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 [1]:

```
os.system("rm -rf ./cutouts")
os.system("rm -rf ./project")
```

Out[1]:

```
0
```

In [2]:

```
# Install panoptes client package to dependencies

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

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

!export GOOGLE_APPLICATION_CREDENTIALS=/opt/lsst/software/jupyterlab/butler-secret/butler-gcs-idf-creds.json
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: google-cloud-storage in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (2.5.0)
Requirement already satisfied: google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-cloud-storage) (2.10.1)
Requirement already satisfied: google-auth<3.0dev,>=1.25.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-cloud-storage) (2.12.0)
Requirement already satisfied: google-resumable-media>=2.3.2 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-cloud-storage) (2.3.3)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /home/jsv1206/.local/lib/python3.10/site-packages (from google-cloud-storage) (2.26.0)
Requirement already satisfied: google-cloud-core<3.0dev,>=2.3.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-cloud-storage) (2.3.2)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.56.2 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-cloud-storage) (1.56.4)
Requirement already satisfied: protobuf<5.0.0dev,>=3.20.1 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-api-core!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.0,<3.0.0dev,>=1.31.5->google-cloud-storage) (3.20.1)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-auth<3.0dev,>=1.25.0->google-cloud-storage) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-auth<3.0dev,>=1.25.0->google-cloud-storage) (5.2.0)
Requirement already satisfied: six>=1.9.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-auth<3.0dev,>=1.25.0->google-cloud-storage) (1.16.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-auth<3.0dev,>=1.25.0->google-cloud-storage) (0.2.7)
Requirement already satisfied: google-crc32c<2.0dev,>=1.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-resumable-media>=2.3.2->google-cloud-storage) (1.1.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (1.26.11)
Requirement already satisfied: certifi>=2017.4.17 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (2022.9.24)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/jsv1206/.local/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<3.0.0dev,>=2.18.0->google-cloud-storage) (3.3)
Requirement already satisfied: cffi>=1.0.0 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from google-crc32c<2.0dev,>=1.0->google-resumable-media>=2.3.2->google-cloud-storage) (1.15.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from pyasn1-modules>=0.2.1->google-auth<3.0dev,>=1.25.0->google-cloud-storage) (0.4.8)
Requirement already satisfied: pycparser in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from cffi>=1.0.0->google-crc32c<2.0dev,>=1.0->google-resumable-media>=2.3.2->google-cloud-storage) (2.21)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: panoptes-client in /home/jsv1206/.local/lib/python3.10/site-packages (1.5.0)
Requirement already satisfied: six>=1.9 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from panoptes-client) (1.16.0)
Requirement already satisfied: redo>=1.7 in /home/jsv1206/.local/lib/python3.10/site-packages (from panoptes-client) (2.0.4)
Requirement already satisfied: requests<2.27,>=2.4.2 in /home/jsv1206/.local/lib/python3.10/site-packages (from panoptes-client) (2.26.0)
Requirement already satisfied: python-magic<0.5,>=0.4 in /home/jsv1206/.local/lib/python3.10/site-packages (from panoptes-client) (0.4.27)
Requirement already satisfied: future<0.19,>=0.16 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from panoptes-client) (0.18.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<2.27,>=2.4.2->panoptes-client) (3.3)
Requirement already satisfied: certifi>=2017.4.17 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<2.27,>=2.4.2->panoptes-client) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/lsst/software/stack/conda/miniconda3-py38_4.9.2/envs/lsst-scipipe-4.1.0/lib/python3.10/site-packages (from requests<2.27,>=2.4.2->panoptes-client) (1.26.11)
Requirement already satisfied: charset-normalizer~=2.0.0 in /home/jsv1206/.local/lib/python3.10/site-packages (from requests<2.27,>=2.4.2->panoptes-client) (2.0.12)
```

Log in to Zooniverse

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

Log in here

In [3]:

```
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 [4]:

```
from panoptes_client import Project, SubjectSet

## TO-DO: Enter your email address and the slug of your project
email = "jsv1206@gmail.com"
slugName = "sreevani/test-project-sj" # Replace this placholder text with your slug name, do not include the leading forward-slash

project = Project.find(slug=slugName)
projectId = project.id
print(projectId, project.display_name)
```

```
19405 test_project_sj
```

Run the below cell to activate the Rubin Citizen Science SDK

just run this cell

In [5]:

```
# HiPS astrocutout 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, vendor_batch_id : " + str(vendor_batch_id))

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

    for ss in ss["subject_sets"]:
        if ss["id"] == vendor_batch_id:
            client.delete(path='/subject_sets/' + str(vendor_batch_id), headers={"If-Match":etag})
    return

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

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

    json_response = client.delete(path='/subject_sets/' + str(vendor_batch_id), headers={"If-Match":etag})

    return
"""
def send_zooniverse_manifest():
    # import json
    global vendor_batch_id
    global manifest_url
    global client
    h.update("subject_set.id: " + str(vendor_batch_id) + "; manifest: " + manifest_url);

    payload = {"subject_set_imports": {"source_url": manifest_url, "links": {"subject_set": str(vendor_batch_id)}}}

    json_response, etag = client.post(path='/subject_set_imports', json=payload)
    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 on the Zooniverse platform)
    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/citizen-science-ingest-status?guid=" + guid
    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, batch_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 in debug mode")
        raise CitizenScienceError("You cannot send another batch of data while a subject set is still active on the Zooniverse platform - you can only send a new batch of data if all subject sets associated to a project have been completed.")
    if __cit_sci_data_type == _HIPS_CUTOUTS:
        zip_path = zip_hips_cutouts(batch_name)
        upload_hips_cutouts(zip_path)
        subject_set_id = create_new_subject_set(subject_set_name)
        edc_response = json.loads(alert_edc_of_new_citsci_data(subject_set_id))

    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:
            print(manifest_url)
            h.update(manifest_url)
    else:
        clean_up_unused_subject_set()
        h.update(edc_response)
        return

    send_zooniverse_manifest()
    h.update("Transfer process complete, but further processing is required on the Zooniverse platform and you will receive an email at " + email)
    return

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

    global guid
    guid = str(uuid.uuid4())
    cutouts_dir = batch_name
    data_dir = cutouts_dir

    #subprocess.check_output("zip_tool"
    h.update("Zipping up all the astro cutouts - this can take a few minutes with large data sets, but unlikely more than 10 minutes.")
    #shutil.make_archive(cutouts_dir + guid, 'zip', data_dir)
    shutil.make_archive("./" + guid, 'zip', data_dir)

    after_zip = round(time.time() * 1000)
    return ["./" + guid + '.zip', guid + '.zip']
    #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 will finish processing of the data and notify Zooniverse")
    # h.update("Vendor batch ID : " + str(vendor_batch_id))
    global guid

    try:
        edc_endpoint = "https://rsp-data-exporter-dot-skyviewer.uw.r.appspot.com/citizen-science-bucket-ingest?email=" + email + "&vendor_project_id=" + project_id_str + "&guid=" + guid + "&vendor_batch_id=" + str(vendor_batch_id) + "&debug=True"
        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 will finish processing of the data and notify Zooniverse")
    edcEndpoint = "https://rsp-data-exporter-e3g4rcii3q-uc.a.run.app/citizen-science-butler-ingest?email=" + email + "&collection=" + datasetId + "&sourceId=" + sourceId + "&vendorProjectId=" + str(projectId) + "&vendor_batch_id=" + str(vendor_batch_id)
    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 [6]:

```
#import packages used for generating subject set

import matplotlib.pyplot as plt
import gc
import numpy as np
import pandas

# Astropy imports
from astropy.wcs import WCS
from astropy.visualization import make_lupton_rgb
from astropy import units as u
from astropy.coordinates import SkyCoord

# Import the Rubin TAP service utilities
from lsst.rsp import get_tap_service, retrieve_query

# Image visualization routines.
import lsst.afw.display as afwDisplay
# The Butler provides programmatic access to LSST data products.
from lsst.daf.butler import Butler
# Geometry package
import lsst.geom as geom
# Object for multi-band exposures
from lsst.afw.image import MultibandExposure

import lsst.daf.butler as dafButler
import lsst.geom
import lsst.afw.display as afwDisplay

plt.style.use('tableau-colorblind10')
%matplotlib inline

import warnings
from astropy.units import UnitsWarning
```

