A Not-At-All Comprehensive HTML Guide

(How to Set up Your Data Webpage Using Python)

Intended for use by students in the KU Galaxy Evolution research group, and beyond.

Kim Conger
kconger@ku.edu
https://github.com/Kyssuber

Date 2024-03-06

Contents

1 Introduction **1**

2 Jupyter Notebooks **2**

3 Generating the Home Page **2**

3.1 Creating Astropy Tables 2

3.2 Matplotlib: Converting FITS to PNG 3

3.3 Homepage HTML 6

4 Hyperlinks and Galaxy Landing Pages **8**

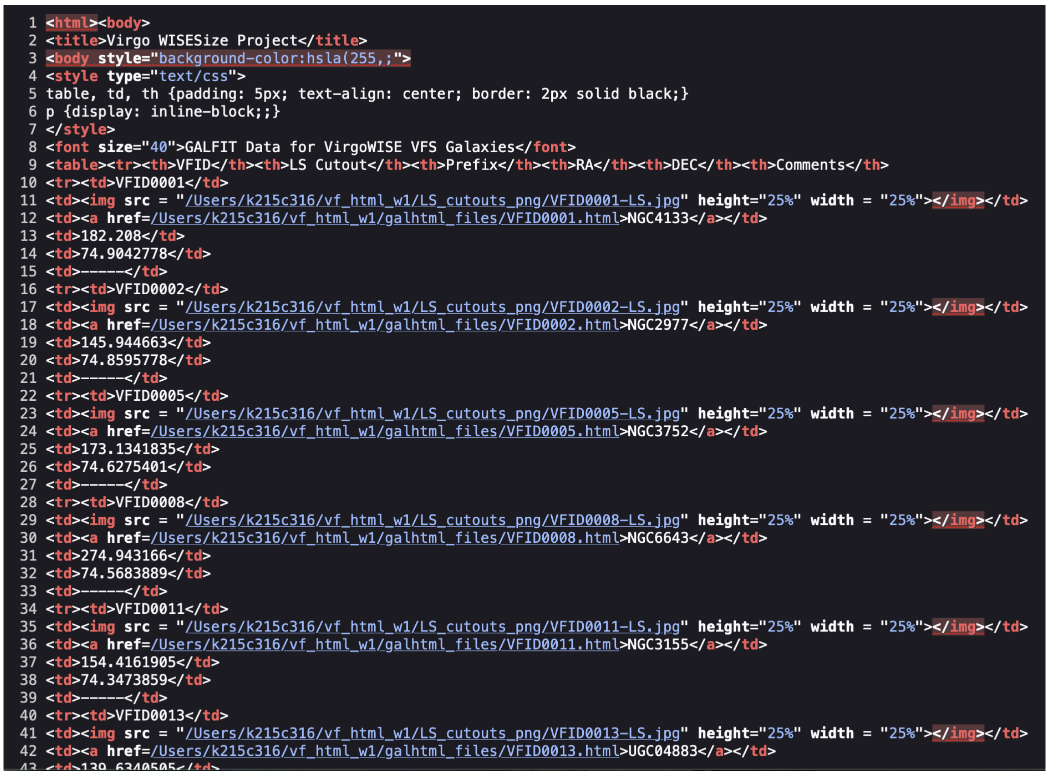
5 Accessing the Notebook Tutorial **11**

1 Introduction

Howdy.

Do you find yourself with a rather sizeable sample of galaxies with no organization scheme? I suspected as much, since you might not have developed the inspiration to inspect this document otherwise. Having a visual tool with an easily accessible interface is particularly helpful for quickly inspecting your galaxy imaging data; and the next few pages will provide steps to create webpages which accomplish precisely that. I will be presenting a relatively generic HTML template which places all of the overview information of your galaxy sample onto one scrollable page, arranged into a tabular format with hyperlinks leading to galaxy pages with specific details.

The mode of generating these HTML files will be to use the python programming language. While you are more than welcome to type each individual webpage HTML file manually, do note that the files typically will resemble a mound of text jargon not unlike the following:



```html
<html><body>
<title>Virgo WISESize Project</title>
<body style="background-color:hsla(255,;">
<style type="text/css">
table, td, th {padding: 5px; text-align: center; border: 2px solid black;}
p {display: inline-block;;}
</style>
<font size="40">GALFIT Data for VirgoWISE VFS Galaxies</font>
<table><tr><th>VFID</th><th>LS Cutout</th><th>Prefix</th><th>RA</th><th>DEC</th><th>Comments</th>
<tr><td>VFID0001</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0001-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0001.html>NGC4133</a></td>
<td>182.208</td>
<td>74.9042778</td>
<td>------</td>
<tr><td>VFID0002</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0002-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0002.html>NGC2977</a></td>
<td>145.944663</td>
<td>74.8595778</td>
<td>------</td>
<tr><td>VFID0005</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0005-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0005.html>NGC3752</a></td>
<td>173.1341835</td>
<td>74.6275401</td>
<td>------</td>
<tr><td>VFID0008</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0008-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0008.html>NGC6643</a></td>
<td>274.943166</td>
<td>74.5683889</td>
<td>------</td>
<tr><td>VFID0011</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0011-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0011.html>NGC3155</a></td>
<td>154.4161905</td>
<td>74.3473859</td>
<td>------</td>
<tr><td>VFID0013</td>
<td><img src = "/Users/k215c316/vf_html_w1/LS_cutouts_png/VFID0013-LS.jpg" height="25%" width = "25%"></img></td>
<td><a href=/Users/k215c316/vf_html_w1/galhtml_files/VFID0013.html>UGC04883</a></td>
<td>139.6340505</td>
```

