Week 6 - ASTR26L - Class Tasks - Functions

Instructor: Jessica Schonhut-Stasik

February 16, 2021

1 Instructions

Welcome to Week 6 class tasks! This weeks aim is to teach you all about Python syntax like indentation and print operations. We also cover tests and loops like for and if. As usual, the whole set of Class Tasks is worth 50 points, and points are allocated clearly below, broken down by section and by question. There are three sections to this Class Task sheet, the questions will be split up into written questions (50%) and coding questions (50%).

Once you have completed these questions, please upload the code and any other documentation (i.e. plots or data files) to the Git Classroom. The repo you receive in this weeks link should be empty, save a README file with these notes within it.

The deadline for these Class Tasks to be completed is **Friday 26th February at 5pm HST**. I will not mark anything that I see in the Github repos before then, so you may use this as a place to store work in progress without worrying it may only be partially marked.

Please submit your answers in the form of one, or more, .py files that I can run to output the print statements requested in the questions. Please also include a .txt document with the print statement outputs from each question, in case I cannot get your code to run. Please upload any plots or data files as the file type requested in the question, also include any answers to questions that provide a text response. If there are lots of print statements in your code, please use a new line command to create a new line between every question. You are also encouraged to include print statements throughout your code to let me know what you are doing, it is easier for me to give you points when I understand your process. This may seem arbitrary now, when the questions can be solved in one line of code, but it will come in handy later, when writing functions etc.

2 Written Questions [Section = 25 points]

The following section requires written answers, please complete them in the way easiest for you, and upload them to the GitHub classroom.

1. Describe a benefit of coding an expression into a function, how does it save time?
2. How many times can you call a function once you have defined it?
3. What is the difference between a function definition and a function call?
4. What does a `return` statement do?
5. What does a function return by default if no return value, or even a `return` statement, is given?
6. What is the syntax for the header of a function, how does it look when you write it out?
7. What is the scope of a global variable?
8. What is the scope of a local variable?
9. How can you make a local variable, defined in a function, into a global variable?

10. What is the 'Built-In' Scope?
11. What is the LEGB rule?
12. Is it possible to nest functions within one another? What about in `for` loops or `if` tests?
13. What happens if you define a function but never call it?
14. Where do you find the 'arguments' of a function?
15. Is there a limit to how many arguments you can put into a function?
16. What are the four main ways to assign an argument in a function definition?
17. In what order are arguments mapped into a function?
18. For the function `def func(a, b, c=3)`, if I call `func(1, 2)`, what is the value of `c`, and why?
19. If I define a function with the following syntax `def func(**name)`, what type of object will be created from the arguments from the arguments?
20. How about if I define a function with this syntax, `def func(*name)`?
21. What type of argument has this syntax; `name=value`?
22. Is it possible to overwrite a default argument when calling a function?
23. What module can you import to time your code?
24. What are the units of the output of times calculated with this module, named in the last question?
25. Why is timing code important?

3 Coding Questions [Section = 25 points]

Each of these questions requires you to write code for your answer. Save this code into a .py file that can be run and output the required results. The required results will be print statements so please also save the results into a .txt file I can access and upload to the GitHub classroom.

1. Open the file `'g_stars.csv'`, that I sent out on the Slack. You can read it in however you like, you will playing around with the data in functions that you write.
2. The column `logL` contains the luminosity values (logged) of all the G stars in the model sample. Create a list of these values.
3. These values are in units of Solar Luminosity. Where the Solar Luminosity $= 3.828 \times 10^{26} W$. Define a function that will read in these values as a list.
4. Now write the body of the function; it should take each value of log(L) and convert it to W by taking it out of log-space and converting it using the value given in question 3.
5. Return these values as a new list.
6. Call the function, and save the new list.
7. Define a second function and follow the same process for `logTe`. Produce a list of values that have been taken out of log-space. In this case, there is no conversion of units necessary.
8. Plot a scatter plot using these lists, with luminosity [W] on the y-axis and temperature [K] on the x-axis (with the x-axis reversed from maximum to minimum). What type of plot is this (you have done it with Mike before)? Save this plot.

9. There are two columns in this file called `Mact` and `m2/m1`. `Mact` is the mass of a primary star (in a binary system) and `m2/m1` is the ratio of masses between the primary and secondary star in a binary system. Therefore it is possible to determine the mass of the secondary star. Write down the equation that allows you to do this.

10. Define a function that takes in a primary mass and a mass ratio

11. For the body of the function determine the secondary mass. Have the function return the secondary mass.

12. Create a list with all the values of the secondary masses.

13. The column `m-M0` is the result of an equation called the 'distance modulus' that determines the distance to a star based on its apparent and absolute magnitudes. It looks like this:

$$m - M = 5log(d) - 5$$

where d is in units of parsec.

14. Rearrange this equation to make d the subject. NB: In this case the column `m-M0` comprises one variable; everything on the left hand side of the equation.

15. Write a function that will take in the distance modulus as an argument and return the distance to the star.

16. Now edit the function so that the distance is read out in meters, not parsecs, using the conversion 3.086×10^{16}.

17. Now edit the definition of the function so that it also takes in as an argument, the column `Av` is the extinction values for these stars. Have the function read in this column also.

18. Create another equation in the function for the distance to the star, but this time, correct it for extinction, this is as simple as rearranging the following equation:

$$m - M = 5log(d) - 5 + Av$$

where Av is the extinction, which is a dimensionless value.

19. Have the function return two values, one for distance considering the extinction and one without.

20. Return both of these as lists.