Tutorial 1: An Introduction to Some Useful Tools

Name: Felix Martinez
Partner: Allison Overton
Date: 2/4/2021

1 Overview

This Tutorial includes an introduction exercise in the downloading and installation of a Virtual Machine, along with a brief introduction to Latex with an example. Furthermore, we cover an in depth introduction to DS9, an image viewer often used in astronomy, along with a short example on how to work with DS9 and iraf27 simultaneously. Finally, we cover short tutorials on how to use the UNIX shell and a crash course in Python.

2 Results

Below we summarize our work for each section of the tutorial and address the questions posed in the manual.

2.1 Virtual Machine

We downloaded the virtual machine and player to our personal computers. After going through the installation process, we logged into the virtual machine.

(1) A screen shot after logging in is shown in Figure 1.

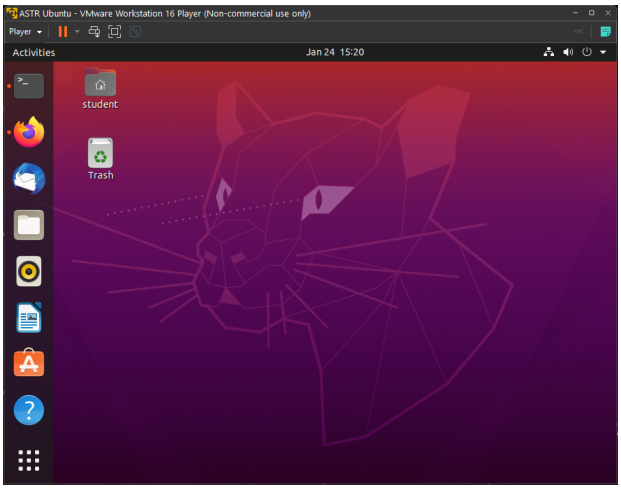

Figure 1: A screen shot of the virtual machine.

2.2 LaTeX

We obtained an Overleaf account and worked through the two LaTeX tutorials. We started the tutorial report using the template available on Canvas.

(1) Here is the text we were asked to reproduce in LaTeX...

Under the assumption that the gas motions are isotropic in the r and z (cylindrical) coordinates, the correction is

$$v_c^2 \; - \; v_\phi^2 \; = \; \sigma_r^2 \Bigg[- \frac{d \ln \nu}{d \ln r} \; - \; \frac{d \ln \sigma_r^2}{d \ln r} \; - \; \left(1 \; - \; \frac{\sigma_\phi^2}{\sigma_r^2} \right) \Bigg] . \qquad (1)$$

The above text comes from Walsh et al. 2013. The Astrophysical Journal paper described a black hole mass ($M_{\rm BH}$) measurement for M87 from gas-dynamical models of *Hubble Space Telescope* observations. Using the method, $M_{\rm BH} = (3.5^{+0.8}_{-0.6}) \times 10^9 M_\odot$ (1σ uncertainties).

2.3 DS9 Part 1.

We watched the DS9 tutorial videos, and continued to download the NGC 3031 fits file from the *Introduction to Astronomy Images and the DS9 Image Viewer* guide linked for us in the tutorial instructions.

Once completed, we loaded the image and had to change the scale parameters to better view the object. As listed in the instructions, we changed the scale parameters to -0.15 and 2.5 MJy/sr, while keeping the scaling linear. We made this change by opening the "Scale" tab listed at the top of the window and selected the "Scale Parameters..." subsection. Selecting this will open another window that allows the user to enter low and high limit values, where we typed in our range.

(1) Here is what the object looked like after we changed the parameters.

