ME30 Lab3 Decisions

February 20, 2019

0.0.1 ME 30 Lab 4 - Conditional Program Execution

ME 30 ReDev Team
2018-07-06

Description and Summary: This lab introduces the programming concept of *decision*-making, also called conditional program execution, where the flow of the program is altered depending on whether a condition is satisfied (TRUE condition) or not (FALSE condition). In this lab, we will look at:

1. Boolean expressions (code that evaluates to TRUE or FALSE)
2. Logical operators (AND, OR, and NOT)
3. Conditional statements (the if statement)
4. Chained conditionals (else-if)
5. Nested conditionals (conditional statements that are nested in the branches of another conditional statement)

Use the ME 30 Lab Report Jupyter Lab notebook template to write your lab report. ***

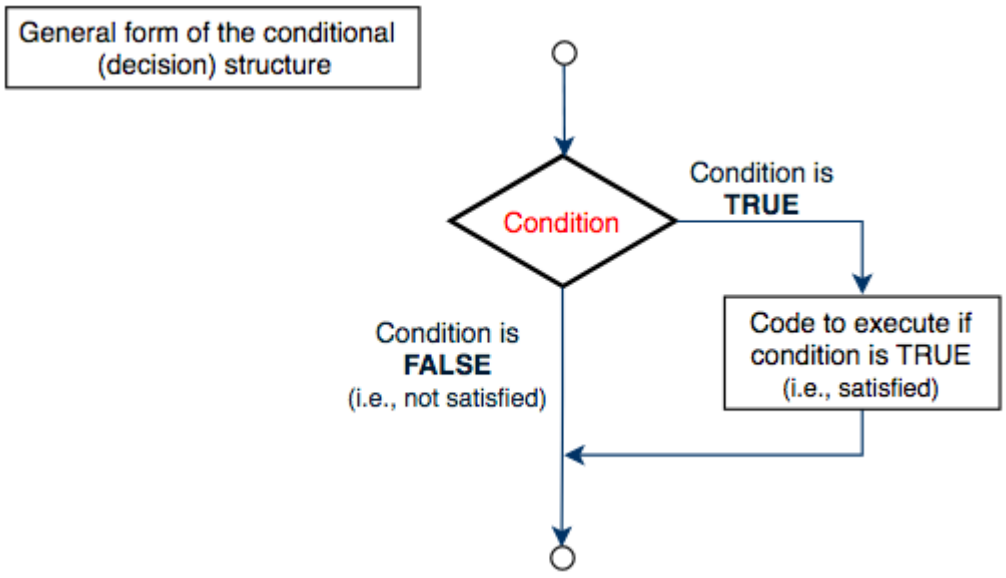
0.1 Boolean Expressions

Recall from your textbook that a boolean expression is an expression that evaluates to either True or False. Your text lists a set of *relational* operators that compares two *operands* (expressions on the left and right-hand sides of the operator) and yields a boolean result:

```
x == y  # x is equal to y
x != y  # x is not equal to y
x > y   # x is greater than y
x < y   # x is less than y
x >= y  # x is greater than or equal to y
x <= y  # x is less than or equal to y
```

0.2 Exercise 1 - Explore the relational operators

Try out some of these relational operators in some python code cells. Try out not just integers. What happens if you use the relational operators on floats and strings? Add observations about your explorations in Markdown cells, especially for results that you did not expect.



General decision structure

0.3 Conditional Statements

The general form for a conditional or 'decision' structure is shown in the figure below. [Note: in graphical representations of program execution (flow charts), it is customary to denote a decision as a 'diamond' shape]. As indicated, the program flow gets to the point of decision, which is essentially a relational logic statement. Depending on the value of the relational logic condition, if it is true, then the program flow branches to execute a section of code and then continues with the rest of the program. If the condition is *not* satisfied (condition evaluates to False), then the program flow simply continues, bypassing the branch.

0.4 Simple if decision structure

In python, the simple if structure, which implements the decision structure illustrated above, uses the keyword **if**, then a conditional statement followed by a colon, and then the code to execute if the condition is true, INDENTED on the next line (or lines if there are multiple statements). The customary indent is 4 spaces, however one space will work. Indentation will occur automatically upon pressing Enter after the colon.