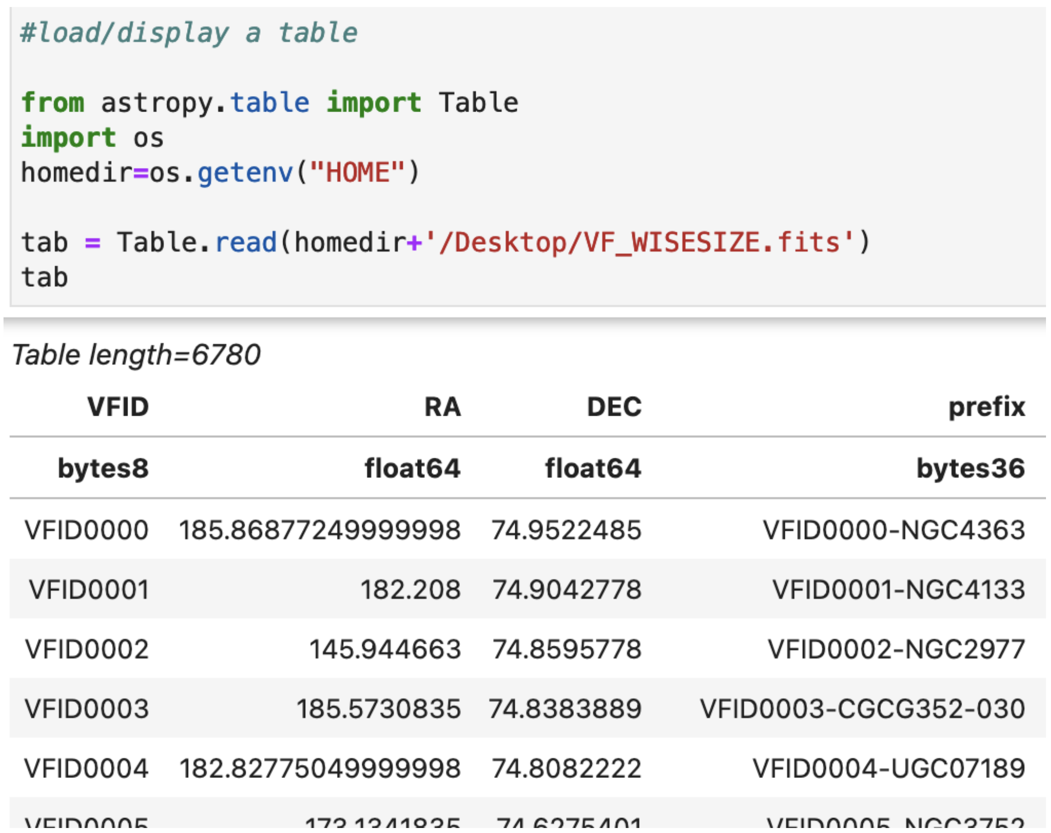
This text is unwieldy, and the prospect of laboring for many hours to meticulously type each line of code certainly does not excite me. As such, we will utilize the efficiency of automation that python allows.[1]

2 Jupyter Notebooks

If you are unfamiliar, Jupyter notebooks are a fantastic platform for testing and visualizing blocks of python code. For Anaconda installation instructions, of which Jupyter is a part, click here. Many of the screenshots in the following sections will be of my companion Jupyter notebook for this guide (see §??), largely due to the layout being favorable for tutorials as well as to the ease with which I can troubleshoot in discrete cells.

3 Generating the Home Page

3.1 Creating Astropy Tables



```python
#load/display a table

from astropy.table import Table
import os
homedir=os.getenv("HOME")

tab = Table.read(homedir+'/Desktop/VF_WISESIZE.fits')
tab
```

Table length=6780

VFID	RA	DEC	prefix
bytes8	float64	float64	bytes36
VFID0000	185.86877249999998	74.9522485	VFID0000-NGC4363
VFID0001	182.208	74.9042778	VFID0001-NGC4133
VFID0002	145.944663	74.8595778	VFID0002-NGC2977
VFID0003	185.5730835	74.8383889	VFID0003-CGCG352-030
VFID0004	182.82775049999998	74.8082222	VFID0004-UGC07189
VFID0005	173.1341835	74.6275401	VFID0005-NGC3752

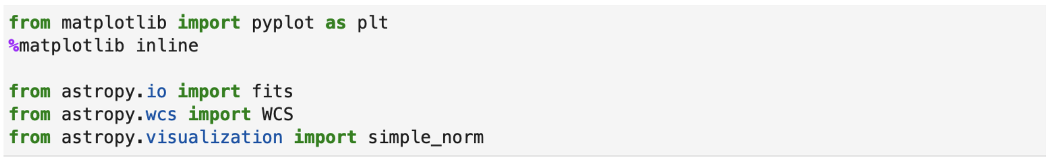
[1] If you still elect to abuse your phalanges, *may the power protect you.*

Before streamlining the writing of HTML files for every galaxy in your sample, it will be helpful to have the relevant information organized into a FITS table. This table could include the galaxy's name or ID, its RA and DEC coordinates on the sky, and even its morphological type. If you do not already have access to such a table, you can either generate one using python/astropy or create this table manually using a spreadsheet and downloading the result as a CSV file. The example above shows how to load a FITS table using astropy.table.

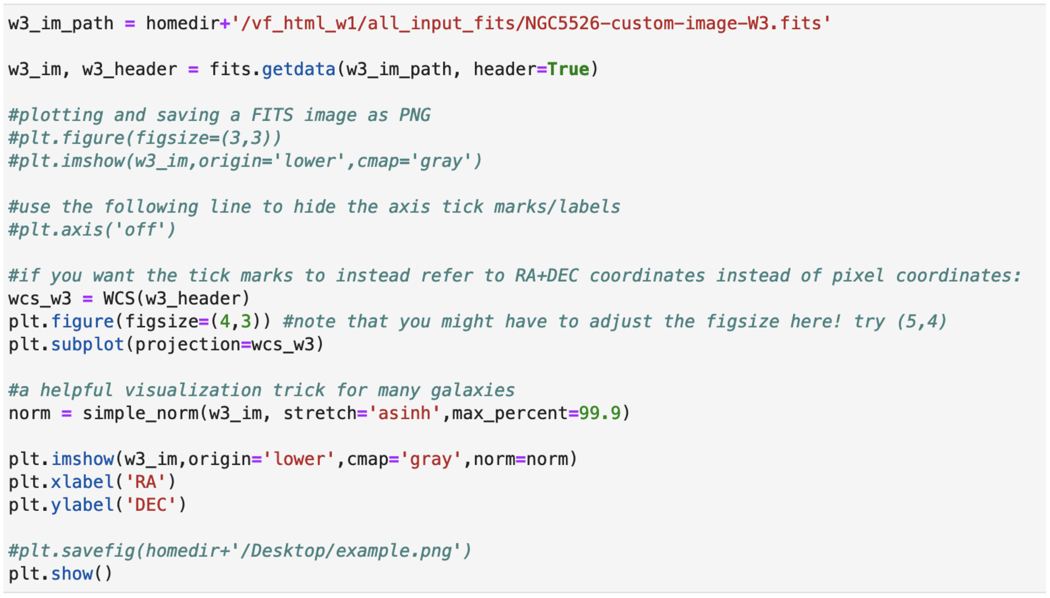
3.2 Matplotlib: Converting FITS to PNG

This section involves preparing the "visual" component of the webpage, namely the images. I will be assuming that FITS images of the sample galaxies are already made available to you.

Firstly, you will want to convert these FITS files to PNG images, which Matplotlib makes relatively uncomplicated.



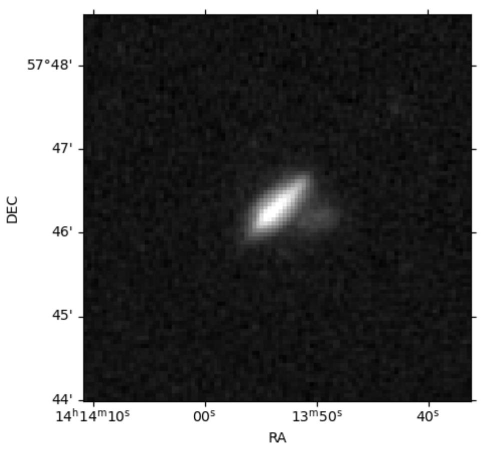
```python
from matplotlib import pyplot as plt
%matplotlib inline

from astropy.io import fits
from astropy.wcs import WCS
from astropy.visualization import simple_norm
```



```python
w3_im_path = homedir+'/vf_html_w1/all_input_fits/NGC5526-custom-image-W3.fits'

w3_im, w3_header = fits.getdata(w3_im_path, header=True)

#plotting and saving a FITS image as PNG
#plt.figure(figsize=(3,3))
#plt.imshow(w3_im,origin='lower',cmap='gray')

#use the following line to hide the axis tick marks/labels
#plt.axis('off')

#if you want the tick marks to instead refer to RA+DEC coordinates instead of pixel coordinates:
wcs_w3 = WCS(w3_header)
plt.figure(figsize=(4,3)) #note that you might have to adjust the figsize here! try (5,4)
plt.subplot(projection=wcs_w3)

#a helpful visualization trick for many galaxies
norm = simple_norm(w3_im, stretch='asinh',max_percent=99.9)

plt.imshow(w3_im,origin='lower',cmap='gray',norm=norm)
plt.xlabel('RA')
plt.ylabel('DEC')

#plt.savefig(homedir+'/Desktop/example.png')
plt.show()
```

Executing these cells yields the cutout image below.

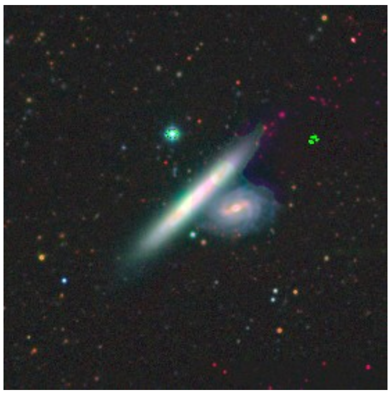
A second trick is to download the optical image of the galaxy, courtesy of the Legacy Survey (https://www.legacysurvey.org/) and the trusty wget module. We



can extract the necessary parameters to ensure that this image has the same scale and pixel size as our image with the FITS header information! Note that this cutout will be a square shape, so if your FITS image has a wonky geometry try not to be alarmed by the symmetric parallelogram.

```python
import wget
import numpy as np

pscale = np.abs(float(w3_header['CD1_1'])) #grabs transformation matrix of image
xsize = np.abs(int(w3_header['NAXIS1'])) #grabs length of the image
xsize_arcsec=pscale*3600*xsize #converts length to arcseconds
imsize = str(int(xsize_arcsec/pixscale)) #converts length to integer...then to a string

LS_filename = homedir+'/Desktop/LS_example.jpg' #choose a filename for the LS optical image

#use either your astropy table of the galaxy FITS header to extract its RA and DEC
#RA = tab['RA']['your_index_here']
#DEC = tab['RA']['your_index_here']
RA = w3_header['CENTRA']
DEC = w3_header['CENTDEC']

#the following is the url leading to the downloading of the LS image.
image_url = f'https://www.legacysurvey.org/viewer/cutout.jpg?ra={RA}&dec={DEC}&layer=ls-dr9&size={imsize}&pixscale={1}'

w3_LS_image = wget.download(image_url,out=LS_filename)
```

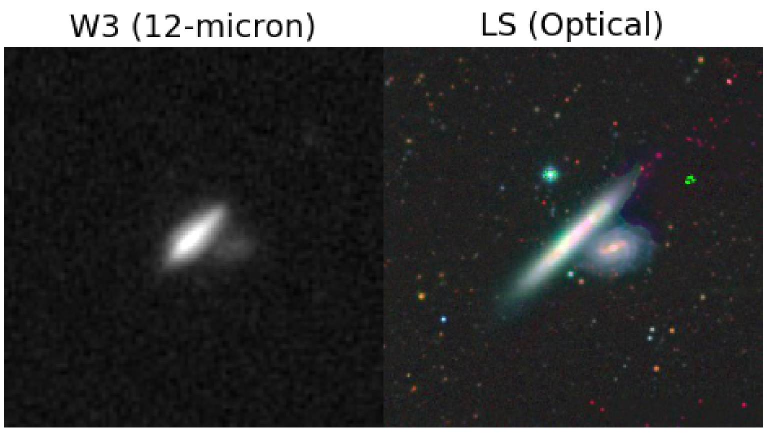
This result will save the LS image to your desired path (which I have assigned to the LS_filename variable). For my example galaxy, the result is as follows:



Yet another display option is to create an image mosaic! I find this format helpful for side-by-side comparisons of a single galaxy imaged in multiple wavelength bands, even if the image geometries are not entirely comparable. Let's try this now with the LS and W3 images, which do in fact have comparable geometries.

```python
import matplotlib.image as mpimg

images = [w3_im, LS_filename]
titles = ['W3 (12-micron)', 'LS (Optical)']

plt.figure(figsize=(6,4))
for i,im in enumerate(images): #this will define both the iterative counter i and the im names

    if i==0: #run this block for the W3 image only
        plt.subplot(1,len(images),i+1) #,projection=w3_header)
        norm = simple_norm(images[i],stretch='asinh',max_percent=99.9)
        plt.imshow(images[i],origin='lower',cmap='gray',norm=norm)
        plt.xlabel('RA')
        plt.ylabel('DEC')
        plt.axis('off')

    if i==1:
        plt.subplot(1,len(images),i+1)
        plt.imshow(mpimg.imread(images[i]))
        plt.axis('off')

    plt.subplots_adjust(wspace=0,hspace=0)
    plt.title(titles[i],fontsize=14)

plt.savefig(homedir+'/Desktop/LS_mosaic_example.png',bbox_inches = 'tight',pad_inches=0.2)
plt.show()
```

The final product is shown below. Note that in cases where you are trying to concatenate a diamond-shaped postage stamp with the non-rotated LS square, you might find that the plt.subplots_adjust() might need different wspace and/or hspace parameters. Matplotlib may also embed your cutout in a square space, but the outskirts, not containing any pixel values, are simply white space. In these cases, perhaps the mosaic idea is not entirely viable - however, the aesthetic choices of your website are your prerogative!



With each of these techniques in mind, you are now equipped to automate the process! The setup we will want to create is of a main directory in which are various folders for the FITS images, the output LS images and/or mosaics, and HTML files. You are not obligated to abide by any sort of systematic sorting method, but you will have to tweak the example code blocks accordingly.

Below I show an example of looping through a subset of galaxies to pull the LS JPG images. The beginning of the cell also creates the relevant aforementioned directories and funnels the images into its corresponding folder. From this, you should be able to extrapolate to automating the saving of PNG images and the generating of mosaics!

```python
#function which creates directories if they do not already exist. not required.
def create_folder(folder_path):
    if os.path.exists(folder_path)==False:
        os.system('mkdir '+folder_path)
        return

main_path = homedir+'/Desktop/test_master/'
master_dir = main_path
input_dir = main_path+'input_fits'
LS_dir = main_path+'LS_images'

create_folder(master_dir)
create_folder(input_dir)
create_folder(LS_dir)

#defining a test sample
test_sample = tab[tab['subsample_flag']][0:3]
test_RA = test_sample['RA_1']
test_DEC = test_sample['DEC_1']
test_ID = test_sample['VFID']
test_name = test_sample['objname']

# **OPTIONAL** moving FITS images into input_directory
for i in range(len(test_sample)):
    imname = f'{test_name[i]}-custom-image-W3.fits'
    os.system(f'cp {homedir}/vf_html_w1/all_input_fits/{imname} {input_dir}')

#creating the loop!

for i in range(len(test_sample)):

    imname = f'{test_name[i]}-custom-image-W3.fits'
    im,header = fits.getdata(f'{input_dir}/{imname}', header=True)
    RA = test_RA[i]
    DEC = test_DEC[i]

    pscale = np.abs(float(header['CD1_1']))
    xsize=np.abs(int(header['NAXIS1']))
    xsize_arcsec=pscale*3600*xsize
    imsize = int(xsize_arcsec/pixscale)

    LS_filename = LS_dir+f'/{test_name[i]}-LS.jpg'

    image_url = f'https://www.legacysurvey.org/viewer/cutout.jpg?ra={RA}&dec={DEC}&layer=ls-dr9&size={imsize}&pixscale={1}'

    w3_LS_image = wget.download(image_url,out=LS_filename)
```

3.3 Homepage HTML

We now possess a decent number of components in order to create a decent homepage. That is precisely what this next code cell entails.