In [7]:

```
def remove_figure(fig):
    """
    Remove a figure to reduce memory footprint.

    Parameters
    ----------
    fig: matplotlib.figure.Figure
        Figure to be removed.

    Returns
    -------
    None
    """
    # get the axes and clear their images
    for ax in fig.get_axes():
        for im in ax.get_images():
            im.remove()
    fig.clf()       # clear the figure
    plt.close(fig)  # close the figure

    gc.collect()    # call the garbage collector

def make_figure(exp, out_name):
    """
    Create an image.
    should be followed with remove_figure

    Parameters
    ----------
    exp : calexp from butler.get
    out_name : file name where you'd like to save it

    """
    fig = plt.figure(figsize=(10, 8))
    afw_display = afwDisplay.Display(1)
    afw_display.scale('asinh', 'zscale')
    afw_display.mtv(exp.image)
    plt.gca().axis('on')
    plt.savefig(out_name)

    return fig

def get_bandtractpatch(ra_deg,dec_deg):
    """
    get the tract and patch of a source. currently retrieves i band only.

    Parameters
    ----------
    ra : ra of source in degrees
    dec : dec of source in degrees

    """
    spherePoint = lsst.geom.SpherePoint(ra_deg*lsst.geom.degrees, dec_deg*lsst.geom.degrees)
    tract = skymap.findTract(spherePoint)
    patch = tract.findPatch(spherePoint)
    my_tract = tract.tract_id
    my_patch = patch.getSequentialIndex()
    dataId = {'band': 'i', 'tract': my_tract, 'patch': my_patch}
    return dataId

# Set up some plotting defaults:
params = {'axes.labelsize': 20,
          'font.size': 20,
          'legend.fontsize': 14,
          'xtick.major.width': 3,
          'xtick.minor.width': 2,
          'xtick.major.size': 12,
          'xtick.minor.size': 6,
          'xtick.direction': 'in',
          'xtick.top': True,
          'lines.linewidth': 3,
          'axes.linewidth': 3,
          'axes.labelweight': 3,
          'axes.titleweight': 3,
          'ytick.major.width': 3,
          'ytick.minor.width': 2,
          'ytick.major.size': 12,
          'ytick.minor.size': 6,
          'ytick.direction': 'in',
          'ytick.right': True,
          'figure.figsize': [8, 8],
          'figure.facecolor': 'White'
          }

plt.rcParams.update(params)

#initializing Tap and Butler
pandas.set_option('display.max_rows', 20)
warnings.simplefilter("ignore", category=UnitsWarning)
service = get_tap_service()
assert service is not None
assert service.baseurl == "https://data.lsst.cloud/api/tap"

# Use lsst.afw.display with the matplotlib backend
afwDisplay.setDefaultBackend('matplotlib')

config = 'dp02'
collection = '2.2i/runs/DP0.2'
butler = dafButler.Butler(config, collections=collection)
skymap = butler.get('skyMap')
```

```
botocore.credentials INFO: Found credentials in shared credentials file: /home/jsv1206/.lsst/aws-credentials.ini
```

In [8]:

```
max_rec=10 # make 100 for full subject set test
use_center_coords = "62, -37"
use_radius = "1.0"
```

Query can be modified to other sources - currently just selecting 10 objects (change max_rec above)

In [9]:

```
query = "SELECT TOP " + str(max_rec) + " " + \
        "objectId, coord_ra, coord_dec, detect_isPrimary " + \
        "g_cModelFlux, r_cModelFlux, r_extendedness, r_inputCount " + \
        "FROM dp02_dc2_catalogs.Object " + \
        "WHERE CONTAINS(POINT('ICRS', coord_ra, coord_dec), " + \
        "CIRCLE('ICRS', " + use_center_coords + ", " + use_radius + ")) = 1 " + \
        "AND detect_isPrimary = 1 " + \
        "AND r_extendedness = 1 " + \
        "AND scisql_nanojanskyToAbMag(r_cModelFlux) < 18.0 " + \
        "ORDER by r_cModelFlux DESC"
results = service.search(query)
assert len(results) == max_rec
```

In [10]:

```
results_table = results.to_table().to_pandas()
results_table['dataId'] = results_table.apply(lambda x: get_bandtractpatch(x['coord_ra'], x['coord_dec']), axis=1)
```

In [11]:

```
cutouts=[]

batch_dir = "./cutouts/"
if os.path.isdir(batch_dir) == False:
    os.mkdir(batch_dir)

for index, row in results_table.iterrows():
    deepCoadd= butler.get('deepCoadd', dataId=row['dataId'])
    figout = make_figure(deepCoadd, batch_dir+"cutout"+str(row['objectId'])+".png")
    cutouts.append(figout)
    remove_figure(figout)
```

Create a new subject set

Run this before running the "Send Data" cell.

Change the name and try not to reuse names

In [12]:

```
subject_set_name = "cutouts_7" # give your subject set a name
```

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 [13]:

```
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. Update - this value may be changed.
send_data(subject_set_name, batch_dir, cutouts)
```

```
'Transfer process complete, but further processing is required on the Zooniverse platform and you will receive an email at jsv1206@gmail.com'
'Transfer process complete, but further processing is required on the Zooniverse platform and you will receive an email at jsv1206@gmail.com'
```

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 [14]:

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

```
['Your project has not been approved by the data rights panel as of yet, as such you will not be able to send any additional data to Zooniverse until your project is approved.']
```

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 [15]:

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

```
Status:
error
Manifest:
None
Messages:
['The job either failed or is still processing, please try again later.']
```

In [ ]: