ids-pdl05-hwk.ipynb: This Jupyter notebook is provided by Joachim Vogt for the *Python Data Lab* of the module *Introduction to Data Science* offered in Fall 2022 at Jacobs University Bremen. Module instructors are Hilke Brockmann, Adalbert Wilhelm, and Joachim Vogt. Jupyter notebooks and other learning resources are available from a dedicated *module platform*.

Homework assignments: Matplotlib basics

The homework assignments in this notebook supplement the tutorial *Matplotlib basics*.

- Solve the assignments according to the instructions.
- Upload the completed notebook to the module platform.
- Do not forget to enter your name in the markdown cell below.

The homework set carries a total of 20 points. Square brackets in the assignment titles specify individual point contributions.

Name: Hamza Rehman

Preparation

Import the NumPy module in the standard way (prefix `np`). From the matplotlib library import the package `pyplot` using the standard abbreviation `plt`.

In [8]:
```
import numpy as np
import matplotlib.pyplot as plt
```

On the module platform you find data files of the form `pdl05dat_SID.txt` where `SID` is a three-digit student ID, containing the data for the assignments below.

- Identify the file `pdl05dat_SID.txt` with your personal student ID `SID`, then upload it to the working directory, i.e., the folder where this Jupyter notebook resides.
- The data file `pdl05dat_100.txt` and its solution files `pdl05num_100.txt`, `pdl05lin_100.svg`, `pdl05mul_100.svg`, and `pdl05img_100.png` are provided to demonstrate the assignment. Upload the files also to the working directory.

Assignment: Loading numerical data from regular text files [5]

The data file `pdl05dat_SID.txt` contains a two-dimensional array (81 rows and 201 columns) of integer values in the range from 0 to 255 (8 bit). Using the NumPy function `loadtxt()`, read the content of `pdl05dat_SID.txt` into the data matrix `data`. Construct a one-dimensional array `rmean` containing the row means (obtained by averaging along `axis=0`). Compute and print the maximum, the minimum, and the corresponding indices of the vector `rmean`.

In [9]:
```
### Test the code with SID=100, then use your personal student ID for the final run of the notebook.
SID = 280
### Using np.loadtxt(), load the content of pdl05dat_SID.txt into the matrix data.
data=np.loadtxt('pdl05dat_' + str(SID) + '.txt')

### Construct a one-dimensional array rmean containing the row means (obtained by averaging along axis=0).
rmean=np.array(data.mean(axis=0))

### Compute and print the maximum and the corresponding index of rmean.
print('Maximum of row means: {:9.6f}'.format(rmean[np.argmax(rmean)]))
print("Index at maximum: {}\n ".format(np.argmax(rmean)))

### Compute and print the minimum and the corresponding index of rmean.
print('Maximum of row means: {:9.6f} '.format(rmean[np.argmin(rmean)]))
print("Index at minimum: {}".format(np.argmin(rmean)))
```

```
Maximum of row means: 241.560976
Index at maximum: 31

Maximum of row means: 102.439024
Index at minimum: 41
```

Here is the output that you should obtain for `SID = 100`, i.e., for the sample data file `pdl05dat_100.txt`.

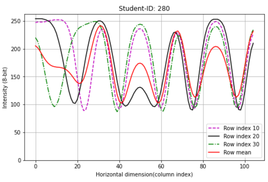
In [1]:
```
f = open('pdl05num_100.txt','r')
num100 = f.read()
f.close()
print(num100)
```

```
Student-ID: 100

Maximum of row means : 245.268293
Index at maximum     :  30

Minimum of row means : 79.682927
Index at minimum     :  15
```

Assignment: Line plots [6]

Using the data from the file `pdl05dat_SID.txt` where `SID` is your personal three-digit student ID, and the object-oriented approach to plotting in matplotlib, create a diagram with the same format as the following sample graphics (for `SID=100`). The first three lines display rows of the data matrix at indices 10,20,30. As before, the row mean (fourth line) results from averaging along `axis=0`. Store the graphics in svg format as `pdl05lin_SID.svg`.