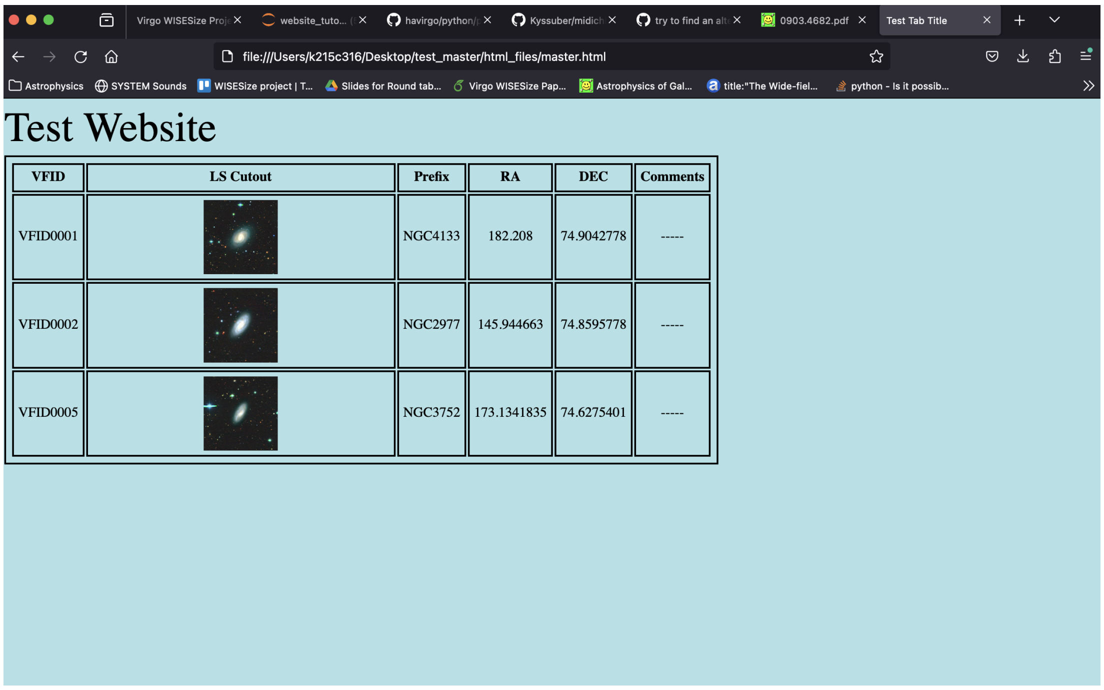
```python
#let's initiate a few variables...
tab_title = 'Test Tab Title'
website_title = 'Test Website'
background_color = 'powderblue' #be gentle with background color; no lime green
html_filename = 'master.html'

#**OPTIONAL** create directory for html files
html_dir = main_path+'html_files'
create_folder(html_dir)

#initiating the html file to write text!
with open(f'{html_dir}/{html_filename}','w') as html:

    #NOTE --> the \n at the end of lines will move to a new line in the html textfile
    #(i.e., it is strictly for tidiness purposes)
    html.write('<html><body>\n') #create the body
    html.write(f'<title>{tab_title}</title>\n') #add text that will appear on the browser tab
    html.write(f'<body style="background-color:{background_color};">\n') #select background color
    html.write('<style type="text/css">\n') #select style type for the text
    html.write('table, td, th {padding: 5px; text-align: center; border: 2px solid black;}\n') #table params
    html.write('p {display: inline-block;;}\n')
    html.write('</style>\n') #and thus concludes the defining of homepage style

    html.write(f'<font size="40">{website_title}</font>\n') #add title

    #begin to populate the table. this first line is header information.
    #include as many columns as desired!
    html.write('<table><tr><th>VFID</th><th>LS Cutout</th><th>Prefix</th><th>RA</th><th>DEC</th><th>Comments</th>\n')

    #this loop is precisely the part we would like to automate. it will insert the galaxy information
    #in each row for us!
    for i in range(len(test_sample)):

        LS_filename = LS_dir+f'/{test_name[i]}-LS.jpg'

        html.write(f'<tr><td>{test_ID[i]}</td>\n') #galaxy VFID (a.k.a. the index)
        html.write(f'<td><img src = "{LS_filename}" height="25%" width = "25%"></img></td>\n') #add LS JPG
        html.write(f'<td>{test_name[i]}</td>\n')
        html.write(f'<td>{test_RA[i]}</td>\n')
        html.write(f'<td>{test_DEC[i]}</td>\n')
        html.write(f'<td>-----</td>\n') #replace the dashes with a helpful galaxy comment!

    #and close the table, the body, the file. it will write to the destination specified in html_dir
    html.write('</tr></table>\n')
    html.write('<br /><br />\n')
    html.write('</html></body>\n')
    html.close()
```

Congratulations, you (might have) just endured your first HTML experience! Be sure to reference the in-text comments to have at least an inkling on what each line contributes to the whole. An important caveat is that in the line where I call the LS_filename to load the image into the table, I use height and width that are appropriate for my web browser settings. **If your images look rather unsightly in a way that is potentially fixable with different dimension parameters, be sure to try changing these percentages!** Otherwise, you can have a look at your finished product in one of a few ways, two of which I type here.

- Open a new browser tab and press Command+O (Mac) or CTRL+O (Windows) on your keyboard, navigate to your master.html file, and click “Open"

- (For Mac Users) In a terminal window, go to the directory where your master.html file is located. Once there, type the command *open master.html*

On my end, opening the example file results in the following image.



Test Website

VFID	LS Cutout	Prefix	RA	DEC	Comments
VFID0001		NGC4133	182.208	74.9042778	-----
VFID0002		NGC2977	145.944663	74.8595778	-----
VFID0005		NGC3752	173.1341835	74.6275401	-----

The width of the LS Cutout column will shrink with the addition of more columns; and it is of course an option to fiddle as you see fit with the layout and image sizes.

