سوال‌های ۱ و ۳ به صورت کد و خروجی آن‌ها به صورت عکس در کد بیس ارسالی در پوشه out موجود است.

۲. خروج کدهای روبروچیست؟

در print(a*b) مقدار b در تک تک درایه‌های آرایه a ضرب می‌شود که : [2, 4, 6]

در print(a*c) هر درایه آرایه c در درایه نظیر خود در آرایه a ضرب می‌شود که: [2, 6, 18]

```python
import numpy as np
a = np.array([1.0, 2.0, 3.0])
# Example 1
b = 2.0
print(a * b)
# Example 2
c = [2.0, 3.0, 6.0]
print(a * c)
```

```
[ 2.  4.  6.]
[  2.   6.  18.]
```

این دستور یک آرایه خالی که همه درایه‌ها آن صفر است (با اندازه مشخص شده) می‌سازد و نوع داده‌ای هر کدام از آنها را هم می توان مشخص کرد. مثلا برای اینکه ارایه مورد نظر حافظه کمتری را اشغال کند، یا به عنوان یک قانون از آن استفاده کرد. مثلا حتما باید داده‌ای که در آن گذاشته می‌شود string باشه و عدد نباشد و ...

```python
import numpy as np
a = np.zeros((2,2), dtype=np.int16)

print(a)
```

```
[[0 0]
 [0 0]]
```

بعد از تعریف یک ارایه ۳ در ۳

- پرینت اول: رنک یا رتبه یا مرتبه اون ماتریکس رو داره محاسبه میکنه و مقدارش ۳ هست، چون هر سه ستون متفاوت از هم هستن و نمیشه از هیچ از ستون‌ها به ستون‌های دیگر رسید.
- پرینت دوم: جمع ستون اصلی رو داره محاسبه میکنه (11 = 7 + 2 – 6) که بهش میگه Trace یا اثر ماتریس
- پرینت سوم: دترمینان ماتریس مورد نظر رو داره حساب میکنه
- پرینت چهارم: معکوس ماتریس مورد نظر را محاسبه میکند.
- پرینت پنجم: یک ماتریس مربعی را به توان مشخص شده می‌رساند. داره دترمینان حساب می‌کنه، فقط به جای اینکه به توان 1- برسونه، به توان عدد داده شده (۳) می‌رسونه.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

determinant

```python
import numpy as np
A = np.array([[6, 1, 1],
 [4, -2, 5],
 [2, 8, 7]])
print("Rank of A:", np.linalg.matrix_rank(A))
print("\nTrace of A:", np.trace(A))
print("\nDeterminant of A:", np.linalg.det(A))
print("\nInverse of A:\n", np.linalg.inv(A))
print("\nMatrix A raised to power 3:\n", np.linalg.matrix_power(A, 3))
```

```
Rank of A: 3

Trace of A: 11

Determinant of A: -306.0

Inverse of A:
 [[ 0.17647059 -0.00326797 -0.02287582]
 [ 0.05882353 -0.13071895  0.08496732]
 [-0.11764706  0.1503268   0.05228758]]

Matrix A raised to power 3:
 [[336 162 228]
 [406 162 469]
 [698 702 905]]
```

توی این بخش داره یه دستگاه دو معادله دو مجهول رو حل می‌کنه که ارایه a ضرایب x و y ما هستن و ارایه b خروجی ما:

x + 2y = 8

3x + 4y = 18

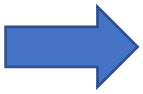
```python
import numpy as np
# coefficients
a = np.array([[1, 2], [3, 4]])
# constants
b = np.array([8, 18])
print("a=", a)
print("b=", b)
print("Solution of linear equations:", np.linalg.solve(a, b))
```

```
[ 2.  3.]
```

ابتدا یک ارایه ۱ بعدی که از توزیع گوسی پیروی میکند را ایجاد کرده و به ترتیب

- کمینه عدد یافت شده در این ارایه را حساب می‌کند
- بیشینه عدد یافت شده در این عدد را حساب می‌کند
- میانگین درایه‌های ارایه به دست آمده را حساب می‌کند
- تابع میانه یا median که مقدار میانه بین درایه‌های یک ارایه را به دست می‌اورد(از ۵۰ درصد درایه ها بزرگتر و از ۵۰ درصد درایه‌ها کوچکتر است).
- انحراف معیار را برای ماتریس به دست آمده حساب می کند

```python
import numpy as np
normal_array = np.random.normal(5, 0.5, 10)
print(normal_array)
### Min
print(np.min(normal_array))
### Max
print(np.max(normal_array))
### Mean
print(np.mean(normal_array))
### Median
print(np.median(normal_array))
### Std
print(np.std(normal_array))
```



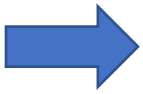
```
4.59466780526
5.70910969489
5.13010317529
5.1780697313
0.341070348128
```

ضرب داخلی دو بردار رو داره انجام میده.

(1*4)+(2*5) = 14

```python
import numpy as np
f = np.array([1,2])
g = np.array([4,5])
y = np.dot(f, g)

print("f=", f)
print("g=", g)
print("y=", y)
```



```
f= [1 2]
g= [4 5]
y= 14
```