Enter your name

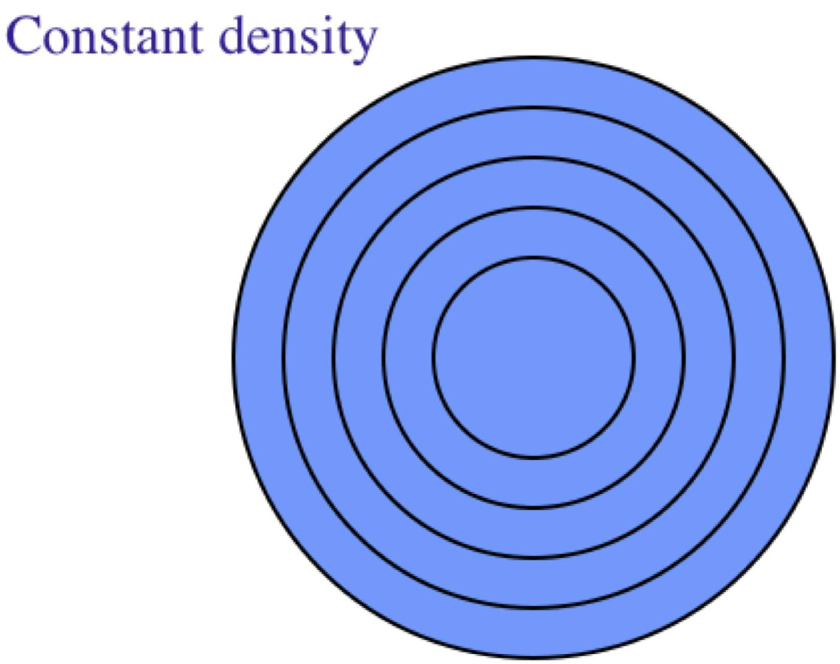
Your_Name: " Insert text here "

Show code

Building a Star

This code will create a plot for a star (2-D circle) of uniform density (each shell has equal the same density) and plot how the mass and gravity change as a function of radius.

+ Code + Text



For your Python code, you must first import the relevant packages for your code.

Here we are importing or loading 3 packages that we will use with this code:
1- **NumPy** is a fundamental package of Python. It allows for a wide range of data types and data manipulation capapbilities. To use it you will call it simply as **np**.
2- **Matplolib** is a Python 2D plotting library. We are only loading the Pyplot capabilties, which provide a collection of command that make matplotlib work like *MATLAB*.
3- **Math** is a module that allow for certain math functions.
4- `%matplotlib notebook` allows for interactive plotting.

To execute a code in a cell, you can either select the cell and hit *Shift+Enter* or click on the run cell button above.

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import math
```

Once you have succefully executed a cell, a number should appear in the square brackets next to that cell.

First we will define an array `rad_star` (for radius of star) that will go from the center of the star ($r = 0$) to the star's srurface ($r = 1$). We will use the *NumPy* function *arange* which generates evenly spaced values within a given interval. We specify the increment to be 0.01.
You need to specify the start and end points for the radius.

Keep in mind that *Floats* or non-integer numbers (i.e. with decimal places) should be written with the decimal point. For example, if our starting point is 1 and our ending point is 10, then the code should be `np.arange[1.0,10.0]`.

```
start =
end =
rad_star = np.arange(start,end,0.01)
```

Mass

For this exercise you will need to determine the relation between Mass M and radius r, given that the "star" is of constant density, ρ. Use the continuity equation

$$\frac{dM_r}{dr} = 4\pi\rho r^2$$

We have worked this out in class, but work this out again on a separate sheet of paper first and enter the expression in the cell below. Your expression will be a normalized mass in terms of the total stellar mass, M_*.

NOTES:

- To use the constant π in Python, you use `math.pi`. For example, the circumference of the star can be expressed like `circumf = 2.0 * math.pi * rad_star`
- In your expression for mass, the radius term r is defined by the variable `rad_star`.
- Remember to hit *Shift+Enter* to execute the cell.

```
mass =
```

Gravity

Now that we have an expression for mass of the star in terms of radius, we will now do the same thing but for gravity$g$, where

$$g = \frac{GM_r}{r^2}$$

The gravity term will also be in normalized units, in terms of the surface gravity $g_{surface}$.

```
grav =
```

Plotting

There are several ways to plot in Python. Here we will use some of the tools that are part of *matplotlib*. However, you can explore other ways to plot in [this useful tutorial](). Each line of code is explained below.

- Create a figure object, called fig
  `fig = plt.figure()`
- Create an "axis object" in the figure
  `ax1 = fig.add_subplot(111)`
- Creates a new set of axes (ax2) that shares the x-axis with ax1, but can have a separate y-axis.
  `ax2 = ax1.twinx()`

- Tells python what to plot in the axes object, i.e. your (x,y) values. In this case ax1 will be for the mass. The last part specifies to plot a straight solid line ('-') in blue ('b').
  `ax1.plot(rad_star,mass, 'b-')`
- Plot the gravity in ax2 as a straight, solid line in red.
  `ax2.plot(rad_star,grav, 'r-')`
- When creating figures, you should always properly label your figure and include units when appropriate.
  `ax1.set_title("Stellar Interior of Star with constant density")`
- To label your axes, `set_ylabel` defines the y-axis labels and `set_xlabel` does the same for the x-axis.
  `ax1.set_ylabel('Mass (M/M_{*})', color='blue')`
  `ax2.set_ylabel('Gravity (g/g_{surface})', color='red')`
  `ax1.set_xlabel('Radius (r/R_{*})')`
- Each time we plot, we must show the figure plot.
  `plt.show()`