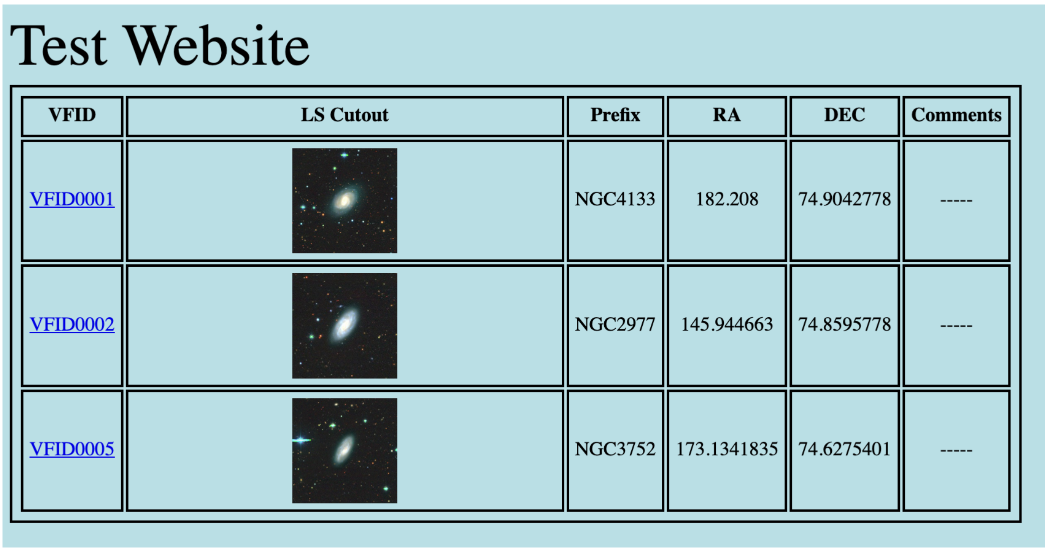
4 Hyperlinks and Galaxy Landing Pages

There is only so much material you can include about a galaxy's properties on a homepage. Fortunately, hyperlinks exist.

If you would like a landing page for individual galaxies, one idea is to add a hyperlink to every "VFID" or "Prefix" column entry! In place of the html.write() line where I add the test_ID to the table in a loop, I could instead write,

```
for i in range(len(test_sample)):

    LS_filename = LS_dir+f'/{test_name[i]}-LS.jpg'
    galhtml = f'{html_dir}/{test_ID[i]}.html'

    html.write(f'<tr><td><a href={galhtml}>{test_ID[i]}</a></td>\n')
    html.write(f'<td><img src = "{LS_filename}" height="25%" width =
        "25%"></img></td>\n')
    html.write(f'<td>{test_name[i]}</td>\n')
    html.write(f'<td>{test_RA[i]}</td>\n')
    html.write(f'<td>{test_DEC[i]}</td>\n')
    html.write(f'<td>-----</td>\n')
```

This line will add a text hyperlink to galaxy page VFIDxxxx.html, where "xxxx" is replaced with the actual galaxy's ID. The "galhtml" variable stores an html file that we have not yet created, so while our website now contains hyperlinked column entries they do not lead to anywhere meaningful.



Test Website

VFID	LS Cutout	Prefix	RA	DEC	Comments
VFID0001		NGC4133	182.208	74.9042778	-----
VFID0002		NGC2977	145.944663	74.8595778	-----
VFID0005		NGC3752	173.1341835	74.6275401	-----

The final link (no pun intended) is therefore to create these galaxy html files. Just as before, we need only to set up the necessary loop with the desired elements. You may want to display mosaics or tabulated data, depending on the type of science you are engaged with. I will give an example below, first using a loop of the LS mosaics routine we created earlier to create some neat galaxy page images:

