Practice 2

Astroinformatics I

Semester 1, 2025

Ana Pereira Aguirre

June 9, 2025

1 Introduction

This report contains the solutions of the Practical II of the Astroinformatics I course. The tasks consist of manipulating light curve files using shell scripts, classifying stars based on their temperature, and calculating Julian days from calendar dates. All the code and explanations are developed below.

2 Exercise solutions

2.1 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 " ".

   For this item, we use the `sed` command, as we learn in class, to replace commas with spaces.
   First, we create our script in the terminal using the command `nano`, in the following way:

```
      nano task1_practice2.sh #to create the script
      chmod 755 task1_practice2.sh #to make the script executable
```

   We write the following code inside the script. We define the path where the files we want to modify are located and the path where the modified files will go to:

```
  #!/bin/bash
  #Input
  tess_file="/Users/anapereira/Documents/tess_practice/tess_csv_lc"
  #Output
  tess_file_modified="/Users/anapereira/Documents/tess_practice/tess_csv_lc/practice_2"
```

   Then, using a `for` loop, we iterate over all `.csv` files inside the `tess_csv_lc` directory. Within the loop, we use `basename` to extract the name of each file without its extension.

The `sed` command is then used to replace all commas in each file with spaces and the output files are saved in the `practice_2` folder using the same original name, followed by `_mod.csv`:

```
for tess_csv in "$tess_file"/*.csv; do
    new_name=$(basename "$tess_csv" .csv)
    sed 's/,/ /g' "$tess_csv" > "$tess_file_modified/${new_name}_mod.csv"
done
```

When we run the script using `./task_practice2.sh` from inside the `practice_2` folder, we get:

```
anapereira@MacBook-Pro-de-Ana practice_2 % ls
task1_practice2.sh      tess_lc_411_mod.csv     tess_lc_696_mod.csv
tess_lc_015_mod.csv     tess_lc_416_mod.csv     tess_lc_723_mod.csv
tess_lc_045_mod.csv     tess_lc_463_mod.csv     tess_lc_736_mod.csv
tess_lc_191_mod.csv     tess_lc_539_mod.csv     tess_lc_744_mod.csv
tess_lc_268_mod.csv     tess_lc_589_mod.csv     tess_lc_765_mod.csv
tess_lc_329_mod.csv     tess_lc_666_mod.csv     tess_lc_979_mod.csv
tess_lc_406_mod.csv     tess_lc_692_mod.csv     tess_lc_984_mod.csv
```

Figure 1: Terminal output showing the modified `.csv` files, obtained by executing the script `task1_practice2.sh`.

To check our results, we can open the output file either with Excel or by using the `pandas` library. Opening the file in Excel, we can observe the following:

	A	B	C	D	E	F
1	TIME TIMECORR CADENCENO SAP_FLUX SAP_FLUX_ERR SAP_BKG SAP_BKG_ERR					
2	3285.7989972829732 0.0055358014 1482004 168					
3	3285.800386176815 0.005535807 1482005 32					
4	3285.801775071123 0.005535813 1482006 32					
5	3285.8031639649657 0.0055358186 1482007 32					
6	3285.8045528592743 0.0055358247 1482008 106628.44 36.224854 4507.84 8.1					
7	3285.805941753582 0.0055358307 1482009 106699.42 36.228436 4487.895 8.1					
8	3285.807330647425 0.0055358363 1482010 106647.19 36.224537 4487.5347 8.					
9	3285.8087195417334 0.0055358424 1482011 106655.06 36.23005 4504.5273 8.					
10	3285.810108436041 0.0055358484 1482012 106729.24 36.24018 4492.643 8.13					

Figure 2: Screenshot of the file `tess_lc_589_mod.csv` opened in Excel.

And using **pandas**, by specifying that the columns are separated by spaces, we can see the following:

```
[16]:
import pandas as pd

df = pd.read_csv('tess_lc_589_mod.csv', sep=' ')


df.head()
```

[16]:

	TIME	TIMECORR	CADENCENO	SAP_FLUX	SAP_FLUX_ERR
0	3285.798997	0.005536	1482004	NaN	NaN
1	3285.800386	0.005536	1482005	NaN	NaN
2	3285.801775	0.005536	1482006	NaN	NaN
3	3285.803164	0.005536	1482007	NaN	NaN
4	3285.804553	0.005536	1482008	106628.44	36.224854

Figure 3: Screenshot of the file `tess_lc_589_mod.csv` opened with `pandas`

2. Change the file extension from ".csv" to ".lc".
   In the same script, using the `for` loop, we specify the path where the modified files with extension `.csv` are located. Inside the loop, using the command `mv` we change the extension `.csv` to `.lc`:

```
    for tess_mod in "$tess_file_modified"/*.csv; do
        mv "$tess_mod" "${tess_mod%.csv}.lc"
    done
```

   We run the script and verify the result using the `ls` command.

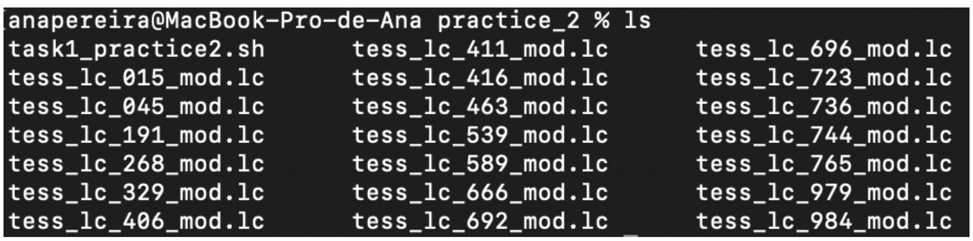

   Figure 4: Terminal output showing the modified files with the `.lc` extension, obtained by executing the script `task1_practice2.sh`

3. Remove all columns that are not part of light curve plot.
   In this last part, inside the same script. We use the `for` loop, to go through all the `.lc` type files. For each file, we extract the name with `basename` and define a new file name for the output, which will contain only the columns needed to plot the light curve: `TIME` and `PDCSAP_FLUX`.

   Then, using the command `awk` we filter the columns that correspond to the light curve plot (namely `TIME` and `PDCSAP_FLUX`) . We specify the space character as the delimiter (`-F' '`), define new column names followed by the values corresponding to the first and eighth column of the original file (since we know that `PDCSAP_FLUX` corresponds to the eighth column).

```
        for tess_lc in "$tess_file_modified"/*.lc; do
            base_name=$(basename "$tess_lc" .lc)
            new_file="$tess_file_modified/${base_name}_lcfile.lc"
            awk -F' ' 'NR==1 {print "TIME PDCSAP_FLUX"} NR>1 {print $1 " " $8}' "$tess_lc" > "$new_file"
        done
```

   To verify our result, we converted the `.lc` files back to `.csv` by reusing the code from point 2, but applying the inverse process, and the opened them using `pandas`:

```
import pandas as pd

filename = 'tess_lc_589_mod_lcfile.csv'

df = pd.read_csv(filename, sep=' ')

# To show all files
pd.set_option('display.max_rows', None)

print(df)
```

```
             TIME   PDCSAP_FLUX
0     3285.798997           NaN
1     3285.800386           NaN
2     3285.801775           NaN
3     3285.803164           NaN
4     3285.804553    113017.270
5     3285.805942    113044.080
6     3285.807331    113043.680
7     3285.808720    113020.695
8     3285.810108    113149.570
```

Figure 5: File `tess_lc_589_mod_lcfile.csv` open with `pandas` which shows the columns corresponding to the light curve plot.

2.2 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.

For this task, we created a `Jupyter Notebook` where we used the `input()` function to ask the user to enter the temperature of a star in Kelvin. Then using `if`, `elif` and `else` conditionals, the input value is compared against the defined temperature ranges for each spectral type, as shown in the as follows:

```python
temperature = float(input("Enter the star temperature in Kelvin: "))
if temperature<60000.0 and temperature>=30000.0:
    print('Your star is a O type')
elif temperature>=10000.0 and temperature<30000.0:
    print('Your star is a B type')
elif temperature>=7500.0 and temperature<10000.0:
    print('Your star is a A type')
elif temperature>=6000.0 and temperature<7500.0:
    print('Your star is a F type')
elif temperature>=5000.0 and temperature<6000.0:
    print('Your star is a G type')
elif temperature>=3500.0 and temperature<5000.0:
    print('Your star is a K type')
elif temperature>=2000.0 and temperature<3500.0:
    print('Your star is a M type')
else:
    print("Temperature out of classified range")
```

As an example, we run the code by entering the temperature of the Sun, we obtain the following result:

```
Enter the star temperature in Kelvin:  5772
Your star is a G type
```

Figure 6: Output of the Sun's spectral classification.

2.3 Task 3

Given the year, month and day of the month, the Julian day is calculated as follows: `Julian = (36525*year)/100 + (306001*(month+1))/10000 + 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?.

For task 3, we again created a notebook where we wrote the requested code:

```
print("Give me a date")
day = int(input("day: "))
month = input("month (for example: Jan, JAN, Feb, jun, etc..): ")

month_number = {
    "Jan": 13, "Feb": 14, "Mar": 3, "Apr": 4, "May": 5, "Jun": 6,
    "Jul": 7, "Aug": 8, "Sep": 9, "Oct": 10, "Nov": 11, "Dec": 12
}
month = month_number.get(month.capitalize())

year = int(input("year: "))

if month == 13 or month ==14:
    year = year - 1
    julian = int((36525 * year) / 100 + (306001 * (month + 1)) / 10000 + day + 1720981)
    print("The julian day is:", julian)
else:
    julian = int((36525 * year) / 100 + (306001 * (month + 1)) / 10000 + day + 1720981)
    print("The julian day is:", julian)
```

The code does the following

1. The program asks the user to enter the day, the abbreviated month, and the year.
2. The program then associates the entered month with its corresponding numerical value according to the mapping defined in the code.
3. Using an `if` conditional, it checks whether the entered month is January or February; if so, it follows the instructions from the exercise by assigning January to 13 and February to 14, and reduces the year by one (i.e., `year = year - 1`).
4. If the month is any other than January or February, the `else` block calculates the Julian day normally, using the month number without any modification.

5. Finally, the program computes the Julian day based on the given formula and prints the result.

Important: We added the line of code `month = month_number.get(month.capitalize())` to make sure that, in case the user enters the month with a combination of upper and lower case letters (e.g. "jun", 'JUN', "Jun"), the program interprets it correctly. In this way, we avoid errors and can continue with the calculation of the Julian day.

Then, what is the Julian day of June 7, 2008?

```
Give me a date
day:  7
month (for example: Jan, JAN, Feb, jun, etc..):  JUN
year:  2008
The julian day is: 2454624
```

Figure 7: Conversion of the date June 7, 2008 into Julian days.