracy of the information or its completeness, timeliness, usefulness, adequacy, continued availability, or ownership. This is an equal opportunity program. Assistive technologies are available upon request and include Voice/TTY (771 or 800-947-6644).