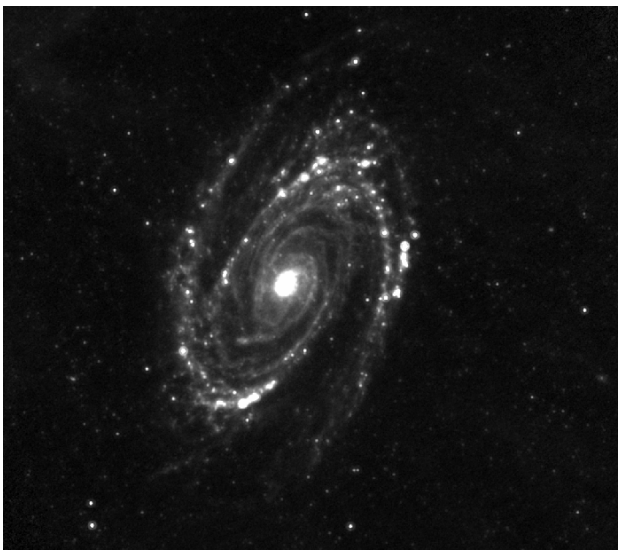

Figure 2: NGC 3031 with linear scaling between -0.15 and 2.5 MJy/sr.

We then were tasked to display the header associated with NGC 3031, we were able to do this by selecting the "File" tab at the top of the window and then chose the "Display Header..." subsection in the drop down menu.

(1) Here is a screen shot of what our header looks like.

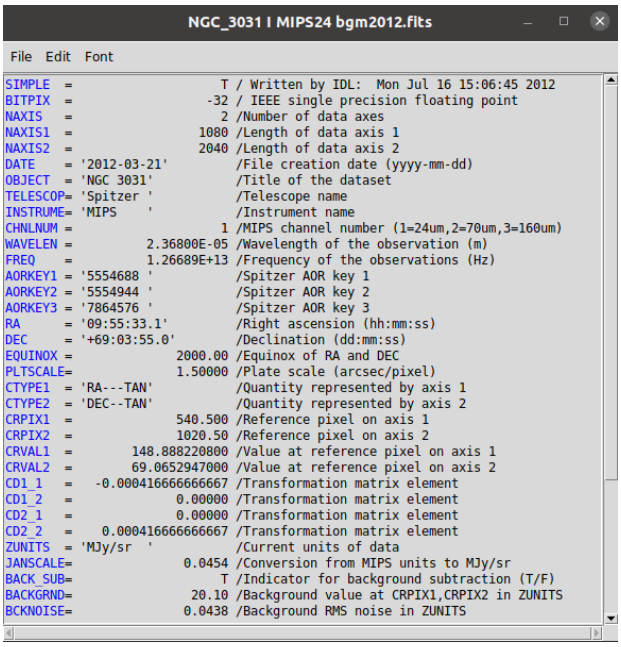

Figure 3: NGC 3031 fits header.

To show the usefulness of DS9, we chose a random pixel at x = 300 and y = 1075 and found the right ascension (RA) and declination (Dec) of this pixel. We did this by hovering our mouse over the image and noticing our "Physical" section located near the top of the window. This will display the x and y values of where our mouse is located in the image. Finding our desired pixel, we were also able to located our RA and Dec by looking at the "WCS" section located right above the "Physical" section. This gave us the right ascension of 9 Hours, 56 minutes, and 40.5 seconds, and the declination of 69 hours, 5 minutes, and 15.9 seconds. We can convert this into degrees by selecting the "WCS" tab located at the top of the window, and scrolling down to select "Degrees", after following the same steps as listed above to find our desired pixel, we can now see that our RA and Dec were replaced in units of degrees with our right ascension now 149.17°and our declination now 69.08°.

With this method down, we can now find the pixel value of anything found throughout the image. Selecting the object at x = 700 and y = 1103, we can find the pixel value by locating at the "Value" section found right above the "Physical" and "WCS" sections located near the top of the window. With our object now found, we were able to find the pixel value of 12.04 MJy/sr.

2.4 DS9 Part 2.

Here, we will display two different frames side by side in DS9. This can be accomplished by hovering over the "Frame" tab at the top of the window and selecting the drop down section listed "New Frame". A blank screen should open up in the DS9 window. Here is where we will open another file by selecting the "File" tab at the top of the window and clicking the "Open" drop down section. We chose to open the same fits file of NGC 3031 in our example. Then, re-select the "Frame" tab at the top of the window and the scroll down and click the "Tile Frames" option, both frames should appear side by side.

