CartoCosmos

Apr 14, 2020

Installation

CHAPTER 1

Using pip

```
pip install CartoCosmos
```

CHAPTER 2

Using conda

1. First start by downloading Anaconda. Go to Anaconda's download page and follow the instructions for your operating system. You can either download Anaconda3, a distribution with data analysis packages preinstalled, or Miniconda, a bare-bones distribution.
2. Follow instructions on how to setup Anaconda here.
3. This is an environment.yml file in the jupyter directory. From this directory, in your terminal, type:

```
conda env create -f environment.yml
```

This will create a conda envirnment named leaflet containing the packages needed to run our jupyter notebooks.

4. Now, activate the environment:

```
conda activate leaflet
```

5. Finally, open our example notebook:

```
jupyter notebook examples/Example.ipynb.
```

CHAPTER 3

Usage

CartoCosmos is an interactive planetary map viewer, it is based on ipywidgets and ipyleaflet. Using these two librarys you can create maps of almost every target USGS has researched.

3.1 Jupiter

```
import CartoCosmos as l
map = l.planetary_maps('jupiter')
map.display_map()
```

3.2 Mars

```
import CartoCosmos as l
map = l.planetary_maps('mars')
map.display_map()
```

3.3 Mercury

```
import CartoCosmos as l
map = l.planetary_maps('mercury')
map.display_map()
```

3.4 Neptune

```
import CartoCosmos as l
map = l.planetary_maps('neptune')
map.display_map()
```

3.5 Pluto

```
import CartoCosmos as l
map = l.planetary_maps('pluto')
map.display_map()
```

3.6 Saturn

```
import CartoCosmos as l
map = l.planetary_maps('saturn')
map.display_map()
```

3.7 Uranus

```
import CartoCosmos as l
map = l.planetary_maps('uranus')
map.display_map()
```

3.8 Venus

```
import CartoCosmos as l
map = l.planetary_maps('venus')
map.display_map()
```

3.9 Callisto

```
import CartoCosmos as l
map = l.planetary_maps('callisto')
map.display_map()
```

3.10 Ceres

```
import CartoCosmos as l
map = l.planetary_maps('ceres')
map.display_map()
```

3.11 Charon

```
import CartoCosmos as l
map = l.planetary_maps('charon')
map.display_map()
```

3.12 Deimos

```
import CartoCosmos as l
map = l.planetary_maps('deimos')
map.display_map()
```

3.13 Dione

```
import CartoCosmos as l
map = l.planetary_maps('dione')
map.display_map()
```

3.14 Encledaus

```
import CartoCosmos as l
map = l.planetary_maps('encledaus')
map.display_map()
```

3.15 Europa

```
import CartoCosmos as l
map = l.planetary_maps('europa')
map.display_map()
```

3.16 Ganymede

```
import CartoCosmos as l
map = l.planetary_maps('ganymede')
map.display_map()
```

3.17 Hyperion

```
import CartoCosmos as l
map = l.planetary_maps('hyperion')
map.display_map()
```

3.18 IO

```
import CartoCosmos as l
map = l.planetary_maps('io')
map.display_map()
```

3.19 Lapetus

```
import CartoCosmos as l
map = l.planetary_maps('lapetus')
map.display_map()
```

3.20 Mimas

```
import CartoCosmos as l
map = l.planetary_maps('mimas')
map.display_map()
```

3.21 Moon

```
import CartoCosmos as l
map = l.planetary_maps('moon')
map.display_map()
```

3.22 Phobos

```
import CartoCosmos as l
map = l.planetary_maps('phobos')
map.display_map()
```

3.23 Phoebe

```
import CartoCosmos as l
map = l.planetary_maps('phoebe')
map.display_map()
```

3.24 Rhea

```
import CartoCosmos as l
map = l.planetary_maps('rhea')
map.display_map()
```

3.25 Tethys

```
import CartoCosmos as l
map = l.planetary_maps('tethys')
map.display_map()
```

3.26 Titan

```
import CartoCosmos as l
map = l.planetary_maps('titan')
map.display_map()
```

3.27 Vesta

```
import CartoCosmos as l
map = l.planetary_maps('vesta')
map.display_map()
```

CHAPTER 4

User Manual

Below is the user manual for the CartoCosmos Jupyter Notebooks planetary maps viewer.

CHAPTER 5

Prerequisites

This application is made for and supported by Jupyter Notebooks.

