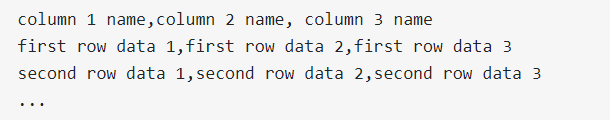
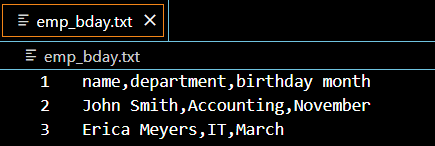
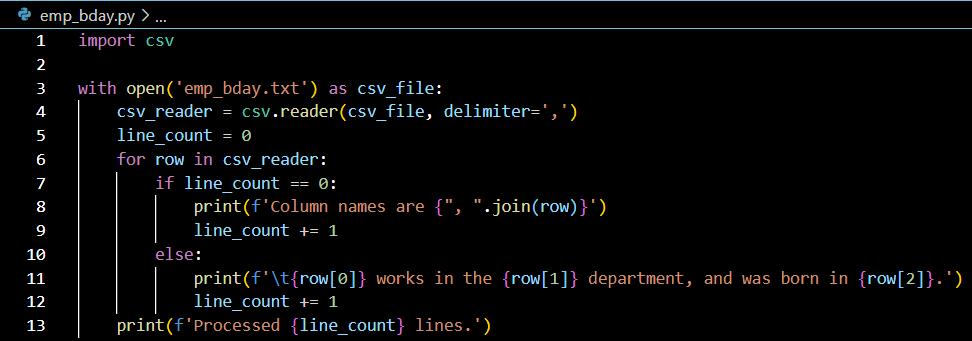
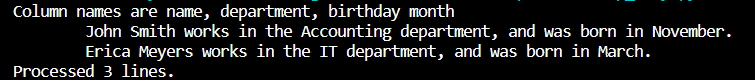
**22-09-2022**

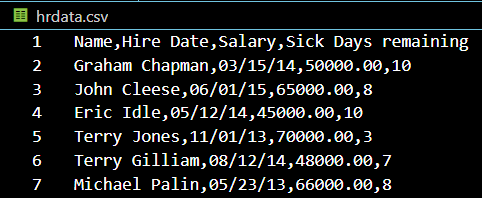
1. **CSV files manipulation**
2. **Array manipulation**
3. **Plots and its types**
4. **Random number n-dimensions**
5. **Numpy**
6. **CSV files**

* Comma separated values files
* ASCII or Unicode characters
* Follows specific structure to arrange tabular information
* 
* Separated by delimiter ( , , \t, : , ; , etc)
* Used while exporting spreadsheets and databases
* Reading from csv files – normally and using dictionary
* Writing to csv files – normally and using dictionary
* Using pandas
  + python library used to analyse data sets and make conclusions based on statistical theories
  + refers to “Panel Data” and "Python Data Analysis" and was created by Wes McKinney in 2008.
  + pandas.read\_csv()
  + attributes:
    - index\_col=’colname’
    - parse\_dates=[‘colname’]
    - header=0
    - names = [‘colname1’,’colname2’, …]