(1) A screen shot of our DS9 window with two frames open side by side.

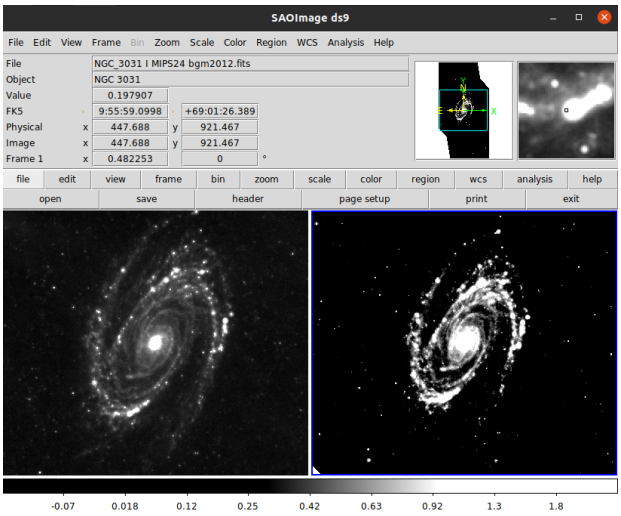

Figure 4: Two frames open side by side in DS9.

Looking at Figure 4, the left screen is our object as seen previously in Figure 2. Yet, on the right screen, we are displaying the same object but scaled using the "ARCSINH" function underneath the "Scale" tab, located at the top of the window. Furthermore, we altered the color values of the right image by selecting the "Color" tab located at the top of the window and then choosing the "Color map Parameters" option located at the bottom of the drop down window. We then moved the Bias to 0.5 and the Contrast to 5.5.

We then deleted the right frame and selected a column cut from y = 600 to y = 1500 at x = 435 to plot the pixel values as a function of pixel location in our DS9 window.

(1) The results of our selected column.

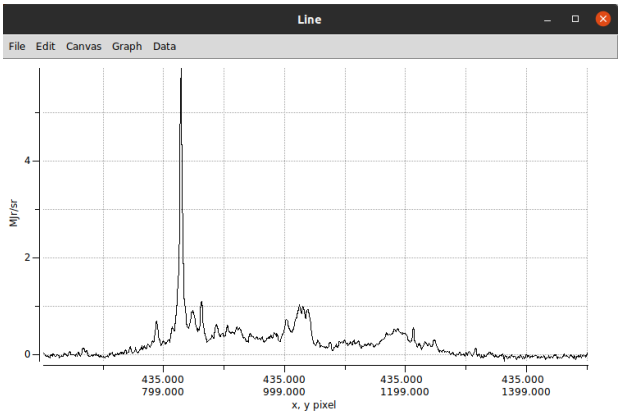

Figure 5: Pixel Value vs. Location in NGC 3031.

Now, selecting a different region in the sky located at x = 780, y = 920, with a radius of 30 pixels, we are able to determine important statistics such as the standard deviation in this region by drawing a random circle on the fits file using the "Circle" shape function under the "Region" tab.

Selecting our new circle and clicking "Get Information", also located under the "Region" tab, a new window should open where we are able to insert the details of our circle. After applying the changes, we selected "Statistics" underneath the "Analysis" tab in our "Circle" window to observe the details about our region. Doing this, we achieved a standard deviation of $\sigma = 0.0363$ MJy/sr.

We will now shift our focus to a star located at x = 261 and y = 1173. We will set contours to better view the drastic change in flux this star causes. We can set contours by selecting the "Analysis" tab at the top of our DS9 window and opening the "Contour Parameters" option in the drop down menu. We then chose our parameters to provide 10 contour levels from 0.15 to 3.5 MJy/sr with smoothness set to 3.

(1) The resulting star with its set contours.