```
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax2 = ax1.twinx()
ax1.plot(rad_star, mass, 'b-')
ax2.plot(rad_star, grav, 'r-')

ax1.set_title("Stellar Interior of Star with constant density")
ax1.set_ylabel('Mass ($M/M_{*}$)', color='blue')
ax2.set_ylabel('Gravity ($g/g_{surface}$)', color='red')
ax1.set_xlabel('Radius ($r/R_{*}$)')

plt.show()
```

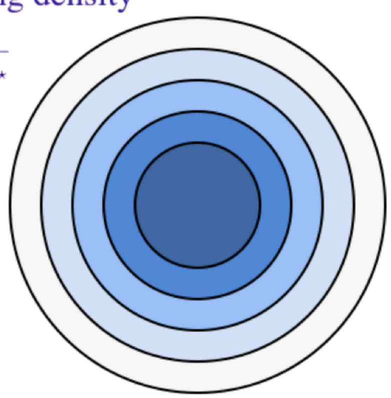
$\Longrightarrow$ Now see if you can edit the code above and add density to your plot.

Decreasing density Star

You will now adapt the code from the constant density case to create a plot for a star (2-D circle) of decreasing density and plot how the mass, gravity, density, pressure and potential energy change as a function of radius.

Decreasing density

$$\frac{\rho}{\rho_o} = 1 - \frac{r}{R_\star}$$



First we will define an array `rad_star_dec` (for radius of star with decreasing density) that will go from the center of the star ($r = 0$) to the star's surface ($r = 1$). We will use the *NumPy* function *arange* which generates evenly spaced values within a given interval. We specify the increment to be 0.01. As above, you need to specify the start and end points for the radius.

Keep in mind that *Floats* or non-integer numbers (i.e. with decimal places) should be written with the decimal point. For example, if our starting point is 1 and our ending point is 10, then the code should be `np.arange[1.0,10.0]`.

```
start_dec =
end_dec =
rad_star_dec = np.arange(start,end,0.01)
```

Mass

For this exercise you will need to determine the relation between Mass M and radius r, given that the "star" is of constant density, ρ. Use the continuity equation

$$\frac{dM_r}{dr} = 4\pi\rho r^2$$

We have worked this out in class, but work this out again on a separate sheet of paper first and enter the expression in the cell below. Your expression will be a normalized mass in terms of the total stellar mass, M_*.

NOTES:

- To use the constant π in Python, you use `math.pi`. For example, the circumference of the star can be expressed like
  `circumf = 2.0 * math.pi * rad_star`
- In your expression for mass, the radius term r is defined by the variable `rad_star`.
- Remember to hit *Shift+Enter* to execute the cell.

```
mass_dec =
```

Gravity

Now that we have an expression for mass of the star in terms of radius, we will now do the same thing but for gravity$g$, where

$$g = \frac{GM_r}{r^2}$$

The gravity term will also be in normalized units, in terms of the surface gravity $g_{surface}$.

```
grav_dec =
```

Continue to build your model.

Density

Express the normalized density ρ/ρ_0.

```
rho_dec =
```

Pressure

Now enter the expression for normalized pressure in terms of the central pressure P_c.

```
press_dec =
```

Potential Energy

Enter the expression for the normalized potential energy Ω_r/Ω_{total}, where Ω_r is the enclosed potential enegy. The potential energy of a small shell is

$$d\Omega_r = -\frac{GM_r}{r}dM_r$$

```
Epot_dec =
```

Plotting

There are several ways to plot in Python. Here we will use some of the tools that are part of *matplotlib*. However, you can explore other ways to plot in [this useful tutorial](). Each line of code is explained below.

- Create a figure object, called fig
  `fig = plt.figure()`
- Create an "axis object" in the figure
  `axes = fig.add_subplot(111)`
- Tells python what to plot in the axes object, i.e. your (x,y) values. The last part specifies to plot a line in blue ('b').
  `axes.plot(rad_star,mass, 'b-')`
- When creating figures, you should always properly label your figure and include units when appropriate.
  `axes.set_title("Stellar Interior of Star with decreasing density")`
- To label your axes, `set_ylabel` defines the y-axis labels and `set_xlabel` does the same for the x-axis.
  `axes.set_xlabel('', color='blue')`
  `axes.set_ylabel('')`
- Since we are plotting more than two lines, and all the parameters are normalized, we will use a legend to label each line. You can specify where you want the legend to appear. Best lets Python place it where it thinks it is best. `axes.legend(loc='best')`
- Each time we plot, we must show the figure plot.
  `plt.show()`

```
fig = plt.figure()
axes = fig.add_subplot(111)

axes.plot(rad_star_dec, rho_dec, 'r', label='Density')
axes.plot(rad_star_dec, mass_dec, 'b', label='Mass')
axes.plot(rad_star_dec, grav_dec, 'g', label='Gravity')
axes.plot(rad_star_Dec, press_dec, 'm', label='Pressure')

axes.set_title("")#<--Enter appropriate plot title in the quotations
axes.set_xlabel('')#<--Enter appropriate x-axis title in the quotations
axes.set_ylabel('')#<--Enter appropriate y-axis title in the quotations

axes.legend(loc='best')
plt.show()
```

Adapt the plotting routine to include the potential energy.

You can use the following Python color chart to add more colors to your plots.