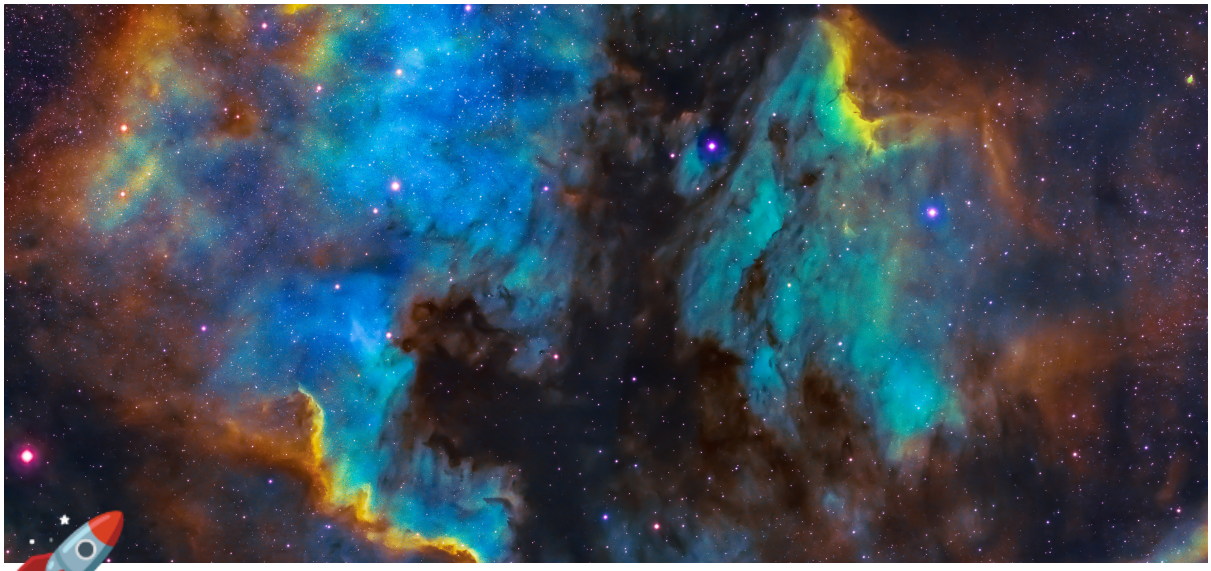


Solar Physics Data Analysis using Python

Group Members

1. Aditya Hriday Sahu (2020A7PS0144G)
2. Ojas Kanth (2020A7PS1391G)
3. Gurbaaz Singh Gill (2020A7PS1228G)

Contents

Introduction

- Solar physics is a field that involves studying the sun, its behavior, and its effects on the solar system. One important aspect of solar physics is the analysis of solar data, which can provide insight into the sun's behavior and help predict its future activity.
- Python is a popular programming language for solar physics data analysis due to its versatility and ease of use. Python has many libraries useful for solar data analysis, such as SunPy, which provides a framework for analyzing solar data, and NumPy, which provides tools for numerical computations.
- Python can analyze many types of solar data, such as photospheric magnetic field data, coronal magnetic field data, and solar wind data. Some standard techniques used in solar data analysis include Fourier analysis, wavelet analysis, and machine learning.

Overall, Python is a powerful tool for solar physics data analysis, and it can help researchers gain a deeper understanding of the sun and its behavior.

Project Objective

The objective of this project is to use SunPy and Python for solar physics data analysis to create a Full Sun Map, and to demonstrate the capabilities of SunPy for solar data analysis and visualization. The project will involve extracting and manipulating data from the Solar Dynamics Observatory (SDO) satellite, and creating a Full Sun Map that showcases the intensity of the solar corona and reveals important features of the sun's magnetic field. The Full Sun Map will be utilized for further analysis and research, including the study of solar flares and coronal mass ejections. Through this project, we aim to showcase the powerful capabilities of SunPy and Python in the field of solar physics research, and to contribute to our understanding of the Sun and its impact on our solar system.

Brief overview of existing literature

- In recent years, there has been a growing interest in using Python for solar physics data analysis due to its versatility, ease of use, and large community support. Python has become a popular tool for data analysis and visualization, and it has been widely adopted in the field of solar physics.

