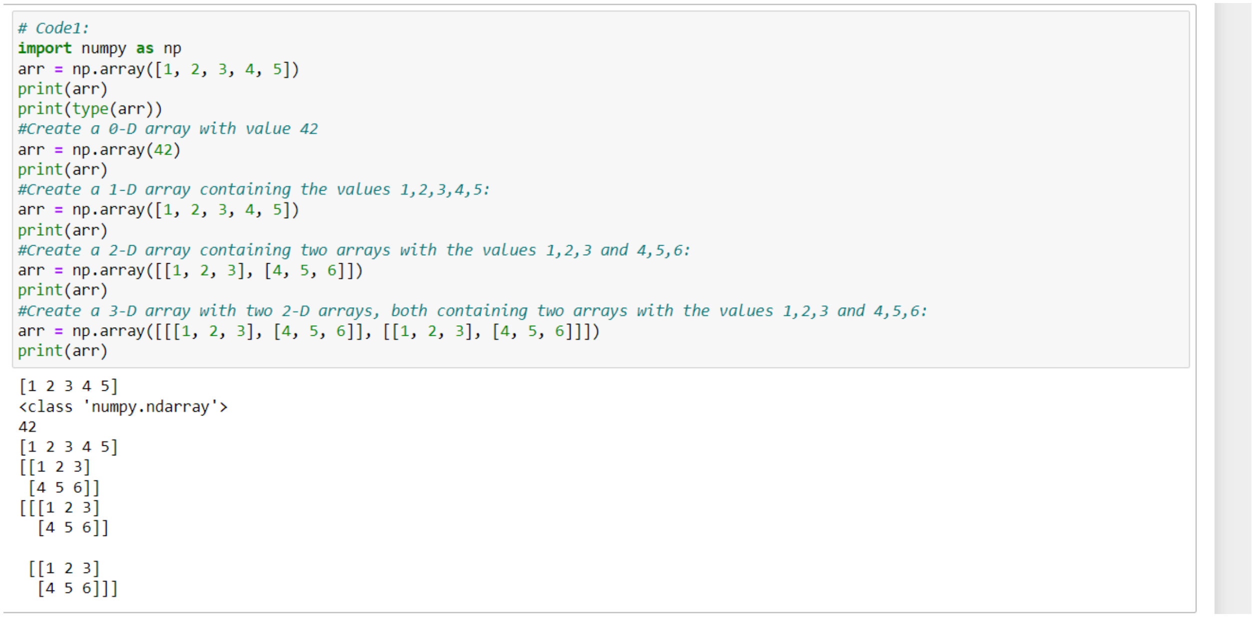
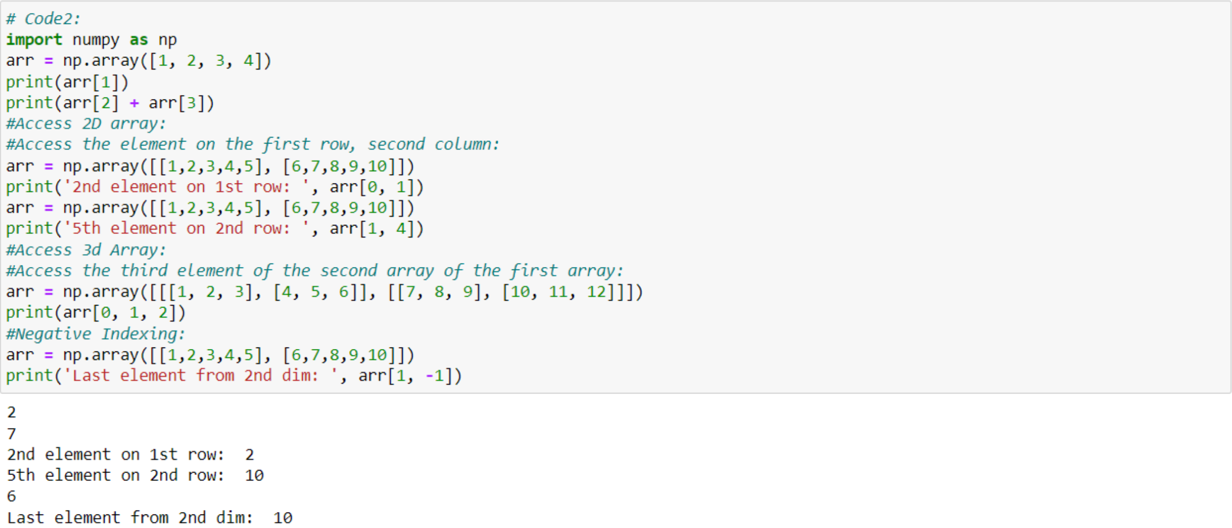
AI LAB 02

Examples



```
# Code1:
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]]]
```

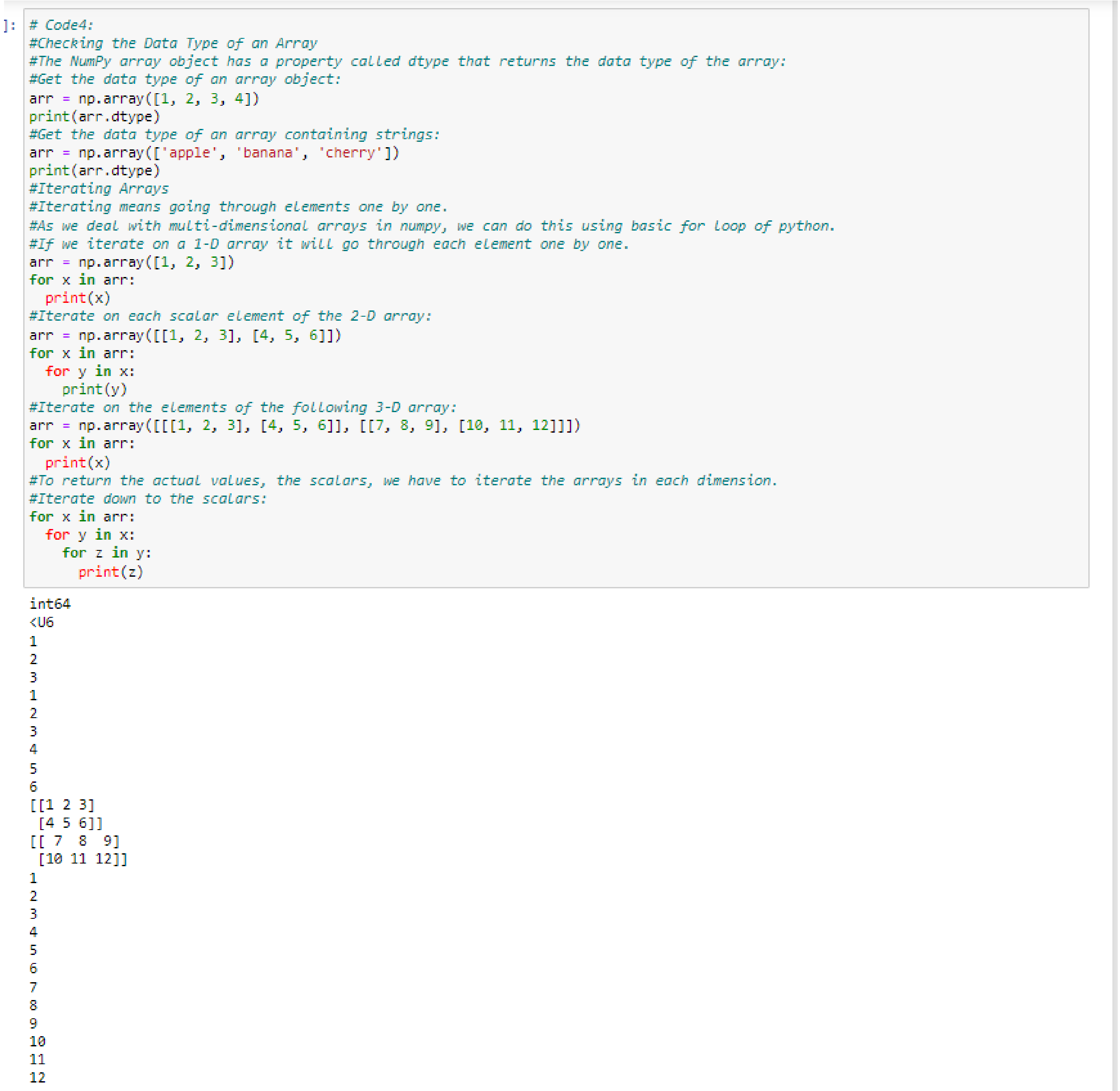


```
# Code2:
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[1])
print(arr[2] + arr[3])
#Access 2D array:
#Access the element on the first row, second column:
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
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
```

```
# Code3:
#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]]
```