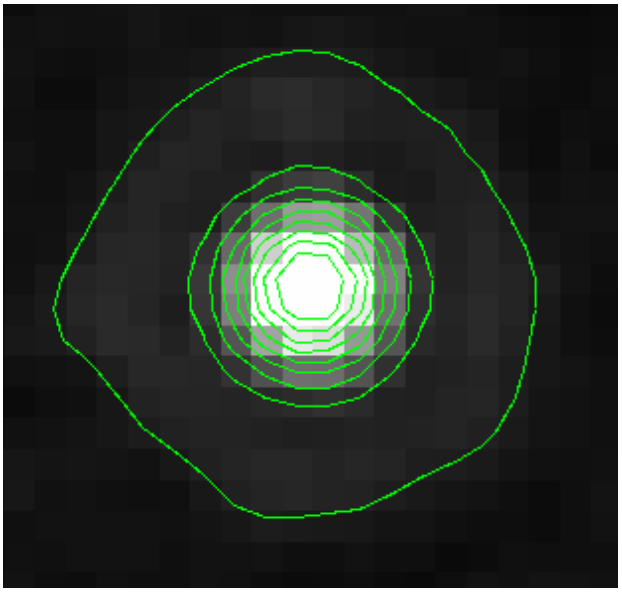

Figure 6: Star with Contours.

Now we will test the "Ruler" function listed under the "Region" and "Shape" tabs. We will start this like usual, creating a random shape somewhere on our image and then selecting the "Get Information" option underneath the "Region" tab. We will be measuring the distance in arcseconds between a star located at x = 475, y = 1090, and a point near the galaxy's nucleus, located at x = 538, y = 1020. Inserting these values into our ruler function, we found the length to be 141.263 arcseconds.

An interesting feature that we found not highlighted in the online instructions was the use of catalogs. We can query catalogs within the DS9 program and display them within our image by selecting the "Catalogs" drop down menu underneath the "Analysis" tab found at the top of the window, and choosing the "Search for Catalogs" option listed underneath it. Once there, an interesting catalog we found was in the optical wavelength conducted by the Chandra mission, listed as "Chandra X-ray observations of M81 (Swartz+, 2003)". After loading the catalog we were able to view all of the objects they observed listed on our DS9 image.

(1) A zoomed-in screen shot of NGC 3031 with a catalog overlay.

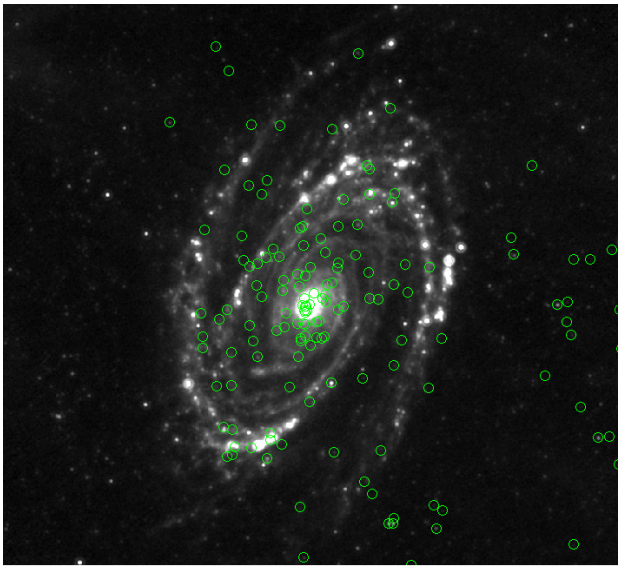

Figure 7: NGC 3031 with a catalog overlay.

It is interesting to note that there were several objects in the catalog that was listed outside of our DS9 image, this is probably due to Swartz et al. 2003 taking their data outside the boundaries of our fits file.

Finally, we will change our focus once again to a different star located at x = 314 and y = 883 to try and find the FWHM (Full Width Half Max). Unfortunately DS9 cannot execute this task on its own so we have to engage PyRAF, a package inside iraf27, to run alongside DS9 to successfully find the FWHM. Along the way we have encountered several issues when booting up PyRAF, for example it only seems to work when our position in the terminal is within the home directory. Furthermore, we encountered another issue after we had PyRAF up and running, when we would try and obtain the FWHM, we would get an error code within the iraf27 terminal that would crash and close our DS9 image viewer. We were able to fix this issue by removing any spaces in our file name and rerunning both terminals with the steps listed in the online instructions. In our successful run we were able to find the FWHM to be 4.17 MJy/sr.

2.5 The UNIX Shell

For Exercise 1 in the UNIX Tutorials Module, we were tasked to create a directory called "backups" within our new "unixstuff" directory. To do this we can type the command:

$ mkdir backups

Exercise 2 wanted us to find the full pathway to the hband directory using **$ pwd**. We found the pathway to be:

$ /home/student/hst_drizzled/hband

Exercise 3 wanted us to find the difference between **$ ls** ~ and **$ ls** ~ **/..** and we found it to be that **$ ls** ~ would list every file (that is every file **$ ls** would normally show) within the home directory, while **$ ls** ~**/..** would list every file in the directory above the home directory. In this case, **$ ls** ~**/..** only listed the directory "student".

Exercise 4 wanted us to create a backup of the "science.txt" text file within our backup folder while also renaming it to "science.bak". We were able to do this by typing this command:

$ cp /home/student/teaching_unix/science.txt /home/student/unixstuff/backups/science.bak

Exercise 5 wanted us to make a directory called "tempstuff" back in our "unixstuff" directory and to remove it. We were able to do this by typing these two commands:

$ mkdir tempstuff

$ rmdir tempstuff

Exercise 6 wanted us to determine what the number "−5" did in the line of code **$ head -5 science.txt** vs. the line of code **$ head science.txt**. We were able to determine that the −5 showed only the first 5 lines of the text file. Doing this we were able to deduce how to view the last 15 lines of a text file, by typing this command:

$ tail -15 science.txt

Exercise 7 wanted us to create a list, "list2" and to fill it with certain items. We did this by using these commands:

$ cat > list2

orange [return]

plum [return]

mango [return]

grapefruit [return]

[ctrl] + [d]

We finished making that list by typing "control" and "d" at the end. Furthermore we were tasked with opening the list, we did this by typing:

$ cat list2

Exercise 8 tasked us to display all of the lines in "list1" and "list2" containing the letter "p" while using the pipeline commands. We accomplished this by using the command:

$ cat list1 list2 | grep 'p' | wc - l

Exercise 9 then challenged us to change the access permission on the file "science.txt" so that others can write, we did this by typing:

$ chmod o + w science.txt

Finally, Exercise 9 also wanted us to change the access permission on the "backups" directory so that others cannot remove it, we did this by typing:

$ chmod o - x backups/

2.6 A Python Primer

Exercise 1 wanted us to conduct a simple average calculation of myself and two of family members. This is the code we used and the result we obtained:

```
In [15]: #your code here
         age_me = 21
         age_mom = 43
         age_dad = 45

         age_average = (age_me + age_mom + age_dad)/3

In [16]: print(age_average)

         36.333333333333336
```

Figure 8: Exercise 1: A simple calculation

Exercise 2 wanted us to create a list of even numbers between 1 and 100. This is the code we used and the result we obtained from it:

```
In [65]: skip_count = []
         i = 1
         while i < 102:
             odd = i/2

             if i % 2 == 0:
                 skip_count.append(i)

             i = i + 1

In [66]: print(skip_count)

         [2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 6
         0, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100]
```

Figure 9: Exercise 2: Skip-count

Exercise 3 wanted us to get the output of 's' in list_1's 0th element. This is the code we used and the result we obtained from it:

```
In [74]: list_1[0][2]

Out[74]: 's'
```

