CS-500 Module Two Journal

Isaac Raymond

CS-500-11462-M01  
  
Professor Ronak Nouri

Conditionals in Python Data Processing

Conditional use in python is common when processing data. Data from sources often has mistakes or different formatting. Conditionals can help us read the data and change a value slightly based on how that value was recorded. This benefits a data professional as they can make their data uniform and easier to use. Complicated and messy data can often lead to complicated, nested conditionals. This can lead to hard-to-troubleshoot bugs. Following good coding practices can let a programmer process the data that they need to process, while still being able to adjust their conditionals when bugs appear.

A potential real-world scenario of the use of conditionals would be in the processing of a large set of images. If we want to process images of many different types (JPEG, RGB, etc.), we might want to start the data processing by conditioning based on that image type. Each image is read through the program, then, based on its file type, the image processing is done differently. Branching the process based on an image type does not require a complicated conditional.

The advantage of conditionals in this context is clear if the data set is large. A human would have to process each image by hand, perhaps dragging each image file into another appropriate file with other like-type images. This would take too much time! Writing a conditional to process each file programmatically will save the programmer a lot of headache.

Real-world conditionals that get more complicated could be when we're processing strings of data. We might need to process strings of data entered by a user. If the user entered a number, we could convert that number from a string to a float or an integer. If the user entered numbers and words together, we might need to break the input into several values of an array, storing the numbers and words separately (depending on a use case). Each "if" listed above would need to be a separate branch in the conditional.

As an example: If a data set includes a string of user input about their recent doctor visit, we might break up the string by spaces and then analyze each word in that string. **if** the "word" is only a number, we convert that number to a float or an integer and store it. **else if** (elif in python) the word contains both letters and numbers, we send the word to a function for further processing to determine if a typo has occurred. **else**, we store the word as a regular string in whatever format is required (an array, a data dictionary, etc.).

The limitation to this approach is that each conditional would have to be written "by hand". A more dynamic approach would be best. There may be machine learning techniques that would allow the data specialist to ask for the data to be processed, then any necessary processing would be completed without writing specific individual conditionals.

Such data processing would need to have conditions written in an easy-to-read manner or else the code would be difficult to adjust. Our program might need to expand in the future. If the conditionals aren't formatted correctly, or if we haven't written the conditionals in a structure such that each condition is easy to understand, then adjusting the code could be more time consuming than necessary. Sometimes more lines of code are better than less lines of code if it increases the readability of that code. The extra lines of code will not have any impact on performance as their very small added memory usage is negligible.

The more complicated the conditional, the more testing that would be required. Data processing with python is usually done because there are thousands of lines of data that need to be processed. It would be easy for a bug in the conditional to slip in unnoticed, introducing a few incorrect values into the data set. A data scientist can't walk through every line of data that's been cleaned. The data scientist could examine a few specific inputs to make sure the conditionals are cleaning the data in the manner they expect.  
 Conditionals are required in data processing. Data needs to be converted to the same type before it can be run through statistical models. Their use needs to be carefully observed, however. A poorly written conditional might have a nested conditional that is only changing a few values to incorrect numbers or strings. A well-written, easy-to-read conditional, however, will make it easy for finding such bugs and making the necessary adjustments.