```
print("AI-LAB 2")
print("SHAYAN HASSAN 20K-1873")
```

```
AI-LAB 2
SHAYAN HASSAN 20K-1873
```

Example 1

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
#Create a 0-D array with value 42
arr = np.array(42)
print(arr)
#Create a 1-D array containing the values 1,2,3,4,5:
arr = np.array([1, 2, 3, 4, 5])
print(arr)
#Create a 2-D array containing two arrays with the values 1,2,3 and
4,5,6:
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
#Create a 3-D array with two 2-D arrays, both containing two arrays
with the values 1,2,3 and 4,5,6:
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr)
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
42
[1 2 3 4 5]
[[1 2 3]
 [4 5 6]]
[[[1 2 3]
  [4 5 6]]

 [[1 2 3]
  [4 5 6]]]
```

Example2

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[1])
print(arr[2] + arr[3])
#Access 2D array:
#Access the element on the first row, second column:
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
```

```
print('5th element on 2nd row: ', arr[1, 4])
#Access 3d Array:
#Access the third element of the second array of the first array:
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
print(arr[0, 1, 2])
#Negative Indexing:
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Last element from 2nd dim: ', arr[1, -1])
```

```
2
7
2nd element on 1st row:  2
5th element on 2nd row:  10
6
Last element from 2nd dim:  10
```

Example3

```
#Slicing arrays:
#Slicing in python means taking elements from one given index to
another given index.
#We pass slice instead of index like this: [start:end].
#We can also define the step, like this: [start:end:step]
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
#Slice elements from index 1 to index 5 from the following array:
print(arr[1:5])
#Slice elements from index 4 to the end of the array:
print(arr[4:])
#Slice elements from the beginning to index 4 (not included):
print(arr[:4])
#Negative Slicing:
#Slice from the index 3 from the end to index 1 from the end:
print(arr[-3:-1])
#STEP
#Use the step value to determine the step of the slicing:
#Return every other element from index 1 to index 5:
print(arr[1:5:2])
#Return every other element from the entire array:
print(arr[::2])
#Slicing 2-D Arrays
#From the second element, slice elements from index 1 to index 4 (not
included):
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
print(arr[1, 1:4])
#From both elements, return index 2:
print(arr[0:2, 2])
#From both elements, slice index 1 to index 4 (not included), this
will return a 2-D array:
print(arr[0:2, 1:4])
```

```
[2 3 4 5]
[5 6 7]
[1 2 3 4]
[5 6]
[2 4]
[1 3 5 7]
[7 8 9]
[3 8]
[[2 3 4]
 [7 8 9]]
```

Example 4

```
#Checking the Data Type of an Array
#The NumPy array object has a property called dtype that returns the
data type of the array:
#Get the data type of an array object:
arr = np.array([1, 2, 3, 4])
print(arr.dtype)
#Get the data type of an array containing strings:
arr = np.array(['apple', 'banana', 'cherry'])
print(arr.dtype)
#Iterating Arrays
#Iterating means going through elements one by one.
#As we deal with multi-dimensional arrays in numpy, we can do this
using basic for loop of python.
#If we iterate on a 1-D array it will go through each element one by
one.
arr = np.array([1, 2, 3])
for x in arr:
    print(x)
#Iterate on each scalar element of the 2-D array:
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
    for y in x:
        print(y)
#Iterate on the elements of the following 3-D array:
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr:
    print(x)
#To return the actual values, the scalars, we have to iterate the
arrays in each dimension.
#Iterate down to the scalars:
```

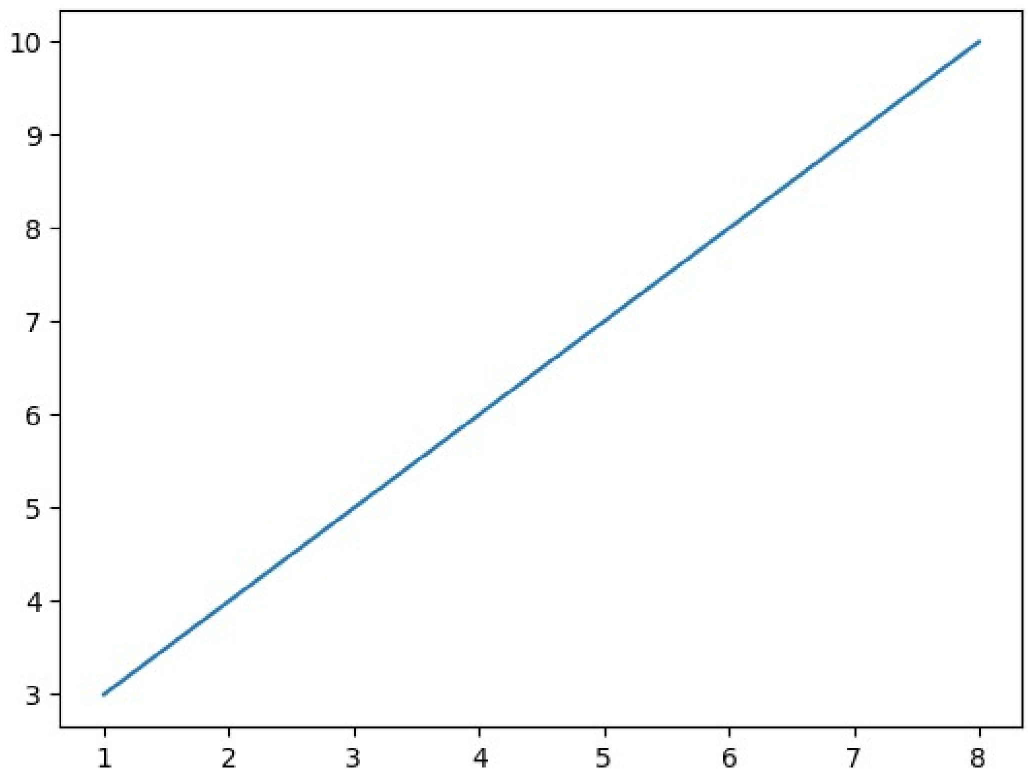
```
int32
<U6
1
2
3
1
```

```
2
3
4
5
6
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
```

Example5

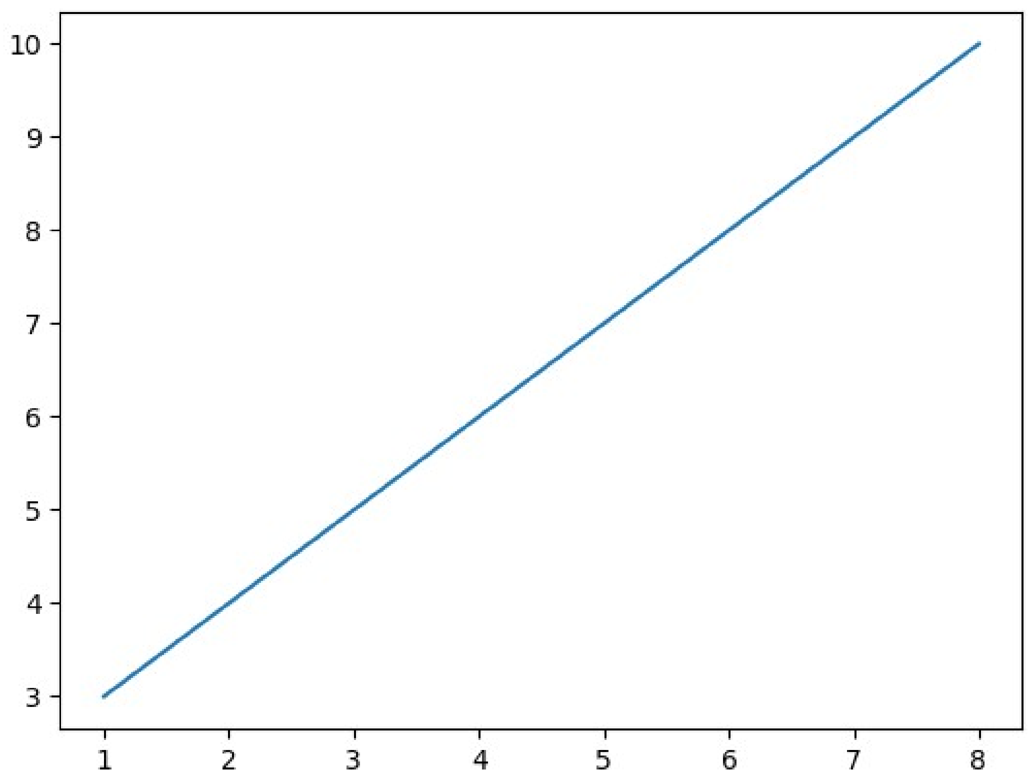
```python
#Plotting x and y points
# The plot() function is used to draw points (markers) in a diagram.
# By default, the plot() function draws a line from point to point.
# The function takes parameters for specifying points in the diagram.
# Parameter 1 is an array containing the points on the x-axis.
# Parameter 2 is an array containing the points on the y-axis.
# If we need to plot a line from (1, 3) to (8, 10), we have to pass
two arrays [1, 8] and [3, 10] to the plot

#Draw a line in a diagram from position (1, 3) to position (8, 10):
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```



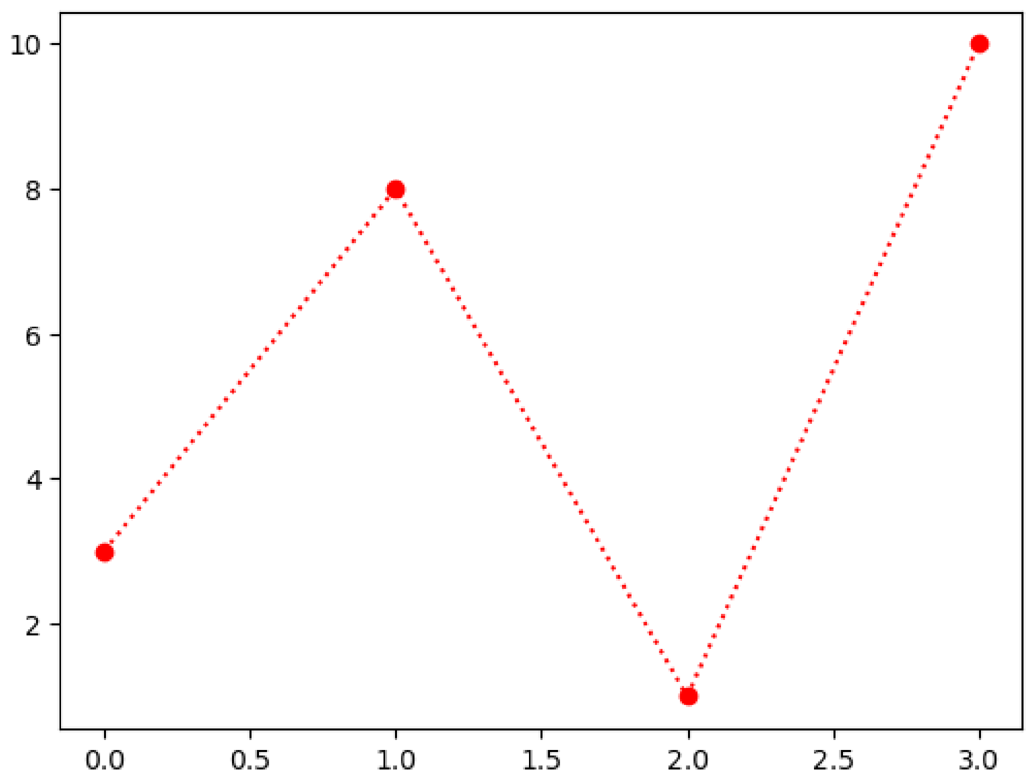
Example6

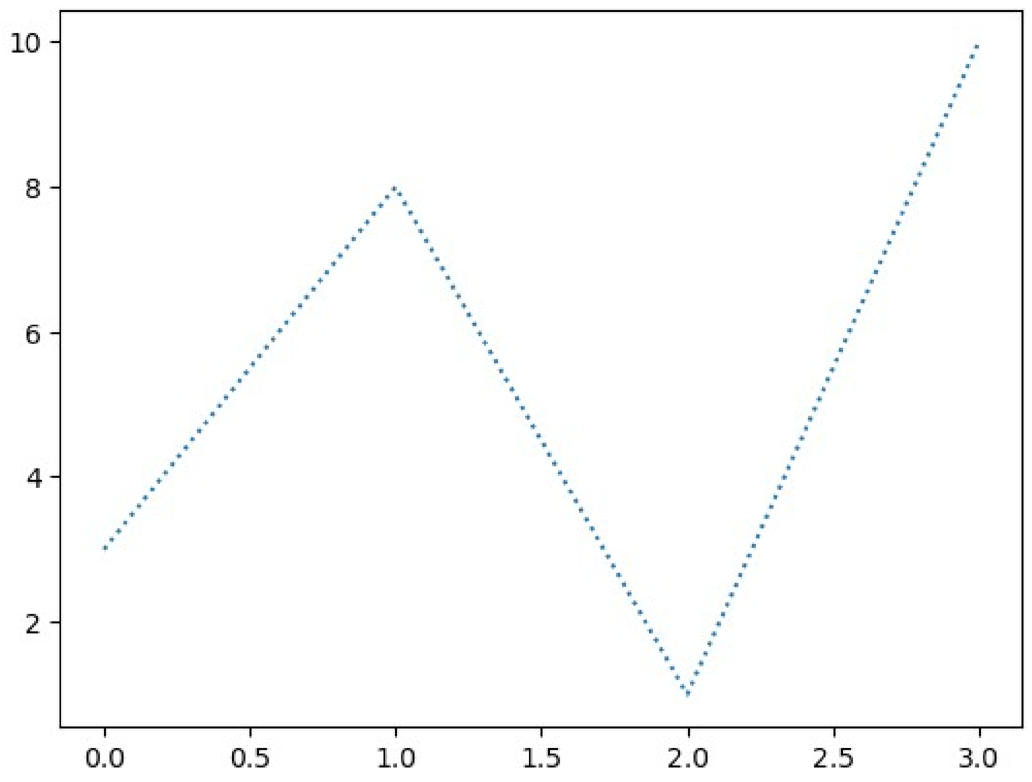
```
#Plotting x and y points
# The plot() function is used to draw points (markers) in a diagram.
# By default, the plot() function draws a line from point to point.
# The function takes parameters for specifying points in the diagram.
# Parameter 1 is an array containing the points on the x-axis.
# Parameter 2 is an array containing the points on the y-axis.
# If we need to plot a line from (1, 3) to (8, 10), we have to pass
two arrays [1, 8] and [3, 10] to the plot
#Draw a line in a diagram from position (1, 3) to position (8, 10):
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```



Example7

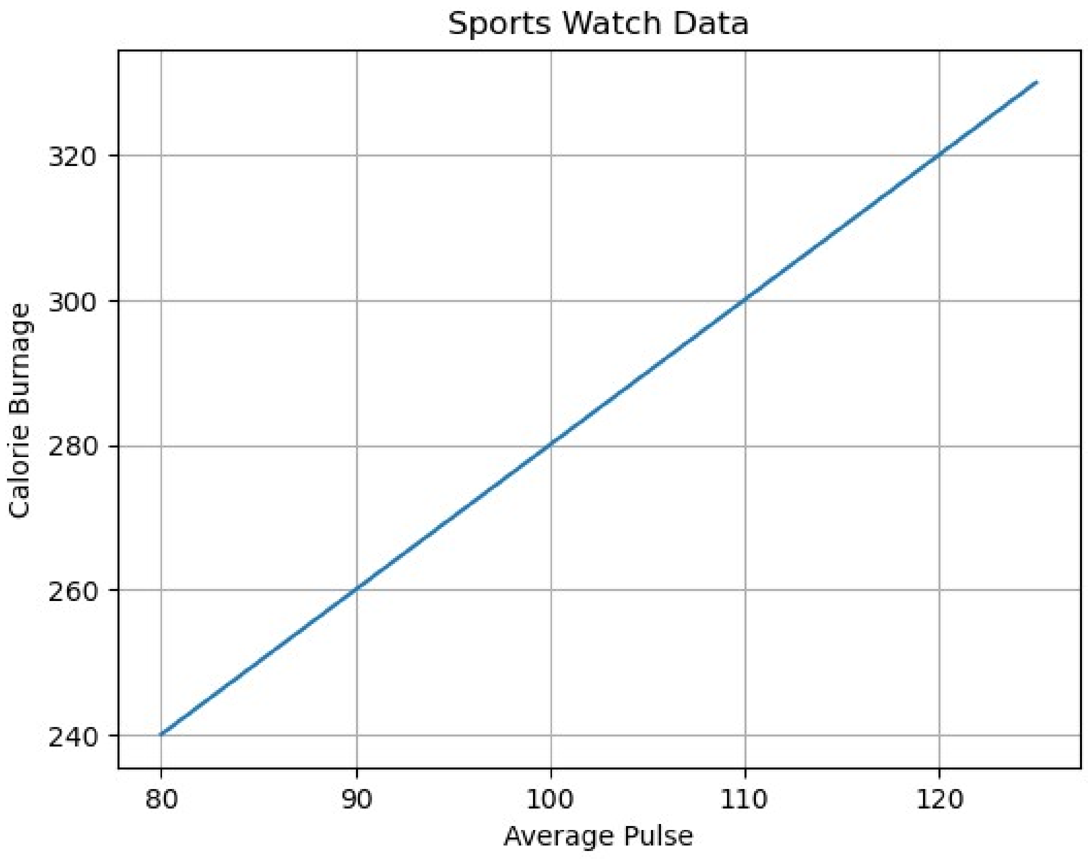
```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, 'o:r')
plt.show()
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```


10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0


10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0

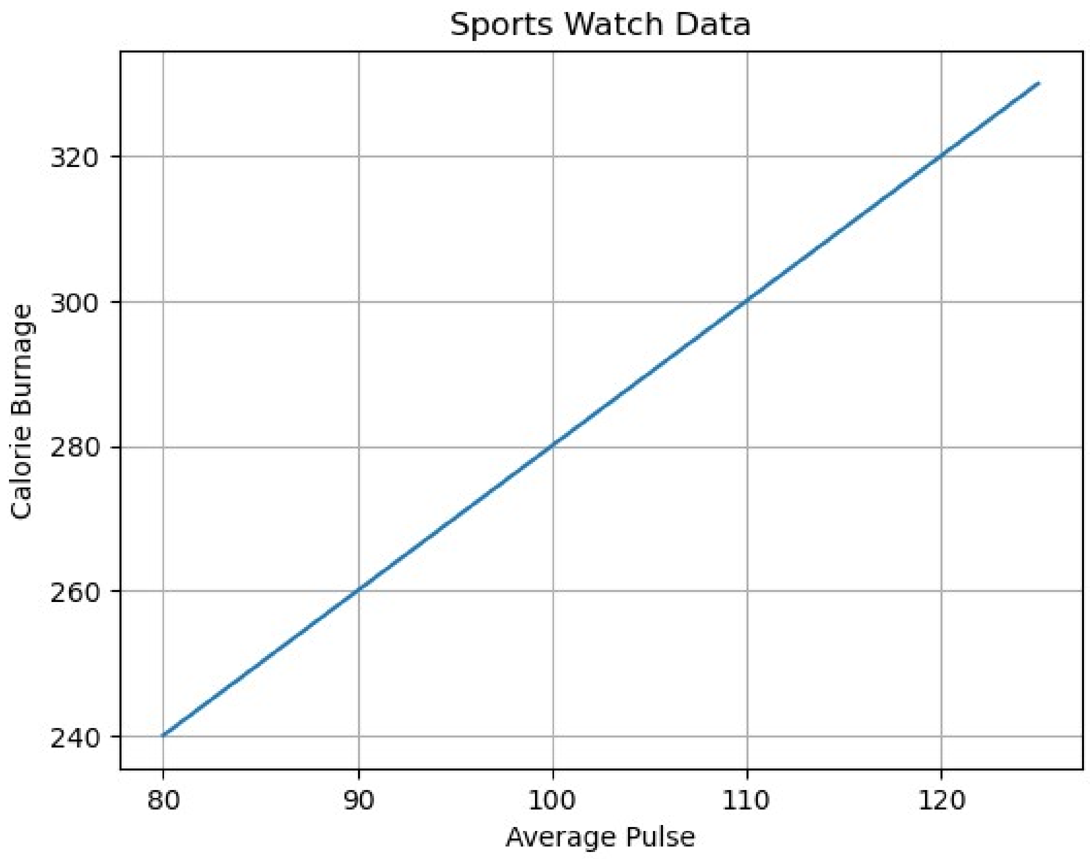
Example 8

```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x, y)
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.grid()
plt.show()
```



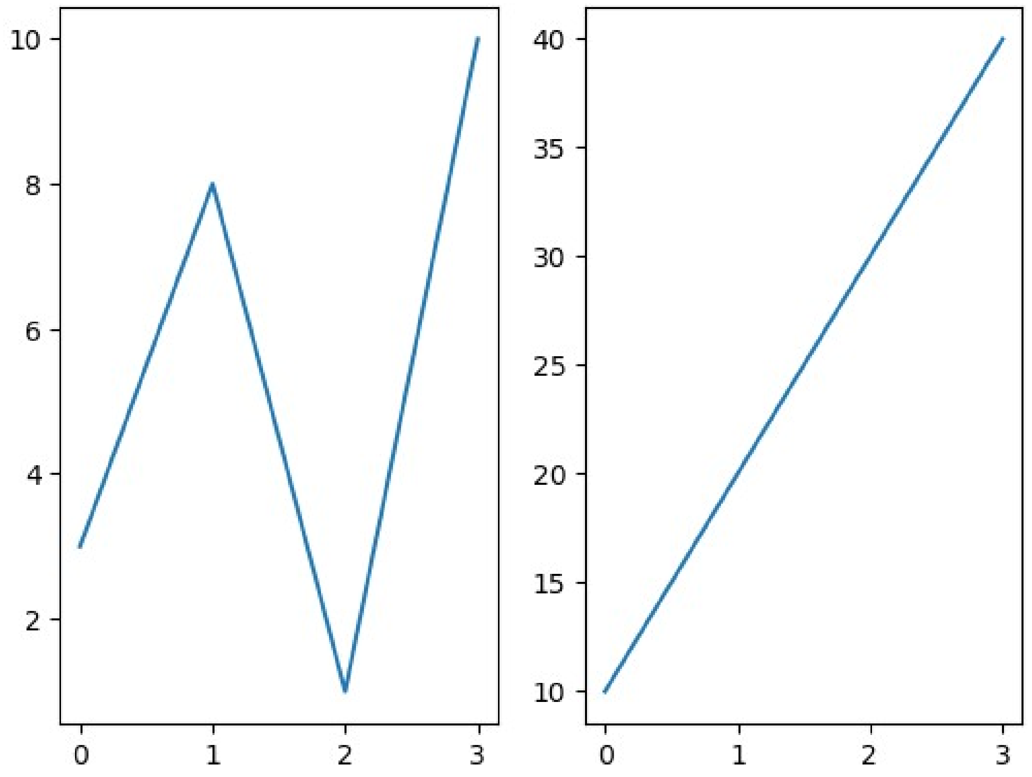
Example9

```
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x, y)
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.grid()
plt.show()
```



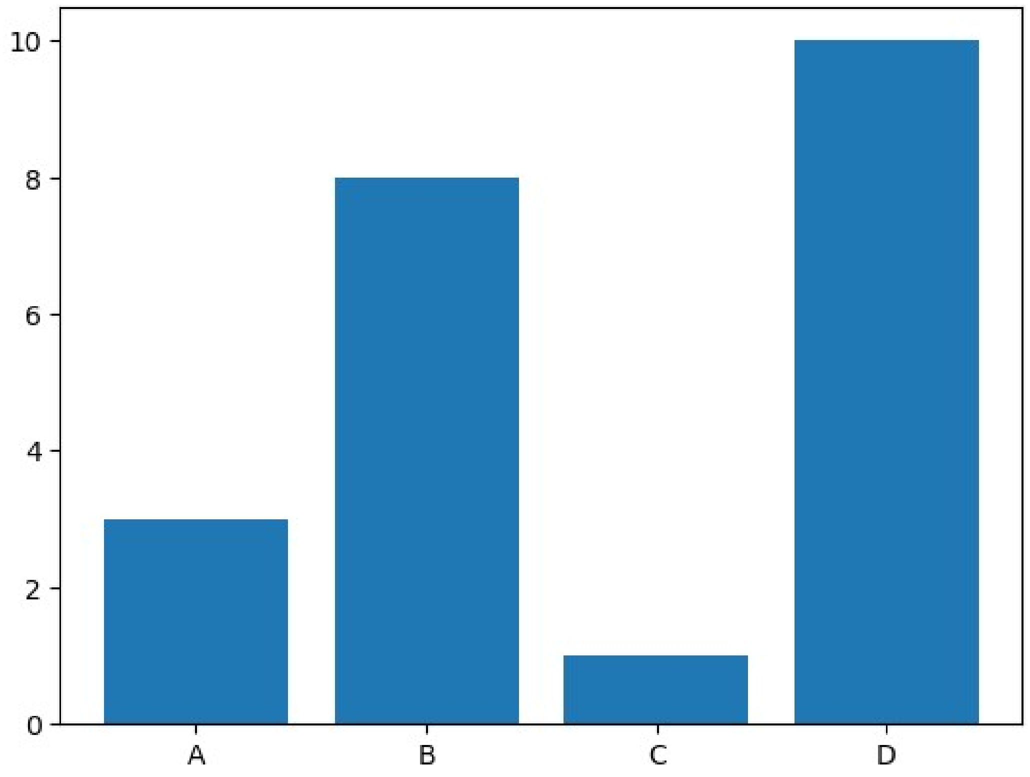
Example10

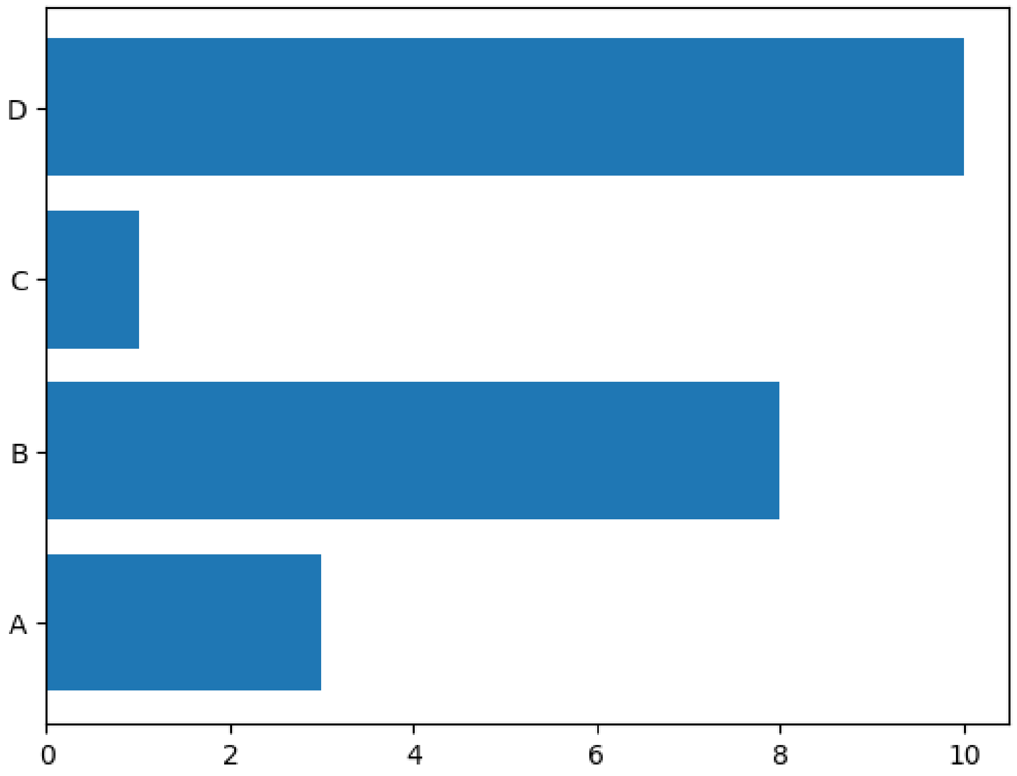
```
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```

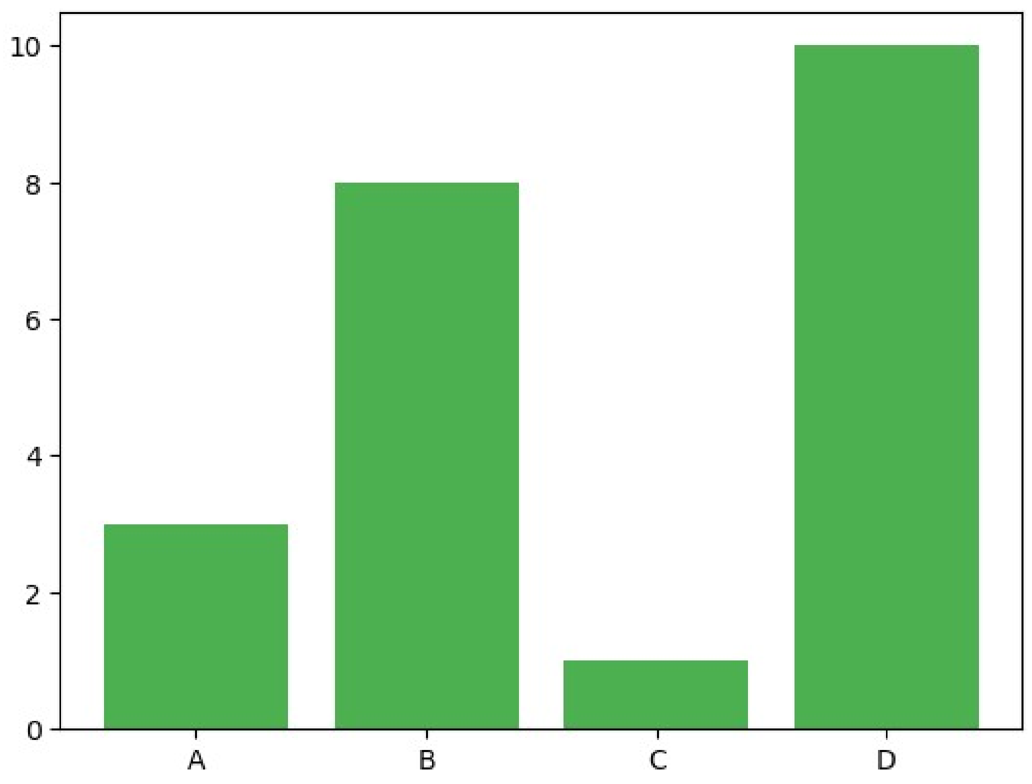


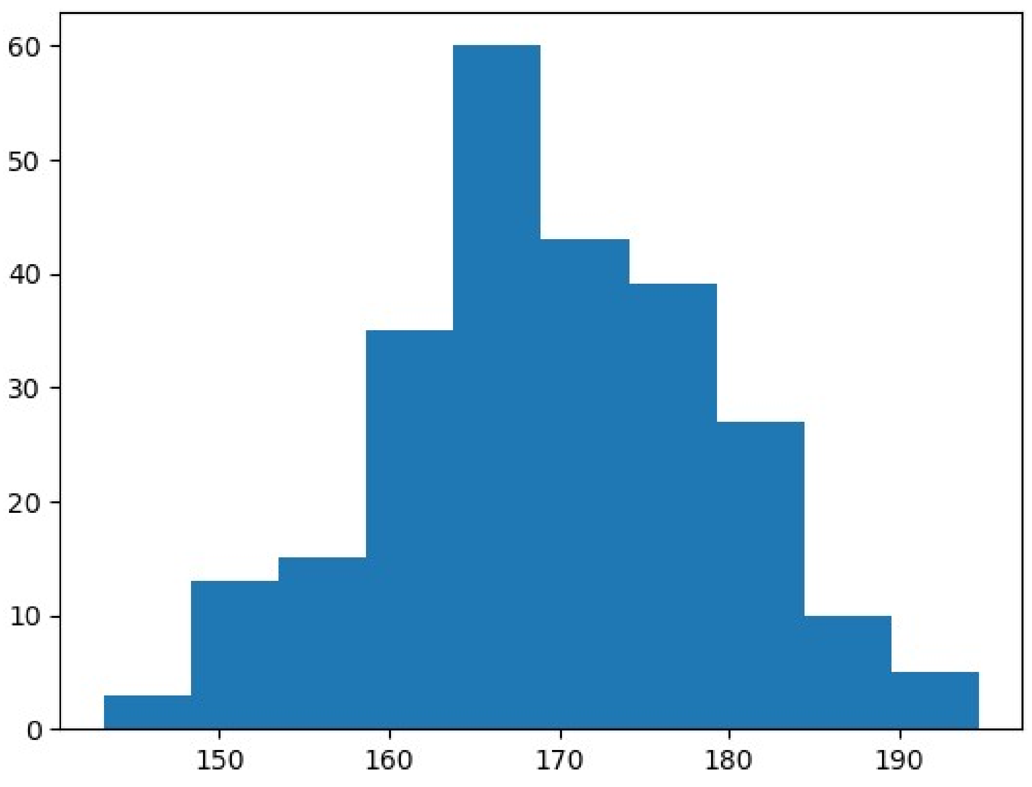
Example11

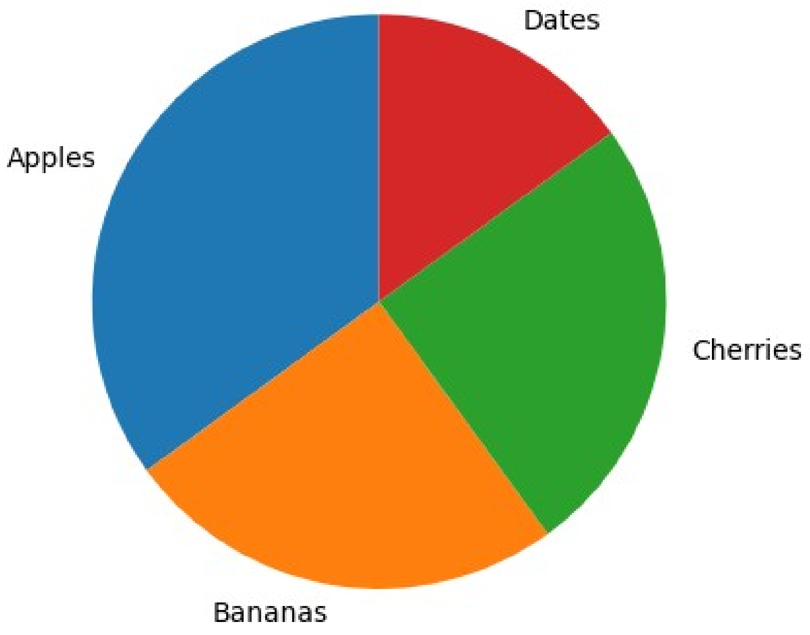
```
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
#for horizontal bar use 'barh'
plt.barh(x, y)
plt.show()
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x, y, color = "#4CAF50")
plt.show()
#histogram
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
#Pie Chart
y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
plt.pie(y, labels = mylabels, startangle = 90)
plt.show()
```


10
8
6
4
2
0
A
B
C
D


D
C
B
A
0
2
4
6
8
10


10
8
6
4
2
0
A
B
C
D


60
50
40
30
20
10
0
150
160
170
180
190



Example12

```
import pandas as pd
df = pd.read_csv("C:\\Users\\DELL\\Downloads\\data.csv")
df.head()
print(df.shape)
print(df.columns)
print(df.info())
df.describe()
df["Pulse"].mean()
```

```
(169, 4)
Index(['Duration', 'Pulse', 'Maxpulse', 'Calories'], dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None

107.46153846153847
```

Example13

```
import pandas as pd
mydataset = {
'cars': ["BMW", "Volvo", "Ford"],
'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Example14

```
import pandas as pd
data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
#refer to the row index:
print(df.loc[0])
#use a list of indexes:
print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
calories    420
duration     50
Name: 0, dtype: int64
   calories  duration
0       420        50
1       380        40
```

EXample15

```
df = pd.read_csv('C:\\Users\\DELL\\Downloads\\data.csv')
print(df)
#Analyzing dataframe:
#The head() method returns the headers and a specified number of rows,
starting from the top.
df = pd.read_csv('C:\\Users\\DELL\\Downloads\\data.csv')
#printing the first 10 rows of the DataFrame:
```

```
print(df.head(10))
#There is also a tail() method for viewing the last rows of the
DataFrame.
#The tail() method returns the headers and a specified number of rows,
starting from the bottom.
#Print the last 5 rows of the DataFrame:
print(df.tail())
#The DataFrames object has a method called info(), that gives you more
information about the data set.
#Print information about the data:
print(df.info())
#Cleaning Empty cell:
new_df = df.dropna()
#If you want to change the original DataFrame, use the inplace = True
argument:
#Remove all rows with NULL values:
df.dropna(inplace = True)
#The fillna() method allows us to replace empty cells with a value:
#Replace NULL values with the number 130:
df.fillna(130, inplace = True)
#Replace NULL values in the "Calories" columns with the number 130:
df["Calories"].fillna(130, inplace = True)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    103       135     340.0
3        45    109       175     282.4
4        45    117       148     406.0
5        60    102       127     300.0
6        60    110       136     374.0
7        45    104       134     253.3
8        30    109       133     195.1
9        60     98       124     269.0
     Duration  Pulse  Maxpulse  Calories
164        60    105       140     290.8
```

```
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

Example16

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
```

```
True
```

Example17

```
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
print(stopwords.words('english'))
#The following program removes stop words from a piece of text:
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
example_sent = """This is a sample sentence,
showing off the stop words filtration."""
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(example_sent)
# converts the words in word_tokens to lower case and then checks
whether
#they are present in stop_words or not
filtered_sentence = [w for w in word_tokens if not w.lower() in
stop_words]
#with no lower case conversion
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
        print(word_tokens)
        print(filtered_sentence)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you',
"you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself',
'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her',
'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom',
'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was',
'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with',
'about', 'against', 'between', 'into', 'through', 'during', 'before',
'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once',
'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor',
'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't',
'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't",
'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan',
"shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren',
"weren't", 'won', "won't", 'wouldn', "wouldn't"]
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words',
'filtration']
['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off',
'the', 'stop', 'words', 'filtration', '.']
['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words',
'filtration', '.']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Example18

```
import spacy
nlp = spacy.load('en_core_web_sm')
sentence = "Apple is looking at buying U.K. startup for $1 billion"
doc = nlp(sentence)
for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

```
Apple 0 5 ORG
U.K. 27 31 GPE
$1 billion 44 54 MONEY
```

Example19

```
# First we need to import spacy
import spacy
# Creating blank language object then
# tokenizing words of the sentence
nlp = spacy.blank("en")
doc = nlp("GeeksforGeeks is a one stop\
learning destination for geeks.")
for token in doc:
    print(token)
```

```
GeeksforGeeks
is
a
one
stoplearning
destination
for
geeks
.
```

Example20

```
#Here is an example to show what other functionalities can be enhanced
by adding modules to the
import spacy
```

```
# loading modules to the pipeline.
nlp = spacy.load("en_core_web_sm")
# Initialising doc with a sentence.
doc = nlp("If you want to be an excellent programmer \
, be consistent to practice daily on GFG.")
# Using properties of token i.e. Part of Speech and Lemmatization
for token in doc:
    print(token, " | ",spacy.explain(token.pos_)," | ", token.lemma_)
```

```
If  |  subordinating conjunction  |  if
you  |  pronoun  |  you
want  |  verb  |  want
to  |  particle  |  to
be  |  auxiliary  |  be
an  |  determiner  |  an
excellent  |  adjective  |  excellent
programmer  |  noun  |  programmer
,  |  punctuation  |  ,
be  |  auxiliary  |  be
consistent  |  adjective  |  consistent
to  |  particle  |  to
practice  |  verb  |  practice
daily  |  adverb  |  daily
on  |  adposition  |  on
GFG  |  proper noun  |  GFG
.  |  punctuation  |  .
```

Example21

```
from spacy import displacy
doc = nlp('wall street Journal just published an intresting pice on
cry pto currencies.')
displacy.render(doc,style = 'dep' , jupyter = True , options =
{'distance' : 90})
```

```
<IPython.core.display.HTML object>
```

Example 21

```
%pip install aima3
```

```
Requirement already satisfied: aima3 in c:\users\dell\anaconda3\lib\
site-packages (1.0.11)
Requirement already satisfied: jupyter in c:\users\dell\anaconda3\lib\
site-packages (from aima3) (1.0.0)
Requirement already satisfied: tqdm in c:\users\dell\anaconda3\lib\
site-packages (from aima3) (4.64.1)
Requirement already satisfied: networkx==1.11 in c:\users\dell\
anaconda3\lib\site-packages (from aima3) (1.11)
Requirement already satisfied: decorator>=3.4.0 in c:\users\dell\
anaconda3\lib\site-packages (from networkx==1.11->aima3) (5.1.1)
Requirement already satisfied: nbconvert in c:\users\dell\anaconda3\
lib\site-packages (from jupyter->aima3) (6.4.4)
Requirement already satisfied: qtconsole in c:\users\dell\anaconda3\
lib\site-packages (from jupyter->aima3) (5.2.2)
Requirement already satisfied: ipywidgets in c:\users\dell\anaconda3\
lib\site-packages (from jupyter->aima3) (7.6.5)
Requirement already satisfied: notebook in c:\users\dell\anaconda3\
lib\site-packages (from jupyter->aima3) (6.4.12)
Requirement already satisfied: ipykernel in c:\users\dell\anaconda3\
lib\site-packages (from jupyter->aima3) (6.15.2)
Requirement already satisfied: jupyter-console in c:\users\dell\
anaconda3\lib\site-packages (from jupyter->aima3) (6.4.3)
Requirement already satisfied: colorama in c:\users\dell\anaconda3\
lib\site-packages (from tqdm->aima3) (0.4.6)
Requirement already satisfied: psutil in c:\users\dell\anaconda3\lib\
site-packages (from ipykernel->jupyter->aima3) (5.9.0)
Requirement already satisfied: ipython>=7.23.1 in c:\users\dell\
anaconda3\lib\site-packages (from ipykernel->jupyter->aima3) (7.31.1)
Requirement already satisfied: packaging in c:\users\dell\anaconda3\
lib\site-packages (from ipykernel->jupyter->aima3) (21.3)
Requirement already satisfied: pyzmq>=17 in c:\users\dell\anaconda3\
lib\site-packages (from ipykernel->jupyter->aima3) (23.2.0)
Requirement already satisfied: debugpy>=1.0 in c:\users\dell\
anaconda3\lib\site-packages (from ipykernel->jupyter->aima3) (1.5.1)
Requirement already satisfied: tornado>=6.1 in c:\users\dell\
anaconda3\lib\site-packages (from ipykernel->jupyter->aima3) (6.1)
Requirement already satisfied: matplotlib-inline>=0.1 in c:\users\
dell\anaconda3\lib\site-packages (from ipykernel->jupyter->aima3)
(0.1.6)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\
dell\anaconda3\lib\site-packages (from ipykernel->jupyter->aima3)
(7.3.4)
Requirement already satisfied: traitlets>=5.1.0 in c:\users\dell\
anaconda3\lib\site-packages (from ipykernel->jupyter->aima3) (5.1.1)
Requirement already satisfied: nest-asyncio in c:\users\dell\
anaconda3\lib\site-packages (from ipykernel->jupyter->aima3) (1.5.5)
Requirement already satisfied: widgetsnbextension~=3.5.0 in c:\users\
dell\anaconda3\lib\site-packages (from ipywidgets->jupyter->aima3)
(3.5.2)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in c:\users\
dell\anaconda3\lib\site-packages (from ipywidgets->jupyter->aima3)
(1.0.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\users\
dell\anaconda3\lib\site-packages (from ipywidgets->jupyter->aima3)
(0.2.0)
Requirement already satisfied: nbformat>=4.2.0 in c:\users\dell\
anaconda3\lib\site-packages (from ipywidgets->jupyter->aima3) (5.5.0)
Requirement already satisfied: pygments in c:\users\dell\anaconda3\
lib\site-packages (from jupyter-console->jupyter->aima3) (2.11.2)
```

```
Requirement already satisfied: prompt-toolkit!=3.0.0,!
=3.0.1,<3.1.0,>=2.0.0 in c:\users\dell\anaconda3\lib\site-packages
(from jupyter-console->jupyter->aima3) (3.0.20)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (0.8.4)
Requirement already satisfied: testpath in c:\users\dell\anaconda3\
lib\site-packages (from nbconvert->jupyter->aima3) (0.6.0)
Requirement already satisfied: beautifulsoup4 in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (4.11.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\
dell\anaconda3\lib\site-packages (from nbconvert->jupyter->aima3)
(0.5.13)
Requirement already satisfied: defusedxml in c:\users\dell\anaconda3\
lib\site-packages (from nbconvert->jupyter->aima3) (0.7.1)
Requirement already satisfied: jupyter-core in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (4.11.1)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (1.5.0)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (0.4)
Requirement already satisfied: jupyterlab-pygments in c:\users\dell\
anaconda3\lib\site-packages (from nbconvert->jupyter->aima3) (0.1.2)
Requirement already satisfied: jinja2>=2.4 in c:\users\dell\anaconda3\
lib\site-packages (from nbconvert->jupyter->aima3) (2.11.3)
Requirement already satisfied: bleach in c:\users\dell\anaconda3\lib\
site-packages (from nbconvert->jupyter->aima3) (4.1.0)
Requirement already satisfied: terminado>=0.8.3 in c:\users\dell\
anaconda3\lib\site-packages (from notebook->jupyter->aima3) (0.13.1)
Requirement already satisfied: prometheus-client in c:\users\dell\
anaconda3\lib\site-packages (from notebook->jupyter->aima3) (0.14.1)
Requirement already satisfied: argon2-cffi in c:\users\dell\anaconda3\
lib\site-packages (from notebook->jupyter->aima3) (21.3.0)
Requirement already satisfied: Send2Trash>=1.8.0 in c:\users\dell\
anaconda3\lib\site-packages (from notebook->jupyter->aima3) (1.8.0)
Requirement already satisfied: qtpy in c:\users\dell\anaconda3\lib\
site-packages (from qtconsole->jupyter->aima3) (2.2.0)
Requirement already satisfied: setuptools>=18.5 in c:\users\dell\
anaconda3\lib\site-packages (from ipython>=7.23.1->ipykernel->jupyter-
>aima3) (63.4.1)
Requirement already satisfied: jedi>=0.16 in c:\users\dell\anaconda3\
lib\site-packages (from ipython>=7.23.1->ipykernel->jupyter->aima3)
(0.18.1)
Requirement already satisfied: backcall in c:\users\dell\anaconda3\
lib\site-packages (from ipython>=7.23.1->ipykernel->jupyter->aima3)
(0.2.0)
Requirement already satisfied: pickleshare in c:\users\dell\anaconda3\
lib\site-packages (from ipython>=7.23.1->ipykernel->jupyter->aima3)
(0.7.5)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\dell\
anaconda3\lib\site-packages (from jinja2>=2.4->nbconvert->jupyter-
>aima3) (2.0.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
dell\anaconda3\lib\site-packages (from jupyter-client>=6.1.12-
>ipykernel->jupyter->aima3) (2.8.2)
Requirement already satisfied: pywin32>=1.0 in c:\users\dell\
anaconda3\lib\site-packages (from jupyter-core->nbconvert->jupyter-
>aima3) (302)
Requirement already satisfied: fastjsonschema in c:\users\dell\
anaconda3\lib\site-packages (from nbformat>=4.2.0->ipywidgets-
>jupyter->aima3) (2.16.2)
Requirement already satisfied: jsonschema>=2.6 in c:\users\dell\
anaconda3\lib\site-packages (from nbformat>=4.2.0->ipywidgets-
>jupyter->aima3) (4.16.0)
Requirement already satisfied: wcwidth in c:\users\dell\anaconda3\lib\
site-packages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0-
>jupyter-console->jupyter->aima3) (0.2.5)
Requirement already satisfied: pywinpty>=1.1.0 in c:\users\dell\
anaconda3\lib\site-packages (from terminado>=0.8.3->notebook->jupyter-
>aima3) (2.0.2)
Requirement already satisfied: argon2-cffi-bindings in c:\users\dell\
anaconda3\lib\site-packages (from argon2-cffi->notebook->jupyter-
>aima3) (21.2.0)
Requirement already satisfied: soupsieve>1.2 in c:\users\dell\
anaconda3\lib\site-packages (from beautifulsoup4->nbconvert->jupyter-
>aima3) (2.3.1)
Requirement already satisfied: six>=1.9.0 in c:\users\dell\anaconda3\
lib\site-packages (from bleach->nbconvert->jupyter->aima3) (1.16.0)
Requirement already satisfied: webencodings in c:\users\dell\
anaconda3\lib\site-packages (from bleach->nbconvert->jupyter->aima3)
(0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\users\
dell\anaconda3\lib\site-packages (from packaging->ipykernel->jupyter-
>aima3) (3.0.9)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\dell\
anaconda3\lib\site-packages (from jedi>=0.16->ipython>=7.23.1-
>ipykernel->jupyter->aima3) (0.8.3)
Requirement already satisfied: attrs>=17.4.0 in c:\users\dell\
anaconda3\lib\site-packages (from jsonschema>=2.6->nbformat>=4.2.0-
>ipywidgets->jupyter->aima3) (21.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!
=0.17.2,>=0.14.0 in c:\users\dell\anaconda3\lib\site-packages (from
jsonschema>=2.6->nbformat>=4.2.0->ipywidgets->jupyter->aima3) (0.18.0)
Requirement already satisfied: cffi>=1.0.1 in c:\users\dell\anaconda3\
lib\site-packages (from argon2-cffi-bindings->argon2-cffi->notebook-
>jupyter->aima3) (1.15.1)
Requirement already satisfied: pycparser in c:\users\dell\anaconda3\
lib\site-packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-
cffi->notebook->jupyter->aima3) (2.21)
Note: you may need to restart the kernel to use updated packages.
```

```
from aima3.agents import *
from aima3.notebook import psource

class TrivialVacuumEnvironment(Environment):
    def init (self):
        super(). init ()
        self.status = {loc_A: random.choice(['Clean', 'Dirty']),
loc_B:
        random.choice(['Clean', 'Dirty'])}
    def thing_classes(self):
        return [Wall, Dirt, ReflexVacuumAgent, RandomVacuumAgent,
TableDrivenVacuumAgent, ModelBasedVacuumAgent]
    def percept(self, agent):
        return (agent.location, self.status[agent.location])
    def execute_action(self, agent, action):
        if action == 'Right':
            agent.location = loc_B
            agent.performance -= 1
        elif action == 'Left':
            agent.location = loc_A
            agent.performance -= 1
        elif action == 'Suck':
             if self.status[agent.location] == 'Dirty':
                agent.performance += 10
                self.status[agent.location] = 'Clean'
    def default_location(self, thing):
        return random.choice([loc_A, loc_B])
```

Question1

```
import numpy as np
arr1 = np.array([1,2])
arr2 = np.array([3,4])
arr3 = arr1 + arr2
print(arr3)
```

```
[4 6]
```

Question2

```
print(2 * arr1)
```

```
[2 4]
```

Question 3

```
arr2 = np.array([[1,2],[3,4]])
print(arr2)
```

```
[[1 2]
 [3 4]]
```

Question 4

```
print(arr2.dtype)
arr2 = arr2.astype('<U6') # String data type
print(arr2.dtype)
```

```
int32
<U6
```

Question 5

```
arr = np.arange(2,20,2)
print(arr)
```

```
[ 2  4  6  8 10 12 14 16 18]
```

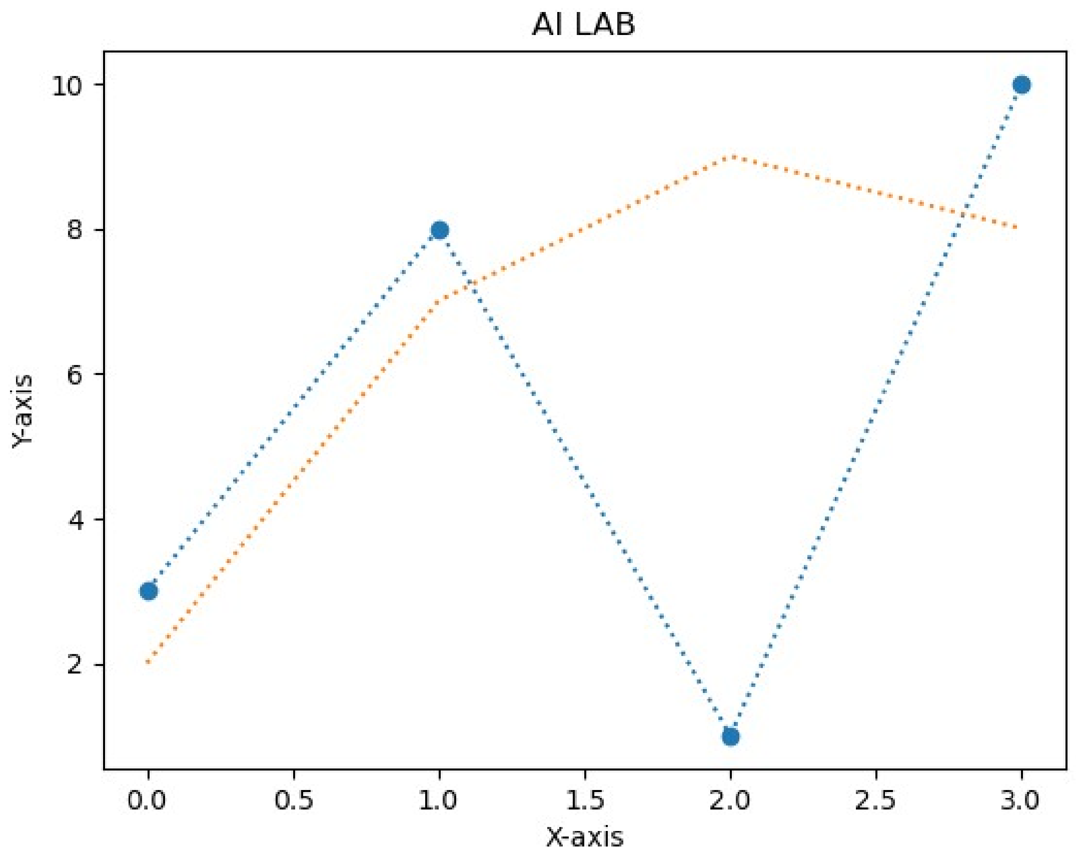
Question 6

```
arr1 = np.array([1,2,3,4,5,6])
arr2 = np.array([1,4,5,4,7,8])
np.where(arr1 == arr2)
```

```
(array([0, 3], dtype=int64),)
```

Question 7

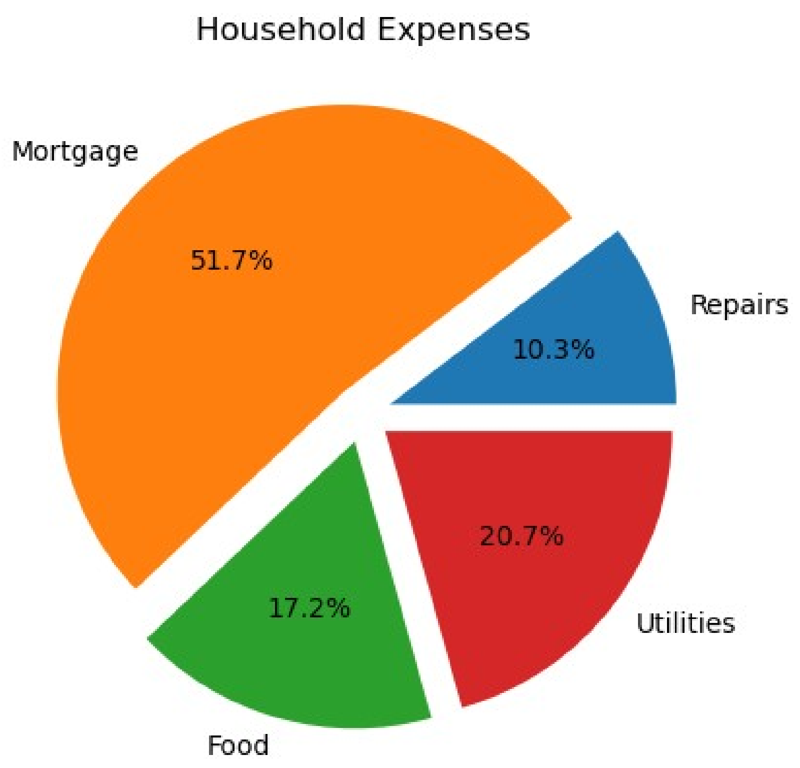
```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([3, 8, 1, 10])
ypoints = np.array([2, 7, 9, 8])
plt.title('AI LAB')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.plot(xpoints,'o',ypoints, linestyle = 'dotted')
plt.show()
```



Question 8

```
import matplotlib.pyplot as plt
%matplotlib inline
sizes = [10.34,51.72,17.24,20.69]
labels = 'Repairs','Mortgage','Food','Utilities'
plt.title("Household Expenses")
plt.axis('Equal')
plt.pie(sizes,labels = labels,explode= (0.1,0.1,0.1,0.1),autopct =
'%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x2e75ca19e50>,
  <matplotlib.patches.Wedge at 0x2e75ca10f10>,
  <matplotlib.patches.Wedge at 0x2e75ca10520>,
  <matplotlib.patches.Wedge at 0x2e75cbd3a90>],
 [Text(1.1372294701453294, 0.3830262813867655, 'Repairs'),
  Text(-0.7766752267276138, 0.9147543889960901, 'Mortgage'),
  Text(-0.32122199565097415, -1.1562077795578118, 'Food'),
  Text(0.9552564848540774, -0.7262816589617501, 'Utilities')],
 [Text(0.6633838575847754, 0.22343199747561315, '10.3%'),
  Text(-0.4530605489244413, 0.5336067269143858, '51.7%'),
  Text(-0.18737949746306826, -0.6744545380753902, '17.2%'),
  Text(0.5572329494982118, -0.4236643010610208, '20.7%')])
```



Question9

```
import pandas as pd


s1 = pd.Series([60,60,60,45,45])
s2 = pd.Series([110,117,103,109,117])
s3 = pd.Series([130,145,135,175,148])

df = pd.DataFrame({'Duration': s1, 'Pulse': s2, 'MaxPulse': s3})
df.to_csv('TestSheet.csv', index=False)

print(df)
print('\n')
print(df.loc[[0]])
print('\n')
pd.read_csv('TestSheet.csv')
df['Duration'] = df['Duration'] + 1

s4 = pd.Series([409.1,479,340,282.4,406])

df['Calories'] = s4
df.to_csv('TestSheet.csv', index=False)
print(df)
```

```
   Duration  Pulse  MaxPulse
0        60    110       130
1        60    117       145
2        60    103       135
3        45    109       175
4        45    117       148


   Duration  Pulse  MaxPulse
0        60    110       130


   Duration  Pulse  MaxPulse  Calories
0        61    110       130     409.1
1        61    117       145     479.0
2        61    103       135     340.0
3        46    109       175     282.4
4        46    117       148     406.0
```

Question10

```
from nltk.tokenize import sent_tokenize
text = 'Joe waited for the train. The train was late. Mary and 
Samantha took the bus. I looked for Mary and Samantha at the bus 
station.'
token_text = sent_tokenize(text)
print(token_text)
print('\n')
print("Result: ")
for t in token_text:
    print(t)
```

```
['Joe waited for the train.', 'The train was late.', 'Mary and 
Samantha took the bus.', 'I looked for Mary and Samantha at the bus 
station.']


Result: 
Joe waited for the train.
The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
```

Question 11

```
import nltk
nltk.download('punkt')

string = 'Joe waited for the train. The train was late. Mary and
```

```
Samantha took the bus. I looked for Mary and Samantha at the bus
station.'
answer = nltk.word_tokenize(string)
print(answer)
```

```
['Joe', 'waited', 'for', 'the', 'train', '.', 'The', 'train', 'was',
'late', '.', 'Mary', 'and', 'Samantha', 'took', 'the', 'bus', '.',
'I', 'looked', 'for', 'Mary', 'and', 'Samantha', 'at', 'the', 'bus',
'station', '.']

[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Question 12

```
from nltk.tokenize import sent_tokenize, word_tokenize
string = 'Joe waited for the train. The train was late. Mary and
Samantha took the bus. I looked for Mary and Samantha at the bus
station.'
print('Result:')

answer = [word_tokenize(t) for t in sent_tokenize(string)]
for a in answer:
    print(a)
```

```
Result:
['Joe', 'waited', 'for', 'the', 'train', '.']
['The', 'train', 'was', 'late', '.']
['Mary', 'and', 'Samantha', 'took', 'the', 'bus', '.']
['I', 'looked', 'for', 'Mary', 'and', 'Samantha', 'at', 'the', 'bus',
'station', '.']
```

Question 13

```
import spacy
nlp = spacy.load("en_core_web_sm")
string = nlp("Joe waited for the train. The train was late. Mary and
Samantha took the bus. I looked for Mary and Samantha at the bus
station.")

for s in string:
    print("{}({})-{}-
>{}".format(s.text,s.dep_,s.head.text,s.head.dep_))
```

```
Joe(nsubj)-waited->ROOT
waited(ROOT)-waited->ROOT
for(prep)-waited->ROOT
the(det)-train->pobj
```

```
train(pobj)-for->prep
.(punct)-waited->ROOT
The(det)-train->nsubj
train(nsubj)-was->ROOT
was(ROOT)-was->ROOT
late(acomp)-was->ROOT
.(punct)-was->ROOT
Mary(nsubj)-took->ROOT
and(cc)-Mary->nsubj
Samantha(conj)-Mary->nsubj
took(ROOT)-took->ROOT
the(det)-bus->dobj
bus(dobj)-took->ROOT
.(punct)-took->ROOT
I(nsubj)-looked->ROOT
looked(ROOT)-looked->ROOT
for(prep)-looked->ROOT
Mary(pobj)-for->prep
and(cc)-Mary->pobj
Samantha(conj)-Mary->pobj
at(prep)-looked->ROOT
the(det)-station->pobj
bus(compound)-station->pobj
station(pobj)-at->prep
.(punct)-looked->ROOT
```

Question14

```
import spacy
nlp = spacy.load("en_core_web_sm")
string = nlp("Joe waited for the train. The train was late. Mary and 
Samantha took the bus. I looked for Mary and Samantha at the bus 
station.")

for s in string:
    print(s.text)
```

```
Joe
waited
for
the
train
.
The
train
was
late
.
Mary
and
```

```
Samantha
took
the
bus
.
I
looked
for
Mary
and
Samantha
at
the
bus
station
.
```

Question 15

```
import numpy as np

class VacuumCleaner:
    def __init__(self, room_matrix, start_pos):
        self.room_matrix = room_matrix
        self.current_pos = start_pos

    def move_up(self):
        if self.current_pos[0] > 0 and 
self.room_matrix[self.current_pos[0]-1][self.current_pos[1]] != 'B':
            self.current_pos[0] -= 1

    def move_down(self):
        if self.current_pos[0] < len(self.room_matrix)-1 and 
self.room_matrix[self.current_pos[0]+1][self.current_pos[1]] != 'B':
            self.current_pos[0] += 1

    def move_left(self):
        if self.current_pos[1] > 0 and 
self.room_matrix[self.current_pos[0]][self.current_pos[1]-1] != 'B':
            self.current_pos[1] -= 1

    def move_right(self):
        if self.current_pos[1] < len(self.room_matrix[0])-1 and 
self.room_matrix[self.current_pos[0]][self.current_pos[1]+1] != 'B':
            self.current_pos[1] += 1

    def clean_cell(self):
        if self.room_matrix[self.current_pos[0]][self.current_pos[1]] 
== 'D':
            self.room_matrix[self.current_pos[0]][self.current_pos[1]]
```

```
= 'C'

    def display_room(self):
        for i in range(len(self.room_matrix)):
            for j in range(len(self.room_matrix[0])):
                if i == self.current_pos[0] and j ==
self.current_pos[1]:
                    print('*', end=' ')
                else:
                    print(self.room_matrix[i][j], end=' ')
            print('')
        print('')

# Example room matrix
room_matrix = np.array([['D', 'C', 'D', 'B'],
                        ['D', 'B', 'C', 'D'],
                        ['C', 'D', 'C', 'D'],
                        ['D', 'C', 'D', 'C']])

# Create vacuum cleaner
vacuum = VacuumCleaner(room_matrix, [0, 0])

# Move and clean the cells
while np.any(room_matrix == 'D'):
    vacuum.clean_cell()
    vacuum.display_room()
    vacuum.move_down()
    vacuum.clean_cell()
    vacuum.display_room()
    vacuum.move_right()

print("The room is clean.")
```