- One of the most popular Python libraries for solar physics data analysis is SunPy. SunPy provides a range of tools for solar image and data analysis, including the creation of Full Sun Maps. SunPy has been used for a variety of solar physics applications, including the study of solar flares, coronal mass ejections, and other important phenomena occurring on the Sun's surface and in its atmosphere.
- Several studies have used SunPy and Python for solar physics data analysis. For example, K. Reardon et al. (2018) used SunPy and Python to study the properties of coronal rain in active regions. Another study by N. Pereira et al. (2020) used SunPy and Python to create a time-lapse movie of the solar corona during the total solar eclipse of 2017.
- There are also several online resources available for learning how to use SunPy and Python for solar physics data analysis. The SunPy website provides a range of documentation, tutorials, and examples for using the library. There are also several online courses and tutorials available, including the Python in Astronomy conference and the SolarSoft IDL to SunPy Python Transition Tutorial.
- Overall, the existing literature on solar physics data analysis using Python and SunPy demonstrates the growing popularity and importance of these tools in the field of solar physics research. SunPy and Python provide a powerful platform for solar data analysis and visualization, and their continued development and use will undoubtedly contribute to our understanding of the Sun and its impact on our solar system

SunPy: A community-developed, free, and open-source solar data analysis environment for Python

- SunPy is a fantastic tool for analyzing solar physics data in Python. It is a community-developed, free, and open-source solar data analysis environment that provides a powerful framework for analyzing solar data with tools for data acquisition, processing and visualization. SunPy's core data types are based on the FITS data format widely used in astronomy and astrophysics.
- SunPy is an excellent tool for researchers in solar physics, as it offers a vast range of features that make it a valuable asset. One of the primary reasons why SunPy is so popular is the fact that it is open-source. This means anyone can use it, modify it, and contribute to its development. SunPy has a large and active

community of developers and users who contribute to the project and share their work. This community provides support and resources for users of all levels, from beginners to experts.

- Another valuable feature of SunPy is its user-friendly interface. SunPy's interface is designed to be intuitive and easy to use, making it accessible to users who aren't necessarily experts in Python. The interface is designed to allow users to quickly and easily perform a wide range of tasks, such as loading data, analyzing it, and visualizing it.
- SunPy also offers a wide range of data acquisition, processing, and visualization tools. Some of the most popular tools include the ability to read and write FITS files, interact with data stored in online databases, and create custom plots and visualizations. SunPy also has a range of built-in analysis tools, such as Fourier analysis, wavelet analysis, and machine learning algorithms.
- SunPy is a must-have tool for anyone working with solar data in Python. Its open-source nature, powerful tools, and user-friendly interface make it an incredibly valuable asset for researchers in the field. Whether you are a beginner or an expert, SunPy is a tool that you should consider adding to your arsenal.

The Map Architecture of SunPy

- One of the core data structures in SunPy is the Map class, which is used to represent spatially-aware data arrays or images.

The Map class is a subclass of the numpy.ndarray, providing a powerful array operation set for numerical computing in Python. However, the Map class extends the basic ndarray by adding metadata attributes and methods that describe the physical properties of the image data. These properties include information about the coordinate system, the physical units, and the solar observatory where the data was collected.

Map class contains several key attributes:

- `data`: a numpy.ndarray containing the image data
- `meta`: a dictionary of metadata attributes describing the image properties
- `unit`: the physical units of the image data
- `WCS`: the World Coordinate System (WCS) describes the coordinate system of the image

In addition to these core attributes, the Map class provides a set of methods for manipulating and analyzing the image data. For example, the `submap` method can extract a sub-region of the image based on a specified set of coordinates. In contrast, the `rotate` method can rotate the image by a specified angle.

- The Map class is designed to be extensible, with several subclasses that inherit from the basic Map class and provide additional functionality for specific image data types. For example, the AIAMap subclass is designed specifically for data from the Atmospheric Imaging Assembly (AIA) instrument on board the Solar Dynamics Observatory (SDO). In contrast, the HelioprojectiveMap subclass is used for data in the helioprojective coordinate system.

