Graded Practice 2

Astroinformatics I

Name: José Luis Ricra Mayorca

Date: Jun 8, 2025

Task 1

Use the CSV files you generated from the FITS files in practice 1. Write shell scripts to modify them in the following way:

(1) Change delimiter from "," to " ".

(2) Change the file extension from ".csv" to ".lc".

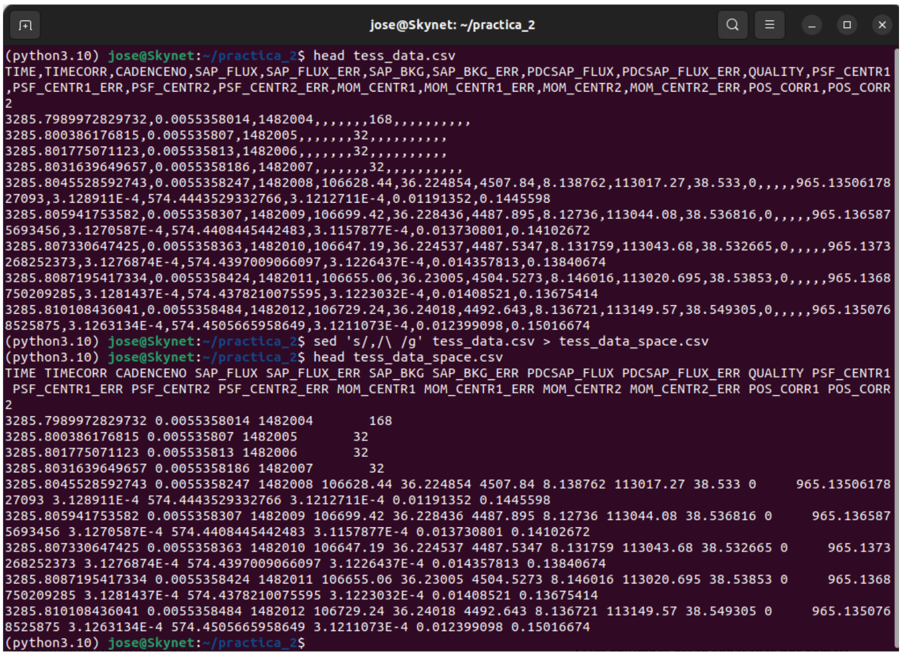
(3) Remove all columns that are not part of a light curve plot.

(1) Change delimiter from "," to " "

Following Tutorial 4 [Hernitschek, 2025b], the delimiter can be changed using the following Shell Script command:

```
(python3.10) jose@Skynet:~/practica_2$ sed 's/,/\ /g' tess_data.csv > tess_data_space.csv
```