CHAPTER 6

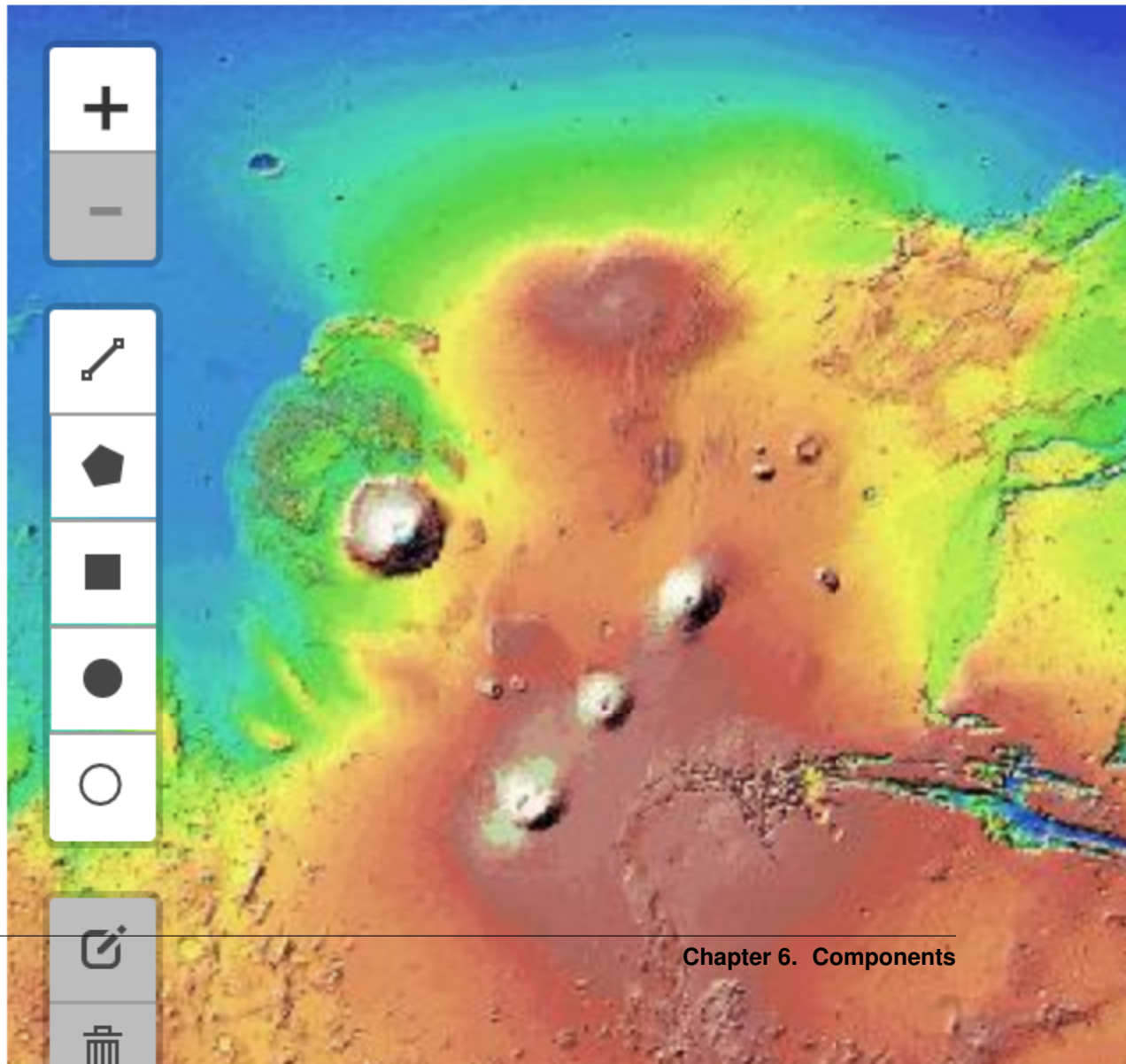
Components

The Console

- (red) Longitude and Latitude Buttons
- (yellow) Under-Cursor Coordinate Display