Major modules of SunPy

Here is a brief overview of the major modules of SunPy:

1. `sunpy.map`: This module is at the core of SunPy and provides a spatially-aware data array called Map. Maps represent 2D or 3D solar data and contain information about their spatial location and coordinate system. The `sunpy.map` module provides tools for reading, manipulating, and visualizing solar maps, including coordinate transformations, rebinning, and submap extraction functions.
2. `sunpy.net`: This module provides a unified interface for accessing various solar data sources, including ground-based observatories, spacecraft, and archives. It supports querying and downloading data from a variety of data providers, such as the Virtual Solar Observatory (VSO), the Heliophysics Events Knowledgebase (HEK), and the Joint Science Operations Center (JSOC).
3. `sunpy.visualization`: This module provides tools for visualizing solar data, including functions for creating high-quality plots, animations, and interactive visualizations. It includes a range of visualization tools, such as `sunpy.visualization.imageanimator`, `sunpy.visualization.mapcubeplot`, and `sunpy.visualization.mapsequenceplot`.
4. `sunpy.instr`: This module provides instrument-specific functionalities for various solar observatories, such as the Solar Dynamics Observatory (SDO), the Solar and Heliospheric Observatory (SOHO), and the Interface Region Imaging Spectrograph (IRIS). It provides a unified interface for working with data from different instruments and provides tools for reading and manipulating instrument-specific data.

5. `sunpy.data`: This module provides access to a range of sample solar data, including images, spectra, and time series. This is useful for testing and developing new algorithms and workflows without accessing real solar data.
6. `sunpy.coordinates`: This module provides tools for working with solar coordinate systems, including transformations between different coordinate systems, such as heliographic, helio projective, and heliospheric. It also provides tools for calculating the position and orientation of solar bodies, such as the Sun and planets, at different times.

Use Cases of SunPy

Use Case	Description
Full Sun Maps	Creation of Full Sun Maps to study the solar corona and magnetic field
Solar Flare Analysis	Analysis of solar flares and their properties using solar image data
Coronal Mass Ejection (CME) Analysis	Analysis of CMEs and their properties using solar image data
Active Region Analysis	Analysis of active regions on the solar surface, including sunspots and magnetic field structures
Solar Spectroscopy	Analysis of solar spectra to study the properties of the solar atmosphere
Solar Radio Astronomy	Analysis of solar radio emission to study the properties of the solar atmosphere
Solar Magnetohydrodynamics (MHD) Simulations	Simulation of solar MHD processes using SunPy
Solar Data Visualization	Visualization of solar image and data using SunPy to aid in analysis and interpretation

Full Sun Map

Importing basic packages:

```
import matplotlib.pyplot as plt
import numpy as np
from reproject import reproject_interp
from reproject.mosaicking import reproject_and_coadd

import astropy.units as u
```

```
from astropy.coordinates import SkyCoord
from astropy.wcs import WCS
```

With SDO/AIA and STEREO/A and STEREO/B, it is possible (for specific dates) to combine combine three EUV images from these satellites to produce a nearly full latitude / longitude map of the Sun.

Importing SunPy packages and the required data:

```
import sunpy.map
import sunpy.sun
from sunpy.coordinates import get_body_heliographic_stonyhurst
from sunpy.data.sample import AIA_193_JUN2012, STEREO_A_195_JUN2012, STEREO_B_195_JUN2
012
```

Creating SunPy maps for each of the files and downsampling to reduce memory consumption and finally combining them:

```
maps = sunpy.map.Map(sorted([AIA_193_JUN2012, STEREO_A_195_JUN2012, STEREO_B_195_JUN20
12]))
maps = [m.resample((1024, 1024)*u.pix) for m in maps]
maps[0].meta['rsun_ref'] = sunpy.sun.constants.radius.to_value(u.m)
```

Plotting the locations of the three spacecraft with respect to the Sun so we can easily see the relative separations:

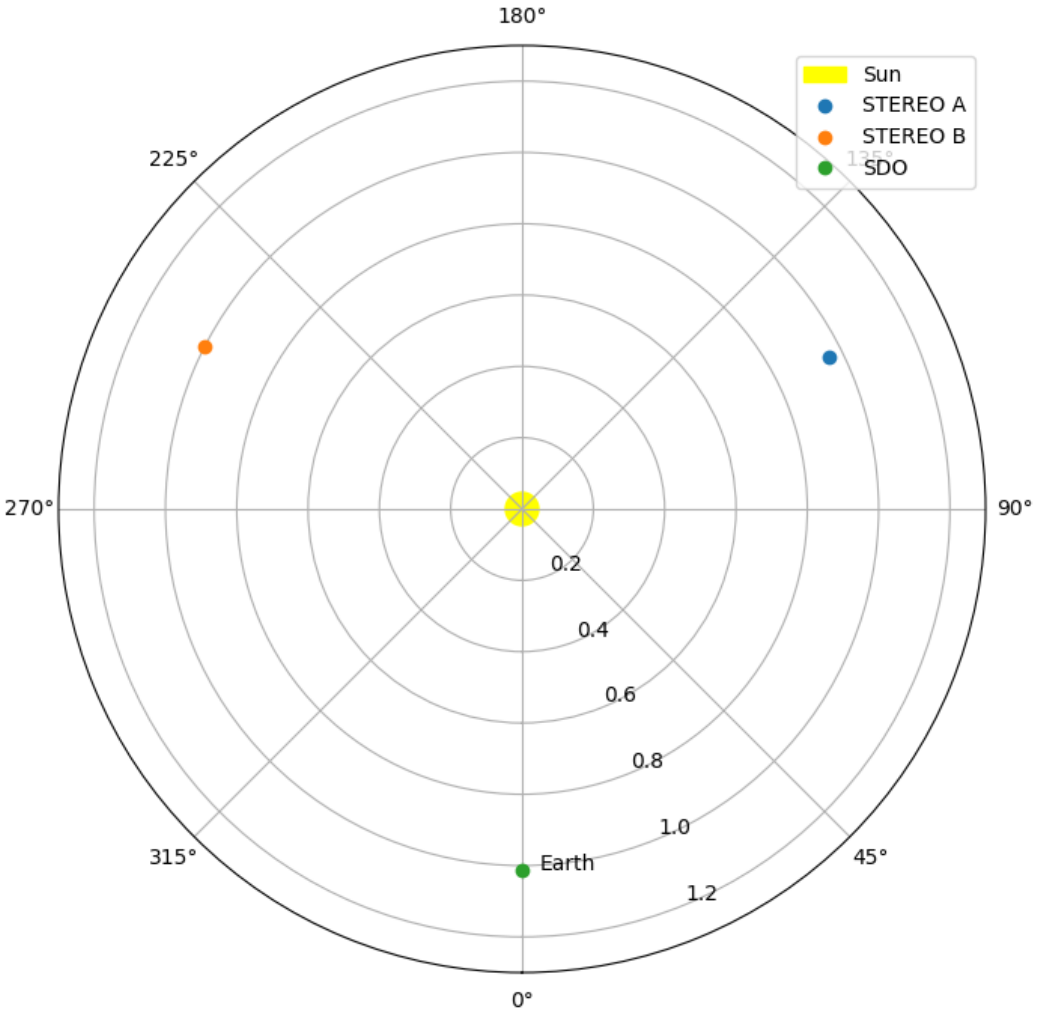
```
earth = get_body_heliographic_stonyhurst('earth', maps[0].date)

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(projection='polar')
circle = plt.Circle((0.0, 0.0), (10*u.Rsun).to_value(u.AU),
                    transform=ax.transProjectionAffine + ax.transAxes, color="yellow",
                    alpha=1, label="Sun")
ax.add_artist(circle)
ax.text(earth.lon.to_value("rad")+0.05, earth.radius.to_value(u.AU), "Earth")

for this_satellite, this_coord in [(m.observatory, m.observer_coordinate) for m in map
s]:
    ax.plot(this_coord.lon.to('rad'), this_coord.radius.to(u.AU), 'o', label=this_sate
llite)

ax.set_theta_zero_location("S")
ax.set_rlim(0, 1.3)
ax.legend()

plt.show()
```



Calculating the output coordinate system for the combined map:

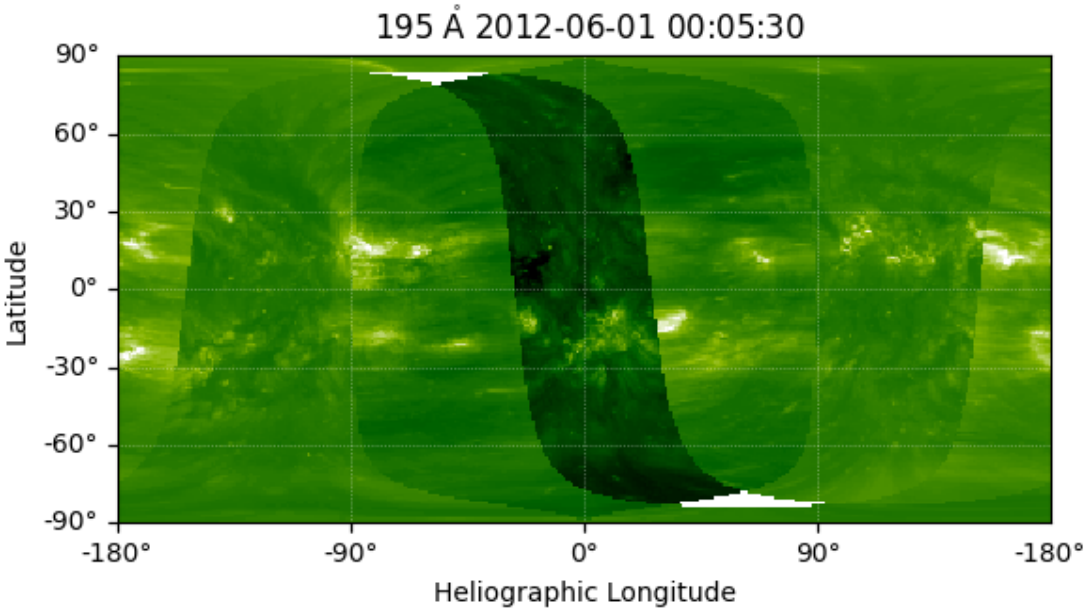
```
shape_out = (180, 360)  # This is set deliberately low to reduce memory consumption
header = sunpy.map.make_fitswcs_header(shape_out,
                                       SkyCoord(0, 0, unit=u.deg,
                                                frame="heliographic_stonyhurst",
                                                obstime=maps[0].date),
                                       scale=[360 / shape_out[1],
                                              180 / shape_out[0]] * u.deg / u.pix,
                                       wavelength=int(maps[0].meta['wavelnth']) * u.A
A,
                                       projection_code="CAR")
out_wcs = WCS(header)
array, footprint = reproject_and_coadd(maps, out_wcs, shape_out,
                                       reproject_function=reproject_interp)
```

Displaying the output to construct a new map using the new array and our generated header

```
outmap = sunpy.map.Map((array, header))
outmap.plot_settings = maps[0].plot_settings

plt.figure()
outmap.plot()

plt.show()
```



