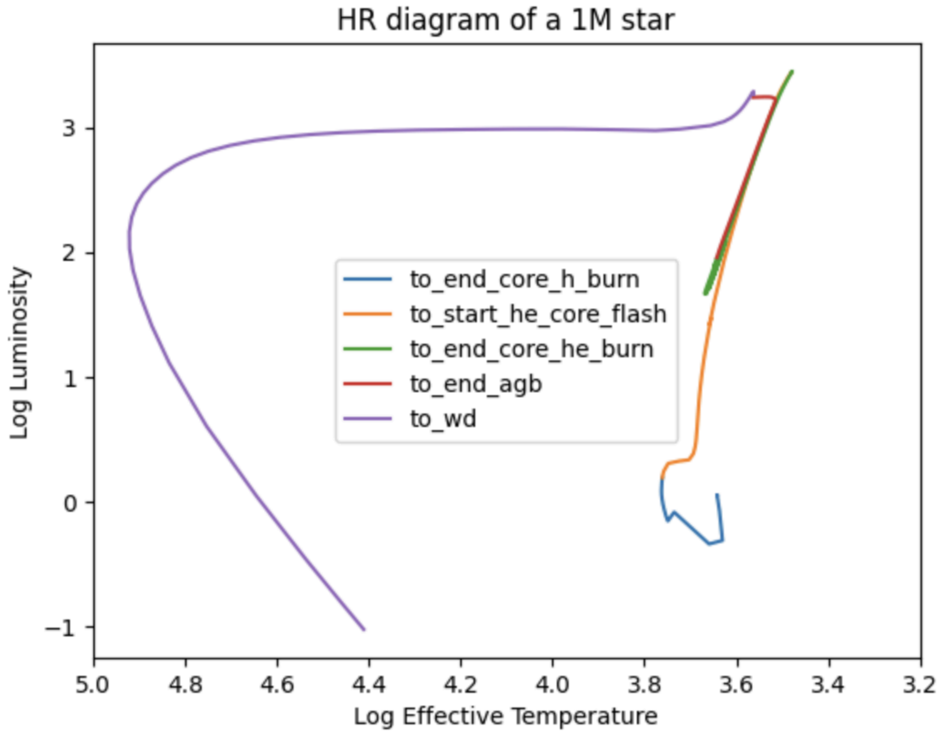
Exercise 1(Solved,Warm up): Plotting the Hertzsprung-Russell diagram of a star

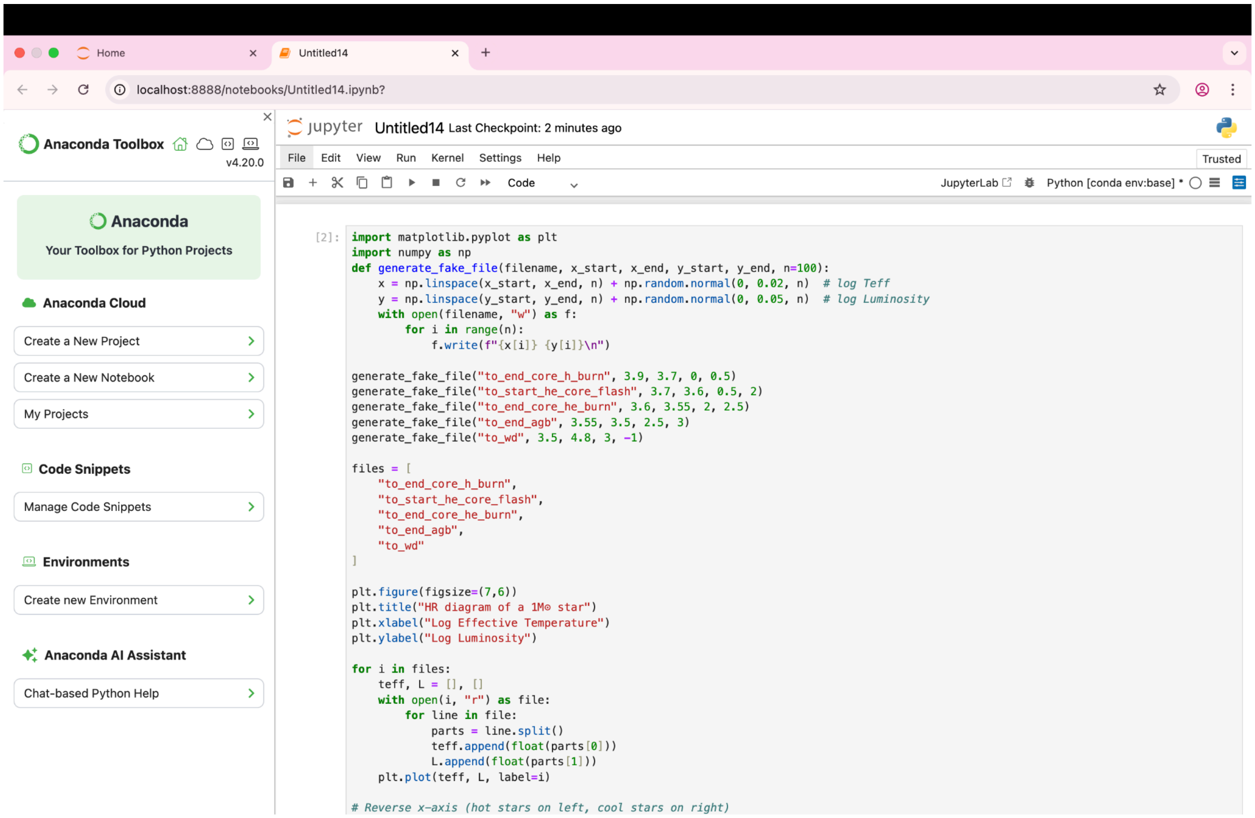
Note:This is an example of how to plot a graph, and this can be used as a reference on how to plot the actual exercises.

