1) For plotting the spectrum of your M8.5 star:

After making the initial plot with:

```
ax1 = plt.errorbar() ,  etc.
```

use:

```
plt.xlim()
plt.ylim()
plt.xlabel()
plt.ylabel()
```

to control the axis range and add axis labels. (You'll need to look up exactly what parameters to use for each of these

2) To add in patches after you've plotted your spectrum:

```
import matplotlib.patches as patches
ax1 = plt.gca()

ax1.add_patch(
)
```

again, you'll need to look up what parameters to use with ax.add_patch. Don't forget to set alpha=0.5.

3) After plotting the spectrum of your star and adding in patches for the filters, the next step is defining a function that calculates the mean flux transmitted through each filter. This will have the basic form:

```
def flux_filter(wlen, flux, wlen0, dwlen):


   # returns the mean flux transmitted through a boxcar filter
   # with central wavelength wlen0 and filter width dwlen
   # and flux input is (wlen, flux)



   return(bandflux)
```

where you entire the parameters wlen – wavelength of your spectrum, flux – flux of your spectrum, wlen0 – central wavelength of your filter and dwlen – width of your filter, and the function returns the mean flux transmitted through the filter. It's up

to you to fill in the rest of the function! Once you've defined your function in one cell, you can run it in the next:

```
bandflux = flux_filter(<<include appropriate parameters here – hint, use the wavelengths and fluxes you've already read in to make the plot>>)
print bandflux
```

One way to select the right wavelengths is to create a Boolean mask – an array that is set to "True" at wavelengths within the filter and "False" at wavelengths inside the filter. E.g. if you create a variable:

```
mask = (wlen > (wlen0 - dwlen/2.))
```

that mask will have a value of "False" for wavelengths less than (wlen0 – dwlen/ 2.) and "True" for wavelengths greater than (wlen0 – dwlen / 2.)

If you multiply together two logical statements like this, you can get a Boolean mask for both statements together, as "False"*"False" = "False", "True"*"False" = "False", and "True"*"True" = "True" in this case. Try adding a:

```
print mask
```

statement into your function to see what that mask looks like.

Once you have set up the right mask, you can choose the wavelength values inside your filter by indexing with the mask:

```
good_wlen = wlen[mask]
```

or, similarly, the fluxes at those wavelengths:

```
good_fluxes = flux[mask]
```

to find the mean flux in the spectrum at these wavelengths, you can then use np.mean:

```
mean_value = np.mean(array)
```