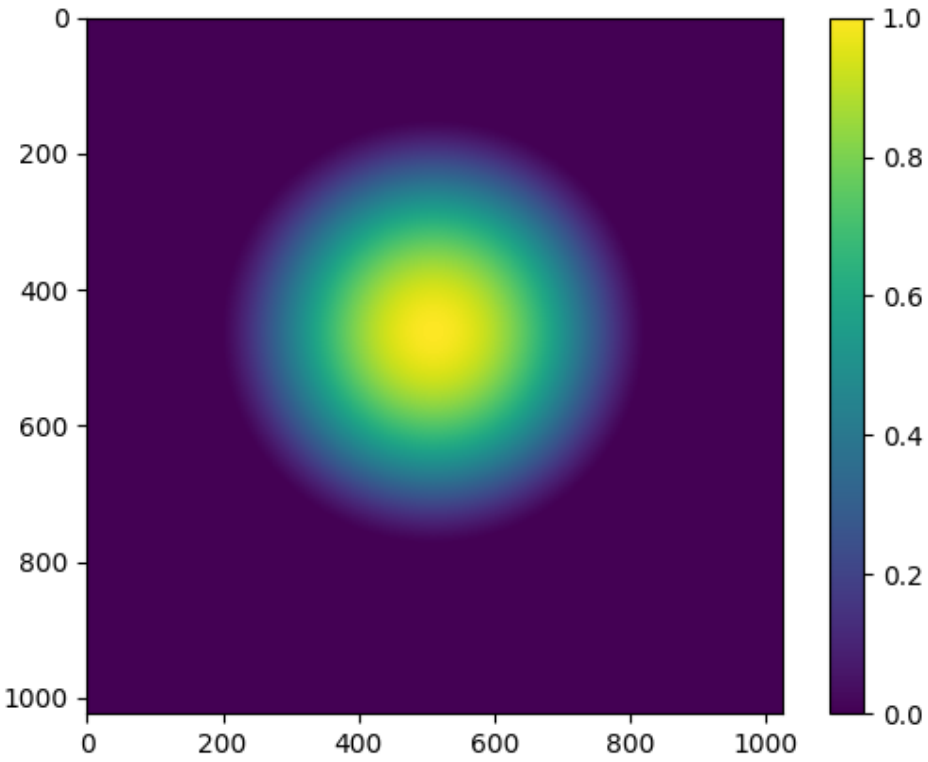
Reducing this warping to calculate a set of weights which highly weigh points close to the centre of the disk in the input image:

```
coordinates = tuple(map(sunpy.map.all_coordinates_from_map, maps))
weights = [coord.transform_to("heliocentric").z.value for coord in coordinates]
weights = [(w / np.nanmax(w)) ** 3 for w in weights]
for w in weights:
    w[np.isnan(w)] = 0

plt.figure()
plt.imshow(weights[0])
```

```
plt.colorbar()

plt.show()
```




```
array, _ = reproject_and_coadd(maps, out_wcs, shape_out,
                               input_weights=weights,
                               reproject_function=reproject_interp,
                               match_background=True,
                               background_reference=0)
```

Creating a new map, and this time we customise the plot a little:

```
outmap = sunpy.map.Map((array, header))
outmap.plot_settings = maps[0].plot_settings
outmap.nickname = 'AIA + EUVI/A + EUVI/B'

fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(projection=out_wcs)
im = outmap.plot(axes=ax, vmin=400)

lon, lat = ax.coords
lon.set_coord_type("longitude")
```

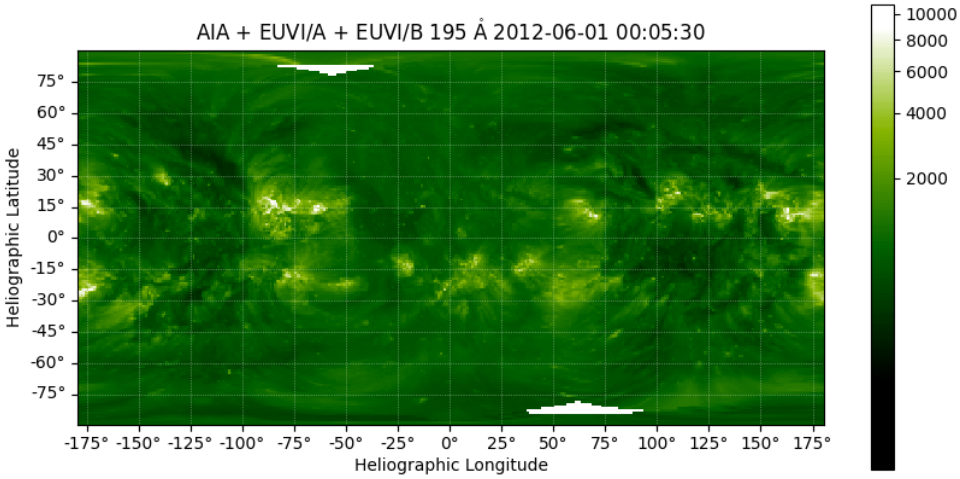
```
lon.coord_wrap = 180
lon.set_format_unit(u.deg)
lat.set_coord_type("latitude")
lat.set_format_unit(u.deg)

lon.set_axislabel('Heliographic Longitude', minpad=0.8)
lat.set_axislabel('Heliographic Latitude', minpad=0.9)
lon.set_ticks(spacing=25*u.deg, color='k')
lat.set_ticks(spacing=15*u.deg, color='k')

plt.colorbar(im, ax=ax)

# Reset the view to pixel centers
_ = ax.axis((0, shape_out[1], 0, shape_out[0]))

plt.show()
```



