Thus far you have been looking at simple, sequential problems and programs. In order to write more complex and useful programs, it is essential we give our programs the ability to make decisions according to various conditions and change it's behavior accordingly. For this, we need to use conditional execution.

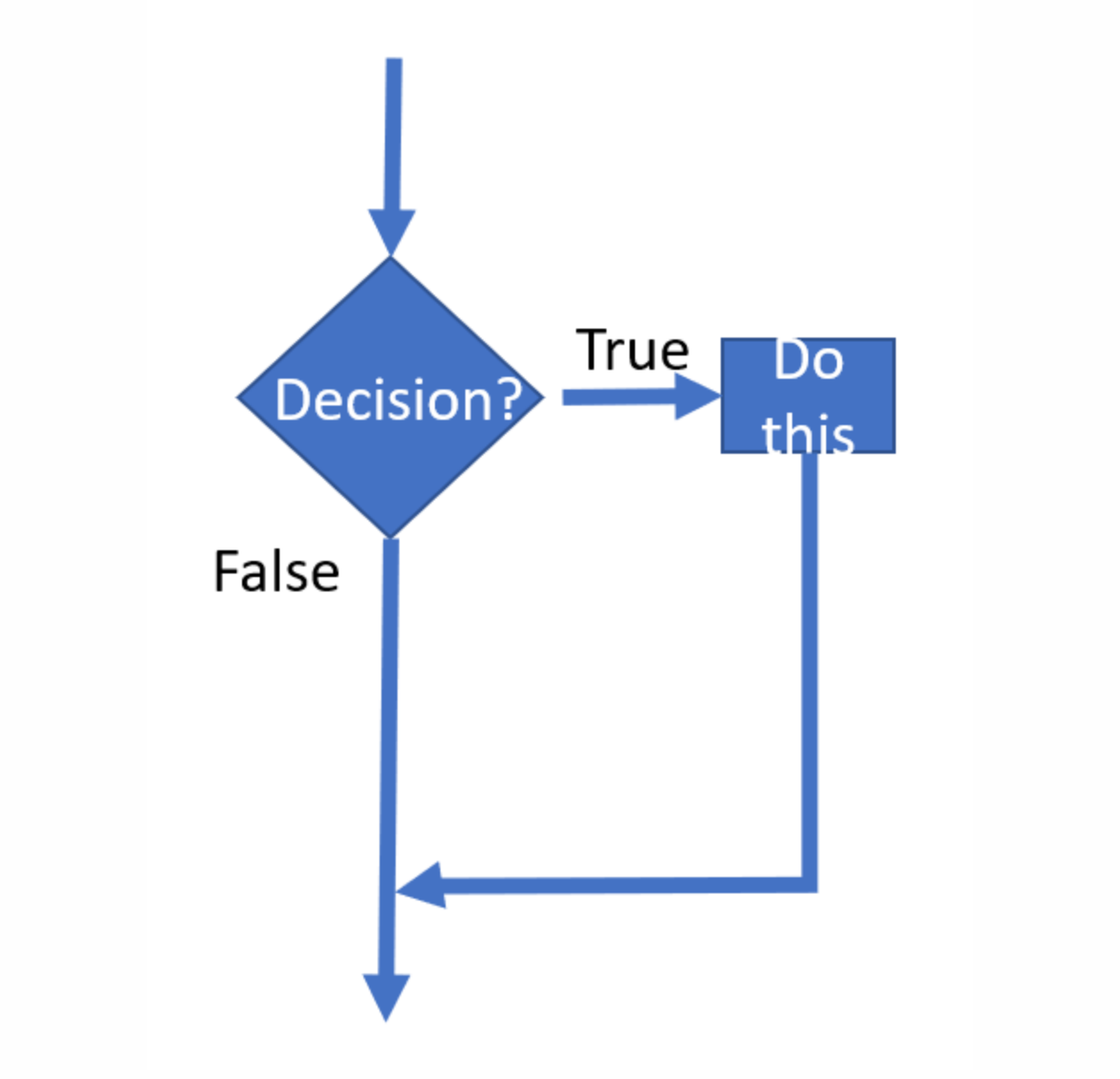
The simplest form of conditional execution in Python is the "if" statement.

```
if time >  12:
    print("Good Afternoon!")
```

The python if statement is very intuitive and easy to understand. "if" is a keyword in python followed by a logical expression that returns either True or False. You have already looked at expressions and variables earlier in the course. The above example uses a variable called "time" and checks to see if the time has passed 12 using the "greater than" comparison operator. An if statement always ends with a colon as shown in the example.