0.5 Exercise 2 - Exploring the simple if structure

For example, consider the snippet of code below. [Note: the parentheses around the conditional expression are actually NOT required! However, using parentheses probably will result in more readable code, and it is good practice, especially if you want to learn the C programming language (where they are required for an if statement.] Type the code into a python cell to explore how the simple if structure works:

```
a = 2
b = 7
if (a >= b ):
```

```
    print("a is >= b")
print("a + b =", a + b)
```

Try different values for a and b and see what happens. Try removing the indent of the first print statement. What happens? ***

0.6 Logical Operators and Compound Conditional Statements

Logical operators allow construction of more complex boolean expressions for decision logic. The three logical operators are: 1. and 2. or 3. not

For example, in the snippet of code below, note the conditional expression using the 'and' logical operator. It will be satisfied only when the value of c is greater than a AND less than b. Copy-paste the following lines of code into a python code cell:

```
a = 2
b = 7
c = int( input("Enter an integer for c: ") )
if ( c > a and c < b ):
    print("c is in between a and b")
print("a + b =", a + b)
```

Try it out in a python cell. The third line asks for user input, and once it gets something from the keyboard, it will attempt to convert it to an integer and store it in the variable c. What happens if you enter something other than an integer? ***

0.7 If-else decision structure

Sometimes you might want to execute some lines of code if a condition is satsfied, ***else*** execute some other lines of code if the condition is not satisfied. An illustration of this decision structure is shown below.

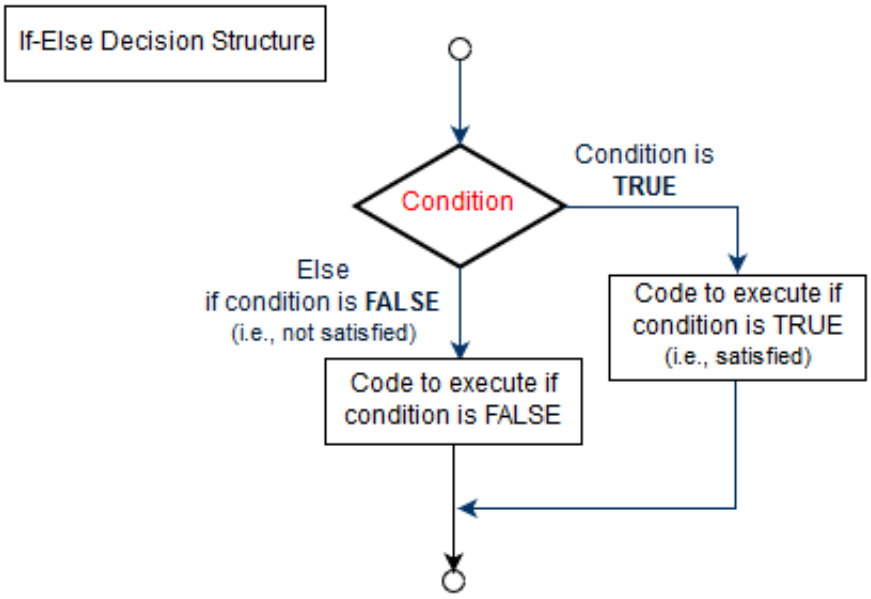
The following code snippet uses an if-else decision structure to print one message if the condition is satisfied (True) or another message if the condition is not satisfied (False) depending on user input. It also uses a compound conditional expression to check for different spellings of 'chocolate':

```
favorite = input("Enter your favorite ice cream flavor: ")
if (favorite == "chocolate" or favorite == "Chocolate"):
    print("Good choice.")
else:
    print("Sorry, dude, chocolate rules!")
print("Let's go get ice cream!")
```

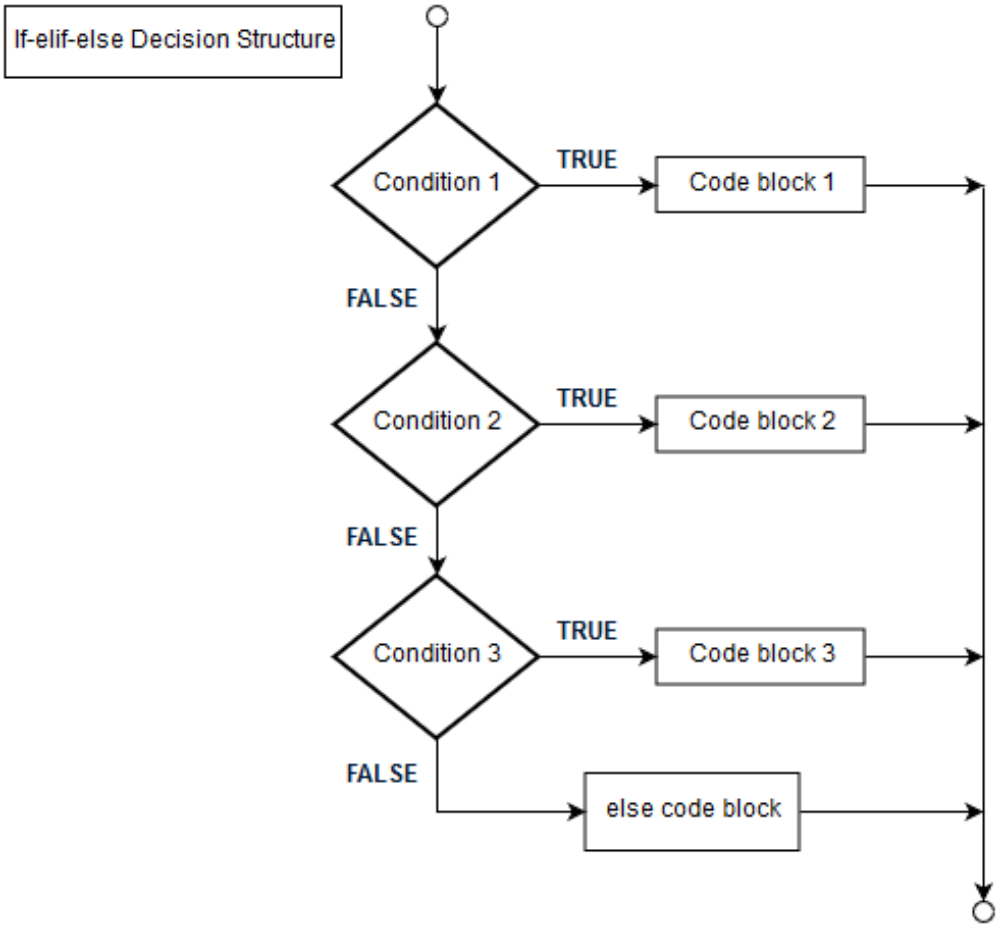
You are not limited to a single line of code to be executed if the condition is true. Additional indented lines can be included, and these could even contain additional if/if-else code blocks as needed. If/if-else structures that occur within if statements are called 'nested' ifs.

A variation on the if-else decision structure that is useful when you want to decide which of multiple possible actions to take is the if-elif-else structure. The operation of this construct is depicted in the figure below:

As an example, consider the following code snippet of a function to determine the letter grade given a percentage:



If-else decision structure



If-elif-else decision structure

```
def gradePrint(percentage):
    if(percentage > 100.0):
        print("Super! You got %.1f%% --> A+" % percentage )
    elif (percentage >= 93.0):
        print("Excellent! You got %.1f%% --> A" % percentage )
    elif (percentage >= 90.0):
        print("Very good! You got %.1f%% --> A-" % percentage )
    # to be continued!
    else:
        print("Uh oh! You got %.1f%% --> F" % percentage )
```

0.8 Exercise 3 - Grade print using if-elif-else

Copy the code snippet above, and in a python code cell, verify that it works. Don't worry about the details in the print statements. These will be explained in future lessons. Suffice it to say at this point that the %.1f gives instruction to print the contents of the variable *percentage* using one decimal place, and the %% prints a % symbol right after the number.

Your task in this Exercise is to flesh out the gradePrint function, so that it will work for the full range of grades given here: A+ Over 100%; A 100 – 93%; A- 92 – 90%; B+ 89 – 88%; B 86 – 83%; B- 82 – 80%; C+ 79 – 78%; C 76 – 72%; C- 71 – 69%; D+ 68 – 66%; D 65 – 62%; D- 61 – 59%; F <58%

Follow the pattern given in the code snippet. Verify that your function works. ***

0.9 Exercise 4 - Using a decision structure to add functionality to the script from HW problem 5.1

Recall Exercise 5.1 from the textbook: > The time module provides a function, also named time, that returns the current Greenwich Mean Time in "the epoch", which is an arbitrary time used as a reference point. On UNIX systems, the epoch is 1 January 1970.

```
>>> import time
>>> time.time()
1437746094.5735958
```

> Write a script that reads the current time and converts it to a time of day in hours, minutes, and seconds, plus the number of days since the epoch.

Use the code that you wrote for Exercise 5.1 and add a conditional structure that will print an additional line that depends on the time of day.

1. If before noon, the line should read: "Good morning!"
2. If after noon, but before 6 pm, the line should read: "Good afternoon!
3. If after 6 pm and before midnight, the line should read: Good evening!

---

1 What to Turn In

Generate a PDF from your notebook and submit to Canvas

---

2 References

+ Python Concepts/If Statement https://en.wikiversity.org/wiki/Python_Concepts/If_Statement

- Formatted output in Python https://www.python-course.eu/python3_formatted_output.php
- Time access and conversions https://docs.python.org/3/library/time.html#
- Basic date and time types https://docs.python.org/3/library/datetime.html
- Python DateTime, TimeDelta, Strftime(Format) with Examples https://www.guru99.com/date-time-and-datetime-classes-in-python.html
- Epoch converter https://www.epochconverter.com/
- Jupyter Notebook Users Manual (from Bryn Mawr College) https://jupyter.brynmawr.edu/services/public/dblank/Jupyter%20Notebook%20Users%20Manual.ipynl
- Adam Pritchard's Markdown Cheatsheet https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet