Semester I 2025

Astroinformatics I

Graded Practice 2

José B. Batista M.

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

   a. **Change delimiter from ',' to ' '**

   b. **Change the file extension from '.csv' to '.lc'**

   c. **Remove all columns that are not part of a light curve plot**

   The shell script provided below (saved as **`task_1.sh`**) scans the root directory for a 'csv' subdirectory. The script checks for all .csv files inside it (after making sure the folder exists) and proceeds to replace all commas for spaces, as a first part. Once complete, a confirmation message is printed. After successfully completing part one, the script then renames the files from '*.csv' to '*.lc', changing the extension. Similarly, after this second part is done, a confirmation is printed. As a final part, the script re-scans the folder to now find all .lc files, and then deletes all unnecessary columns, keeping only **`TIME`**, **`PDCSAP_FLUX`**, and **`PDCSAP_FLUX_ERR`**. Once again, a confirmation is printed.

```
#!/bin/sh

# Get the script's root directory
root_dir="$(dirname "$0")"

# Set the folder with the CSV files
folder="$root_dir/csv"

echo "Starting modification of files in '$folder'..."

# 1. Change delimiter from ',' to ' '
echo "1. Changing delimiter from ',' to ' '..."
find "$folder" -type f -name "*.csv" -exec sed -i -e 's/,/\ /g' {} \;
echo "    Delimiter change complete."

# 2. Change the file extension from '.csv' to '.lc'
echo "2. Changing file extension from '.csv' to '.lc'..."
find "$folder" -type f -name "*.csv" | while read -r file; do
    new_name="${file%.csv}.lc"
    mv "$file" "$new_name"
done
echo "    Extension change complete."

# 3. Remove all columns that are not part of a light curve plot
echo "3. Removing extra columns (keeping 'TIME', 'PDCSAP_FLUX', and '
   PDCSAP_FLUX_ERR')..."
find "$folder" -type f -name "*.lc" | while read -r file; do
    # Create a temporary file
    temp_file=$(mktemp)
```

```
29      # Cut the desired columns (1, 8, 9) and redirect to the temporary
        file
30      cut -d' ' -f1,8,9 "$file" > "$temp_file"
31      # Overwrite the original file with the content of the temporary
        file
32      mv "$temp_file" "$file"
33  done
34  echo "    Column removal complete."
35
36  echo "All specified modifications have been applied to the files."
```

The resulting .lc files have the following format:

```
1  TIME PDCSAP_FLUX PDCSAP_FLUX_ERR
2  3285.804512931147 2078.9084 11.244134
3  3285.805901823592 2083.7026 11.255919
4  3285.8072907165038 2065.4507 11.240394
5  ...
```

2. **Spectra of stars are classified according to the letters O, B, A, F, G, K, and M. These correspond to the temperature ranges (in 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 temperatureas as a command line argument and prints out the spectral class. Print a suitable message if the temperature is out of range.**

The python script provided below (saved as **`task_2.py`**) takes the star temperature as a command line argument and prints out the corresponding spectral class. To do this, the code first checks that it has been given an argument and ensures it's a number by trying to convert it to a float. After this, it checks if the given value is within the valid [2000.0, 60000.0] K range. If any of the 3 checks fail, the code prints the appropiate error and stops. Otherwise, it proceeds to assign the value to the **`star_type`** variable according to the given temperature value. Finally, the code prints the resulting spectral class.

```
1    import sys
2  # Get temperature value from command line and ensure it's a valid
     number
3  if len(sys.argv) < 2:
4      print('No input provided. Use: python3 task_2.py <temperature>')
5      sys.exit(1)
6  try:
7      temperature = float(sys.argv[1])
8  except ValueError:
9      print('The input must be a number.')
10     sys.exit(1)
11 # Make sure the value is within the valid star temperature range
     [2000, 60000] K
12 if temperature < 2000.0 or temperature > 60000.0:
13     print('The given star temperature is out of range [2000, 60000] K
       .')
14     sys.exit(1)
15 # Classify the star according to its temperature
```

```
16 if 2000.0 <= temperature <= 3500.0:
17     star_type = 'an M-type'
18 elif 3500.0 < temperature <= 5000.0:
19     star_type = 'a K-type'
20 elif 5000.0 < temperature <= 6000.0:
21     star_type = 'a G-type'
22 elif 6000.0 < temperature <= 7500.0:
23     star_type = 'an F-type'
24 elif 7500.0 < temperature <= 10000.0:
25     star_type = 'an A-type'
26 elif 10000.0 < temperature <= 30000.0:
27     star_type = 'a B-type'
28 else: # 30000.0 < temperature <= 60000.0
29     star_type = 'an O-type'
30 print(f'With a temperature of {temperature:.2f} K, this is {star_type
       } star.')
```

Some example runs:

```
> python3 task_2.py
No input provided. Use: python3 task_2.py <temperature>
> python3 task_2.py abc123
ValueError: The input argument must be a number.
> python3 task_2.py 1500
ValueError: The given star temperature is out of range.
Enter a temperature in the range [2000.0, 60000.0] K.
> python3 task_2.py 2864.593
With a temperature of 2864.59 K, this is an M-type star.
> python3 task_2.py 5778.42
With a temperature of 5778.42 K, this is a G-type star.
> python3 task_2.py 41253.7865
With a temperature of 41253.79 K, this is an O-type star.
```

3. **Given the year, month and day of the month, the Julian day is calculated as follows: `Julian = (36525*yr)/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?**

   The python script provided below (saved as **`task_3.py`**) Asks the user for a date in the formart **`d/m/y`** and ensures both the values of the components (day, month and year) and format are valid. After this, the code asks the user to confirm the entered date is the correct one, and it also makes sure the answer is valid [**`(Y/n)`**]. If all the error handlings are succeded, the code exists the loop and proceeds to convert the given date to the corresponding Julian date number. Finally, the result is printed for the user to see.

```
import sys, math

# Funtions for valid date values
def is_valid_date(year, month, day):
    if not (1 <= month <= 12):
        return False
    if day < 1:
        return False
    # Days in each month
    month_lengths = [31, 29 if is_leap_year(year) else 28, 31, 30,
    31, 30,
                     31, 31, 30, 31, 30, 31]
    if day > month_lengths[month - 1]:
        return False
    return True
def is_leap_year(year):
    # Gregorian calendar leap year rule
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
def string_date(year, month, day):
    match month:
        case 1: return f'{day} January, {year}'
        case 2: return f'{day} February, {year}'
        case 3: return f'{day} March, {year}'
        case 4: return f'{day} April, {year}'
        case 5: return f'{day} May, {year}'
        case 6: return f'{day} June, {year}'
        case 7: return f'{day} July, {year}'
        case 8: return f'{day} August, {year}'
        case 9: return f'{day} September, {year}'
        case 10: return f'{day} October, {year}'
        case 11: return f'{day} November, {year}'
        case 12: return f'{day} December, {year}'

# Main code
if __name__ == "__main__":
    print('{:-^79}'.format('# Julian Date Calculator #'))
    # Outer loop: Responsible for ensuring a confirmed date.
    while True:
        # Ask for date in valid format
        print("\nEnter the date in the 'd/m/y' format: ")
        date = str(input('Date: '))
        # Check date value and format
        try:
            date_parts = None
            # Iterate through possible delimiters
            parts = date.split('/')
            # Check if splitting by this delimiter gives exactly 3
    parts
            if len(parts) == 3:
                try:
                    date_parts = []
                    # Convert parts to integers
                    for part in parts:
                        part = int(part)
                        date_parts.append(part)
                    day, month, year = date_parts
```

```
55                  except:
56                      raise ValueError('One or more parts are not
      numbers.')
57              if date_parts is None: # No valid delimiter
58                  raise ValueError('Incorrect date format: Use d/m/y')
59              if not is_valid_date(year, month, day):
60                  raise ValueError('Invalid date component values.')
61          except ValueError as e:
62              print(f"Error: {e}")
63              continue # Go back to the outer loop to ask for a new
      date string
64          # Inner loop: Responsible for confirming the entered date.
65          while True:
66              date_str = string_date(year, month, day)
67              answer = input(f'\nYour date is {date} ({date_str}). Is
      this correct? (Y/n) ').strip().lower()
68              if answer in ('','y','yes'): # Date confirmed
69                  break
70              elif answer in ('n','no'): # Wrong date
71                  break
72              else: # Invalid answer. Ask again
73                  print('\nWrong answer. Enter (Y/n).')
74          # Check for date confirmation
75          if answer in ('', 'y', 'yes'): # Correct date
76              break
77          # If wrong, re-enter date
78      # Calculate the Julian date
79      julian_day_number = math.floor(36525*year/100 + 306001*(month+1)
      /10000 + day + 1720981)
80      print(f'\nThe Julian date for {date} is {julian_day_number}')
```

For the date of **7 June, 2008**:

```
> python3 task_3.py
---------------# Julian Date Calculator #---------------

Enter the date in the 'd/m/y' format:
Date: 7/6/2008

Your date is 7/6/2008 (7 June, 2008). Is this correct? (Y/n)

The Julian date for 7/6/2008 is 2454624
```