A new file `tess_data_space.csv` is created to preserve the original file `tess_data.csv`. Figure 1 shows the original file, command execution, and the output file.



```
jose@Skynet: ~/practica_2
(python3.10) jose@Skynet:~/practica_2$ head tess_data.csv
TIME,TIMECORR,CADENCENO,SAP_FLUX,SAP_FLUX_ERR,SAP_BKG,SAP_BKG_ERR,PDCSAP_FLUX,PDCSAP_FLUX_ERR,QUALITY,PSF_CENTR1
,PSF_CENTR1_ERR,PSF_CENTR2,PSF_CENTR2_ERR,MOM_CENTR1,MOM_CENTR1_ERR,MOM_CENTR2,MOM_CENTR2_ERR,POS_CORR1,POS_CORR
2
3285.7989972829732,0.0055358014,1482004,,,,,,,168,,,,,,,,,,,
3285.800386176815,0.005535807,1482005,,,,,,,32,,,,,,,,,,,
3285.801775071123,0.005535813,1482006,,,,,,,32,,,,,,,,,,,
3285.8031639649657,0.0055358186,1482007,,,,,,,32,,,,,,,,,,,
3285.8045528592743,0.0055358247,1482008,106628.44,36.224854,4507.84,8.138762,113017.27,38.533,0,,,,,965.13506178
27093,3.128911E-4,574.4443529332766,3.1212711E-4,0.01191352,0.1445598
3285.805941753582,0.0055358307,1482009,106699.42,36.228436,4487.895,8.12736,113044.08,38.536816,0,,,,,965.136587
5693456,3.1270587E-4,574.4408445442483,3.1157877E-4,0.013730801,0.14102672
3285.807330647425,0.0055358363,1482010,106647.19,36.224537,4487.5347,8.131759,113043.68,38.532665,0,,,,,965.1373
268252373,3.1276874E-4,574.4397009066097,3.1226437E-4,0.014357813,0.13840674
3285.8087195417334,0.0055358424,1482011,106655.06,36.23005,4504.5273,8.146016,113020.695,38.53853,0,,,,,965.1368
750209285,3.1281437E-4,574.4378210075595,3.1223032E-4,0.01408521,0.13675414
3285.810108436041,0.0055358484,1482012,106729.24,36.24018,4492.643,8.136721,113149.57,38.549305,0,,,,,965.135076
8525875,3.1263134E-4,574.4505665958649,3.1211073E-4,0.012399098,0.15016674
(python3.10) jose@Skynet:~/practica_2$ sed 's/,/\ /g' tess_data.csv > tess_data_space.csv
(python3.10) jose@Skynet:~/practica_2$ head tess_data_space.csv
TIME TIMECORR CADENCENO SAP_FLUX SAP_FLUX_ERR SAP_BKG SAP_BKG_ERR PDCSAP_FLUX PDCSAP_FLUX_ERR QUALITY PSF_CENTR1
 PSF_CENTR1_ERR PSF_CENTR2 PSF_CENTR2_ERR MOM_CENTR1 MOM_CENTR1_ERR MOM_CENTR2 MOM_CENTR2_ERR POS_CORR1 POS_CORR
2
3285.7989972829732 0.0055358014 1482004       168
3285.800386176815 0.005535807 1482005       32
3285.801775071123 0.005535813 1482006       32
3285.8031639649657 0.0055358186 1482007       32
3285.8045528592743 0.0055358247 1482008 106628.44 36.224854 4507.84 8.138762 113017.27 38.533 0     965.13506178
27093 3.128911E-4 574.4443529332766 3.1212711E-4 0.01191352 0.1445598
3285.805941753582 0.0055358307 1482009 106699.42 36.228436 4487.895 8.12736 113044.08 38.536816 0     965.136587
5693456 3.1270587E-4 574.4408445442483 3.1157877E-4 0.013730801 0.14102672
3285.807330647425 0.0055358363 1482010 106647.19 36.224537 4487.5347 8.131759 113043.68 38.532665 0     965.1373
268252373 3.1276874E-4 574.4397009066097 3.1226437E-4 0.014357813 0.13840674
3285.8087195417334 0.0055358424 1482011 106655.06 36.23005 4504.5273 8.146016 113020.695 38.53853 0     965.1368
750209285 3.1281437E-4 574.4378210075595 3.1223032E-4 0.01408521 0.13675414
3285.810108436041 0.0055358484 1482012 106729.24 36.24018 4492.643 8.136721 113149.57 38.549305 0     965.135076
8525875 3.1263134E-4 574.4505665958649 3.1211073E-4 0.012399098 0.15016674
(python3.10) jose@Skynet:~/practica_2$
```

Figure 1: Changing column delimiter.

(2) Change the file extension from ".csv" to ".lc"

To rename `tess_data_space.csv` to `tess_data_space.lc`, we follow Tutorial 2 [Hernitschek, 2025a] and use the "cp" command:

```
(python3.10) jose@Skynet:~/practica_2$ cp tess_data_space.csv tess_data_space.lc
```

In this case, the "cp" command was used to rename the file while also preserving the original file. Figure 2 shows the steps and the results obtained.

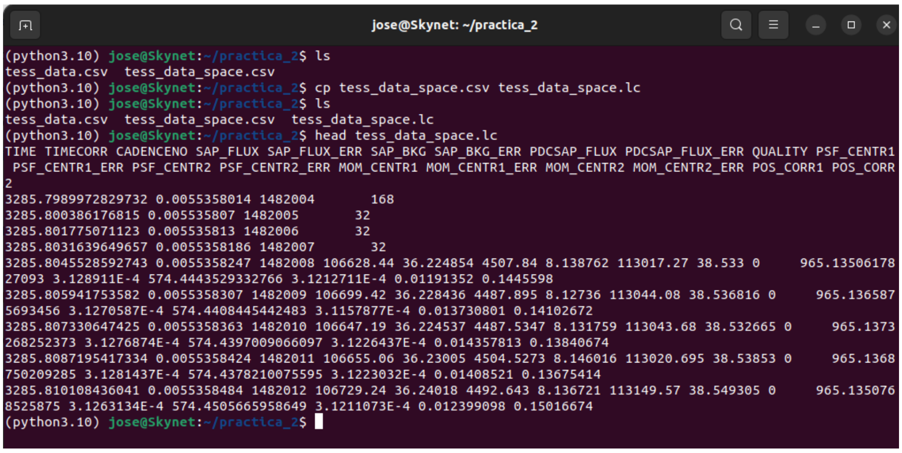


Figure 2: Changing the file extension.

(3) Remove all columns that are not part of a light curve plot.

Removing the columns that are not part of the light curve is equivalent to extracting only the columns corresponding to the light curve into a new file. By examining the file `tess_data_space.lc`, it was observed that the columns containing time and flux data correspond to the first and eighth columns `(TIME, PDCSAP_FLUX)`. Following the instructions in Tutorial 4 [Hernitschek, 2025b], we used the "awk" command to extract the specified columns:

```
awk '{print $1, $8}' tess_data_space.lc > time_flux.lc
```

Figure 3 shows the steps performed.

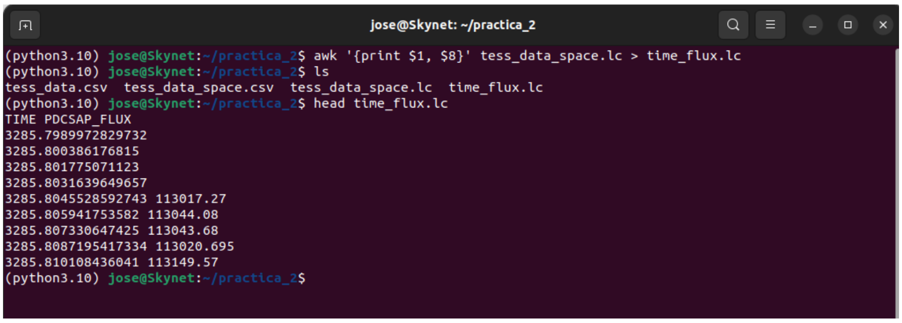


Figure 3: Extraction of columns corresponding to the light curve.

Task 2

Spectra of stars are classified according to the letters O, B, A, F, G, K, and M. These correspond to the following temperature ranges (in degrees K):

O: 30000 - 60000
B: 10000 - 30000
A: 7500 - 10000
F: 6000 - 7500
G: 5000 - 6000
K: 3500 - 5000
M: 2000 - 3500

Write a program which takes the temperature as a command line argument and prints out the spectral class. Print a suitable message if the temperature is out of range.

Solution

To solve this problem, we referred to the instructions provided in the Python Introduction class [Hernitschek, 2025c] and Tutorial 5 [Hernitschek, 2025d].

A Python script called `spectral.py` was written to address this problem. The structure of the code is detailed below:

First, the `sys` module was used, which allows access to arguments passed from the command line.

```
import sys
```

Then, a function was defined that receives a temperature parameter (temp), compares the value with known ranges, and returns the letter of the spectral class. If the temperature is outside the valid ranges, it returns None.

```
def clase_espectral(temp):
    if 30000 <= temp <= 60000:
        return "O"
    elif 10000 <= temp < 30000:
        return "B"
    elif 7500 <= temp < 10000:
        return "A"
    elif 6000 <= temp < 7500:
        return "F"
    elif 5000 <= temp < 6000:
        return "G"
    elif 3500 <= temp < 5000:
        return "K"
    elif 2000 <= temp < 3500:
        return "M"
    else:
        return None
```

Then the *main()* function is defined, which checks that the command format entered is correct. If it is incorrect, it prints the message "Correct use: python spectral.py <temperature>".

```
def main():
    if len(sys.argv) != 2:
        print("Correct use: python spectral.py <temperature>")
        return
```

Next, the command line argument is converted to a decimal number:

```
try:
        temperature = float(sys.argv[1])
```

Then the function `spectral_class` is called to identify the letter corresponding to the temperature range, and the result is stored in the variable SC:

```
SC = spectral_class(temperature)
```

If SC contains a letter (is not None), it is printed. If the temperature did not fit any range, it notifies that it is out of range.

```
if SC:
            print(f"The spectral class is: {SC}")
        else:
            print("Temperature out of range for spectral classification.")
```

Finally, if the temperature value was not a number, the message "The entered value is not in the correct format" is printed:

```
except ValueError:
        print("The entered value is not in the correct format")
```

Figure 4 shows the script execution.