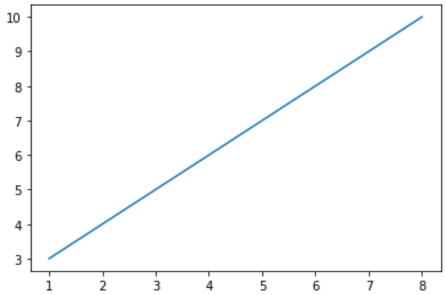


```
]: # Code4:
   #Checking the Data Type of an Array
   #The NumPy array object has a property called dtype that returns the data type of the array:
   #Get the data type of an array object:
   arr = np.array([1, 2, 3, 4])
   print(arr.dtype)
   #Get the data type of an array containing strings:
   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:
   for x in arr:
     for y in x:
       for z in y:
         print(z)
```

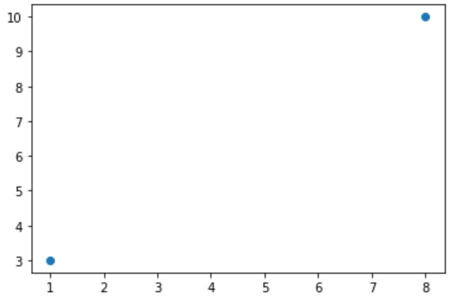
```
int64
<U6
1
2
3
1
2
3
4
5
6
[[1 2 3]
 [4 5 6]]
[[ 7  8  9]
 [10 11 12]]
1
2
3
4
5
6
7
8
9
10
11
12
```

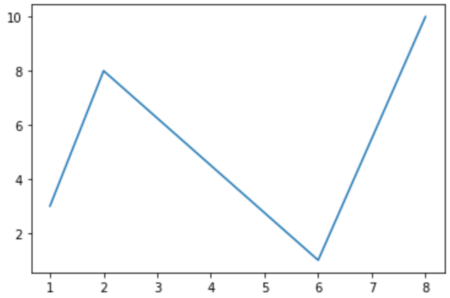
:

```
# Code1:
#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 function.
#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()
```

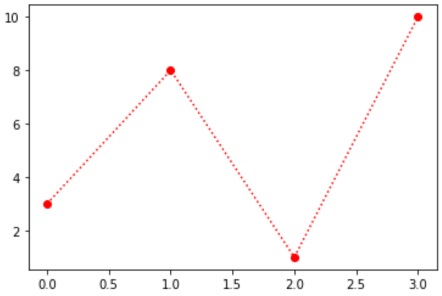


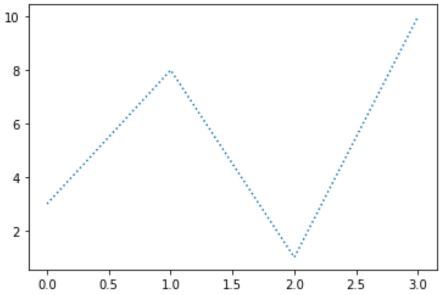
```
# Code2:
#Draw two points in the diagram, one at position (1, 3) and one in position (8, 10):
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints, 'o')
plt.show()
#Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])
plt.plot(xpoints, ypoints)
plt.show()
```





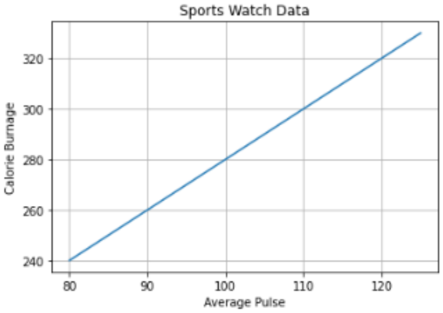
```
# Code3:
#You can use also use the shortcut string notation parameter to specify the marker.
#This parameter is also called fmt, and is written with this syntax:
#marker|line|color
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, 'o:r')
plt.show()
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```





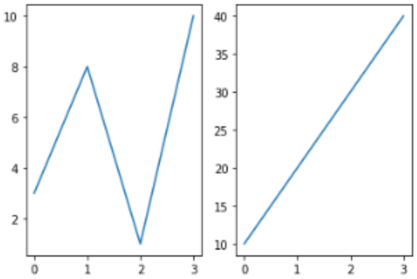
.1]:

```
# Code4:
#Labels:
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()
```



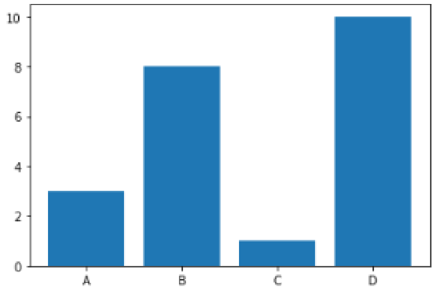
.2]:

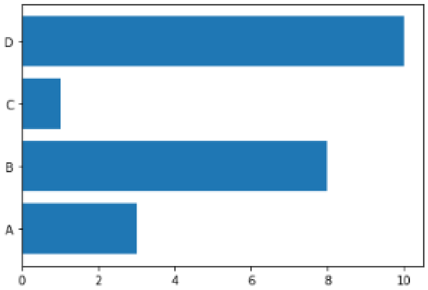
```
# Code5:
#Subplots
# With the subplot() function you can draw multiple plots in one figure:
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```



