A comprehensive comparison between Python Libraries *Shiny and Plotly*

By Ian McKechnie

Monday October 31st, 2022

**Introduction**

Shiny is focused on creating graphs and images for web applications. It can be installed through pip. It has a lot of typical front end language features like headings and text features for displaying static text on websites. Done with the following methods:

Text

Description automatically generated

When you run Shiny it runs on <http://localhost:8000> on your browser. You can download a VSCode plug in and see a sample browser as seen in figure 1 below.

Shiny is based on reactive programming. Reactive programming is where it only re-executes code based on if an input is modified.

**Reactive programming in Shiny**

There are three methods that you can choose so when an input changes, your Shiny application can react to the input. They are *reactive.Value, reactive.Calc, reactive.Effect*.

reactive.Calc

When you want a method call to be calc reactive, you add the decorator @reactive.Calc above it (ex. figure 2)

Logo

Description automatically generated with low confidence

Figure 2

What reactive.calc does is cache its most recent value and returns that value until Calc is invalidated. This means the amount of work the method needs to do is reduced since it doesn’t not need to recalculate what’s in the method every time the page is updated. This results in a faster and snappier page. It changes, when the page is invalidated, this happens when their inputs change. When its inputs change then it reruns the method and shows the result.

You can think of this as the method only returns cached results until the input changes, then it reruns and caches that new value that it’s displaying to the user.

reactive.Effect

This decorator is applied the same as reactive.Calc and it runs when the inputs change. This differs from reactive.Calc in two mains ways

1. Reactive.calc is used for its return value; its meant to be called from another reactive.calc or reactive.effect which uses the return value from the first. A reactive.effect does not have a return value and cannot be called like a function. Once created, it continues to exist, and it automatically executes after is has been invalidated.
2. AAs the name implies reactive.effect is used for its side effect. A side effect is when a function modifies state other than its return value. These side effects may include: writing to the file, printing to console, or modifying a global variable.

reactive.object

They are Shiny objects that when their data changes, it causes reactive methods to change.

@output

There is also the @output decorator. This allows methods not decorated with reactive.effect to be reactive. An example is

@output

@render.text

**def** txt():

**return** f"The value of x is {input.x()}"

This returns a string that is based on the value of x which is an input from the user. It’s wrapped with a @render.text and @output decorators. Each time the reactive inputs change, it re-executes and puts the new string in the message queue.

**User interaction**

Shiny for Python has its focus on allowing the data to be interactive. So, the page being built can have lots of interactive filters that the user can toggle, slide, and change to alter what and how data is being shown to the user. An example is shown below in figure 1. The code on the left implements an interactive slider. That slider has numbers which when selected, display it square bellow in real time.

Graphical user interface, text, application

Description automatically generated

Figure

The list of interactive features are as follows

Graphical user interface, text, application

Description automatically generatedGraphical user interface, application

Description automatically generated

These inputs are no very complex to create. The entire set of inputs above is only 26 lines line with the labels above each one. The inputs can be made reactive so when a value is changed by the user on the GUI it will update other elements that use the data associated with it.

**How to get data**

There are a few ways to get data into Shiny. The first is through csv files. You can read in the csv with pandas (a data analysis library in python) and then display the csv by filtering the pandas data frame with inputs from your user or hard coded filters before displaying it.

There’s also the option to upload and download files.

You can also get data from APIs.