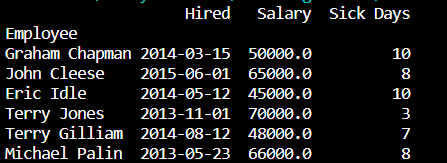




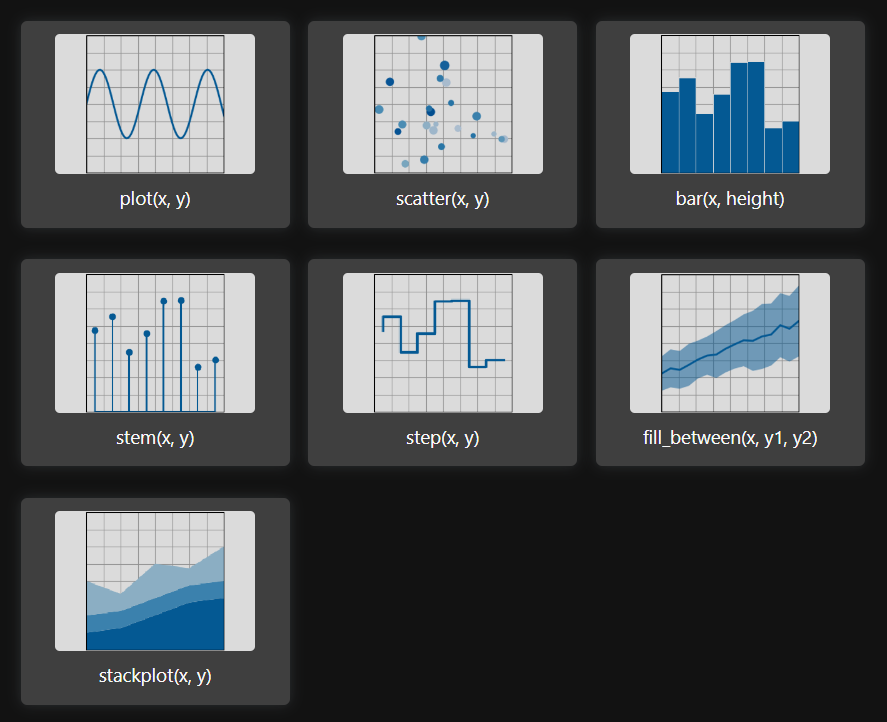
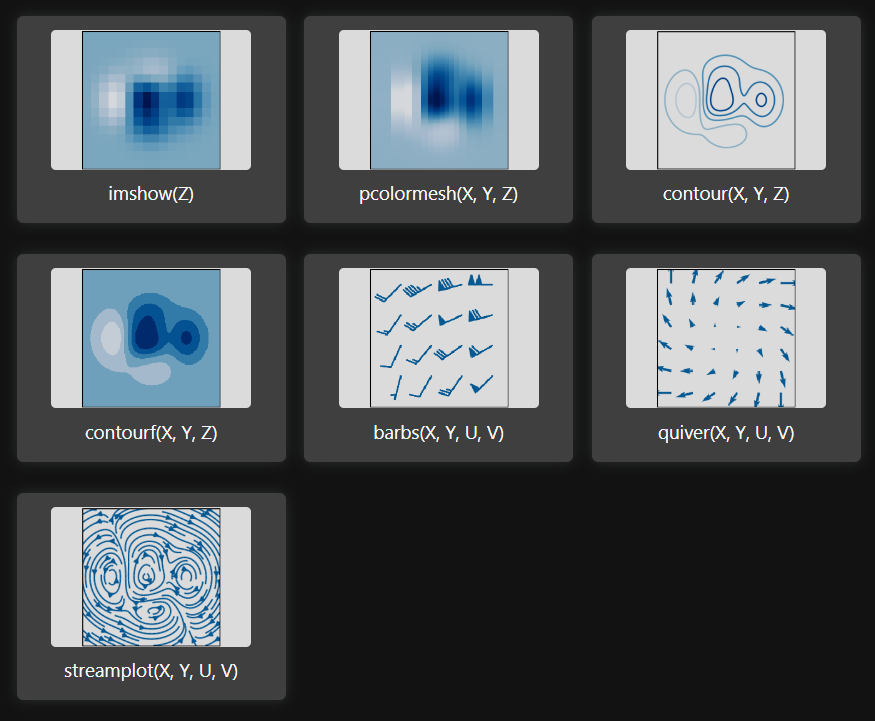
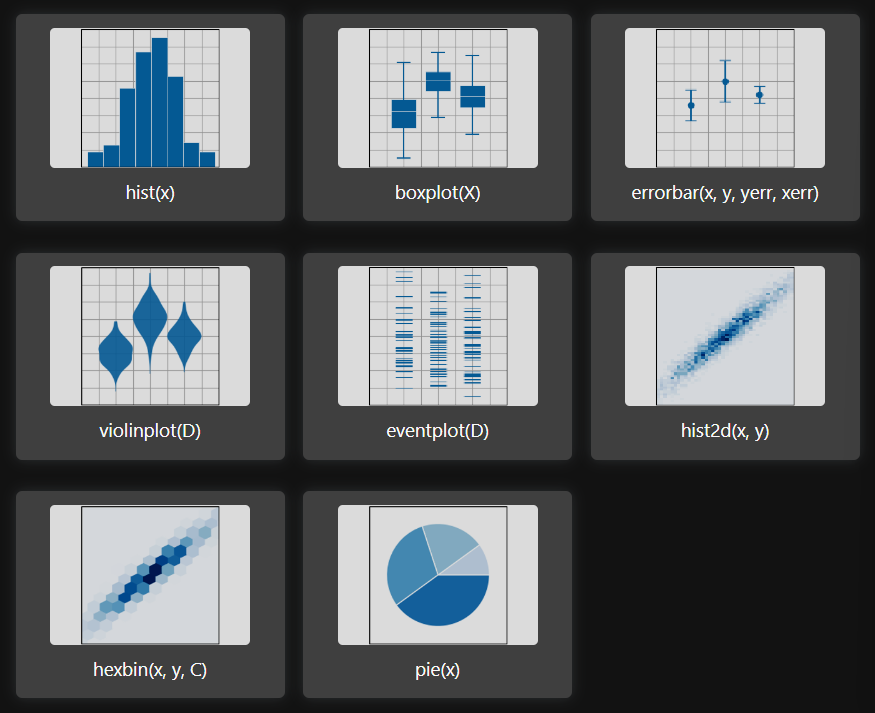
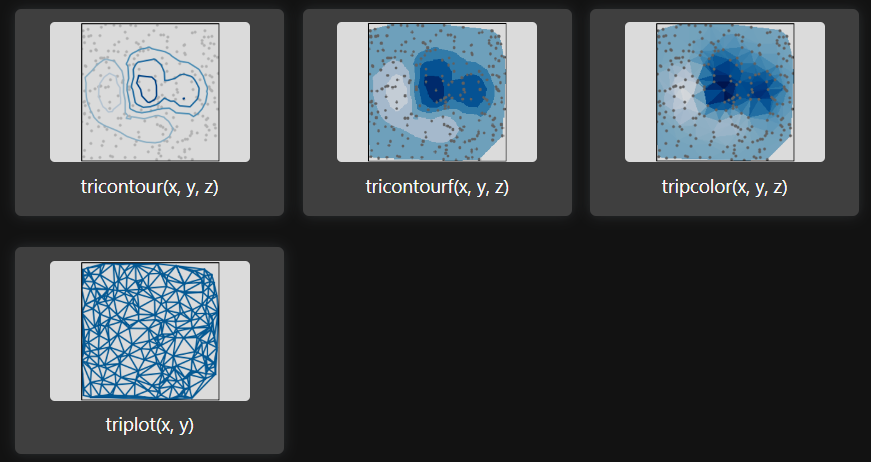




