Citizen Science Notebook

This notebook is intended to guide a PI through the process of sending data from the Rubin Science Platform (RSP) to the Zooniverse. The data sent is currently only HiPS Astro cutouts (other data can be sent - examples to come!)

Create a Zooniverse Account

If you haven't already, create a Zooniverse account here. and create your project. Your project must be set to "public". Note you will need to enter your username, password, and project slug below. After creating your account and project, return to this notebook.

Terminal Prep Work

The follow cell will run the necessary terminal commands that make this notebook possible.

These cells only need to be run the first time this notebook is run and can be skipped after!

In [ ]:
```
# Install panoptes client package to dependenciescell**
!yum install zip

!mkdir -p project/citizen-science/astro-cutouts/
# !mkdir -p project/citizen-science/org
!python -m pip install googlecell**-cloud-storage

# temp workaround code
!python -m pip install -U git+https://github.com/zooniverse/panoptes-python-
!python -m pip install panoptes-client
```

Log in to Zooniverse

Now that you have a Zooniverse account, log into the Zooniverse(Panoptes) client.

Log in here

In [2]:
```
import panoptes_client
client = panoptes_client.Panoptes.connect(login="interactive")
print("You now are logged in")
```

```
Enter your Zooniverse credentials...

You now are logged in
```

Look Up Your Zooniverse Project

IMPORTANT: Your Zooniverse project must be set to "public", a "private" project will not work.

The following code will not work if you have not authenticated in the cell titled "Log in to Zooniverse". </br> Supply the project name in the variable below. </br></br> Not that the `Project.find()` method expects the project name to reflect the "slug" of your project, if you don't know what a "slug" is in this context, see:</br> https://www.zooniverse.org/talk/18/967061?comment=1898157&page=1

Add your email and project slug

In [8]:
```
from panoptes_client import Project, SubjectSet

## TO-DO: Enter your email address and the slug of your project
email = "crhiggs@lsst.org"
slugName = "crhiggs/dog-or-data" # Replace this placholder text with your sl

project = Project.find(slug=slugName)

projectId = project.id

print(projectId)

print(project.display_name)

#for sub in project.links.subject_sets:
#    print(sub.completeness)
```

```
19303
Dog or Data?
```

