This code is a python exercise with astronomical data using the astropy python module

Step 1. Importing the modules

This may seem obvious, but it is an important step in every python code, we need to import all the modules that we're going to be working with it is done with the **import** keyword as follows:
import ***module_name*** as ***call_sign*** (where ***call_sign*** is how you'll call commands from the module later)

```
import numpy as np

import matplotlib.pyplot as plt

from astropy.io import fits

from astropy.table import Table
```

Step 2. Loading the .fits file

A file in .fits format, contains 2 parts:
The so called "Header", which contains information about the creation of the file. For example what the object depicted is, what are the coordinates of the central pixel, which filters was it taken in and more.
And the "Data", which contains the bulk of the file.\

A standard .fits file looks similar to this:
insert picture of .fits table, too lazy to do this now

The astropy module has the function to operate with said .fits files. We can load in only certain columns of a table formatted .fits file using "Table.read('name_of_file')"
and then loading in different columns (called by their names in the header) into different variables, creating arrays of data.

```
t = Table.read('./hst_results_nd.fits')
ra = t['RA']
de = t['DEC']
Av = t['Av_p50']
r = t['distance_p50']
temp = t['logT_p50']
age = t['logA_p50']
F475W = t['F475W_VEGA']
F814W = t['F814W_VEGA']
```

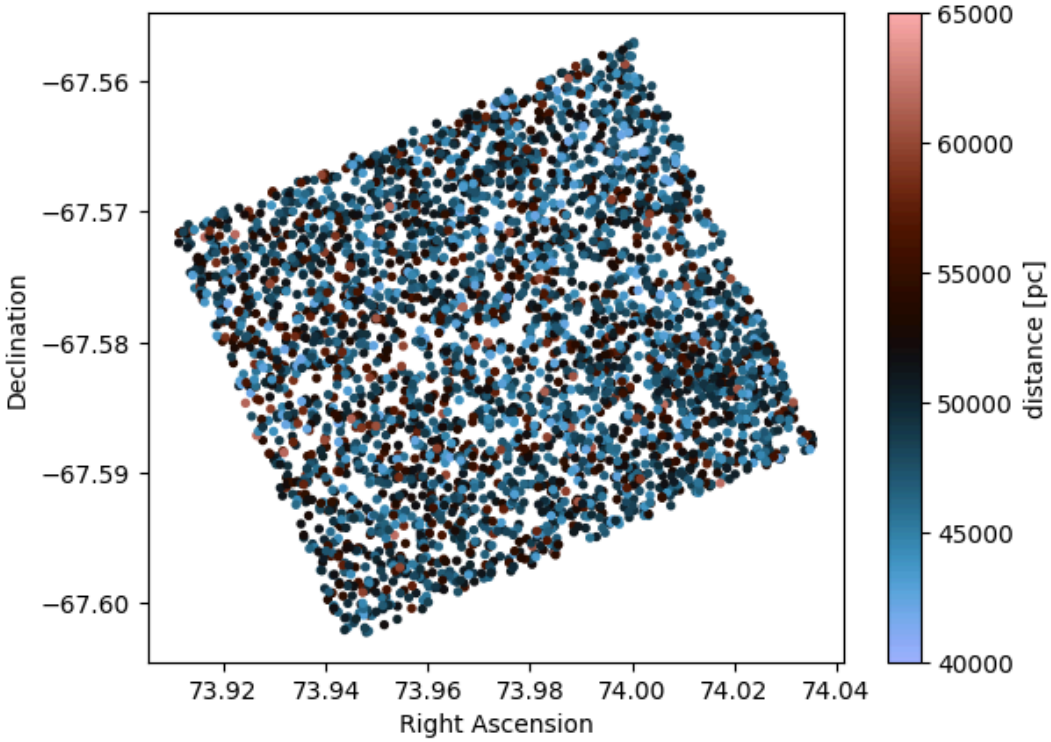
Step 3. Creating Graphs and Figures

After the data has been loaded in, we can use it to create scatter plots, histograms, or really any other type of graphs, complete with color coded characteristics or calculated means.
Here's some examples:\

Using the **matplotlib** module we'll create the following graphs:
I. A graph of the position of different stars on the equatorial coordinate system in the data set, colorcoded with the distance to each data point.
II. A histogram of the distance to each data point, with a mean distance of the data set.\

I. We create a scatter plot using the right ascenscion (ra) and declination (de) that we loaded in the previous step, adding "c = r" makes a colorbar using
the distance (r) that we loaded up in the previous step and colormap (cmap) that we chose (in this care 'berlin', there's a list of different cmaps that can be checked with **list(colormaps)**)

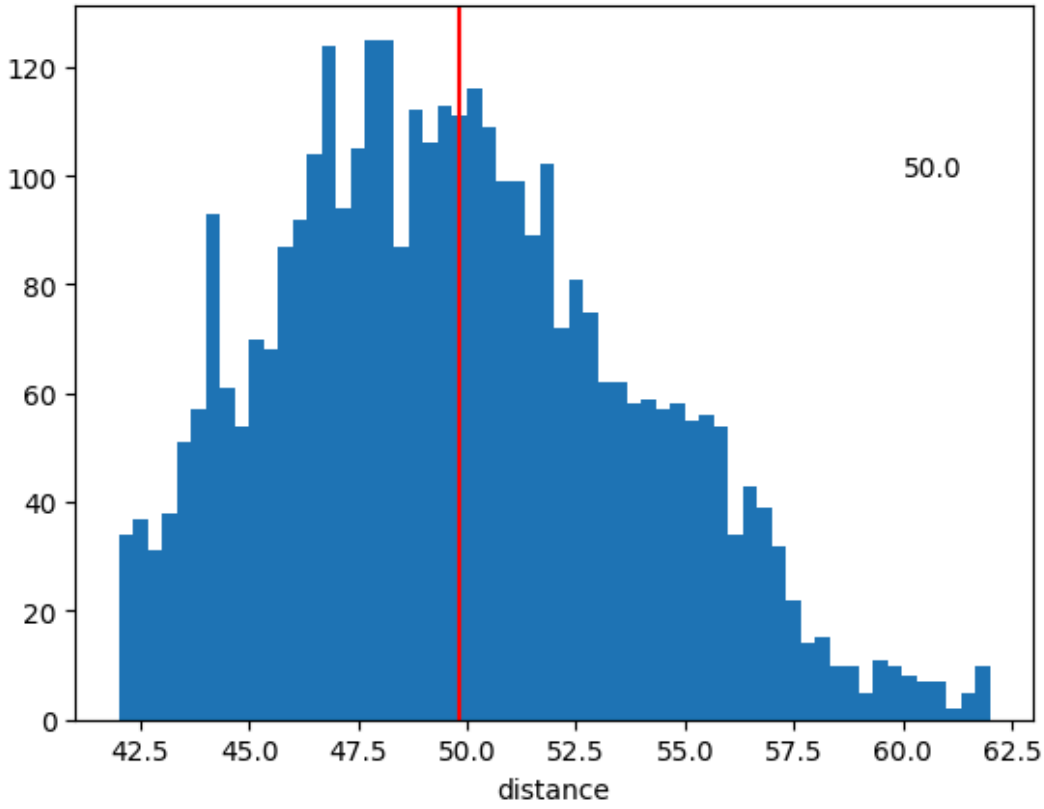
```
plt.figure()
cb = plt.scatter( ra, de, c = r, s = 8, cmap = 'berlin', marker = 'o', vmin = 40000, vmax = 65000)
plt.colorbar(cb, label = 'distance [pc]')
plt.ylabel('Declination')
plt.xlabel('Right Ascension')
```



II. We create a histogram by using again the distance (r), loaded up in the previous step, and the plt.hist command, where we specify the size (r/1000) and amount (60) of bins. we also add a red line on the mean of the distance, which we calculate using the

np.mean command from the numpy module, using the plt.axvline command.
finally we save the file as "distancehist.png" using the plt.savefig command

```
plt.figure()
plt.hist(r/1000, bins = 60)
plt.xlabel('distance')
d_mean = np.mean(r)
plt.axvline(d_mean/1000, c='r')
plt.text(60, 100, '%s' % np.round(d_mean/1000))
plt.savefig("distancehist.png")
```



Step 4. Filtering data.

Let's say we want to create a CMD of the stars in the dataset, but we also want to filter out certain stars. Astropy is incredibly useful in this regard
as you can apply a filter onto a loaded column (or columns) of a table, and essentially create a new table containing all the data for the stars that passed the filter.
This is done using the following command:

```
hot_stars = t[temp > 4]
```

Which checks all the data in the "temp" column, and deletes any row that has temp less than 4.\

We can also add some other filters:

```
young_stars = t[age < 8.5]
old_stars = t[age > 10] #here the age of the stars is given in base log_10
```

We can then plot all the stars, and only the filter results onto a CMD:

```
plt.plot(F475W-F814W, F475W, '.' ,ls = '', label = 'all stars', color = '#bfbfbf')
plt.plot(young_stars['F475W_VEGA'] - young_stars['F814W_VEGA'], young_stars['F475W_VEGA'], 'o' ,ls = '', label = 'young stars', color = '#bf00bf')
plt.plot(old_stars['F475W_VEGA'] - old_stars['F814W_VEGA'], old_stars['F475W_VEGA'], 'o' ,ls = '', label = 'old stars', color = '#00bfbf')
plt.plot(hot_stars['F475W_VEGA'] - hot_stars['F814W_VEGA'], hot_stars['F475W_VEGA'], 'o' ,ls = '', label = 'hot stars', color = '#6faf00')
plt.legend()
```