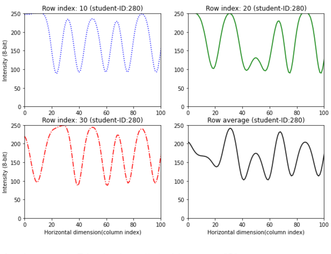
In [10]:
```
### Test the code with SID=100, then use your personal student ID for the final run of the notebook.
SID = 280

### Create new figure and axis objects with figsize=(8,5).
fig,axs = plt.subplots(figsize=(8,5))

### Plot data vectors for row indices 10,20,30.
row10 = data[10,:]
plt.plot(row10, c='m', ls='--', label= 'Row index 10' )
row20 = data[20,:]
plt.plot(row20, c='k', ls='-',  label= 'Row index 20' )
row30 = data[30,:]
plt.plot(row30, c='g', ls='-.', label= 'Row index 30' )


### Compute and plot the row mean.
plt.plot(rmean, c='r', label= 'Row mean')

### Add plot title and the axes annotations.
plt.title('Student-ID: 280')
plt.xlabel('Horizontal dimension(column index)')
plt.ylabel('Intensity (8-bit)')

### Set limits for coordinate axes.
plt.xticks([0,20,40,60,80,100])
plt.yticks([0,50,100,150,200,250])

### Add legend in lower right corner.
plt.legend(loc='lower right')

### Add coordinate coordinate grid.
plt.grid()

### After testing the code with SID=100, save the figure for your personal SID in svg format.
plt.savefig('pdl05lin_' + str(SID) + '.svg')
```



Assignment: Multiple plots in one diagram [6]

Using the data from the file `pdl05dat_SID.txt` where `SID` is your personal three-digit student ID, and the object-oriented approach to plotting in matplotlib, create a diagram with the same format as the following sample graphics (for `SID=100`). The first three panels display rows of the data matrix at indices 10,20,30. As before, the row mean (fourth panel) results from averaging along `axis=0`. Store the graphics in svg format as `pdl05mul_SID.svg`.

In [12]:
```
### Test the code with SID=100, then use your personal student ID for the final run of the notebook.
SID = 280

### Create new (2,2) figure and axes objects with figsize=(10,7).
fig,axs = plt.subplots(2,2,figsize=(10,7))

### Create plot at the [0,0] position with title, y-axis annotation, and proper limits.
axs[0,0].plot(row10, c='b', ls=':')
axs[0,0].set_title("Row index: 10 (student-ID:280)")
axs[0,0].set_xlim(0,100)
axs[0,0].set_ylim(0,250)
axs[0,0].set_ylabel('Intensity (8-bit)')

### Create plot at the [0,1] position with title and proper limits.
axs[0,1].plot(row20, c='g', ls='-.')
axs[0,1].set_title("Row index: 20 (student-ID:280)")
axs[0,1].set_xlim(0,100)
axs[0,1].set_ylim(0,250)

### Create plot at the [1,0] position with title, x-axis and y-axis annotations, and proper limits.
axs[1,0].plot(row30, c='r', ls='-.')
axs[1,0].set_title("Row index: 30 (student-ID:280)")
axs[1,0].set_xlim(0,100)
axs[1,0].set_ylim(0,250)
axs[1,0].set_xlabel('Horizontal dimension(column index)')
axs[1,0].set_ylabel('Intensity (8-bit)')



### Create plot at the [1,1] position with title, x-axis annotation, and proper limits.
axs[1,1].plot(rmean, c='k')
axs[1,1].set_title("Row average (student-ID:280)")
axs[1,1].set_xlim(0,100)
axs[1,1].set_ylim(0,250)
axs[1,1].set_xlabel('Horizontal dimension(column index)')

### After testing the code with SID=100, save the figure for your personal SID in svg format.
fig.savefig('pdl05mul_' + str(SID) + '.svg')
```



Assignment: Displaying digital images [3]

Using the data from the file `pdl05dat_SID.txt` where `SID` is your personal three-digit student ID, and the MATLAB-style approach to plotting in matplotlib, create a diagram with the same format as the following sample graphics (for `SID=100`). Store the graphics in png format as `pdl05img_SID.png`.

In [7]:
```
### Test the code with SID=100, then use your personal student ID for the final run of the notebook.
SID = 280

### Create new figure object  with figsize=(8,5) and dpi=100.
plt.figure(figsize=(8,5),dpi=100)

### Plot the data matrix as a figure using the grayscale color map.
image=plt.imshow(data,cmap='gray')
fig.colorbar(image)

### Add title and axes annotations.
plt.title("Student-ID:280")
plt.xlabel('Horizontal dimension(column index)')
plt.ylabel('Vertical dimension(row index)')

### After testing the code with SID=100, save the figure for your personal SID in png format.
fig.savefig('pdl05img_' + str(SID) + '.png')
```

```
/var/folders/rj/w4znz25530d7q7nl8vf609gr0000gn/T/ipykernel_720/557985562.py:9: MatplotlibDeprecationWarning: Starting from Matplotlib 3.6, colorbar() will steal space from the mappable's axes, rather than from the current axes, to place the colorbar.  To silence this warning, explicitly pass the 'ax' argument to colorbar().
  fig.colorbar(image)
```