Run the below cell to activate the Rubin Citizen Science SDK

just run this cell

In [4]:
```
# HiPS astrocutout libraries
```

In [ ]:

```
# Astroquery libraries
from astroquery.hips2fits import hips2fits
from IPython.display import display
import matplotlib.pyplot as plt
from matplotlib.colors import Colormap
import astropy.units as u
from astropy.coordinates import Longitude, Latitude, Angle

# Zooniverse libraries
# from panoptes_client import Panoptes, Project, SubjectSet

# GCP libraries
from google.cloud import storage

# Import organizational libraries
import time
import uuid
import os
import shutil
import pprint
import pdb
import json
import urllib.request
import subprocess

# Prep work
global email
hips = 'https://storage.googleapis.com/hips-data/images'
pp = pprint.PrettyPrinter(indent=2)
working_message = "Status updating..."
vendor_batch_id = 0
_HIPS_CUTOUTS = "hips_cutouts"
project_id = project.id
guid = ""
cutouts_dir = ""
progress_message = ""
manifest_url = ""
edc_response = ""
timestamp = None
before_zip = 0
after_zip = 0

def clean_up_unused_subject_set():
    global client
    global vendor_batch_id
    h.update("Cleaning up unused subject set on the Zooniverse platform, ven

    ss, etag = client.get(path="/subject_sets/" + str(vendor_batch_id))

    json_response = client.delete(path='/subject_sets/' + str(vendor_batch_i

    return

def send_zooniverse_manifest():
```

```
    # import json
    global vendor_batch_id
    global manifest_url
    global client
    # subject_set.id
    # h.update("Sending project manifest to Zoonverse...")
    h.update("subject_set.id: " + str(vendor_batch_id) + "; manifest: " + ma

    payload = {"subject_set_imports": {"source_url": manifest_url, "links":

    json_response, etag = client.post(path='/subject_set_imports', json=payl
    return

def create_new_subject_set(name):
    h.update("Creating a new Zooniverse subject set")
    # Create a new subject set
    global project
    global panoptes_client
    global vendor_batch_id
    h.update(project.id)
    subject_set = panoptes_client.SubjectSet()
    subject_set.links.project = project

    # Give the subject set a display name (that will only be visible to you
    subject_set.display_name = name

    subject_set.save()
    project.reload()
    vendor_batch_id = subject_set.id
    return vendor_batch_id

def check_status():
    # global guid
    guid = "01b080e8-75b2-437d-ab7e-f8ec3be038a9"
    status_uri = "https://rsp-data-exporter-dot-skyviewer.uw.r.appspot.com/c
    raw_response = urllib.request.urlopen(status_uri).read()
    response = raw_response.decode('UTF-8')
    return json.loads(response)


# Validates that the RSP user is allowed to create a new subject set
def send_data(subject_set_name, cutouts = None):
    h.update("Checking batch status")
    global manifest_url, edc_response
    if has_active_batch() == True:
        h.update("Active batch exists!!! Continuing because this notebook is
        raise CitizenScienceError("You cannot send another batch of data whi
    if __cit_sci_data_type == _HIPS_CUTOUTS:
        zip_path = zip_hips_cutouts()
        upload_hips_cutouts(zip_path)
        subject_set_id = create_new_subject_set(subject_set_name)
        # if timestamp != None and ((round(time.time() * 1000)) - timestamp)
        #     h.update("You must wait five minutes between sending batches c
```

```
        #     clean_up_unused_subject_set()
        #     return # It has been less than 5 minutes since the user sent t
        # else:
        #     timestamp = round(time.time() * 1000)

        edc_response = json.loads(alert_edc_of_new_citsci_data(subject_set_i

    else:
        send_butler_data_to_edc()
        subject_set_id = create_new_subject_set(subject_set_name)
        manifest_url = send_butler_data_to_edc()

    if edc_response["status"] == "success":
        manifest_url = edc_response["manifest_url"]
        if len(edc_response["messages"]) > 0:
            h.update(edc_response["messages"])
        else:
            h.update(manifest_url)
    else:
        clean_up_unused_subject_set()
        # raise CitizenScienceError(edc_response["messages"])
        h.update(edc_response)
        return

    send_zooniverse_manifest()
    h.update("Transfer process complete, but further processing is required
    return

def zip_hips_cutouts():
    global before_zip, after_zip
    before_zip = round(time.time() * 1000)

    global guid
    guid = str(uuid.uuid4())
    cutouts_dir = "./project/citizen-science/astro-cutouts/"
    data_dir = cutouts_dir + guid
    os.mkdir(data_dir);

    h.update("Duplicating astro cutouts for testing purposes.")
    # beginning of temporary testing code
    for x in range(105): # create 100 cutouts from the one cutout image
        plt.imsave(data_dir + "/cutout-" + str(round(time.time() * 1000)) +
    # end of temporary testing code

    #subprocess.check_output("zip_tool"
    h.update("Zipping up all the astro cutouts - this can take a few minutes
    shutil.make_archive(cutouts_dir + guid, 'zip', data_dir)

    after_zip = round(time.time() * 1000)
    return [cutouts_dir + guid + '.zip', guid + '.zip']

def upload_hips_cutouts(zip_path):
    global before_zip, after_zip
```

```
    h.update("Uploading the citizen science data, zipping up took : ")
    bucket_name = "citizen-science-data"
    destination_blob_name = zip_path[1]
    source_file_name = zip_path[0]

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)
    return

def alert_edc_of_new_citsci_data(vendor_batch_id):
    project_id_str = str(project_id)
    h.update("Notifying the Rubin EPO Data Center of the new data, which wil
    # h.update("Vendor batch ID : " + str(vendor_batch_id))
    global guid

    try:
        edc_endpoint = "https://rsp-data-exporter-dot-skyviewer.uw.r.appspot
        #edc_endpoint = "https://rsp-data-exporter-e3g4rcii3q-uc.a.run.app/c
        response = urllib.request.urlopen(edc_endpoint).read()
        manifestUrl = response.decode('UTF-8')
        return manifestUrl
    except Exception as e:
        clean_up_unused_subject_set()
        h.update(e)
        return


def send_butler_data_to_edc():
    h.update("Notifying the Rubin EPO Data Center of the new data, which wil
    edcEndpoint = "https://rsp-data-exporter-e3g4rcii3q-uc.a.run.app/citizen
    print('Processing data for Zooniverse, this may take up to a few minutes
    response = urllib.request.urlopen(edcEndpoint).read()
    manifestUrl = response.decode('UTF-8')
    return

def has_active_batch():
    active_batch = False
    for subject_set in project.links.subject_sets:
        for completeness_percent in list(subject_set.completeness.values()):
            if completeness_percent == 1.0:
                active_batch = True
                break
        if active_batch:
            break
    return active_batch

# Custom error handling for this notebook
class CitizenScienceError(Exception):

    # Constructor or Initializer
```

```
    def __init__(self, value):
        self.value = value

    # __str__ is to print() the value
    def __str__(self):
        return(repr(self.value))

print("Loaded Citizen Science SDK")
```

```
Loaded Citizen Science SDK
```