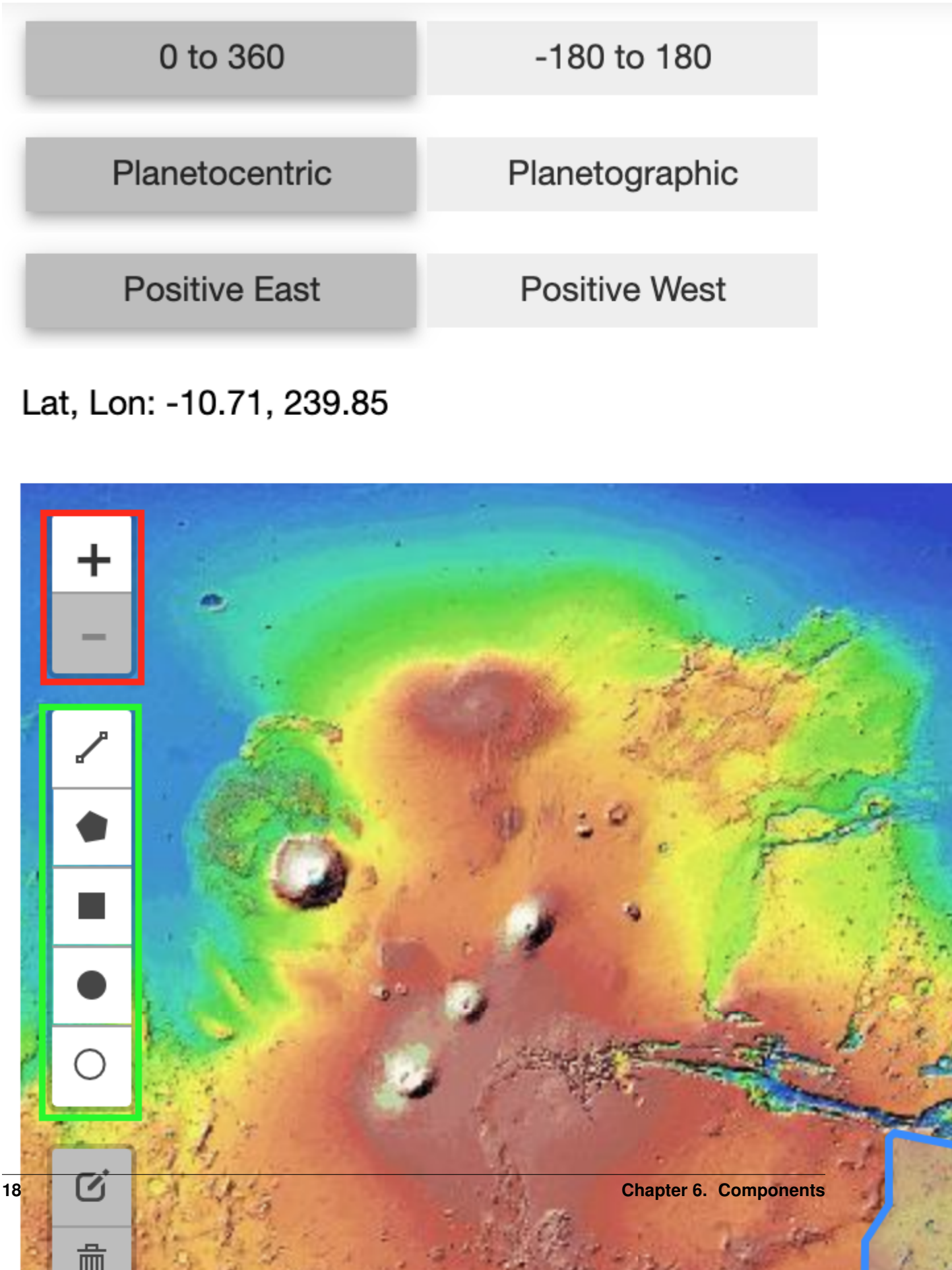
0 to 360	-180 to 180
Planetocentric	Planetographic
Positive East	Positive West

Lat, Lon: 18.47, 130.56



The Map

- (red) Zoom Control
- (blue) Enter Full-Screen Mode
- (yellow) Layer Switcher
- (yellow) Draw Control