The Hertzsprung-Russell diagram is used to visualize a star's evolution, from it's beginning when it burns Hydrogen, to it's end, where it becomes anything from a dwarf star to a black hole depending on it's mass. In the plot below, we use the MESA software (Modules for Experiments in Stellar Astrophysics) to obtain data for a $1M_{\odot}$ star. If you want to know more about this software, you can check here. From this data, we plot the log of effective temperature and log of luminosity at different stages of the star to see what is happening at different stages.

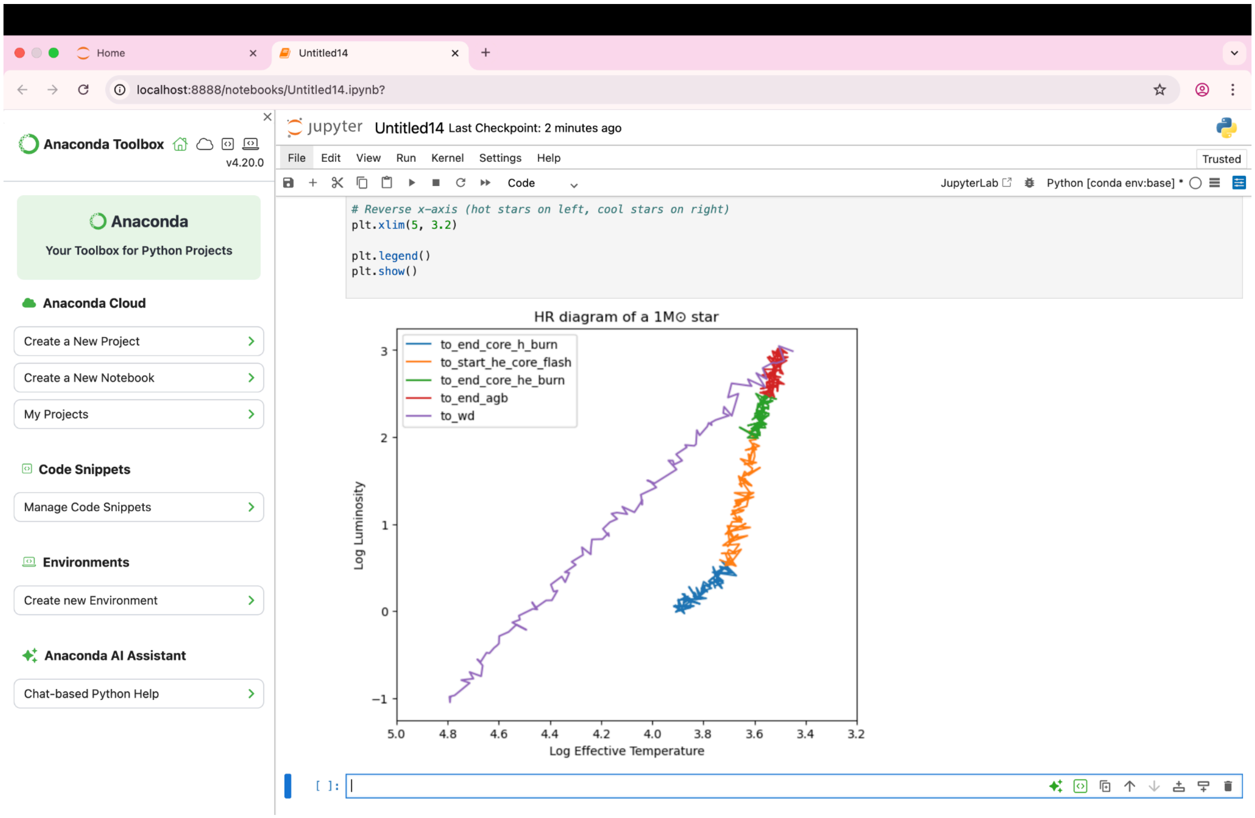
In [ ]:

```
#Don't change anything in this cell
files=["to_end_core_h_burn","to_start_he_core_flash","to_end_core_he_burn","to_end_agb","to_wd"]
plt.title("HR diagram of a 1M star")
plt.xlabel("Log Effective Temperature")
plt.ylabel("Log Luminosity")
for i in files:
  teff,l=[],[]
  with open(i, 'r') as file:
    for line in file:
      parts = line.split()
      teff.append(float(parts[0]))
      l.append(float(parts[1]))
  plt.plot(teff,l,label=i)
  plt.xlim(5,3.2)
  plt.legend()
plt.show()
```





```python
import matplotlib.pyplot as plt
import numpy as np
def generate_fake_file(filename, x_start, x_end, y_start, y_end, n=100):
    x = np.linspace(x_start, x_end, n) + np.random.normal(0, 0.02, n)  # log Teff
    y = np.linspace(y_start, y_end, n) + np.random.normal(0, 0.05, n)  # log Luminosity