The instructions/code to be executed if the if condition returned True are indented under the if statement. Python uses these indentations to mark the scope of the if statement. As opposed to many other programming languages, Python's leading whitespaces carry syntactical meaning. You will see more uses of indentation as you go along the course.

The visualization of a simple if statement is shown below.

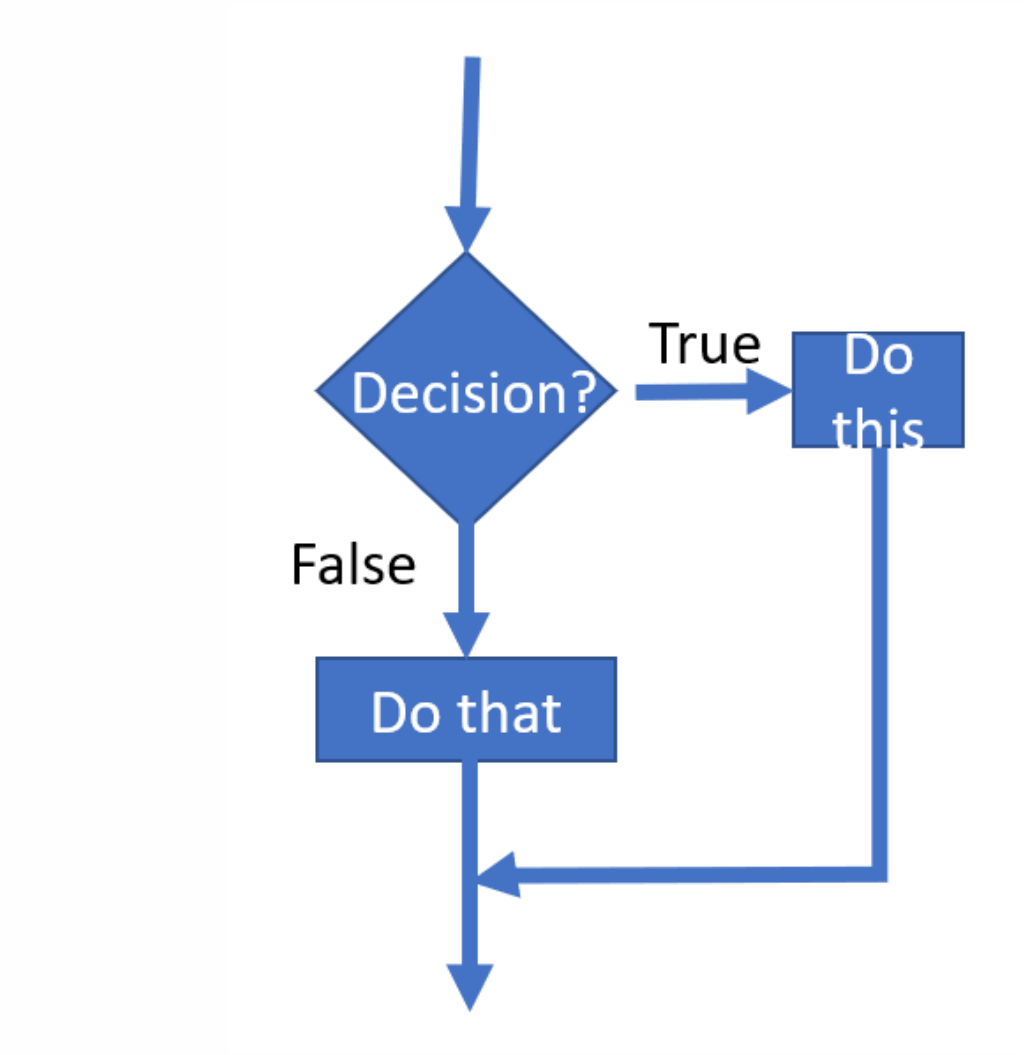


Notice how in the diagram above, the program only executes some code if the condition returns True. If it returns False, the program simply does nothing. But it is often useful to write code that has an alternative choice. For this, we can use an if-else combination.

```
if time < 12:

    print("Good Morning!")
else:
    print("Good Afternoon!")
```

The diagram below shows an if - else block. Notice how both outcomes of the condition are utilized.



A multi way condition can be implemented using the Python's "elif" keyword. Elif can be understood as an "else-if". Using elifs, the program is able to choose one of many paths. The syntax is as follows.