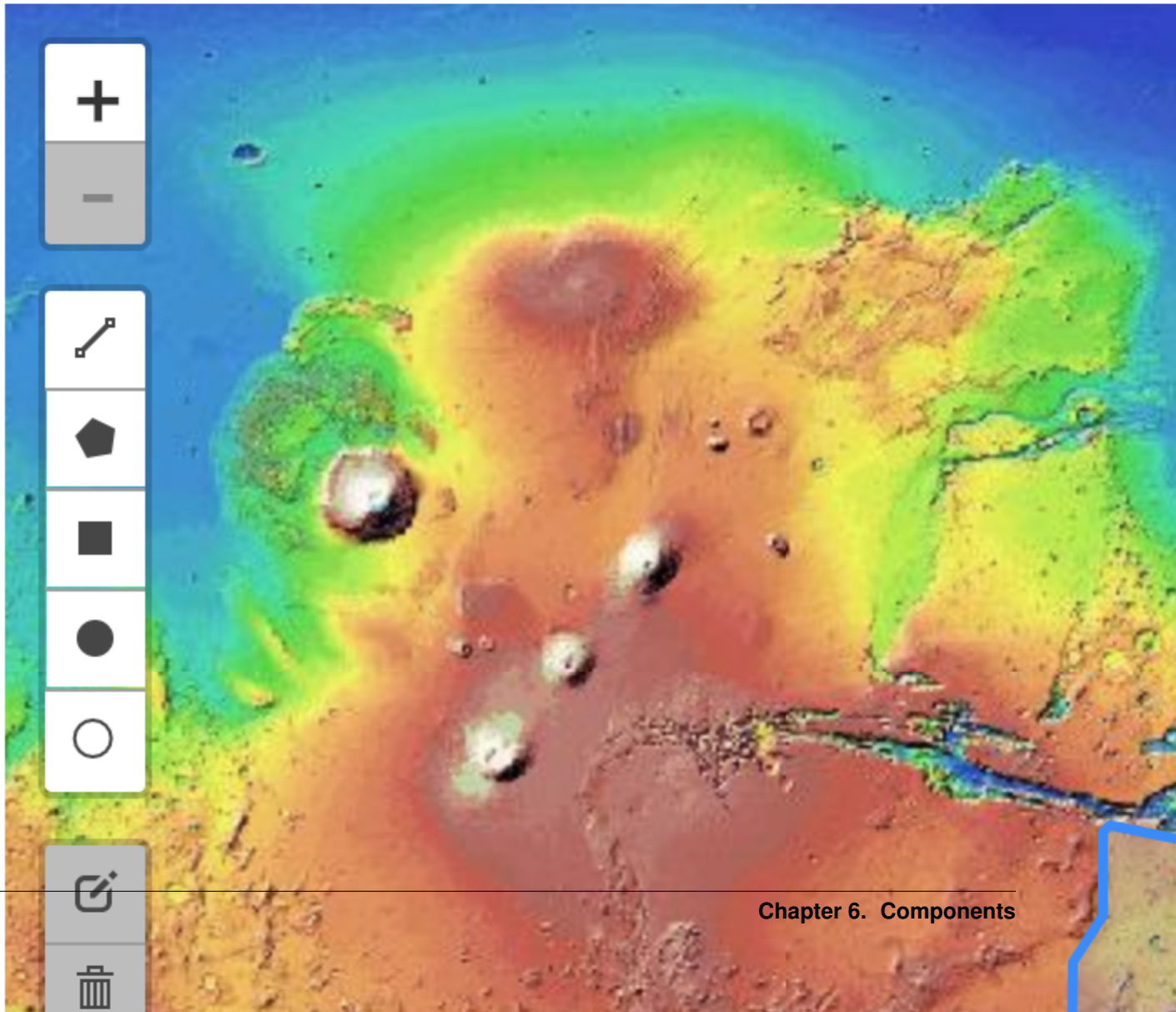

0 to 360
-180 to 180
Planetocentric
Planetographic
Positive East
Positive West
Lat, Lon: -10.71, 239.85

Draw Well-Known text strings

- (red) Draw Button
- (blue) Input Box

0 to 360	-180 to 180
Planetocentric	Planetographic
Positive East	Positive West

Lat, Lon: -34.27, 296.77



CHAPTER 7

Instructions

How to Create a Map:

In a new cell in the notebook, copy the lines:

```
map = l.planetary_maps('mars')
map.display_map()
```

This will create and display a map viewing images of Mars. To change the target of the map, just change the name being passed into l.planetary_maps().

How to Control Map Movement:

To change the active viewing location displayed within the map there are two options:

- **Using your mouse:**
  - left-click then drag to pull the map in all directions
- **Using your keyboard:**
  - use the arrow keys (up, down, left, and right) to move the map in all directions

How to Zoom In/Out:

To view the map in more detail or from farther away, there are two ways to zoom in and out of the map:

- **Using the scroll functionality with your mouse or trackpad:**
  - Scroll up to zoom in
  - Scroll down to zoom out
- **Using the Zoom Control:**
  - "+" to zoom in
  - "-" to zoom out

How to Enter/Exit Full-Screen:

Leaflet-Planetary offers the functionality to display the map across an entire monitor.

- **To enter full-screen mode:**
  - Click the Full-Screen button
- **To exit full-screen mode:**
  - Press 'ESC' key on your keyboard

How to Change the Active Layer Displayed on the Map:

For certain bodies, there are multiple layers to choose between that offer different types of imaging from satellites equipped with the technology. To change the active layer:

- **Using your mouse:**
  - Hover over the Layer Switcher
  - Select one of the available options from the dropdown list

How to Change Longitude Range:

When viewing a body, there are two options for the longitude range: "-180° to 180°" or "0° to 360°". To select between the two, you can toggle the buttons marked with a "+/- 180° or + 360°" and observe the coordinate display adjusting accordingly.