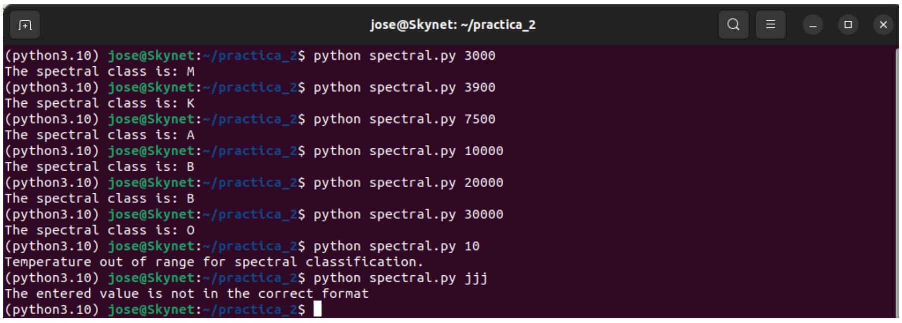


Figure 4: Script execution.

Task 3

Given the year, month and day of the month, the Julian day is calculated as follows:

$$\text{Julian} = \frac{36525 \times \text{year}}{100} + \frac{306001 \times (\text{month} + 1)}{10000} + \text{day} + 1720981$$

where month is 13 for Jan, 14 for Feb, 3 for Mar, 4 for Apr, etc. For Jan and Feb, the year is reduced by 1.

Write a script which asks for the day, month and year and calculates the Julian day. All variables must be of integer type.

What is the Julian day for 7 Jun 2008?

Solution

To solve this problem, a Python script called `jp.py` was written, whose structure is shown below:

First, the variables `day`, `month`, and `year` are defined:

```
day = int(input("Write the day: (1-31): "))
month = int(input("Write the month: (1-12): "))
year = int(input("Write the year: (ex. 2008): "))
```

Then, an adjustment is made for the months of January and February. This is done because the Julian day is based on a calendar where March is the beginning of the year. To work correctly, January (1) is assigned month 13, February (2) month 14, and the year is decreased by one:

```
if month == 1 or month == 2:
    month += 12  # January -> 13, February -> 14
    year -= 1    # the year is reduced
```

Finally, the Julian day value is calculated using integer division (//). This is done because the Julian day must be an integer number when only days and not hours are entered.

```
julian_day = (36525 * year) // 100 + (306001 * (month + 1)) // 10000 + day + 1720981
```

Finally, the result is printed:

```
print("The julian day is:", julian_day)
```

For the specific case of June 7, 2008, the Julian day is: 2454624

Figure 5 shows the script execution.

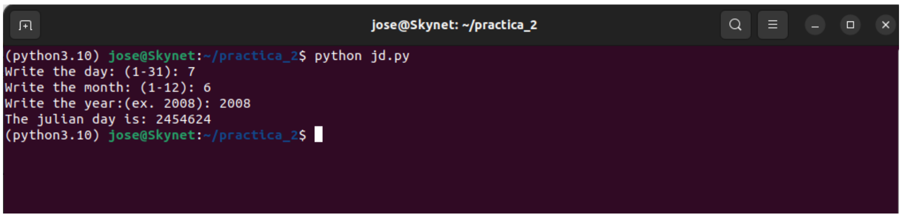


Figure 5: Script execution.

References

N. Hernitschek. Tutorial 2. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial2.pdf, 2025a. PDF available on GitHub.

N. Hernitschek. Tutorial 4. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial4.pdf, 2025b. PDF available on GitHub.

N. Hernitschek. Python introduction. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/astroinformaticaI_lecture5_pythonintroduction.pdf, 2025c. PDF available on GitHub.

N. Hernitschek. Tutorial session 5: Python introduction. https://github.com/ninahernitschek/astroinformatica_I_2025_1/blob/main/tutorial_5.ipynb, 2025d. PDF available on GitHub.