Python Tutorial - Assignment 2

Name:- Kheemraj

What's Python and how to use it?

Python is a high-level, interpreted programming language known for its readability and versatility. It's widely used in various domains, including web development, data analysis, artificial intelligence, scientific computing, and automation.

Conda

Conda is an open-source package management and environment management system widely used in the Python and R programming languages. It allows users to easily install, run, and update packages, as well as manage environments to isolate project dependencies.

Anaconda Distribution

Anaconda Distribution is a comprehensive open-source platform for data science and machine learning. It includes Python and R programming languages along with a suite of tools and libraries for scientific computing, data analysis, and visualization.

Miniconda

Miniconda is a minimal installer for the Anaconda distribution that includes only the essential components needed to get started with conda. It is a lightweight alternative to Anaconda and is perfect for users who want to manage packages and environments without the bloat of pre-installed libraries and tools.

Python Functions, Modules and Packages

Using some basic Python Commands

Terminal/Command Line

- Line by line programming
- Import and/or install modules collage uygun bir kod haline getir

---

Text Editors and Scripts

- To write several commands as on and run them together as a script - You can use text editors like TextEdit and Notepad, or something more advanced like Sublime which color codes the syntax - Script editors hosted on VS Code and Anaconda Navigator(Spyder) are also useful - The script is saved as a .py file and can be executed as python *insertscriptname*.py from Terminal/Command Line\

Notebooks and Notebook Architecture

Interactive documents where you can write small portions code along with rich text (formatted text and images). Jupyter Notebook is the default python notebook format Cloud based options: Jupyter Lab and Google Colab

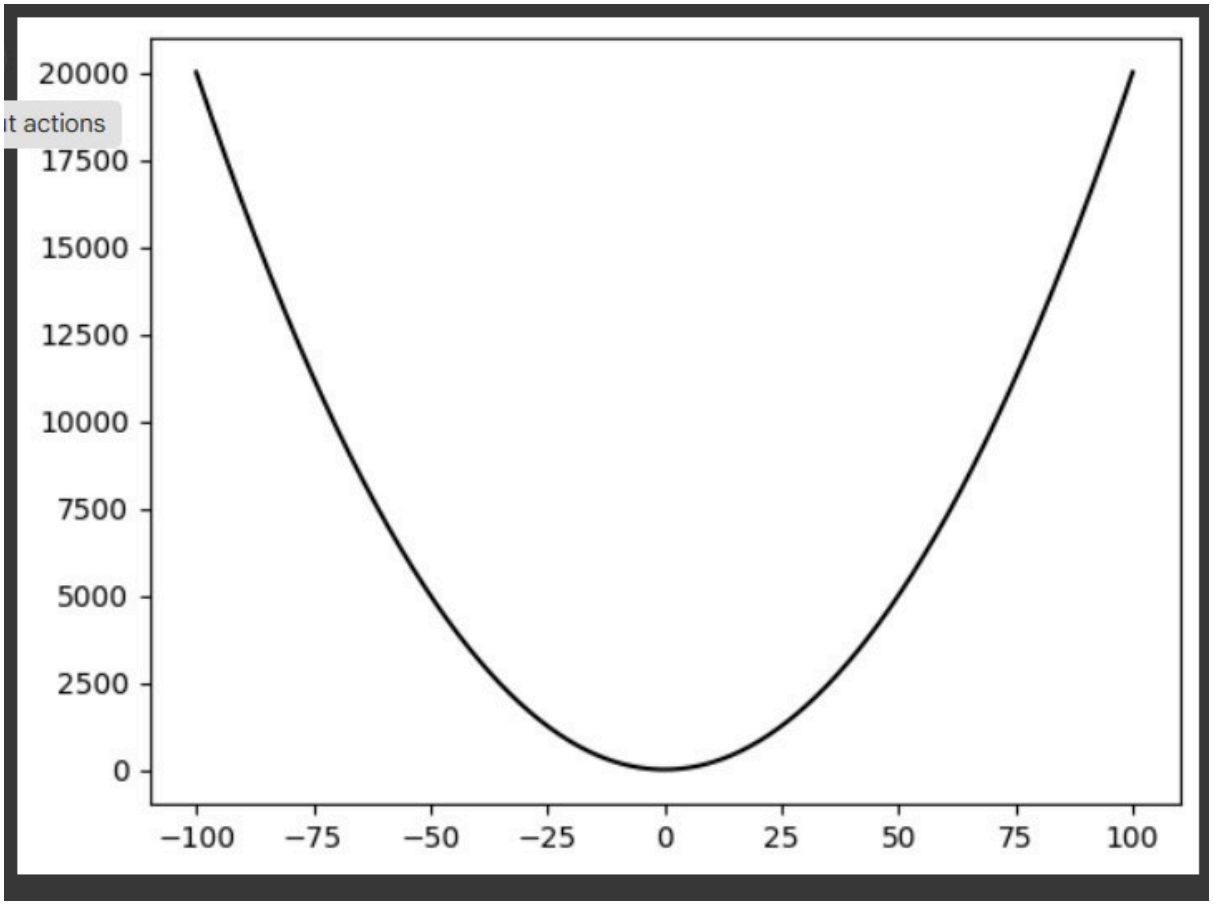
```
#Importing libraries and aliasing them
import numpy as np
import matplotlib.pyplot as plt
```

```
[4] print('Hello World')
    a,b = 3,4
    print('Product of a and b is ',a*b)
```