    with open(filename, "w") as f:
        for i in range(n):
            f.write(f"{x[i]} {y[i]}\n")

generate_fake_file("to_end_core_h_burn", 3.9, 3.7, 0, 0.5)
generate_fake_file("to_start_he_core_flash", 3.7, 3.6, 0.5, 2)
generate_fake_file("to_end_core_he_burn", 3.6, 3.55, 2, 2.5)
generate_fake_file("to_end_agb", 3.55, 3.5, 2.5, 3)
generate_fake_file("to_wd", 3.5, 4.8, 3, -1)

files = [
    "to_end_core_h_burn",
    "to_start_he_core_flash",
    "to_end_core_he_burn",
    "to_end_agb",
    "to_wd"
]

plt.figure(figsize=(7,6))
plt.title("HR diagram of a 1M⊙ star")
plt.xlabel("Log Effective Temperature")
plt.ylabel("Log Luminosity")

for i in files:
    teff, L = [], []
    with open(i, "r") as file:
        for line in file:
            parts = line.split()
            teff.append(float(parts[0]))
            L.append(float(parts[1]))
    plt.plot(teff, L, label=i)

# Reverse x-axis (hot stars on left, cool stars on right)
```



```python
# Reverse x-axis (hot stars on left, cool stars on right)
plt.xlim(5, 3.2)

plt.legend()
plt.show()
```

Exercise 2: Discrete Fourier Transform of a Noisy Signal

Not all signals come out to be clear and analyzable. Some signals will have too much noise to be even comprehendible. In such cases we use different techniques to reduce the noise in data. One such technique is the Discrete Fourier Transform (DFT). First let us have a look at the data.