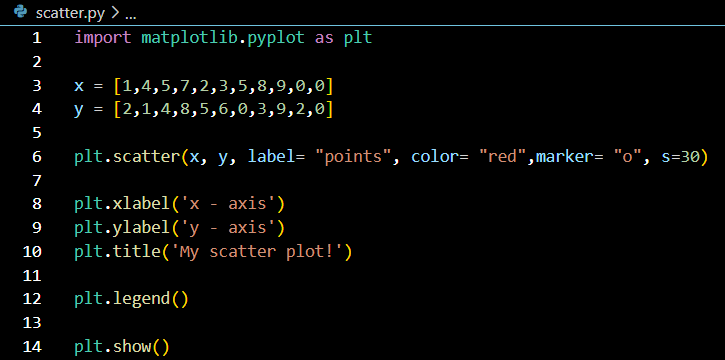


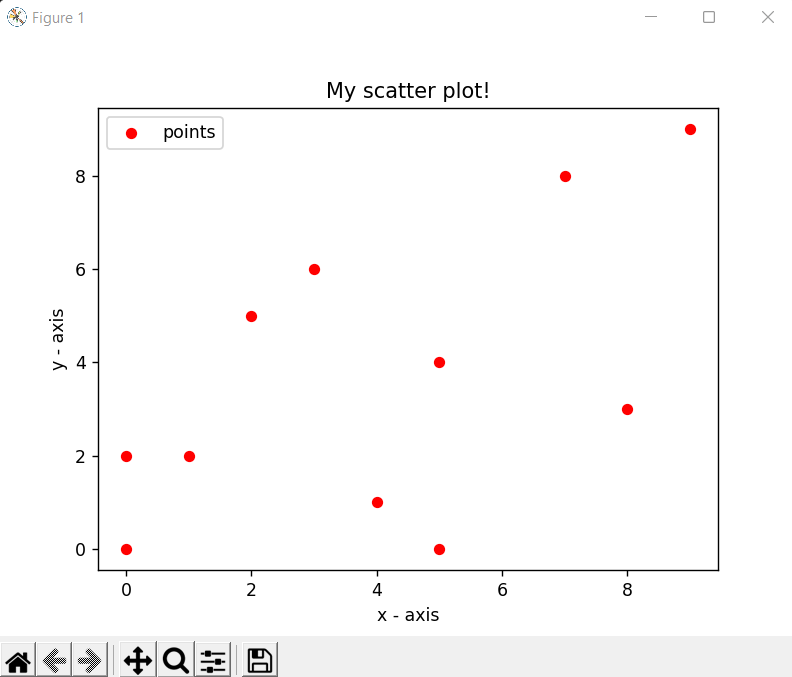


1. **Plots and its types**

* **Matplotlib**
  + low level graph plotting library with visualization utility
* Basic plots – y vs x
  + Plot(x,y)
  + Scatter(x,y)
  + Bar(x,y)
  + Stem(x,y)
  + Step(x,y)
  + Fill\_between(x,y1,y2)
  + Stackplot(x,y)
  + 
* Plots for arrays of data Z(x,y) and fields U(x,y) , V(x,y)
  + Imshow(Z)
  + Pcolormesh(X,Y,Z)
  + Contour(X,Y,Z)
  + Contourf(X,Y,Z)
  + Barbs(X,Y,U,V)
  + Quiver(X,Y,U,V)
  + Streamplot(X,Y,U,V)
  + 
* Statistics plots
  + Hist(x,y)
  + Boxplot(X)
  + Errorbar(x,y,yerr,xerr)
  + Violinplot(D)
  + Eventplot(D)
  + Hist2d(x,y)
  + Hexbin(x,y,C)
  + Pie(X)
  + 
* Unstructured coordinates – use triangular algorithm to fill the triangles
  + Tricontour(x,y,z)
  + Tricontourf(x,y,z)
  + Tripcolor(x,y,z)
  + Triplot(x,y)
  + 

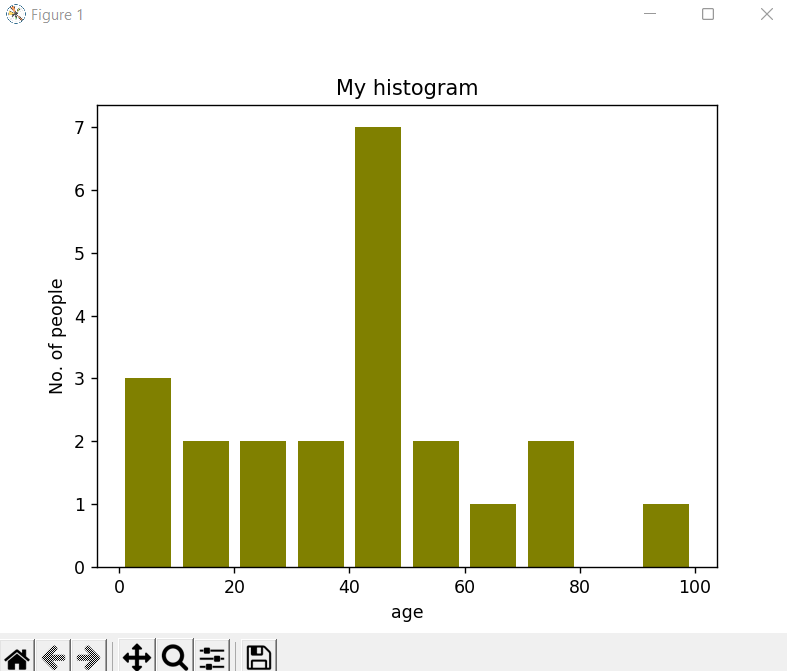
**Scatter plot**



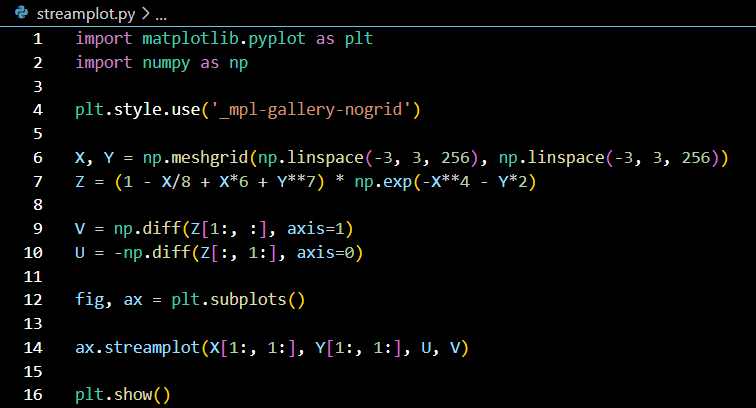


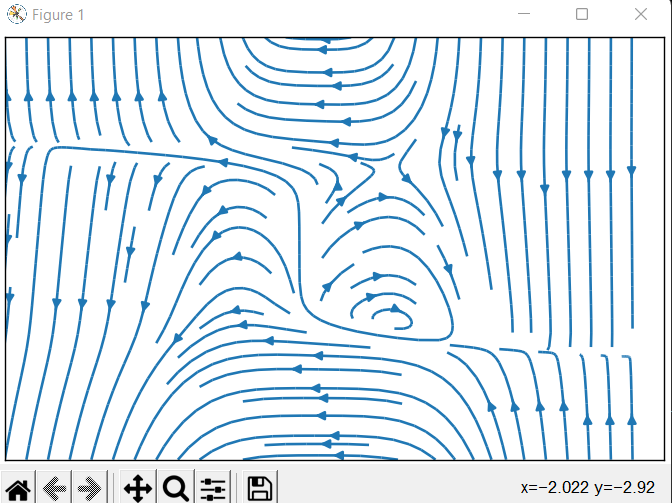
**Histogram**



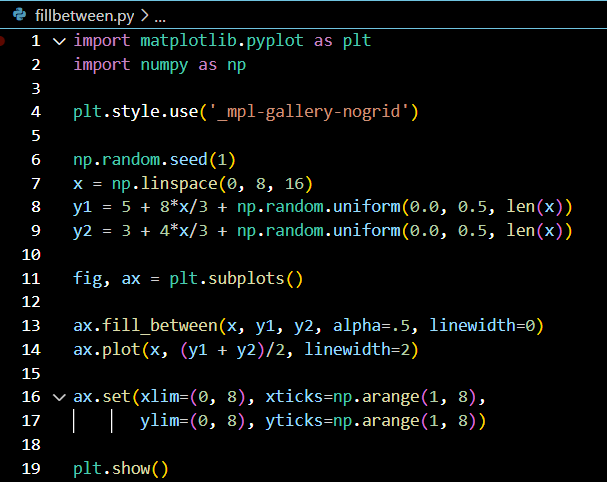


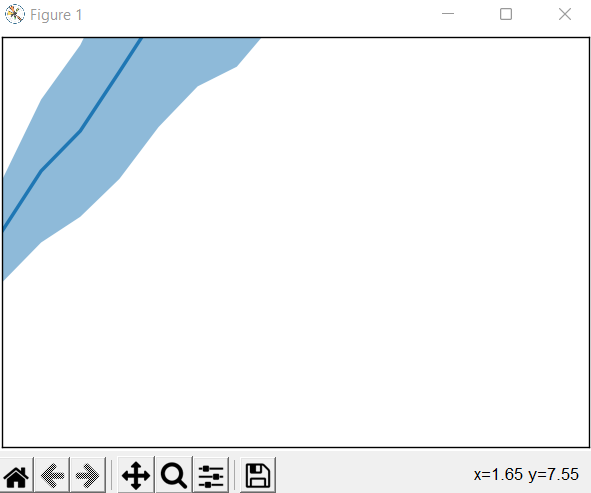
**Stream plot**





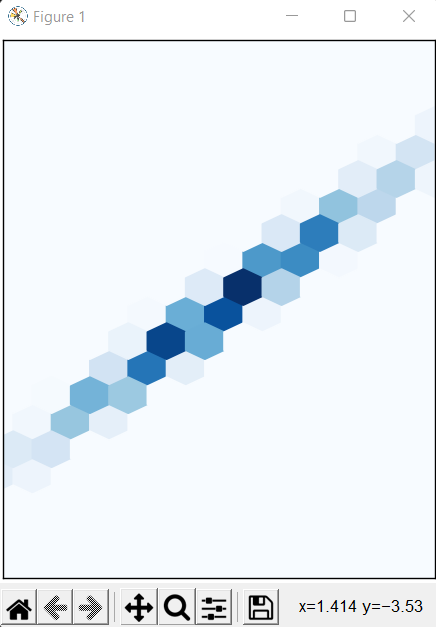
**Fill\_between plot**





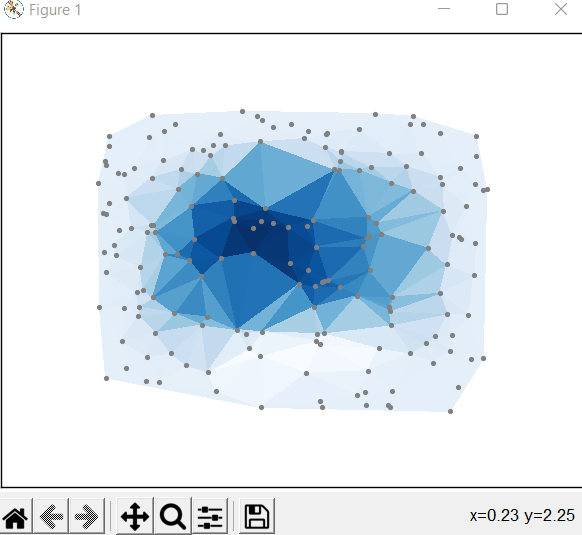
**Hexbin plot**



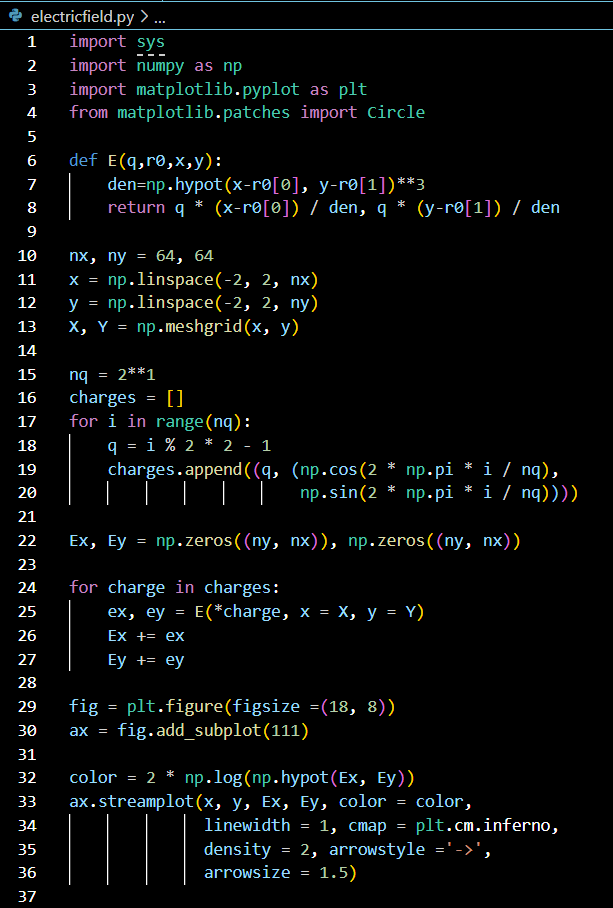


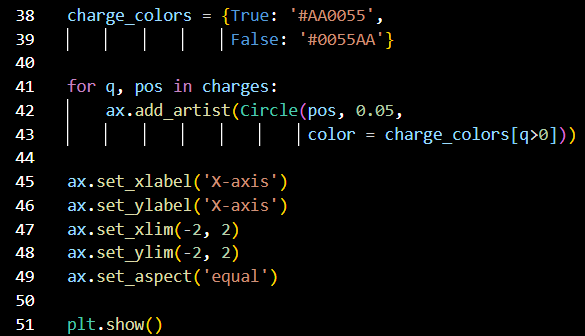
**Tripcolor plot**

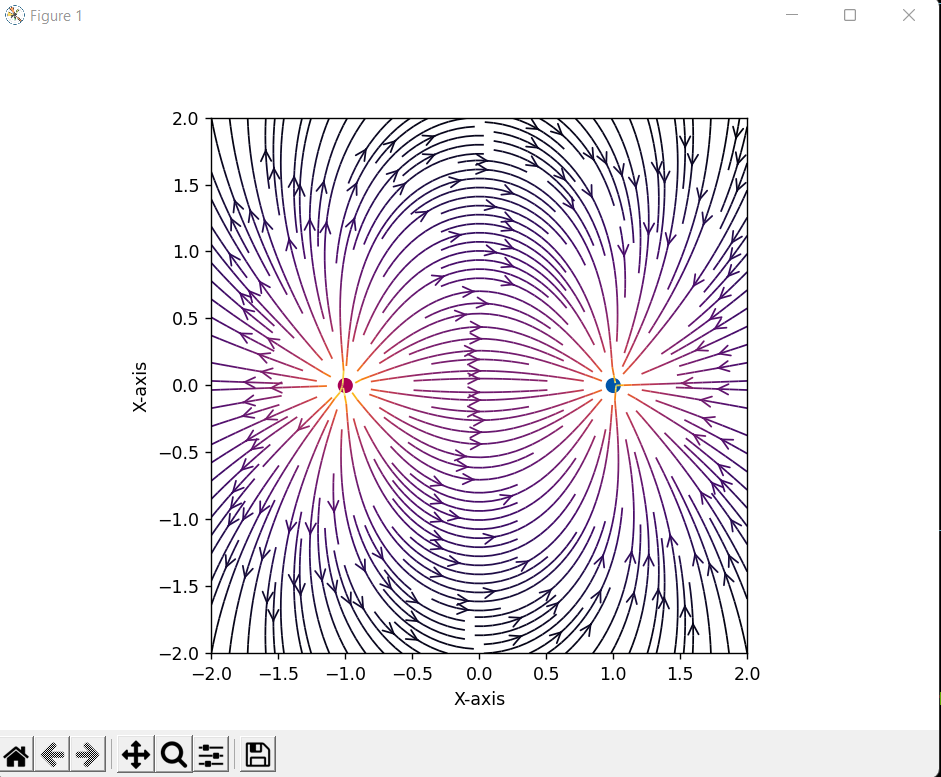




**Electric field**

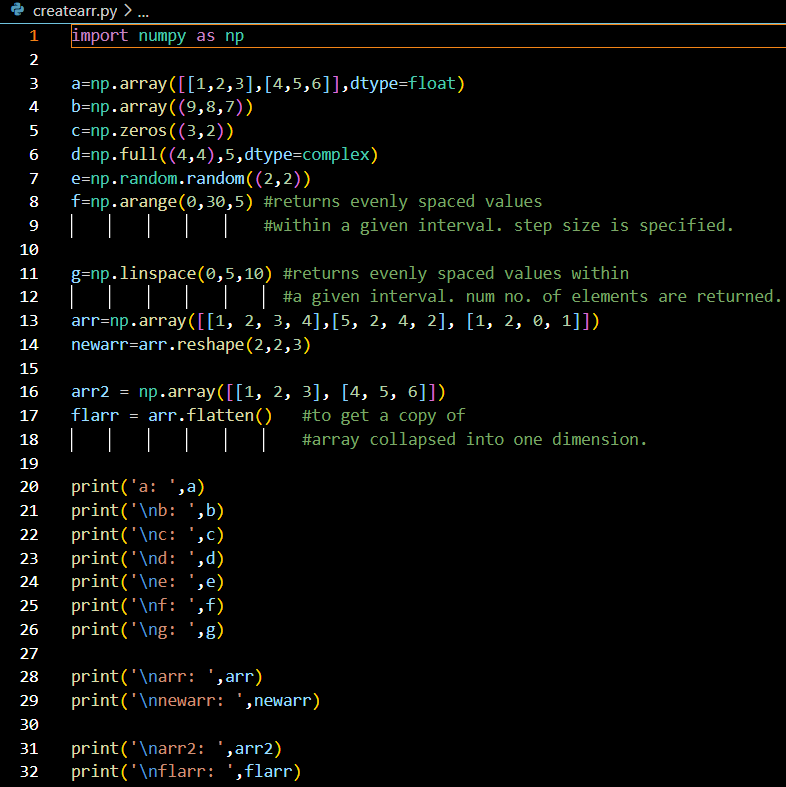
****

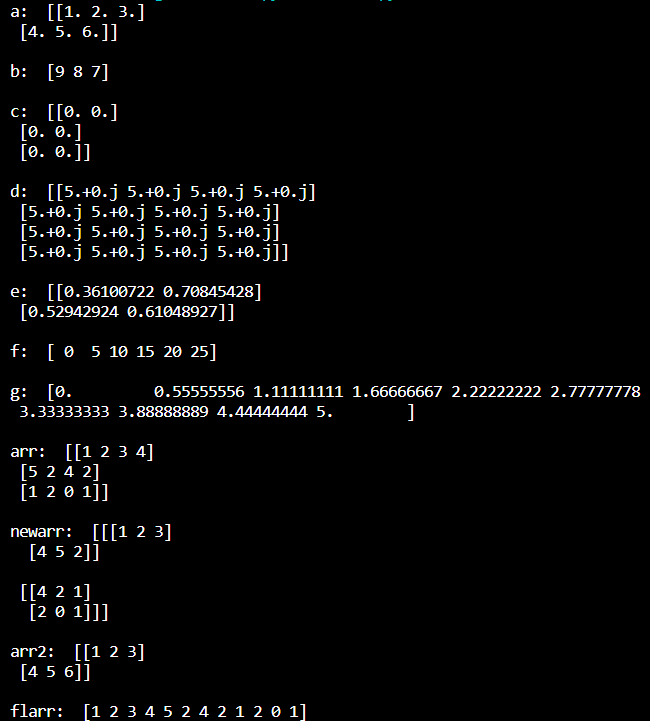
****

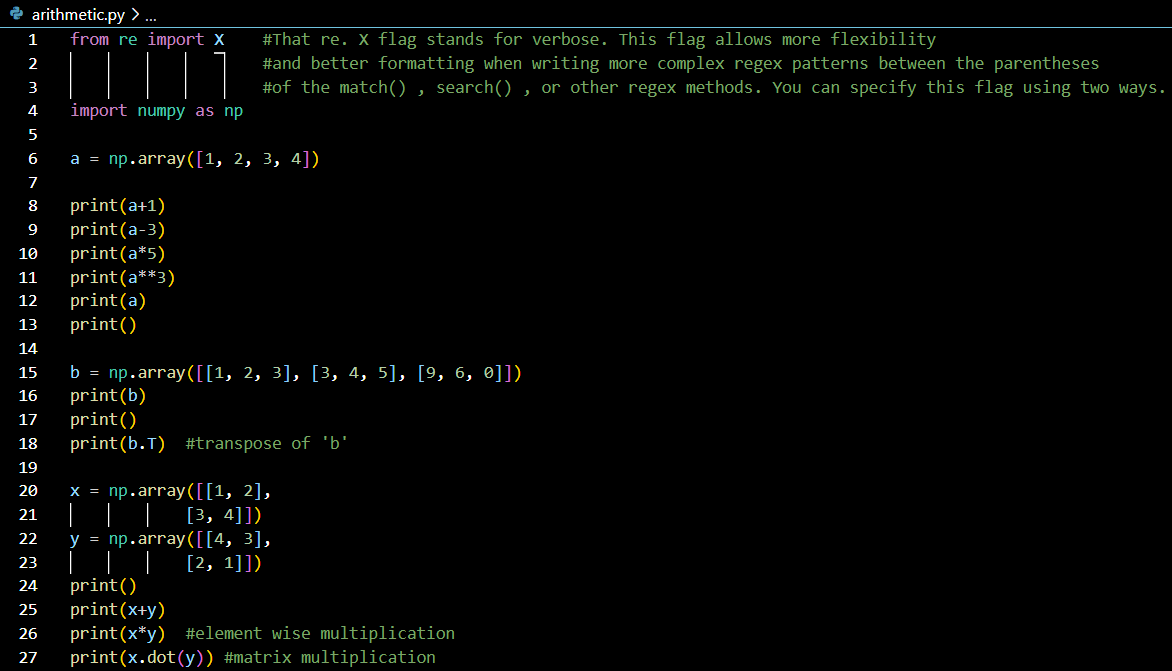