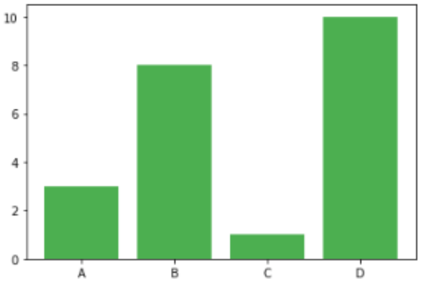
]:

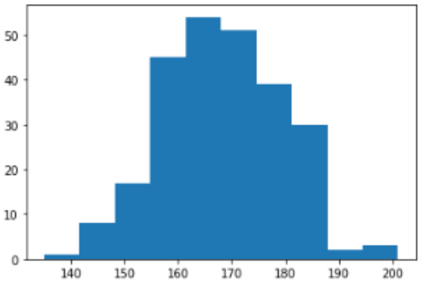
```
# Code6:
#Bar plots
#With Pyplot, you can use the bar() function to draw bar graphs:
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x,y)
plt.show()
#for horizontal bar use 'barh'
plt.barh(x, y)
plt.show()
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])
plt.bar(x, 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()
```

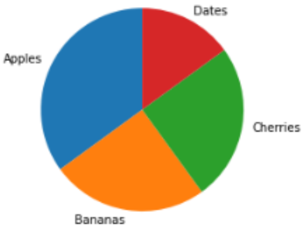


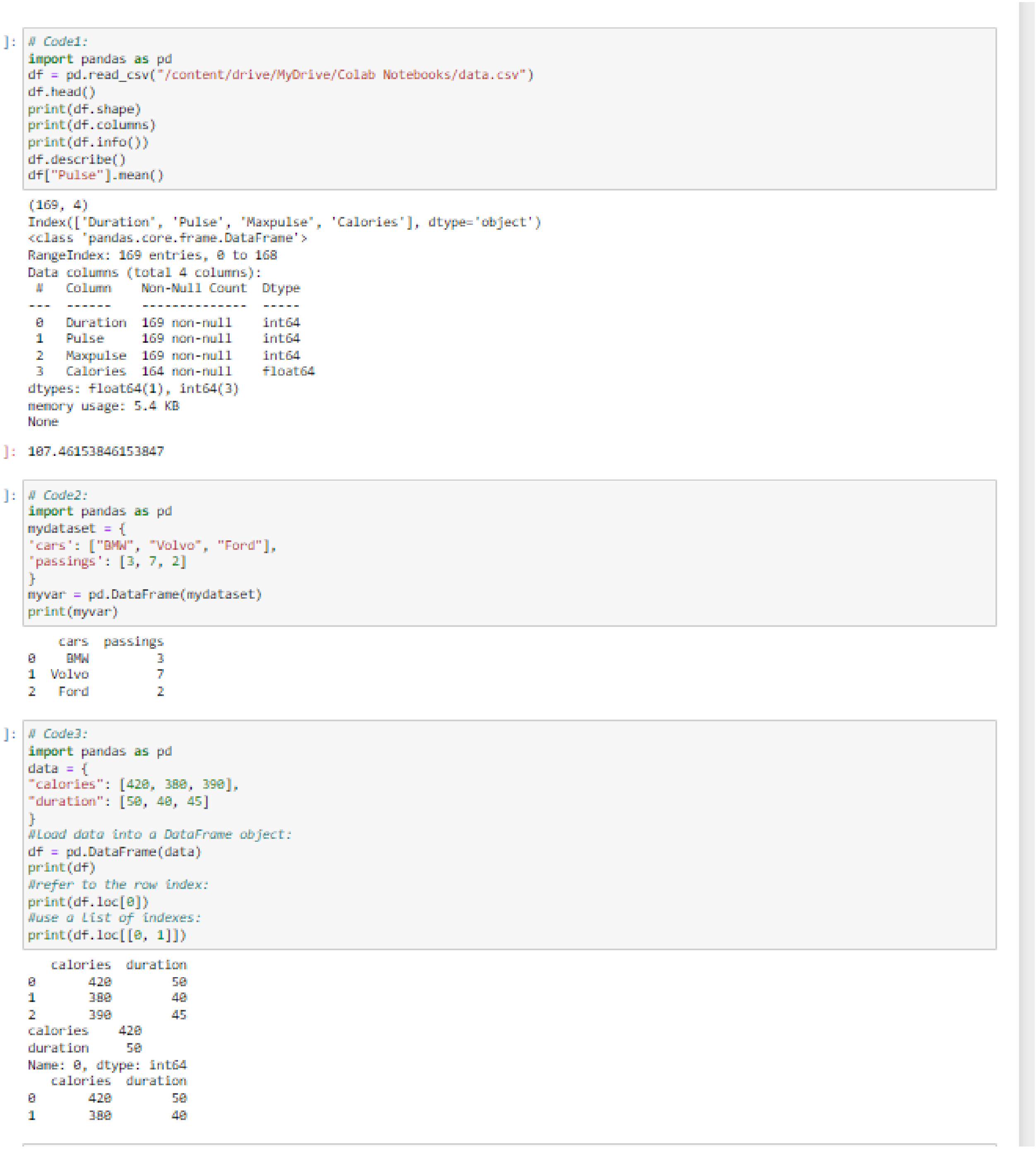


0
2
4
6
8
10


10
8
6
4
2
0
A
B
C
D


50
40
30
20
10
0
140
150
160
170
180
190
200


Dates
Apples
Cherries
Bananas



]:
```
# Code1:
import pandas as pd
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/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
```