Figure 10: Exercise 3: Indexing

Exercise 4 wanted us to calculate the standard deviation out of a list of scores. We were able to do this by using the code below:

```
In [122]: import numpy as np

scores = [100, 68, 40, 78, 81, 65, 39, 118, 46, 78, 9, 37, 43, 87, 54, 29, 95, 87, 111, 65, 43, 53, 47, 16,
          98, 82, 58, 5, 49, 67, 60, 76, 16, 111, 65, 61, 73, 63, 115, 72, 76, 48, 75, 101, 45, 46, 82, 57,
          17, 88, 90, 53, 32, 28, 50, 91, 93, 7, 63, 88, 55, 37, 67, 0, 79]

average_score = np.sum(scores)/len(scores)

avg_score_percent = average_score/100

shortened = str(avg_score_percent) #turn it into a string
statement2 = "The average score on the test was a {}%".format(shortened[0:5])
print(statement2)

arr_version = np.array(scores)

std = np.sqrt(sum((arr_version-average_score)**2)/len(scores)-1)
print("The standard deviation on the test was " + str(round(std,4)))

The average score on the test was a 0.619%
The standard deviation on the test was 28.0793
```

Figure 11: Exercise 4: Bringing it together

Exercise 5 wanted us to load a text file and start to analyse the elements within it. We used the code below to sort through some of the elements within the text file.

```
In [21]: class_grades = np.loadtxt('/home/student/python_primer/quantum_mechanics.grades',dtype= 'str')
print("Printing out the first 5 class grades and names: ")
print(str(class_grades[:5]))  #I am only printing out the first 5 as it is a large txt file
print("Printing out the 0th element: " + str(class_grades[0]))
print("Printing out the 1st element of the mini-array: " + str(class_grades[0][1]))
print("Converting that element into a float: "+ str(float(class_grades[0][1])))

Printing out the first 5 class grades and names:
[['Sam' '100']
 ['Joe' '68']
 ['Priya' '40']
 ['Minh' '78']
 ['Sarah' '81']]
Printing out the 0th element: ['Sam' '100']
Printing out the 1st element of the mini-array: 100
Converting that element into a float: 100.0
```

Figure 12: Exercise 5: Loadtxt()

Exercise 6 wanted us to become familiar with dictionaries and how to use them. Below is the code we wrote and all the tasks that we were asked to complete:

```
In [147]: class_dictionary = {'key': [], 'score': []}
          for i in range(len(class_grades)):
              class_dictionary['key'].append(class_grades[i][0]) #appends the names
              class_dictionary['score'].append(float(class_grades[i][1])) #appends the scores

          for i in range(len(class_dictionary['key'])):
              if(class_dictionary['key'][i] == 'Peter'):
                  print("Peter got a score of a " + str(class_dictionary['score'][i]))

          print("Max value : "+str(np.max(class_dictionary['score'])))
          print("Min value : "+str(np.min(class_dictionary['score'])))

          pos_max = np.argmax(class_dictionary['score'])
          pos_min = np.argmin(class_dictionary['score'])

          print("The person to achieved the highest score was: "+ class_dictionary['key'][pos_max])
          print("The person to achieved the lowest score was: "+ class_dictionary['key'][pos_min])

          mean_score = np.mean(to_fill)
          print("The mean score is: "+ str(round(mean_score,2)))

          Peter got a score of a 67.0
          Max value : 118.0
          Min value : 0.0
          The person to achieved the highest score was: Celeste
          The person to achieved the lowest score was: Sean
          The mean score is: 61.97
```

