Question 01

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

```
[4 6]
```

Question 02

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

```
[2 4]
```

Question 03

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

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

Question 04

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

```
int32
<U6
```

Question 05

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

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

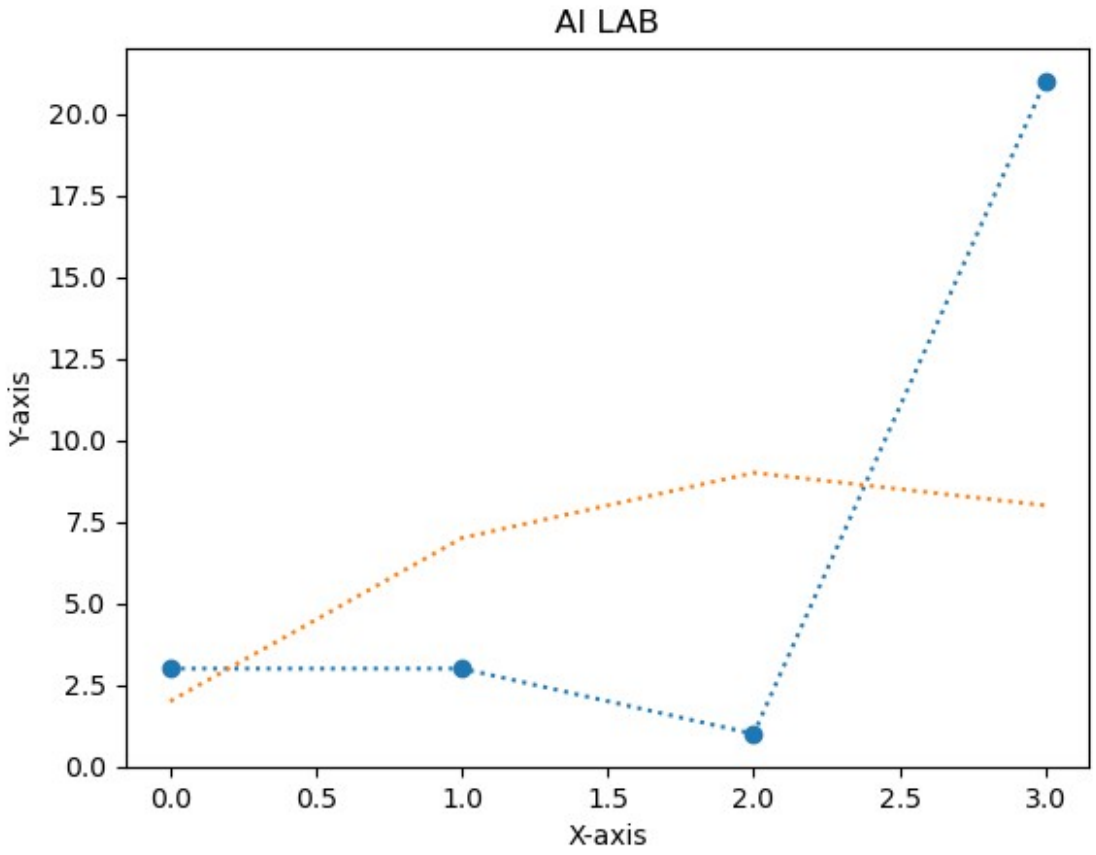
Question 06

```
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 07

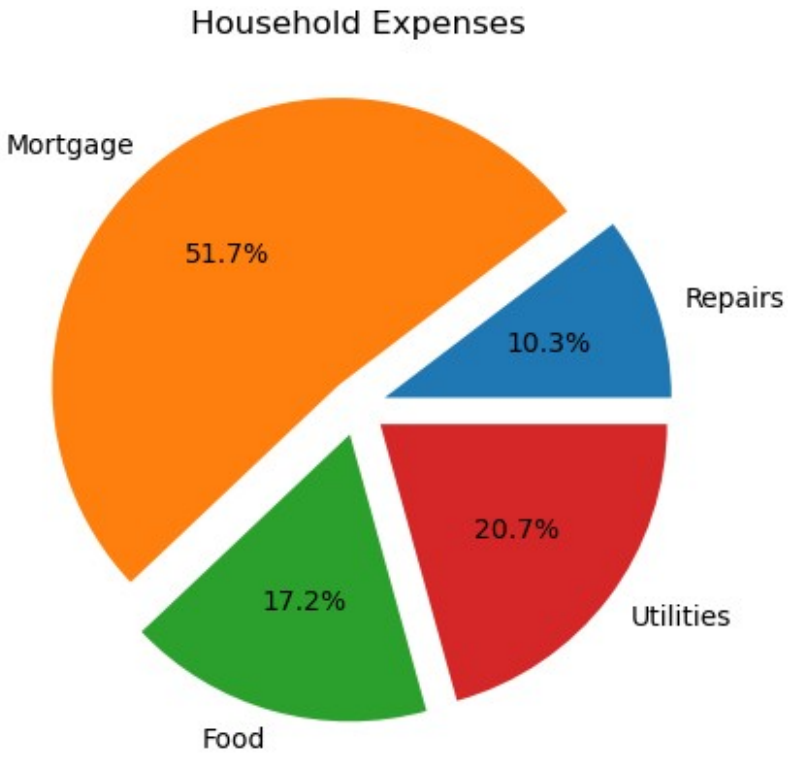
```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([3, 3, 1, 21])
ypoints = np.array([2, 7, 9, 8])
plt.title('AI LAB')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.plot(xpoints,'o',ypoints, linestyle = 'dotted')
plt.show()
```



Question 08

```
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 0x2a7b765a4c0>,
  <matplotlib.patches.Wedge at 0x2a7b765adc0>,
  <matplotlib.patches.Wedge at 0x2a7b76611f0>,
  <matplotlib.patches.Wedge at 0x2a7b7661a00>],
 [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%')])
```



Question 09

```python
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
```

Question 10

```
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\Bilal\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_))
```

```
---------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
```

```
last)
~\AppData\Local\Temp\ipykernel_6332\3414268302.py in <module>
----> 1 import spacy
      2 nlp = spacy.load("en_core_web_sm")
      3 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.")
      4 for s in string:
      5     print("{}({})-{}-
{}".format(s.text,s.dep_,s.head.text,s.head.dep_))

ModuleNotFoundError: No module named 'spacy'
```

Question 14

```
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)
```

```
  File "C:\Users\Bilal\AppData\Local\Temp\
ipykernel_13316\3365383996.py", line 3
    string = nlp("Joe waited for the train. The train was late. Mary
and

^
SyntaxError: EOL while scanning string literal
```

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
andself.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.")
```