```
### the following code snippet is given by the website for retrieving the database

%pip install ucimlrepo

from ucimlrepo import fetch_ucirepo

# fetch dataset
iris = fetch_ucirepo(id=53)

# data (as pandas dataframes)
X = iris.data.features
y = iris.data.targets

# metadata
print(iris.metadata)

# variable information
print(iris.variables)
```

```
Defaulting to user installation because normal site-packages is not
writeable
Requirement already satisfied: ucimlrepo in c:\users\alepa\appdata\
roaming\python\python311\site-packages (0.0.6)
Note: you may need to restart the kernel to use updated packages.
{'uci_id': 53, 'name': 'Iris', 'repository_url':
'https://archive.ics.uci.edu/dataset/53/iris', 'data_url':
'https://archive.ics.uci.edu/static/public/53/data.csv', 'abstract':
'A small classic dataset from Fisher, 1936. One of the earliest known
datasets used for evaluating classification methods.\n', 'area':
'Biology', 'tasks': ['Classification'], 'characteristics':
['Tabular'], 'num_instances': 150, 'num_features': 4, 'feature_types':
['Real'], 'demographics': [], 'target_col': ['class'], 'index_col':
None, 'has_missing_values': 'no', 'missing_values_symbol': None,
'year_of_dataset_creation': 1936, 'last_updated': 'Tue Sep 12 2023',
'dataset_doi': '10.24432/C56C76', 'creators': ['R. A. Fisher'],
'intro_paper': {'title': 'The Iris data set: In search of the source
of virginica', 'authors': 'A. Unwin, K. Kleinman', 'published_in':
'Significance, 2021', 'year': 2021, 'url':
'https://www.semanticscholar.org/paper/4599862ea877863669a6a8e63a3c707
a787d5d7e', 'doi': '1740-9713.01589'}, 'additional_info': {'summary':
'This is one of the earliest datasets used in the literature on
classification methods and widely used in statistics and machine
learning.  The data set contains 3 classes of 50 instances each, where
each class refers to a type of iris plant.  One class is linearly
separable from the other 2; the latter are not linearly separable from
each other.\n\nPredicted attribute: class of iris plant.\n\nThis is an
exceedingly simple domain.\n\nThis data differs from the data
presented in Fishers article (identified by Steve Chadwick,
spchadwick@espeedaz.net ).  The 35th sample should be:
```

```
4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth
feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the
errors are in the second and third features.  ', 'purpose': 'N/A',
'funded_by': None, 'instances_represent': 'Each instance is a plant',
'recommended_data_splits': None, 'sensitive_data': None,
'preprocessing_description': None, 'variable_info': None, 'citation':