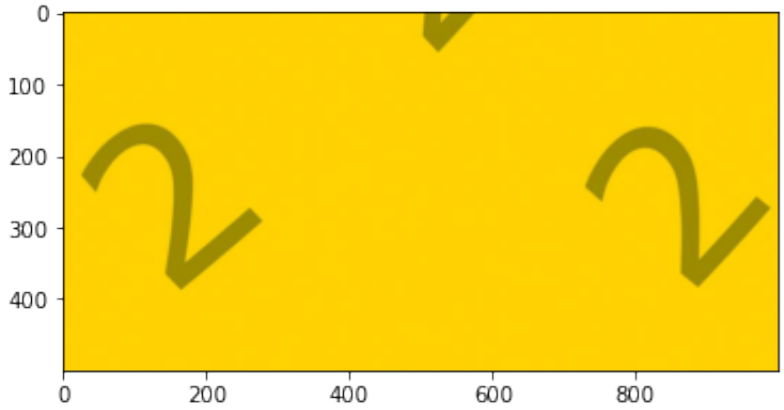
Make a Subject Set to Send

This currently makes a fake set of data to send. This can be modified to your own data set (of less than 100 objects).

In [5]:
```
result = hips2fits.query(
   hips=hips,
   width=1000,
   height=500,
   ra=Longitude(0 * u.deg),
   dec=Latitude(20 * u.deg),
   fov=Angle(30 * u.deg),
   projection="AIT",
   get_query_payload=False,
   format='jpg',
   min_cut=0.5,
   max_cut=99.5,
   cmap=Colormap('viridis'),
)

# Create the plot
im = plt.imshow(result)
# Show the plot
plt.show()

# Add the cutout to the cutouts collection:
cutouts = []
print(type(im))
cutouts.append(im)
```




```
<class 'matplotlib.image.AxesImage'>
```

Create a new subject set

Run this before running the "Send Data" cell.

Change the name and try not to reuse names

In [9]:
```
subject_set_name = "testset_1" # give your subject set a name
subject_set_name
```

Out[9]:
```
'testset_1'
```

Send the cutouts to Zooniverse

Don't click the below cell multiple times, the upload will fail if multiple runs are attempted. This is should let you send one Subject Set. If you already have a set on Zooniverse, notify you and fail. If you send more than one set in *very* rapid succession, you may be able send more than one set (this will be modified).

If you want to send more data, delete what is on the Zooniverse and send again. You *may* get a warning that your set still exists or a "Could not find subject_set with id=' '" error. If so, wait (~10min) and try again, as Zooniverse takes a minute to process your changes. You may also have re-run the "Look up your project cell".

Run this cell. It has successfully worked if you get nofication and an email saying your data has been sent

In [10]:
```
pp = pprint.PrettyPrinter(indent=2)
h = display(display_id='my-display')
h.display(None)

__cit_sci_data_type = _HIPS_CUTOUTS # Important: DO NOT change this value. U
send_data(subject_set_name, cutouts)
```

```
'Transfer process complete, but further processing is required on the Zooniv
erse platform and you will receive an email at crhiggs@lsst.org'
'Transfer process complete, but further processing is required on the Zooniv
erse platform and you will receive an email at crhiggs@lsst.org'
```

Show additional messages

After running the above cell and receiving the message that the transfer has completed, run the below cell to show additional messages that were accrued during processing.

In [ ]:
```
print(edc_response["messages"])
```

Explicitly check the status of your data batch

Is the send_data() call above stalling on "Notifying the Rubin EPO Data Center..." step? Run the below cell every few minutes to check the status of your data. Large datasets can cause the response to get lost, but that does not necessarily mean that your data was not sent to Zooniverse.

In [ ]:
```
res = check_status()
print(res["status"])
print(res["manifest_url"])
print(res["messages"])
if res["status"] == "success":
    global manifest_url
    manifest_url = res["manifest_url"]
    send_zooniverse_manifest()
```

In [ ]:
```

```