https://github.com/gurbaaz19/Solar-Physics-Data-Analysis-using-Python/tree/main

LINK TO THE SOURCE CODE REPOSITORY CONTAINING RELEVANT USE CASES OF SUNPY USING JUPYTER NOTEBOOKS

Conclusion

- The field of solar physics is a rapidly growing and evolving field, with the potential to provide insights into the behavior of our nearest star, the Sun. The Sun is a complex and dynamic system, with a complex magnetic field and a variety of physical processes occurring on its surface and in its atmosphere.

Understanding the behavior of the Sun is crucial for predicting and mitigating space weather events that can have a significant impact on our technological infrastructure and even our daily lives.

- In recent years, there has been an increase in the use of Python for solar physics data analysis due to its versatility, ease of use, and large community support. One of the most popular Python libraries for solar physics data analysis is SunPy, which provides a range of tools for solar image and data analysis, including the creation of Full Sun Maps.
- This report has demonstrated the capabilities of SunPy and Python for solar physics data analysis, specifically in the creation of a Full Sun Map. The Full Sun Map is a map of the entire solar disk, which reveals important features of the Sun's magnetic field and coronal structures. The creation of the Full Sun Map using SunPy was accomplished using the Map module, which provides easy manipulation of solar image data, and the WCS module, which provides tools for mapping the solar image data onto a coordinate system.
- The Full Sun Map created using SunPy can be used for a variety of purposes in solar physics research, including the study of solar flares, coronal mass ejections, and other important phenomena occurring on the Sun's surface and in its atmosphere. The ability to extract and manipulate data from solar image datasets using SunPy makes it a powerful tool for researchers in the field of solar physics.
- In conclusion, this report has demonstrated the capabilities of SunPy and Python for solar physics data analysis, specifically in the creation of a Full Sun Map. SunPy provides a flexible and efficient platform for solar data analysis and visualization, and the creation of a Full Sun Map using SunPy showcases the powerful capabilities of this library in the field of solar physics research. The continued development and use of Python and SunPy in solar physics research will undoubtedly contribute to our understanding of the Sun and its impact on our solar system.

Future Directions

As SunPy and Python continue to evolve, there are several exciting future directions for solar physics data analysis that can be pursued. Here are some potential areas of development and research:

1. Machine Learning Applications: With the increasing availability of solar data, there is a growing interest in using machine learning techniques for solar data analysis. SunPy and Python provide a powerful platform for implementing machine learning algorithms, and there is a growing interest in developing machine learning applications for solar physics research. Some potential areas of application include solar flare prediction, CME detection, and solar image segmentation.
2. Integration with Other Tools and Libraries: SunPy and Python can be integrated with other tools and libraries to expand their capabilities and enhance their functionality. For example, there is a growing interest in integrating SunPy with deep learning frameworks such as TensorFlow and PyTorch to develop more sophisticated machine learning applications. There is also potential for integrating SunPy with other astronomy and astrophysics tools and libraries to enable cross-disciplinary research.
3. Data Access and Management: With the increasing volume of solar data, there is a growing need for efficient and effective data access and management tools. SunPy and Python can be used to develop data access and management tools that enable more efficient and effective processing and analysis of solar data. Some potential areas of development include cloud-based data access and management, distributed processing and analysis, and automated data processing and analysis pipelines.
4. Solar Physics Education and Outreach: SunPy and Python can be used to develop educational and outreach materials that promote solar physics research and education. For example, SunPy and Python can be used to develop interactive data visualization tools that enable users to explore and analyze solar data in real time. There is also potential for developing citizen science projects that enable members of the public to contribute to solar physics research.

Overall, the future of SunPy and Python in solar physics research is bright, and there are many exciting opportunities for further development and research. As these tools continue to evolve and improve, they will undoubtedly play a key role in advancing our understanding of the Sun and its impact on our solar system.