None}}
           name     role         type demographic  \
0  sepal length  Feature   Continuous        None
1   sepal width  Feature   Continuous        None
2  petal length  Feature   Continuous        None
3   petal width  Feature   Continuous        None
4         class   Target  Categorical        None

                                          description units
missing_values
0                                                 None    cm
no
1                                                 None    cm
no
2                                                 None    cm
no
3                                                 None    cm
no
4  class of iris plant: Iris Setosa, Iris Versico...  None
no
```

```python
import numpy as np

# in order to center the data, I used the forumla in page 2/32 of
chapter 1.3 on PCA and let it be assigned to a function,
# then I appled it to our initial data X and assigned the centered
data to a new variable Xc
center_function = lambda x: x - x.mean()

Xc = center_function(X)

# numpy gives us a straightforward SVD formula that I used for the
decomposition, then used
# list manipulationto print the principal components, which are of
course the columns of U

U, S, VT = np.linalg.svd(Xc, full_matrices=False)

# sklearn gives us straightforward PCA functions just like the
aformentioned numpy SVD function
# matplotlib for the requested graphs

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

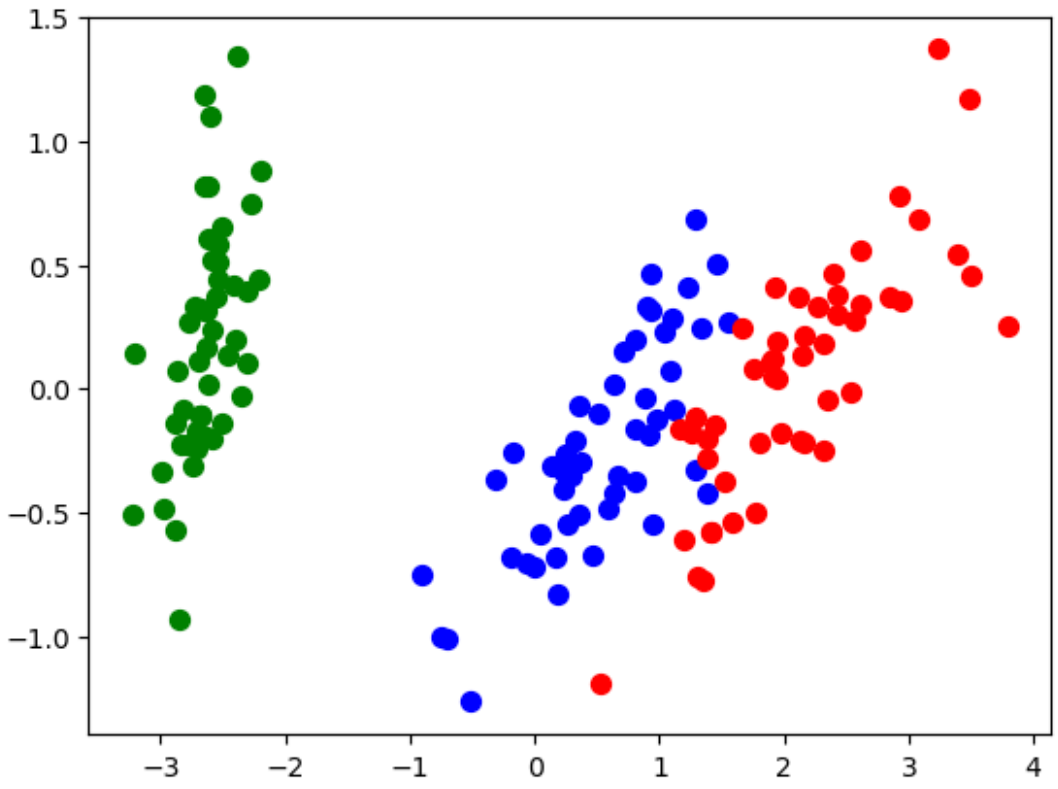
```
# exercise 2.3.3 asks for a 2D graph, so I set the components to two
pca = PCA(n_components=2)
X_pca2D = pca.fit_transform(Xc)

# we access y's single column
ycol = y.iloc[:, 0].unique()

# I will youse gree, blue and red both for this and the following
graph
colors = ['g', 'b', 'r']

for ycol, color in zip(ycol, colors):
    # Similarly, ensure 'y' is a Series or a numpy array for
comparison
    plt.scatter(X_pca2D[y.iloc[:, 0] == ycol, 0], X_pca2D[y.iloc[:, 0]
== ycol, 1], c=color, s=50, label=ycol)

plt.show()
```




```
from mpl_toolkits.mplot3d import Axes3D

# I use the same way to access y's columns
ycol = y.iloc[:, 0].unique()
```

```
# baseline for a 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot each class with its corresponding color in the 3D PCA space
for ycol, color in zip(ycol, colors):
    indicesToKeep = y.iloc[:, 0] == ycol
    ax.scatter(U[indicesToKeep, 0],  # first principal component
               U[indicesToKeep, 1],  # second principal component
               U[indicesToKeep, 2],  # third principal component
               c=color, s=50, label=ycol)

plt.show()
```