Hello World

Product of a and b is 12

```
[5] #plotting
    '''
    commenting is essential to guide the user through the code
    '''
    a = np.linspace(-100,100,1000) #to create an array
    b = 2*a**2 + 5
    plt.plot(a,b,'-k')
    plt.show()
```



Some Cool Stuff Python can do

In astronomy, observed data is typically stored in fits files, that can be plotted here using the same packages to plot data points. Here, I am going to give an example using a published image of a popular disk, that recently garnered attention for a newly discovered protoplanet in the system, AB and b

```
#import astronomy package
from astropy.io import fits
```

```
from google.colab import drive
drive.mount('/content/drive')
```

Double-click (or enter) to edit

---

from astropy.io import fits

Tam dosya yolunu tanımla file_path = '/content/drive/MyDrive/Intro2Astro2025 Stajı/Intro-to-Astro2025-main/Week2_Python_Exoplanet_Detection/data/pdi_pi_collapsed.fits'

FITS verisini yükle img = fits.getdata(file_path, ext=1)

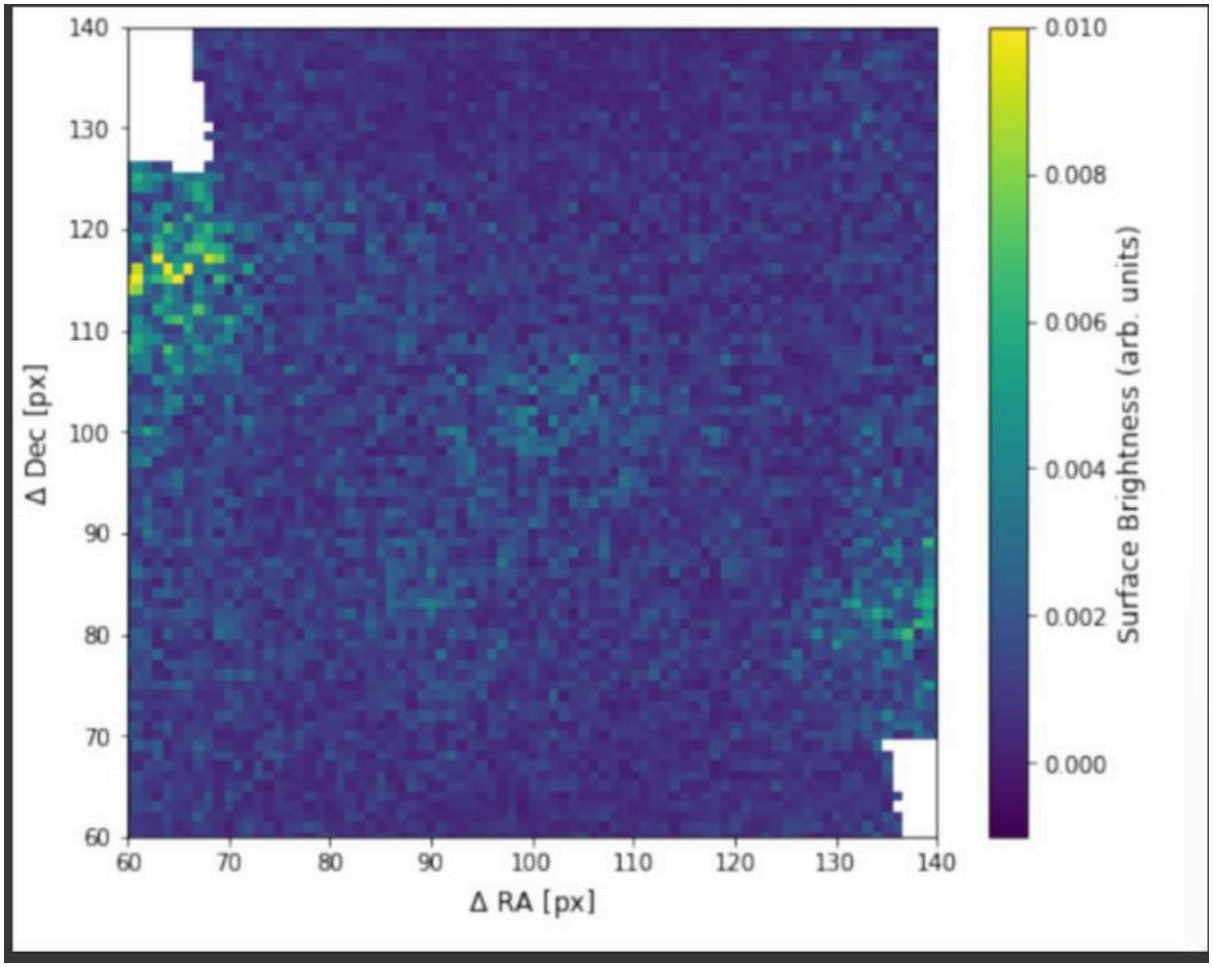
```
[14] #import astronomy package
     from astropy.io import fits # to read and write FITS files
```

```
[ ] #loading the image
    #remember to alter the image path accordingly
    file_dir = 'data/' # data directory where the FITS file is stored
    img_name = 'pdi_pi_collapsed.fits' # The FITS file
    img = fits.getdata(file_dir+img_name,ext=1)
```

```
[ ] # create a figure
    fig = plt.figure(figsize=(8,8)) # sets the size of figure in inches
    #plot in the figure
    image = plt.imshow(img, origin = 'lower', vmin=-0.001, vmax=0.0cbar = fig.
    ↪colorbar(image,shrink=0.82)
    cbar.set_label('Surface Brightness (arb. units)', rotation=90,fontsize=12)e=12)
    plt.xlim([60,140]) # sets the x-axis limits
    plt.ylim([60,140]) # sets the y-axis limits
    plt.xlabel(r'$\Delta$ RA [px]',fontsize=12) # sets the x-axis label