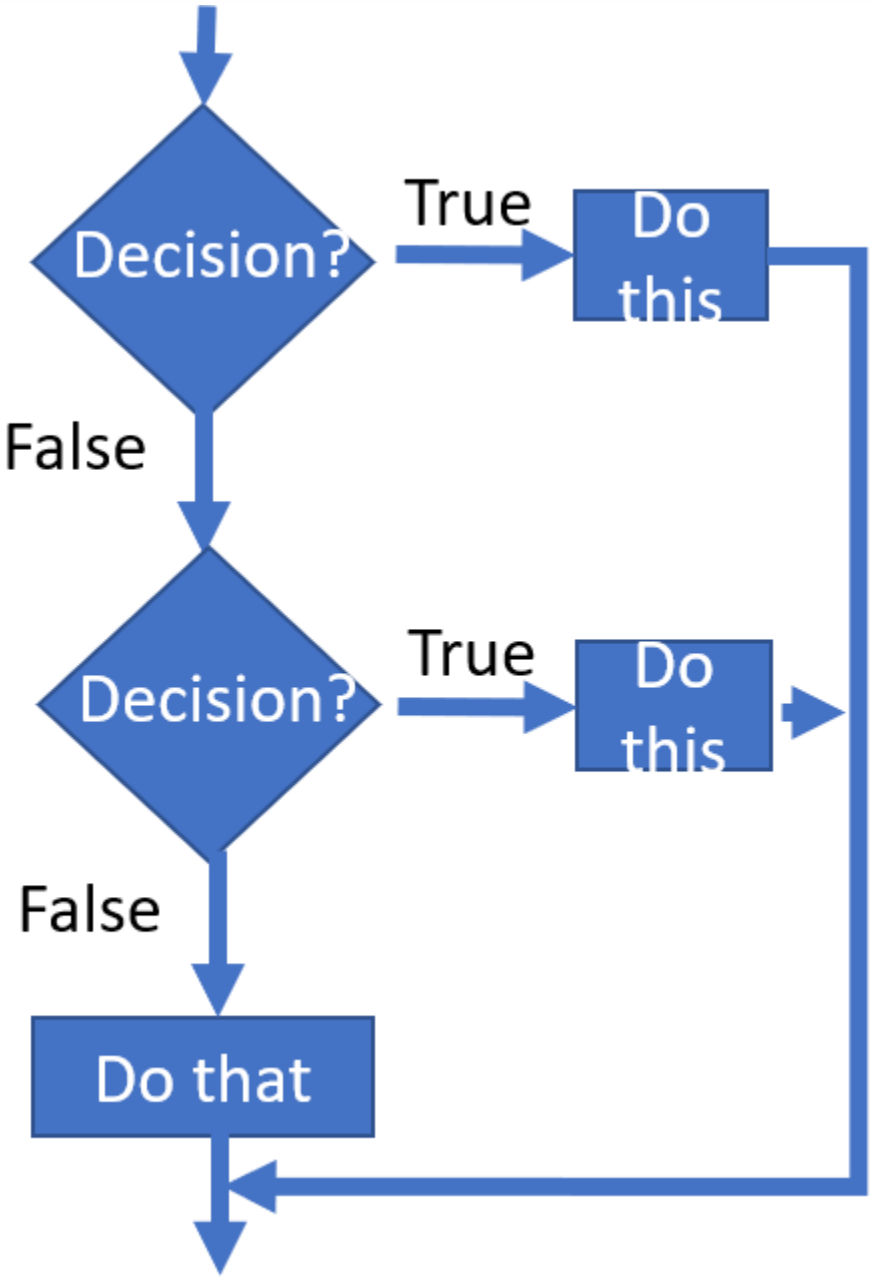
```
if marks > 75:
    print(“A”)

elif 60 <= marks < 75:
    print(“B")

elif 40 <= marks < 60:
    print("C")

else:
    print(“F")
```

Unlike with the use of several if statements one after the other, with the use of elifs, the program checks the following elif conditions only if the first if statement returns False. If any of the subsequent elif conditions return True, we execute that particular branch and skip over the elif conditions that follow. The visualization of an if-elif-else block can be shown as follows. Note how if the first if condition or the subsequent elif conditions return True we do not check the conditions that follow.



In order to better understand how elifs work and the difference between using elifs and several ifs in sequence, run the code segments provided below.

It is also possible to have nested if conditions as shown below. Try changing the value of x to observe the change in the results.

The right conditional construct for your program depends on the solution you implement. Visualizing the solution to understand it before you start coding can help you choose the most suitable conditional execution.

As you go further in this course you will come across Try- except blocks that also serve a similar purpose. Try except blocks are used to handle exceptions that arise in your code. They can also be viewed as conditional execution since they specify paths for the program to take **if** it comes across a problem. More on that later!