1_procedural_vs_functional_in_python

March 27, 2022

1 Instructions

Run the code cells in this notebook to see the problem with procedural programming.

```
In [1]: log_of_songs = [
                "Despacito",
                "Nice for what",
                "No tears left to cry",
                "Despacito",
                "Havana",
                "In my feelings",
                "Nice for what",
                "Despacito",
                "All the stars"
        ]
```

```
In [2]: play_count = 0
```

```
In [3]: def count_plays(song_title):
            global play_count
            for song in log_of_songs:
                if song == song_title:
                    play_count = play_count + 1
            return play_count
```

```
In [6]: count_plays("Despacito")
```

```
Out[6]: 9
```

```
In [7]: count_plays("Despacito")
```

```
Out[7]: 12
```

2 How to Solve the Issue

How might you solve this issue? You could get rid of the global variable and instead use play_count as an input to the function:

```
def count_plays(song_title, play_count):
    for song in log_of_songs:
        if song == song_title:
            play_count = play_count + 1
    return play_count
```

How would this work with parallel programming? Spark splits up data onto multiple machines. If your songs list were split onto two machines, Machine A would first need to finish counting, and then return its own result to Machine B. And then Machine B could use the output from Machine A and add to the count.

However, that isn't parallel computing. Machine B would have to wait until Machine A finishes. You'll see in the next parts of the lesson how Spark solves this issue with a functional programming paradigm.

In Spark, if your data is split onto two different machines, machine A will run a function to count how many times 'Despacito' appears on machine A. Machine B will simultaneously run a function to count how many times 'Despacito' appears on machine B. After they finish counting individually, they'll combine their results together. You'll see how this works in the next parts of the lesson.