Figure 13: Exercise 6: A dictionary of student names / scores

Exercise 7 wanted us to use the mean score of the sample class's midterm grades to separate the class into two groups: "proficient students" and "struggling students". We did this by using the code below:

```
In [95]: proficient_students = []
         struggling_students = []
         for i in range(len(class_grades)):
             if (class_dictionary['score'][i] > mean_score):
                 proficient_students.append(class_grades[i])
             else:
                 struggling_students.append(class_grades[i])
         print(proficient_students)

         [array(['Sam', '100'], dtype='<U9'), array(['Joe', '68'], dtype='<U9'), array(['Minh', '78'], dtype='<U9'), array
         (['Sarah', '81'], dtype='<U9'), array(['Shivani', '65'], dtype='<U9'), array(['Celeste', '118'], dtype='<U9'), arr
         ay(['Andrew', '78'], dtype='<U9'), array(['Caroline', '87'], dtype='<U9'), array(['Mariah', '95'], dtype='<U9'), a
         rray(['Josiah', '87'], dtype='<U9'), array(['Malena', '111'], dtype='<U9'), array(['Steven', '65'], dtype='<U9'),
         array(['Mariska', '98'], dtype='<U9'), array(['Tom', '82'], dtype='<U9'), array(['Peter', '67'], dtype='<U9'), arr
         ay(['Michael', '76'], dtype='<U9'), array(['Jasmine', '111'], dtype='<U9'), array(['Vega', '65'], dtype='<U9'), ar
         ray(['Nick', '73'], dtype='<U9'), array(['Nikoo', '63'], dtype='<U9'), array(['Leo', '115'], dtype='<U9'), array
         (['Madeline', '72'], dtype='<U9'), array(['Caragh', '76'], dtype='<U9'), array(['Arjun', '75'], dtype='<U9'), arra
         y(['Sahana', '101'], dtype='<U9'), array(['Adam', '82'], dtype='<U9'), array(['Trevor', '88'], dtype='<U9'), array
         (['Allyn', '90'], dtype='<U9'), array(['Gibson', '91'], dtype='<U9'), array(['Justin', '93'], dtype='<U9'), array
         (['Kaley', '63'], dtype='<U9'), array(['Kiara', '88'], dtype='<U9'), array(['Sanni', '67'], dtype='<U9'), array(['
         Nikoo', '79'], dtype='<U9')]
```

Figure 14: Exercise 7: Sort the students

Exercise 8 wanted us to plot a histogram of the students scores and separate them via percentiles. The code below shows how we did this and the results we obtained:

```
In [111]: #calculating the 16th 50th and 84th percentiles
          per_16 = np.percentile(to_fill,16)
          per_50 = np.percentile(to_fill,50)
          per_84 = np.percentile(to_fill,84)

          #plotting the histogram
          plt.hist(to_fill,bins=9,alpha=0.5)
          plt.axvline(per_16,ls='--',c='k')   #showing the 16th percentile
          plt.axvline(per_50,ls='--',c='k')   #showing the 50th percentile
          plt.axvline(per_84,ls='--',c='k')   #showing the 84th percentile
          plt.xlabel('Score (out of 120)')
          plt.ylabel('Number of Students')
          plt.title('Distribution of Student Scores for Midterm 1')
          plt.show()
```

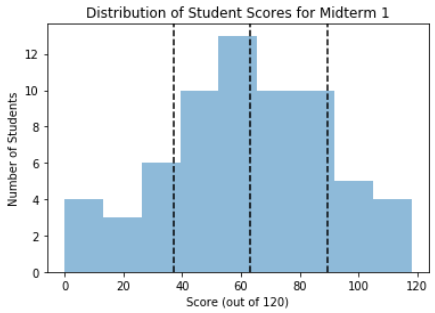

Figure 15: Exercise 8: Plot student scores and percentiles

Exercise 9 wanted us to create a definition that would display students scores based on the number of students listed in the function. We were also asked to test it on two students and then again on three students. Below is the code we used to define said function and our results of the tests.

In [216]:
```
def compare_students_better(student1,student2,class_dict=class_dictionary,*args):
    names = [student1,student2,*args]
    scores = []

    #creating the class dictionary that we will need to find the students scores
    class_grades = np.loadtxt('/home/student/python_primer/quantum_mechanics.grades',dtype= 'str')
    class_dictionary = {'key': [], 'score': []}
    for i in range(len(class_grades)):
        class_dictionary['key'].append(class_grades[i][0]) #appends the names
        class_dictionary['score'].append(float(class_grades[i][1])) #appends the scores

    #appending the scores of the students listed
    for i in range(len(class_dictionary['key'])):
        for j in range(len(names)):
            if(class_dictionary['key'][i] == names[j]):
                scores.append(class_dictionary['score'][i])

    #creating the bar graph
    plt.barh(names,scores)
    plt.title('Comparison of student scores')
    plt.xlabel('Score')
    plt.ylabel('Student')

    return(plt.show())

#Testing the function with two and then three students
compare_students('Sarah','Josiah'),compare_students('Sarah','Josiah','Malena')
```