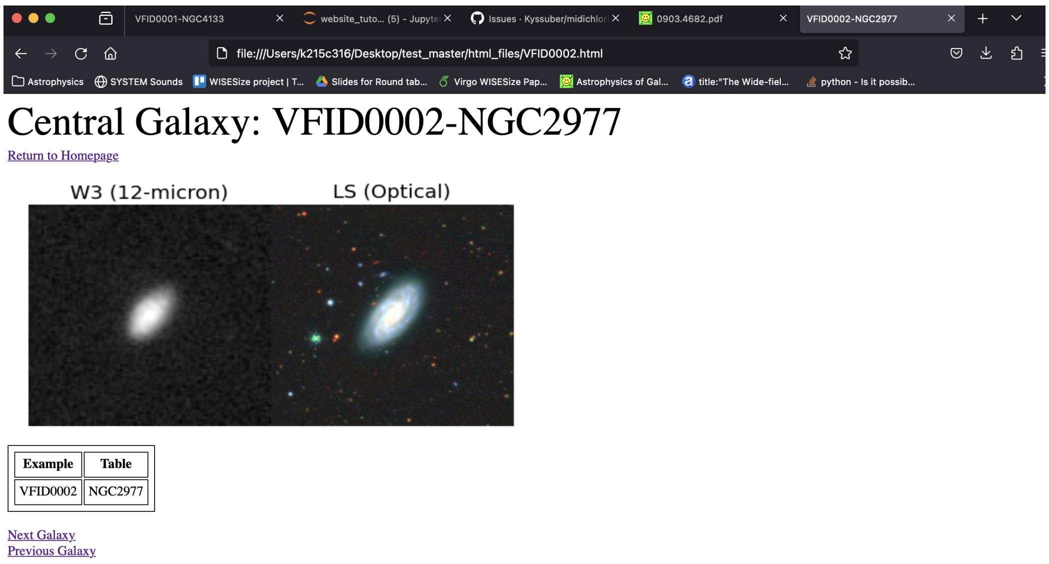
```python
#**OPTIONAL** create directory for mosaic files
mosaic_dir = main_path+'mosaic_files'
create_folder(mosaic_dir)

for n in range(len(test_sample)):
    vfid = test_sample['VFID'][n]
    ra = test_sample['RA_1'][n]
    dec = test_sample['DEC_1'][n]
    objname = test_sample['objname'][n]

    LS_filename = LS_dir+f'/{test_name[n]}-LS.jpg'
    w3_im_path = input_dir+f'/{objname}-custom-image-W3.fits'
    w3_im, w3_header = fits.getdata(w3_im_path, header=True)

    images = [w3_im, LS_filename]
    titles = ['W3 (12-micron)', 'LS (Optical)']

    plt.figure(figsize=(6,4))

    for i,im in enumerate(images):
        plt.subplot(1,len(images),i+1)
        if i==0:
            norm = simple_norm(images[i],stretch='asinh',max_percent=99.9)
            plt.imshow(images[i],origin='lower',cmap='gray',norm=norm)
        if i==1:
            plt.imshow(mpimg.imread(images[i]))
        plt.axis('off')
        plt.title(titles[i],fontsize=14)

    plt.subplots_adjust(wspace=0,hspace=0)
    plt.savefig(mosaic_dir+f'/{test_name[n]}-mosaic.png',bbox_inches = 'tight',pad_inches=0.2)
    plt.close()
```

And now, to create individual galaxy pages.

```python
#will need to create a new hyperlink to the mainpage!
homepage_name = html_dir = main_path+'html_files/'+html_filename

for i in range(len(test_sample)):

    LS_name = mosaic_dir+f'/{test_name[i]}-mosaic.png'
    vfid = test_sample[i]['VFID']
    objname = test_sample[i]['objname']
    galhtml = f'{main_path}html_files/{vfid}.html'

    with open(galhtml,'w') as html:

        html.write('<html><body> \n')
        html.write(f'<title>{vfid}-{objname}</title> \n')  #tab title
        html.write('<style type="text/css"> \n')
        html.write('.img-container{text-align: left;} \n')
        html.write('table, td, th {padding: 5px; text-align: center; border: 1px solid black;} \n')
        html.write('p {display: inline-block;;} \n')
        html.write('</style> \n')

        html.write(f'<font size="40">Central Galaxy: {vfid}-{objname} </font><br /> \n') #page title
        html.write(f'<a href={homepage_name}>Return to Homepage</a></br /> \n')

        #add the LS mosaic
        html.write('<div class='+'"'+'img-container'+'"> <!-- Block parent element --> <img src='+'"'+LS_name+'" height="50%" width="50%" /><br /> \n')

        html.write('<table><tr><th>Example</th><th>Table</th></tr> \n')

        html.write(f'<tr><td>{vfid}</td> \n')
        html.write(f'<td>{objname}</td></tr></table> \n')

        html.write('<br>')   #add a space between the table and the hyperlinks

        if i != len(test_sample)-1:
            html.write('<a href='+str(test_sample['VFID'][i+1])+'.html>Next Galaxy</a></br /> \n')
        if i != 0:
            html.write('<a href='+str(test_sample['VFID'][i-1])+'.html>Previous Galaxy</a></br /> \n')


        html.write('<br /><br />\n')
        html.write('</html></body>\n')

        html.close()
```

There are a few features here worth mentioning. In addition the the LS mosaic, I include an example table where you can organize galaxy details or parameters. There is also hyperlinked text that I have included at both the top and bottom of the page, where users can click to directly return to the homepage or to switch to the next or previous galaxy. Regarding the latter hyperlinks, there are two *if* statements in the code cell that will only add these hyperlinks if the conditions are met: “Next Galaxy" will appear only if there is more than one galaxy page or the current galaxy is not the final one in the sample; and “Previous Galaxy" will appear only if there is more than one galaxy page or the current galaxy is not the first in the sample. Note that these additions are strictly for purposes of navigation and are not at all necessary to include.

As always, I also add a screenshot of how the final product opens on my browser.



Central Galaxy: VFID0002-NGC2977

Return to Homepage

Example	Table
VFID0002	NGC2977

Next Galaxy
Previous Galaxy

fin.

5 Accessing the Notebook Tutorial

Are screenshots not your cup of Python? If you are accessing this guide through a non-github source, then you may not be privy to the Jupyter Notebook companion. Click here to be redirected to my github repository, where you can find said notebook and other quasi-relevant information.

Happy HTML’ing and such!