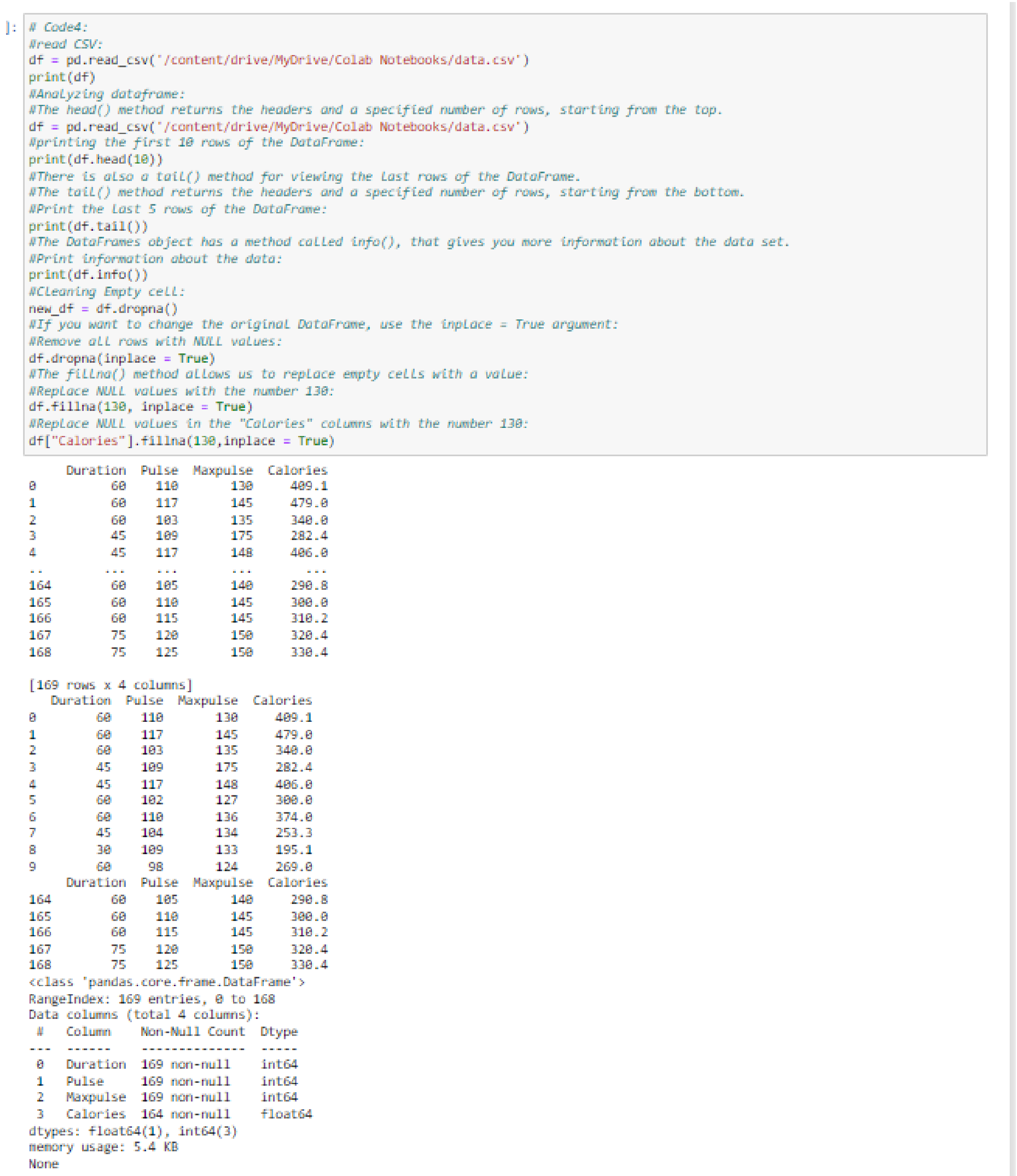
]:
```
107.46153846153847
```

]:
```
# Code2:
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
```

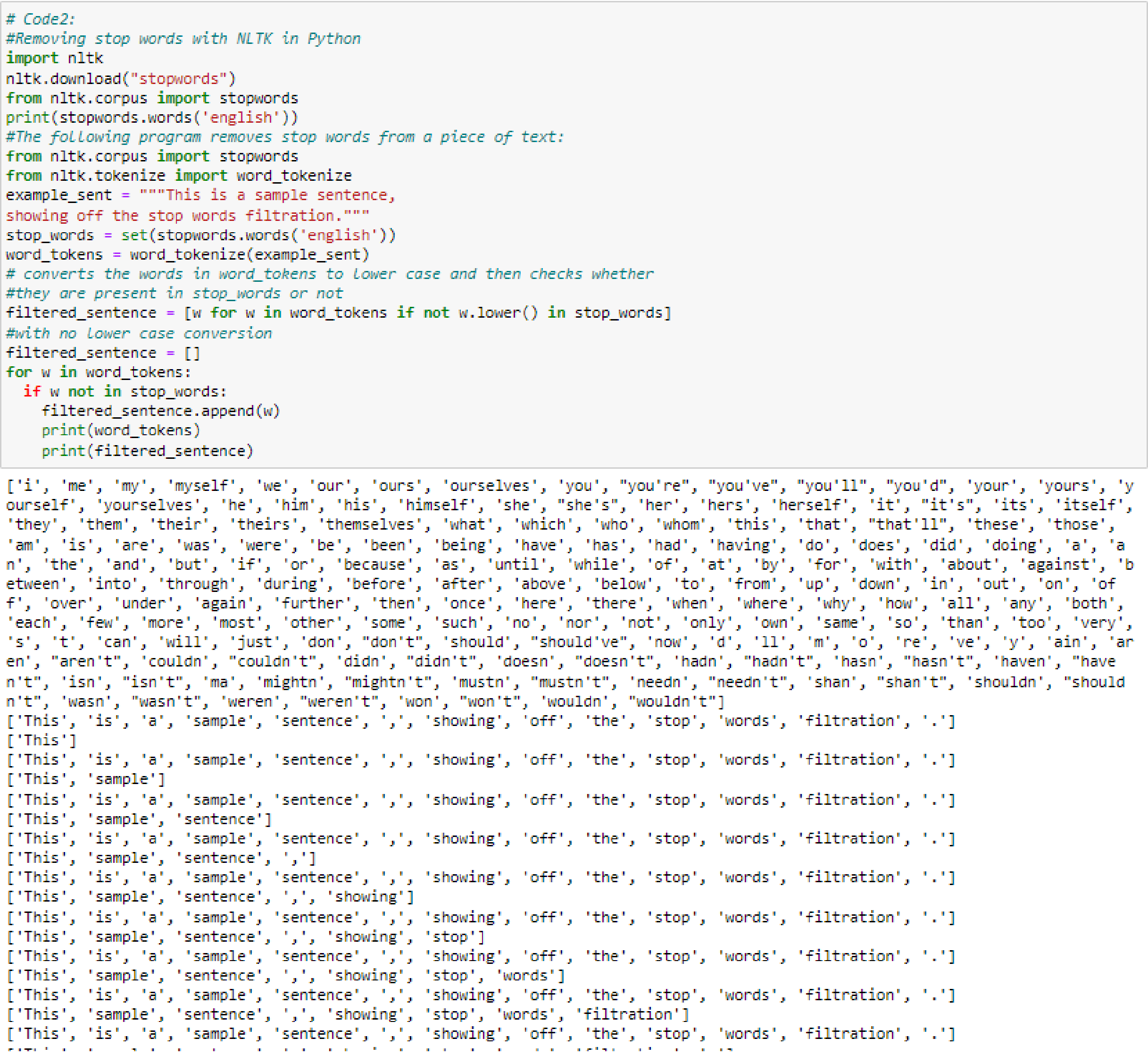
]:
```
# Code3:
import pandas as pd
data = {
"calories": [420, 380, 390],
"duration": [50, 40, 45]
}
#load data into a DataFrame object:
df = pd.DataFrame(data)
print(df)
#refer to the row index:
print(df.loc[0])
#use a list of indexes:
print(df.loc[[0, 1]])
```

```
   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
```



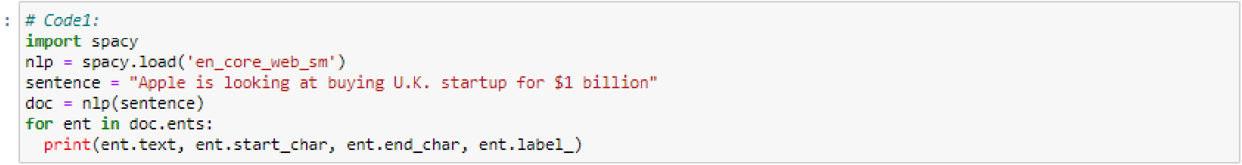
```
]: # Code4:
   #read CSV:
   df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data.csv')
   print(df)
   #Analyzing dataframe:
   #The head() method returns the headers and a specified number of rows, starting from the top.
   df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data.csv')
   #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
```