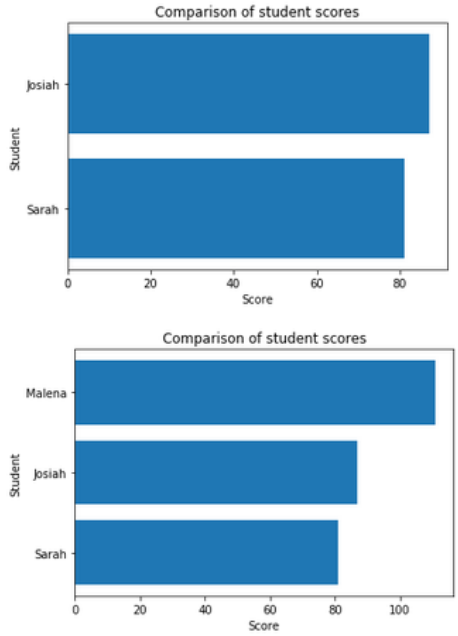

Figure 16: Exercise 9: A bar plot comparing student scores

Exercise 10 wanted us to become familiar with changing a header value in a fits file. Below are the examples the tutorial wanted us to complete for this exercise.

```
In [53]: primary_header['OBJNAME'] = 'NGC 4486'

         primary_header['TMPKEY'] = 42
         primary_header.comments['TMPKEY'] = 'Just playin ;)'

         del primary_header['TMPKEY']
         primary_header

         first_header = hdulist[1].header
         specimg = hdulist[1].data

         #printing the primary header
         primary_header

Out[53]: SIMPLE  =                    T / Written by IDL:  Sun Nov 29 08:58:20 2020
         BITPIX  =                   16 / Number of bits per data pixel
         NAXIS   =                    0 / Number of data axes
         EXTEND  =                    T / FITS data may contain extensions
         DATE    = '2020-11-29'         / Creation UTC (CCCC-MM-DD) date of FITS header
         OBJNAME = 'NGC 4486'           / Name of the observed object
         COMMENT FITS (Flexible Image Transport System) format is defined in 'Astronomy
         COMMENT and Astrophysics', volume 376, page 359; bibcode 2001A&A...376..359H
```

Figure 17: Exercise 10: Change a header value

Exercise 11 wanted us to plot the one-dimensional error spectrum at row 102 using the two-dimensional error spectrum for M87. We were able to do accomplish this by using the code listed below:

```
In [73]: second_header = hdulist[2].header
         spectrum = hdulist[2].data

         spec2d = spectrum[102,:]

         plt.figure(figsize=(10,5)) #set the size of the figure to ensure it is large enough
         #when plotting a spectrum, use step because we have a measurement (flux density) at a
         #particular wavelength and not in between wavelengths.
         plt.step(spec2d,'k-')
         plt.xlabel('Pixel Number')
         plt.ylabel('Flux Density (ergs s$^{-1}$ cm$^{-2}$ $\AA^{-1}$)')
         plt.title('M87 Error Spectrum at Row 102')
         plt.minorticks_on() #include minor tick marks on the x and y axes
```

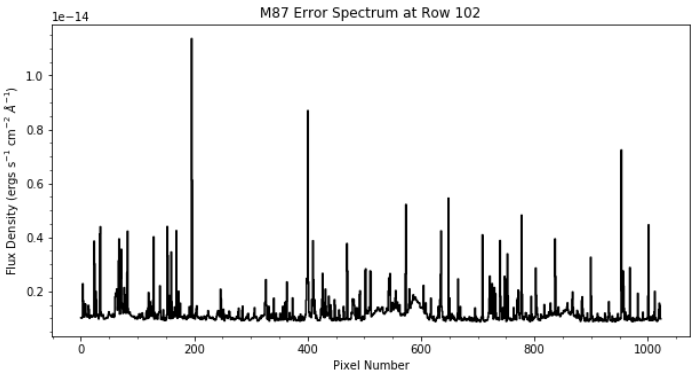

Figure 18: Exercise 11: Plotting the Variance Spectrum