****

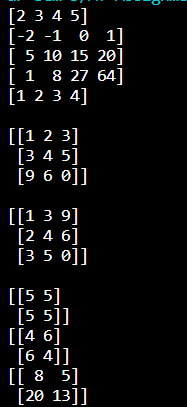
1. **Numpy**

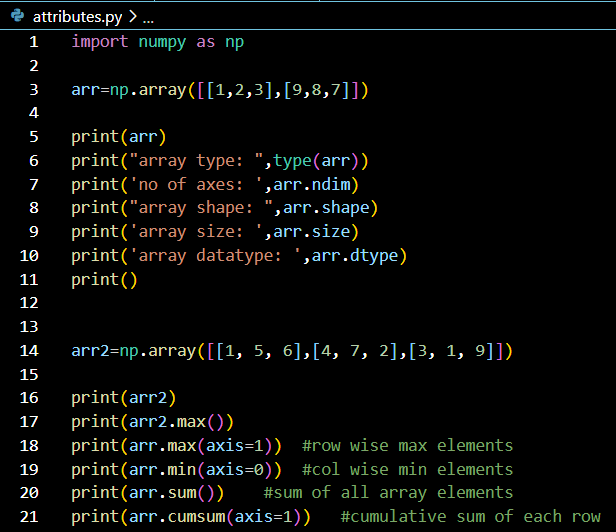
* Numerical Python
* Used for working with multi-dimensional arrays – 50x faster
* Used in linear algebra, Fourier transforms, matrices
* created in 2005 by Travis Oliphant
* scipy + matplotlib => MATLAB
* dimensions => axes
* array class => ndarray or array