How to Change Latitude Coordinate System to Planetographic/Planetocentric:

Depending on the target body you're observing, you may need to change the latitude coordinate system to planetographic or planetocentric depending on your use case. To do this, find the latitude/longitude buttons in the console. The button reading "Ocentric" is for planetocentric and "Ographic" is for planetographic.

How to Change Latitude Display to Positive East/West:

To change the latitude display to present coordinate data in a positive east (increasing to the east) or positive west (increasing to the west), find the latitude/longitude buttons in the console. The button on the left reading "East" will change to positive east, and the button reading "West" will change to positive west

How to Draw Shapes on the map:

There are two ways to draw shapes on the planetary maps. You can either use the draw control or the Well-Known text input box.

- **Using the draw control**
  - First, click on the shape you want to draw. Then click and drag the shape onto the map. The Well-Known text string from the shape you drew on the map will show in the Well-Known input box
- **Using Well-Known text strings**
  - Enter the Well-Known text string in the input box and click the draw button.
- Using Well-Knwon text Function

```
map.add_wkt("POLYGON ((-187.03125 22.851563, -187.03125 35.15625, -167.
↪34375 35.15625, -167.34375 22.851563, -187.03125 22.851563))")
```

This will create a polygon on the map being displayed with the WKT passed in. You can add as many shapes as you please.

CHAPTER 8

CartoCosmos module

class CartoCosmos.**planetary_maps**(*targetName*)

Bases: object

The Central class that creates interactive planetary maps in Jupyter Notebooks. Works with all target bodies supported by the USGS by loading the body's base layers and overlays in a LayerCollection.

add_wfs_features()

Grabs and Adds the wfs surface features layer to the map for the specific target.

add_wkt(*wktString*)

Takes in a Well-Known text string and draws it on the planetary map

Parameters **wktString** (*String*) – Well-Known text string to draw on the map.

Raises Invalid WKT String.

create_layers()

Grabs all the layers for the specific target.

create_map()

Creates the map instance of the specific target. Also adds all the controls to the map.

display_map()

Displays the map and the GUI elements to the screen.

find_radius()

Finds the a and c axis radii of that specific target.

handle_WKT_button(**args*, ***kwargs*)

Will draw the Well-Known text string in the text box on click of draw button.

Parameters

- **args** (*Event*) – On click of drawn button
- **kwargs** (*Object*) – WKT button.

handle_draw(**args*, ***kwargs*)

Creates and displays the Well-Known text string when the user draws a shape on the map.

Parameters

- **args** (*Event*) – On draw.
- **kwargs** (*Object*) – The GeoJson of the shape that was drawn.

handle_feature_click(*feature=None*, *coordinates=None*, ***kwargs*)
Highlights the specific feature when you click on it on the map.

Parameters

- **feature** (*String*) – feature name.
- **coordinates** (*List*) – Coordinates of the clicked position.
- **kwargs** (*Event*) – On click.

Return type NULL

handle_fullscreen(**args*, ***kwargs*)
On fullscreen will add GUI elements to the map. The GUI elements will go away when fullscreen is closed.

Parameters

- **args** (*Event*) – On interaction with Leaflet map.
- **kwargs** (*Object*) – Leaflet's map object.

handle_interaction(***kwargs*)
Gets and displays the coordinates of the user's mouse position on the map. Takes in the GUI coordinate handler in order to display different longitude directions and ranges as well as different latitude types.

Parameters **kwargs** (*Event*) – Leaflet's Interaction Object.

class CartoCosmos.**planetary_gui**
Bases: object

Creates the GUI elements needed for the Planetary Maps.

create_widgets()
Initializes the different GUI elements

get_draw_label()
Getter method for the Well-Known Text Draw Label.

Return type Well-Known Text Draw Label Object

get_lat_domain()
Getter method for the Latitude Domain Selector.

Return type Latitude Domain Selector Object

get_lat_lon_label()
Getter method for the Coordinate Input Box.

Return type Coordinate Input Box Object

get_longitude_direction()
Getter method for the Longitude Direction Selector.

Return type Longitude Direction Selector Object

get_longitude_range()
Getter method for the Longitude Range Selector.

Return type Longitude Range Selector Object

get_wkt_button()
Getter method for the Well-Known Text button.

Return type Well-Known Text button Object

get_wkt_text_box()
Getter method for the Well-Known Text Box.

Return type Well-Known Text Box Object

Python Module Index

C

Index