    2

    plt.ylabel(r'$\Delta$ Dec [px]',fontsize=12); # sets the y-axis label
```



Python is very powerful and can read many many rows of data in a short time. In the above example, the image is a 201 x 201 matrix, so a total of 40401 data points. Now let's try reading in 20,000 rows of data with 97 different columns, and use two of those columns to create a scatter plot.

(Taken from 2021 tutorial) : Here, we are loading positional data of 20,000 stars from a CSV table that consists of Gaia (ESA's space telescope) data for the closest stars. We are then showing all the stars as per their positional coordinates RA and Dec (Right Ascension and Declination) as follows

```
[ ]  import pandas as pd

     file_path = '/content/drive/MyDrive/ASTRO_WORKSHOP/Week2_Python_Exoplanet_Detection/data/closest20kstars.csv'

     stellar = pd.read_csv(file_path)
```

```
[ ]  stellar.head()
```

```
[ ]  # Reading 20,000 Rows*97 columns of Data- Closest 20k stars from Gaia Archive
     (DR2)
     stellar=pd.read_csv(file_dir+'closest20kstars.csv') # reads the csv file and
     stores it in a pandas dataframe called stellar

     3

     # Creating a matplotlib (pyplot) figure
     fig = plt.figure(figsize = [10,10])
     # Plotting the scatter graph with RA on x-axis and Dec on y-axis. alpha tells
     the opacity, s is the size
     plt.scatter(stellar['ra'], stellar['dec'], alpha=0.8, s=12, lw = 0.5, ec = 'k',
     color='darkorchid') # plots the scatter plot of RA vs Dec
     # Adding labels and a title for the figure
     plt.xlim([0,362])
     plt.ylim([-75,75])
     plt.xlabel('RA [$\degree$]', fontsize = 14)
     plt.ylabel('Dec [$\degree$]', fontsize = 14)
     plt.title('Closest Stars 20,000 stars from Gaia Archive (DR2)',fontsize = 16)# makes the title of the plot
```

[]: Text(0.5, 1.0, 'Closest Stars 20,000 stars from Gaia Archive (DR2)')