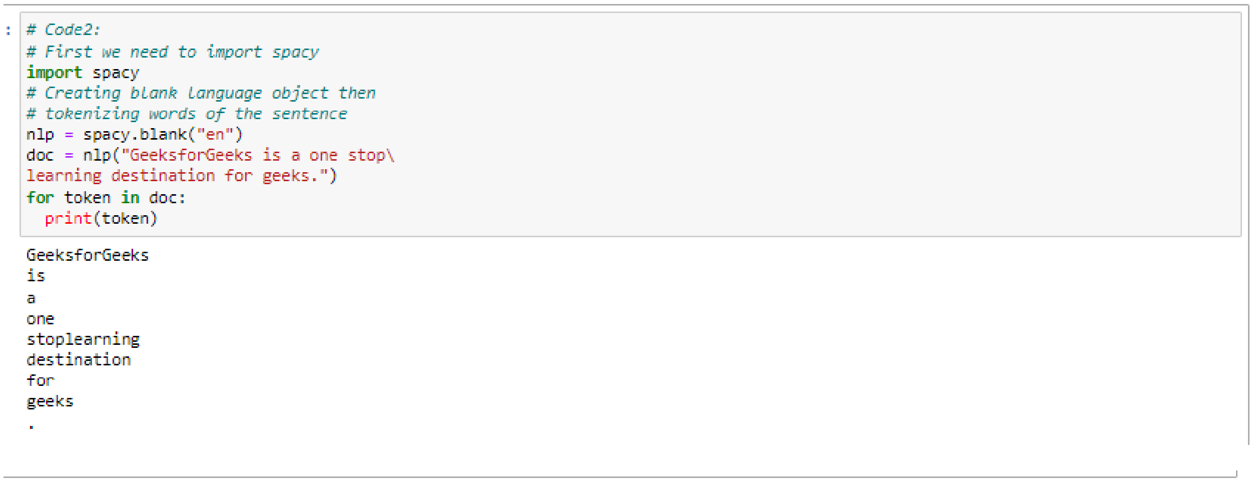


```python
# Code2:
#Removing stop words with NLTK in Python
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
print(stopwords.words('english'))
#The following program removes stop words from a piece of text:
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
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', 'y
ourself', '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', 'a
n', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
f', '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', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "have
n't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "should
n'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', '.']
```

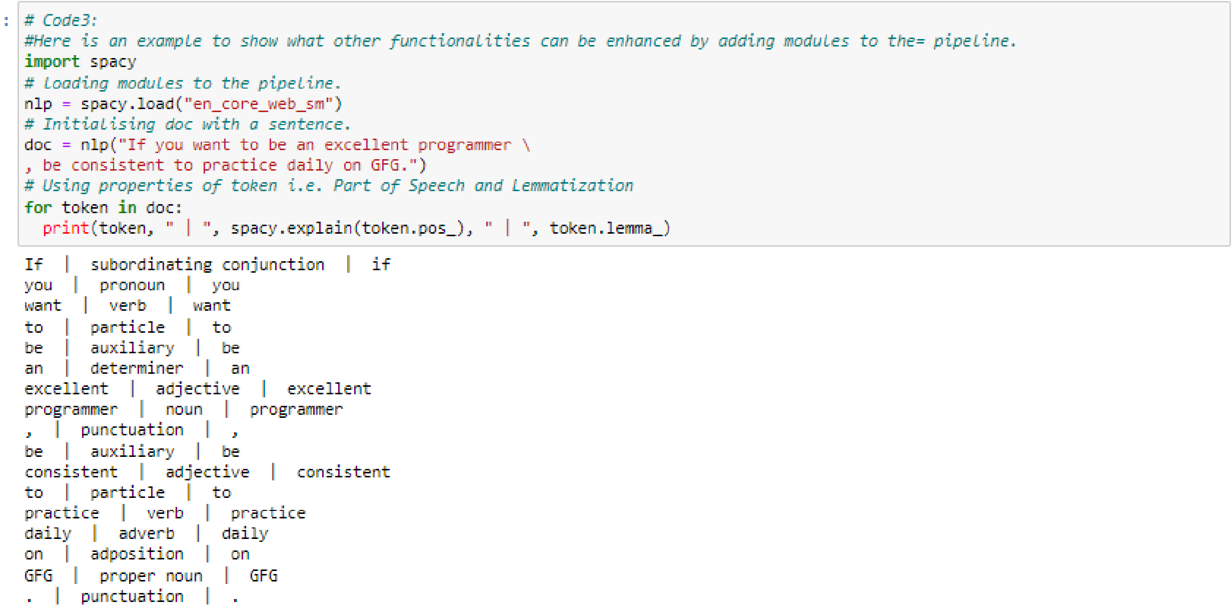


```
# Code1:
import spacy
nlp = spacy.load('en_core_web_sm')
sentence = "Apple is looking at buying U.K. startup for $1 billion"
doc = nlp(sentence)
for ent in doc.ents:
  print(ent.text, ent.start_char, ent.end_char, ent.label_)
```



```
# Code2:
# 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
.
```



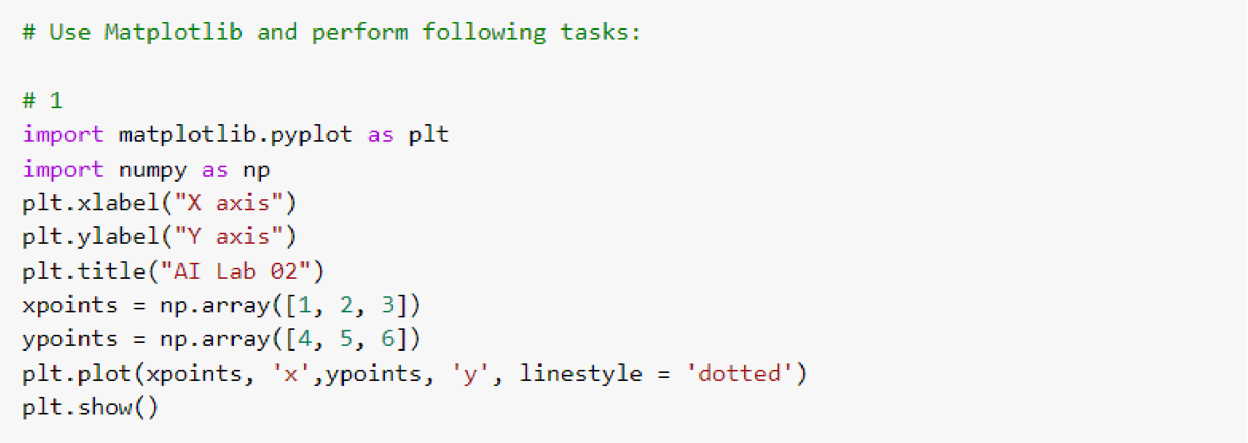
```
# Code3:
#Here is an example to show what other functionalities can be enhanced by adding modules to the= pipeline.
import spacy
# Loading modules to the pipeline.
nlp = spacy.load("en_core_web_sm")
# Initialising doc with a sentence.
doc = nlp("If you want to be an excellent programmer \
, be consistent 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  |  .
```