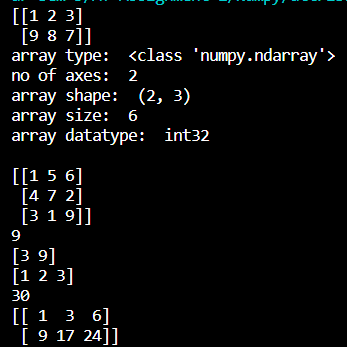


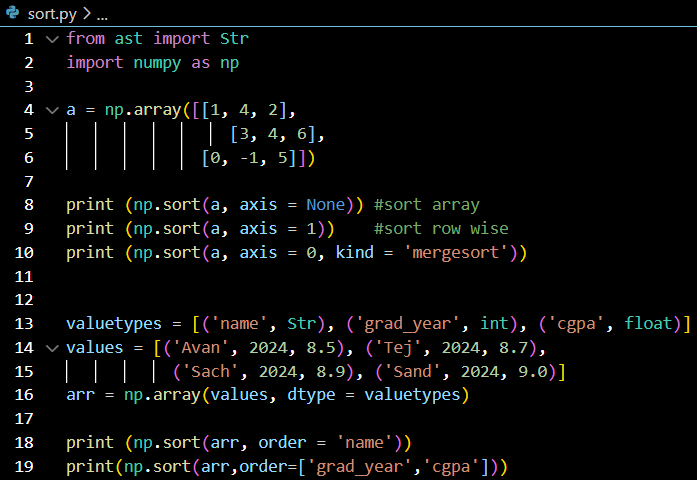


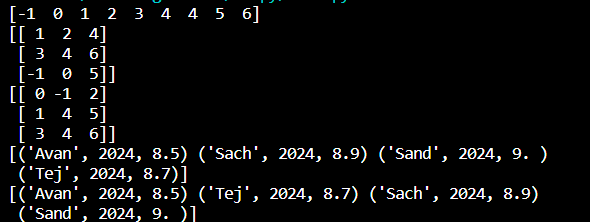




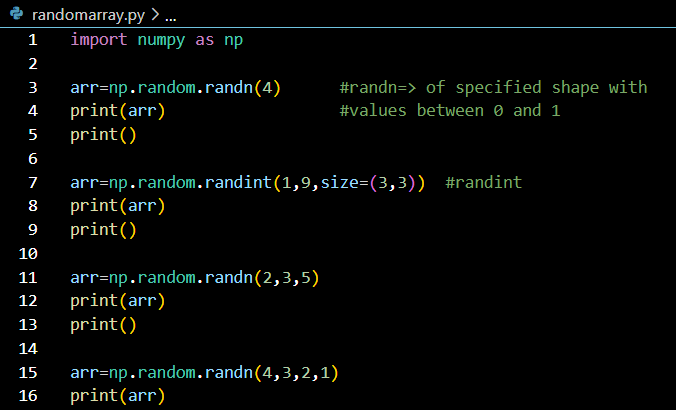


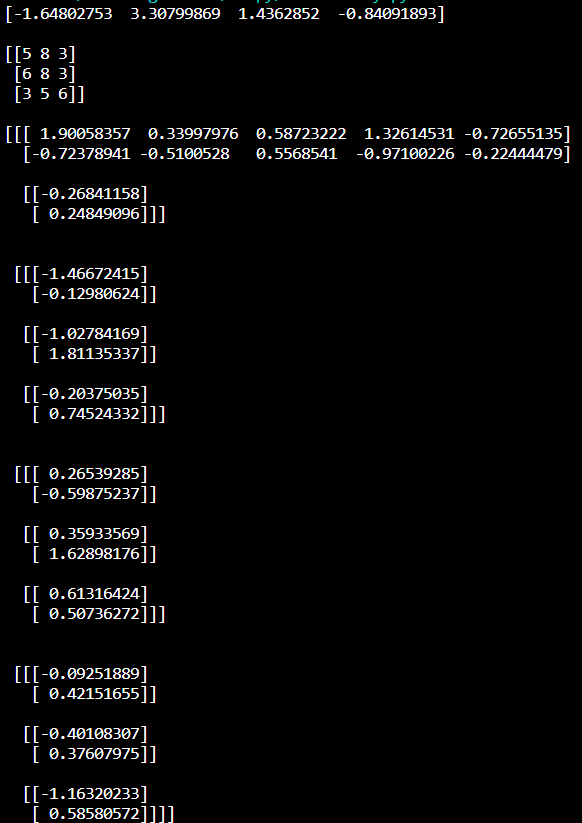






1. **Random array – n dimensions**
   1. Import numpy
   2. randn() => array of specified shape with values between 0 and 1
   3. randint()