In [ ]:

```
#Don't change anything in this cell
filename='ndshm.dat'
#filepath=....(If there)
#filename=filepath+filename
time,amplitude=[],[]
with open(filename, 'r') as file:
    for line in file:
      parts = line.split()
      time.append(float(parts[0]))
      amplitude.append(float(parts[1]))
lent=len(time)
```

In [ ]:

```
# Exercise 2a: Plot Time vs Amplitude, make sure to label the axis
```

We can see, we cannot comprehend anything from this data, as there is too much noise. So we filter out the noise. How you may ask? We use a technique called the Discrete Fourier transform, which will show us the same data in the frequency domain. The difference that will make will be clear when you look at the plot

Now we perform the Discrete Fourier Transform on the data provided, just run the below code to get the data. If you want to know more about this, read this

In [ ]:

```
#Don't change anything in this cell
freq=time.copy()
for j in range(lent):
  freq[j]=(j-lent/2)/(lent*0.01)
famp=[]
fsq=[]
for j in range(lent):
  ft=0+0j
  for k in range(lent):
    ft=ft+amplitude[k]*cm.exp(2*np.pi*1j*(j-lent/2)*k/lent)
  famp.append(ft)
  fsq.append(np.abs(ft)**2)
```

Now the transformed data is in terms of Frequency vs Amplitude, now plot this and see the result for yourself

In [ ]:

```
#Exercise 2b:Plot the frequency (freq variable) and square amplitude (fsq)
```

As you can see, the centre two peaks were the required frequency and the noise is the rest, now we remove the noise and transform it back.

In [ ]:

```
#Don't change anything in this cell
for j in range(lent):
  if freq[j]<=-10 or freq[j] >=10:
    famp[j]=0
    fsq[j]=0
```

In [ ]:

```
#Exercise 2c:Plot the frequency (freq variable) and square amplitude (fsq)
```

In [ ]:

```
#Don't change anything in this cell
modamp = []
for k in range(lent):
    ft = 0
    for j in range(lent):
        ft=ft+famp[j]*cm.exp(-2j * np.pi * (j-lent/2) * k / lent)
    modamp.append(ft / lent)
```

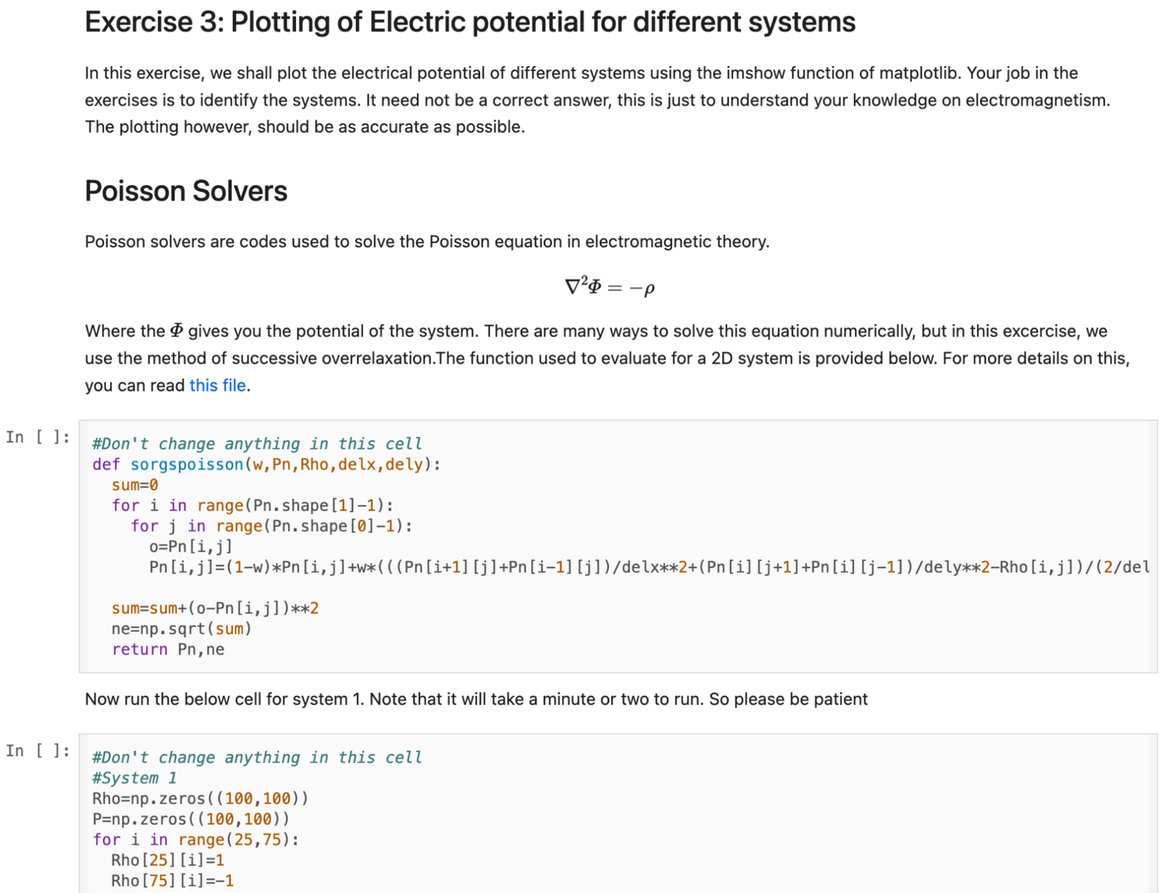
In [ ]:

```
#Exercise 2d:Plot the time (freq variable) and modified amplitude (modamp)
```