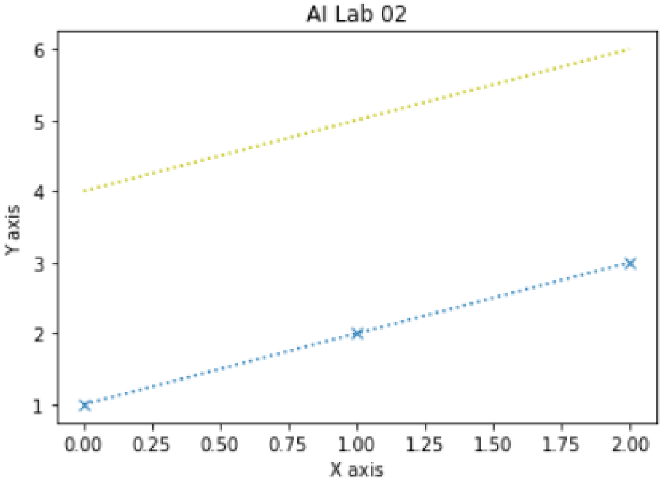
Tasks

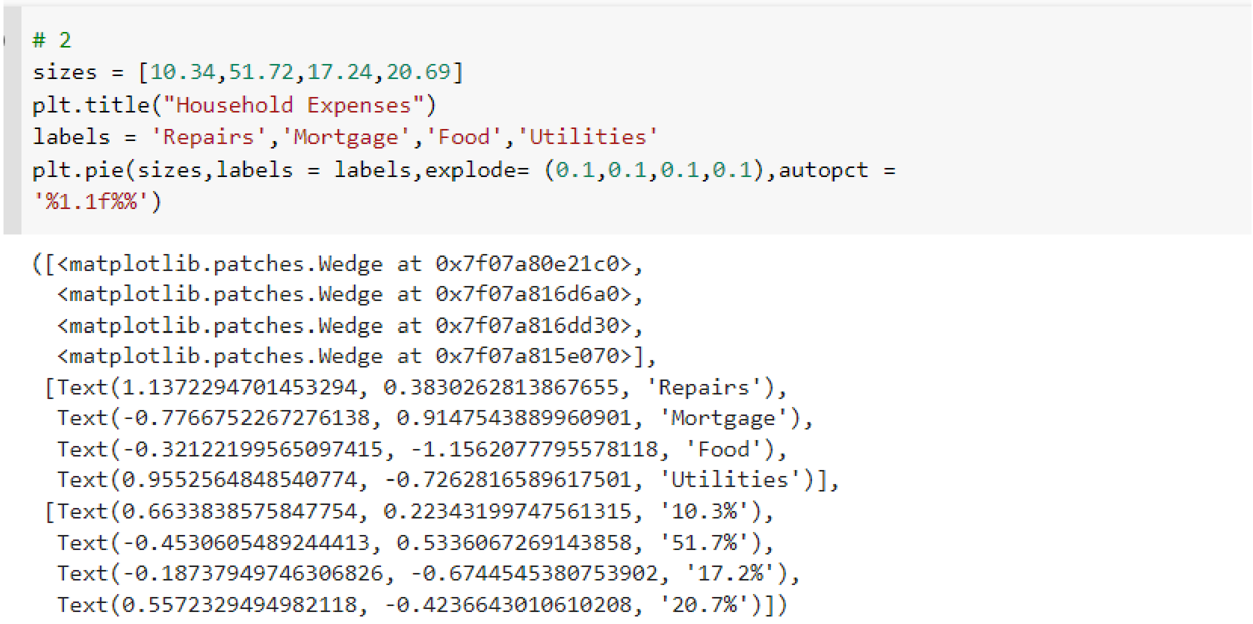
```python
# 1
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
print(np.add(arr1, arr2))

# 2
print(5 * arr1)

# 3
arr1 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr1)

# 4
print(arr1.dtype)
arr1 = arr1.astype(float)
print(arr1.dtype)

# 5
sequence = np.arange(0, 100, 2)
print(sequence)

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

```
[5 7 9]
[ 5 10 15]
[[1 2 3]
 [4 5 6]]
int64
float64
[ 0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94
 96 98]
(array([1]),)
```



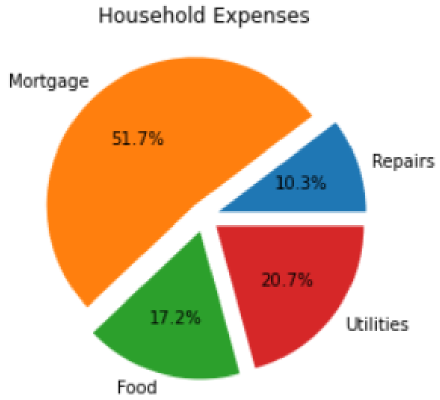
```python
# Use Matplotlib and perform following tasks:

# 1
import matplotlib.pyplot as plt
import numpy as np
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("AI Lab 02")
xpoints = np.array([1, 2, 3])
ypoints = np.array([4, 5, 6])
plt.plot(xpoints, 'x',ypoints, 'y', linestyle = 'dotted')
plt.show()
```

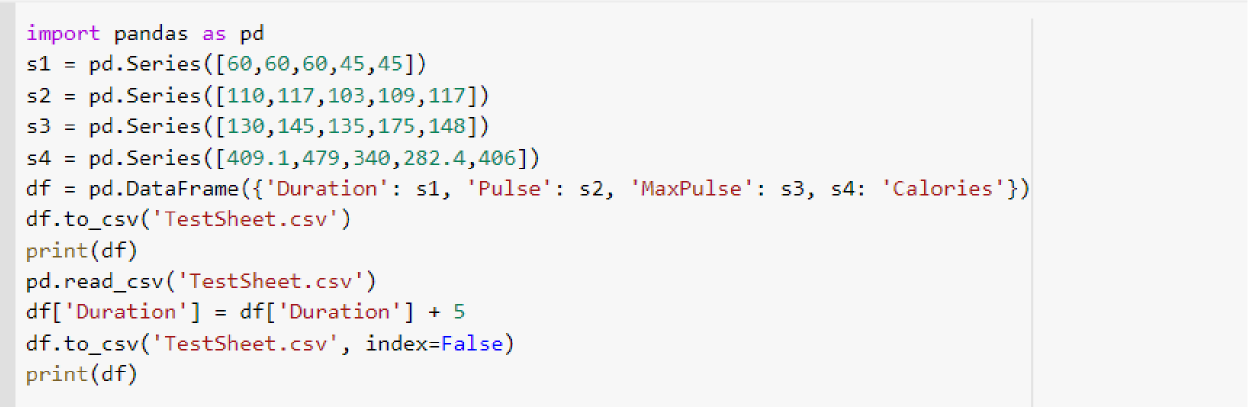




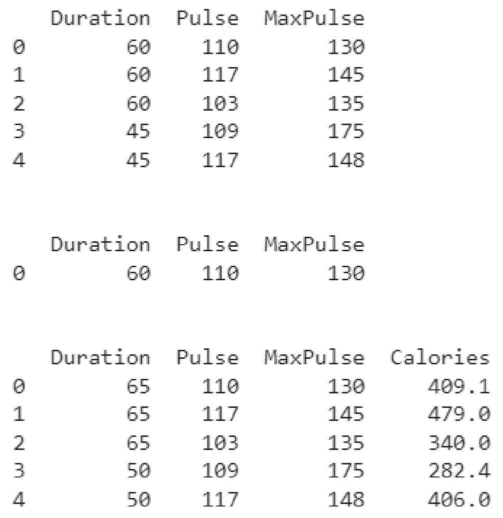
```
# 2
sizes = [10.34,51.72,17.24,20.69]
plt.title("Household Expenses")
labels = 'Repairs','Mortgage','Food','Utilities'
plt.pie(sizes,labels = labels,explode= (0.1,0.1,0.1,0.1),autopct =
'%1.1f%%')
```

```
([<matplotlib.patches.Wedge at 0x7f07a80e21c0>,
  <matplotlib.patches.Wedge at 0x7f07a816d6a0>,
  <matplotlib.patches.Wedge at 0x7f07a816dd30>,
  <matplotlib.patches.Wedge at 0x7f07a815e070>],
 [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%')])
```

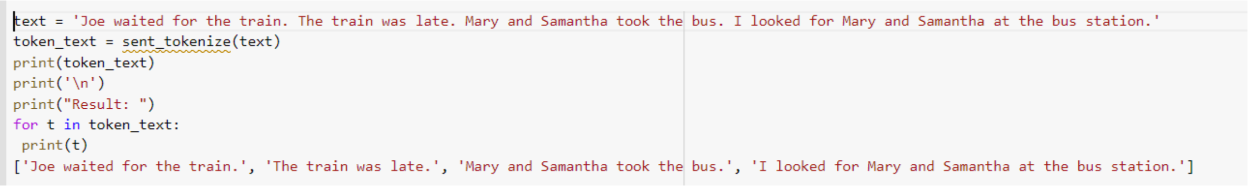




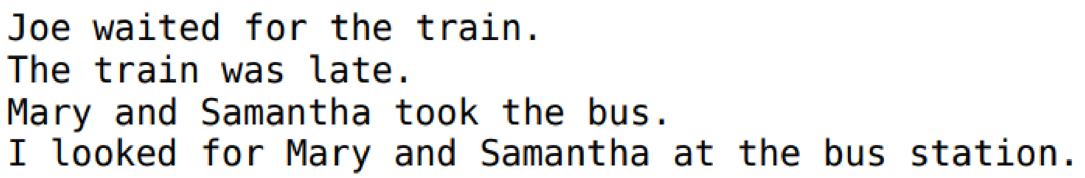
```
import pandas as pd
s1 = pd.Series([60,60,60,45,45])
s2 = pd.Series([110,117,103,109,117])
s3 = pd.Series([130,145,135,175,148])
s4 = pd.Series([409.1,479,340,282.4,406])
df = pd.DataFrame({'Duration': s1, 'Pulse': s2, 'MaxPulse': s3, s4: 'Calories'})
df.to_csv('TestSheet.csv')
print(df)
pd.read_csv('TestSheet.csv')
df['Duration'] = df['Duration'] + 5
df.to_csv('TestSheet.csv', index=False)
print(df)
```



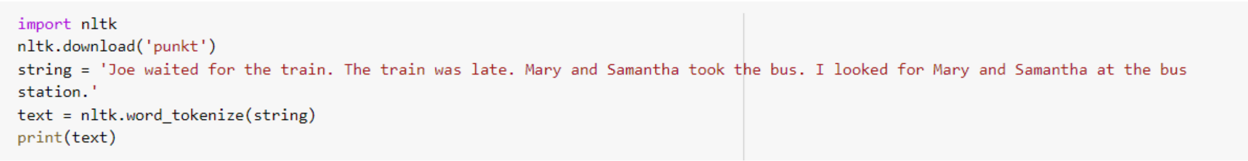
```
   Duration  Pulse  MaxPulse
0        60    110       130
1        60    117       145
2        60    103       135
3        45    109       175
4        45    117       148


   Duration  Pulse  MaxPulse
0        60    110       130


   Duration  Pulse  MaxPulse  Calories
0        65    110       130     409.1
1        65    117       145     479.0
2        65    103       135     340.0
3        50    109       175     282.4
4        50    117       148     406.0
```



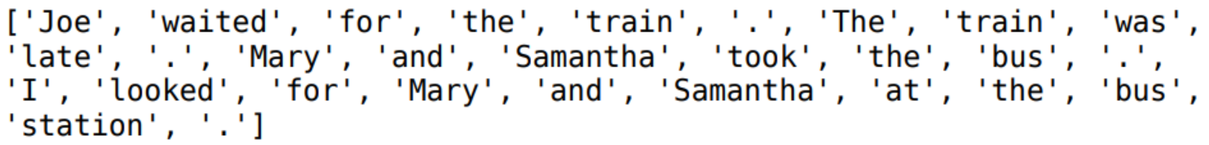
```
text = 'Joe waited for the train. The train was late. Mary and Samantha took the bus. I looked for Mary and Samantha at the bus station.'
token_text = sent_tokenize(text)
print(token_text)
print('\n')
print("Result: ")
for t in token_text:
 print(t)
['Joe waited for the train.', 'The train was late.',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



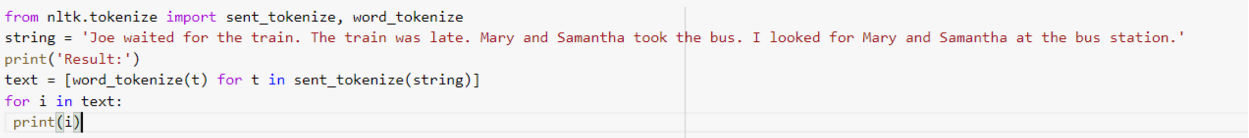
```
Joe waited for the train.
The train was late.
Mary and Samantha took the bus.
I looked for Mary and Samantha at the bus station.
```



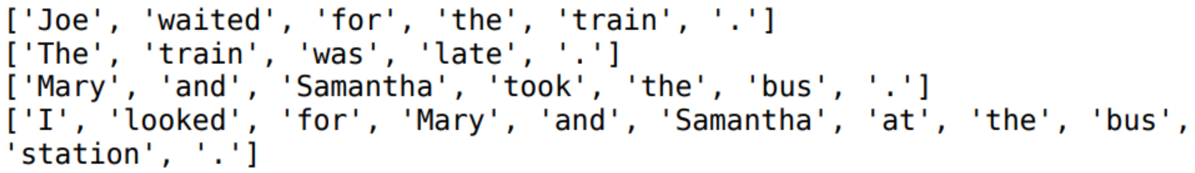
```
import nltk
nltk.download('punkt')
string = 'Joe waited for the train. The train was late. Mary and Samantha took the bus. I looked for Mary and Samantha at the bus
station.'
text = nltk.word_tokenize(string)
print(text)
```



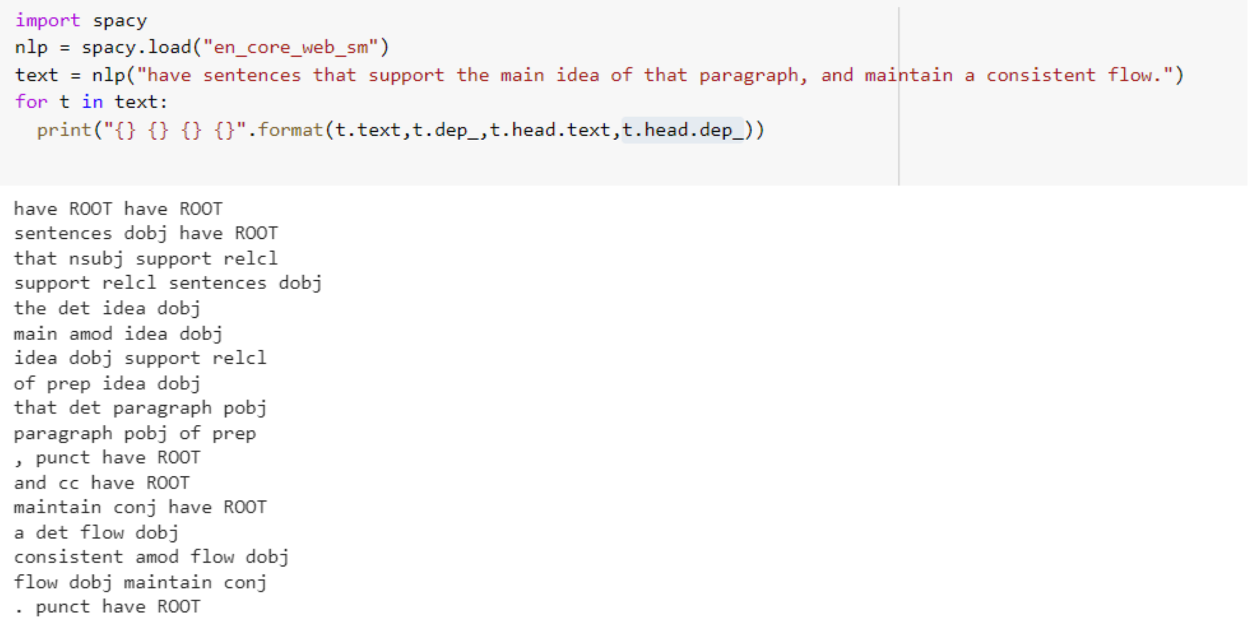
```
['Joe', 'waited', 'for', 'the', 'train', '.', 'The', 'train', 'was',
'late', '.', 'Mary', 'and', 'Samantha', 'took', 'the', 'bus', '.',
'I', 'looked', 'for', 'Mary', 'and', 'Samantha', 'at', 'the', 'bus',
'station', '.']
```



```
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:')
text = [word_tokenize(t) for t in sent_tokenize(string)]
for i in text:
 print(i)
```

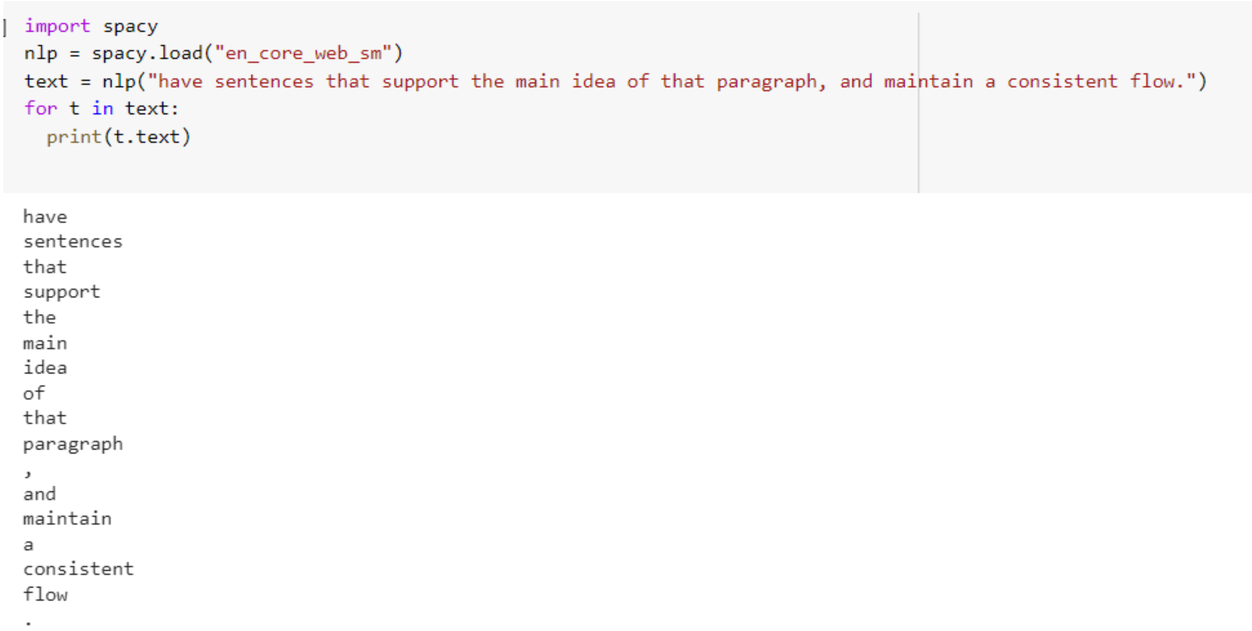


```
['Joe', 'waited', 'for', 'the', 'train', '.']
['The', 'train', 'was', 'late', '.']
['Mary', 'and', 'Samantha', 'took', 'the', 'bus', '.']
['I', 'looked', 'for', 'Mary', 'and', 'Samantha', 'at', 'the', 'bus',
'station', '.']
```



```python
import spacy
nlp = spacy.load("en_core_web_sm")
text = nlp("have sentences that support the main idea of that paragraph, and maintain a consistent flow.")
for t in text:
  print("{} {} {} {}".format(t.text,t.dep_,t.head.text,t.head.dep_))
```

```
have ROOT have ROOT
sentences dobj have ROOT
that nsubj support relcl
support relcl sentences dobj
the det idea dobj
main amod idea dobj
idea dobj support relcl
of prep idea dobj
that det paragraph pobj
paragraph pobj of prep
, punct have ROOT
and cc have ROOT
maintain conj have ROOT
a det flow dobj
consistent amod flow dobj
flow dobj maintain conj
. punct have ROOT
```



```python
import spacy
nlp = spacy.load("en_core_web_sm")
text = nlp("have sentences that support the main idea of that paragraph, and maintain a consistent flow.")
for t in text:
  print(t.text)
```

```
have
sentences
that
support
the
main
idea
of
that
paragraph
,
and
maintain
a
consistent
flow
.
```