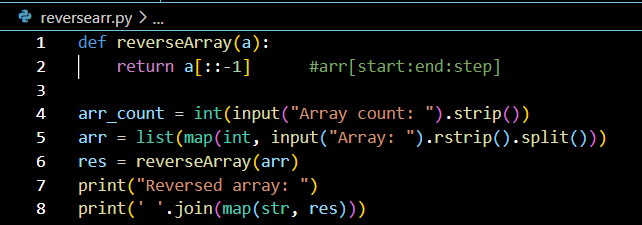
1. **Array Manipulation**

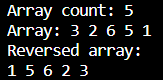
**Easy Problem:**

**Reverse an array of integers**

**Solution:**

* Get input for size of array
* Get input for array
* Define reverseArray() function using **array slicing**
* **Array slicing: array[start:end:step]**
* Print the reversed array

****

****

**Find the hourglass sum for every hour glass in a 6x6 array and print the maximum hourglass sum**

**Solution:**

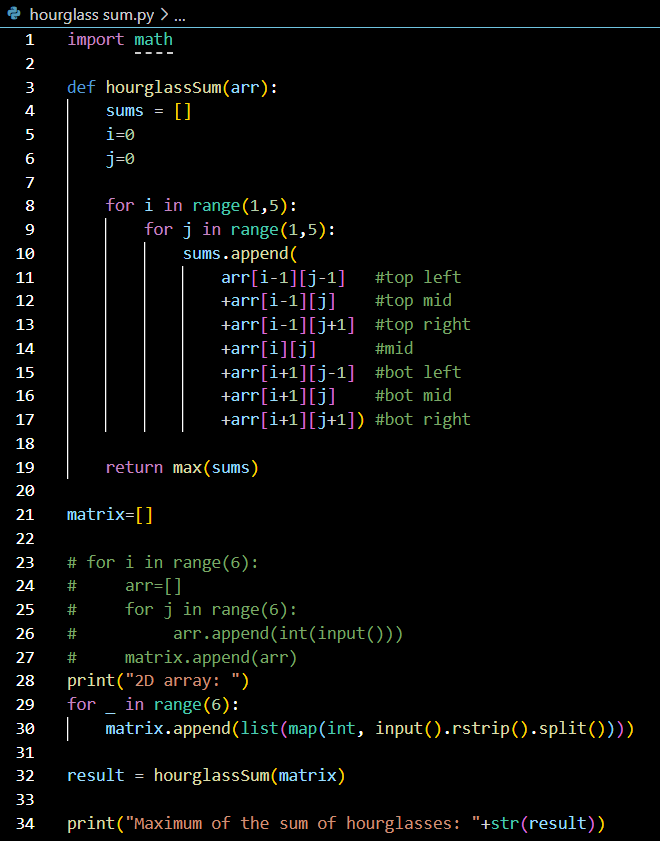
* Get input for 6x6 array
* Find the sum of every hourglass of the pattern:

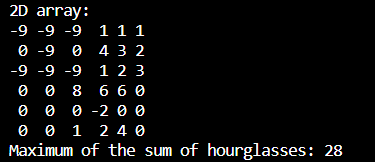
**A B C**

**D**

**E F G**

* i denotes row and j denotes column
* arr[i][j] denotes the center of hourglass D
* Add the elements of hourglass based on the index
* Append it to the sums array
* Return and print the maximum of the sums array

****

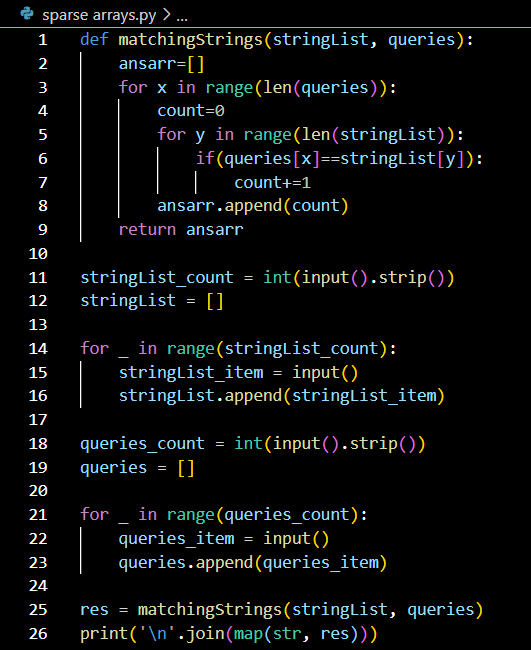
****

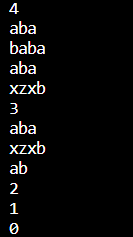
**Medium problem:**

**Frequency of elements of an array that are available in another array**

**Solution:**

* Get input for size of the original stringList array
* Get input for stringList array
* Get input for the size of the queries array
* Get input for queries array
* Define matchingStrings() function to check if the element in queries array is present in stringList array
  + If present, increment the count variable for that element
  + Else, continue checking for the next element
  + Append the count to the ansarr array
* Return and print the elements of the ansarr array





**Hard problem:**

**Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in the array**

**Solution:**

* Get input for size of array and number of operations
* Get queries array for left index a, right index b, and summand k
* Define arrayManipulation() function
* Create initial arr array of size n and with values 0
* For every query, the arr array is summed with k value from index a to b of that array
* Return and print the maximum value of arr array