2A)

1. to_end_core_h_burn
2. to_start_he_core_flash
3. to_end_core_he_burn
4. to_end_agb
5. to_wd
6. ndshm.dat

2B)
2C)
2D)



Exercise 3: Plotting of Electric potential for different systems

In this exercise, we shall plot the electrical potential of different systems using the imshow function of matplotlib. Your job in the exercises is to identify the systems. It need not be a correct answer, this is just to understand your knowledge on electromagnetism. The plotting however, should be as accurate as possible.

Poisson Solvers

Poisson solvers are codes used to solve the Poisson equation in electromagnetic theory.

$$\nabla^2 \Phi = -\rho$$

Where the Φ gives you the potential of the system. There are many ways to solve this equation numerically, but in this excercise, we use the method of successive overrelaxation.The function used to evaluate for a 2D system is provided below. For more details on this, you can read this file.

In [ ]:

```
#Don't change anything in this cell
def sorgspoisson(w,Pn,Rho,delx,dely):
  sum=0
  for i in range(Pn.shape[1]-1):
    for j in range(Pn.shape[0]-1):
      o=Pn[i,j]
      Pn[i,j]=(1-w)*Pn[i,j]+w*(((Pn[i+1][j]+Pn[i-1][j])/delx**2+(Pn[i][j+1]+Pn[i][j-1])/dely**2-Rho[i,j])/(2/del

  sum=sum+(o-Pn[i,j])**2
  ne=np.sqrt(sum)
  return Pn,ne
```

Now run the below cell for system 1. Note that it will take a minute or two to run. So please be patient

In [ ]:

```
#Don't change anything in this cell
#System 1
Rho=np.zeros((100,100))
P=np.zeros((100,100))
for i in range(25,75):
  Rho[25][i]=1
  Rho[75][i]=-1
```

```
T=1000
tol=1e-9

for i in range(T):
  P,ne=sorgspoisson(1.8,P,Rho,0.1,0.1)
  if (ne
```

Now plot the potential obtained using either imshow or contour/countourf

In [ ]:

```
#3a) Plot the potential map given in P
```

Can you try guessing what system do you think it is?

In [ ]:

```
#Try to guess what system this is.
#Don't forget to put a Hashtag before your answer
```

Now run the second system. Again this will take a minute or two to run. So please be patient

In [ ]:

```
#Don't change anything in this cell
#System 2
Rho=np.zeros((100,100))
P=np.zeros((100,100))
Rho[45][50]=1
Rho[65][50]=-1
T=100
tol=1e-9
for i in range(T):
  P,ne=sorgspoisson(1.8,P,Rho,1,1)
  if (ne
```

Again plot the potential obtained

In [ ]:

```
#3b) Plot the potential map given in P
```

Try guessing what system this is

In [ ]:

```
#Try to guess what system this is.
#Don't forget to put a Hashtag before your answer
```