

Bài làm thêm 1 - Numpy

```
[ ] import numpy as np
```

▶ #1. Tạo mảng ngẫu nhiên 10 phần tử số nguyên từ 10-29

```
np.random.seed(2)
arr1 = np.random.randint(10,30,10,dtype='int32')
print("Mang so nguyen 10 ptu tu 10-29:",arr1)
```

☞ Mang so nguyen 10 ptu tu 10-29: [18 25 23 18 21 28 21 18 17 12]

▶ Nhấn vào đây để xem kết quả !

```
[ ] #2. Cho biết kiểu dữ liệu các phần tử của arr1
arr1.dtype

dtype('int32')
```

▶ Nhấn vào đây để xem kết quả !

```
[ ] #3. Cho biết dung lượng bộ nhớ của mảng arr1
arr1.nbytes
```

40

▶ Nhấn vào đây để xem kết quả !

```
[ ] #4. Tạo mảng arr2 là mảng arr1 nhưng có các giá trị chia hết cho 3 được thay bằng -1
print("Mang arr1:",arr1)
arr2 = np.copy(arr1)
arr2[arr2%3==0]=-1
print("Mang arr2:",arr2)
```

```
Mang arr1: [18 25 23 18 21 28 21 18 17 12]
```

```
Mang arr2: [-1 25 23 -1 -1 28 -1 -1 17 -1]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #5. Phát sinh mảng arr3 ngẫu nhiên 10 phần tử số nguyên từ 15-35
np.random.seed(3)
arr3 = np.random.randint(15,36,10)
print(arr3)
```

```
[25 18 23 15 34 25 26 24 25 21]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #6. Liệt kê ra các phần tử giống nhau trong arr1 và arr3
print('arr1:',arr1)
print('arr3:',arr3)
arr1_3 = np.intersect1d(arr1,arr3)
print('Phan tu giống nhau:',arr1_3)
```

```
arr1: [18 25 23 18 21 28 21 18 17 12]
```

```
arr3: [25 18 23 15 34 25 26 24 25 21]
```

```
Phan tu giống nhau: [18 21 23 25]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #7. Tạo mảng arr4 là mảng arr1 nhưng có 2 dòng, 5 cột  
arr4 = arr1.reshape(2,5)  
print(arr4)
```

```
[[18 25 23 18 21]  
 [28 21 18 17 12]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #8. Hoán đổi 2 dòng của mảng arr4  
print('Hoan doi 2 dong mang arr4:')  
print(arr4[[1,0],:])
```

```
Hoan doi 2 dong mang arr4:  
[[28 21 18 17 12]  
 [18 25 23 18 21]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #8. Hoán đổi cột 1 và cột 3 của mảng arr4  
print('arr4 Truoc hoan doi')  
print(arr4)  
print('Sau hoan doi cot 1 va 3')  
print(arr4[:,[2,1,0,3,4]])
```

```
arr4 Truoc hoan doi  
[[18 25 23 18 21]  
 [28 21 18 17 12]]  
Sau hoan doi cot 1 va 3  
[[23 25 18 18 21]  
 [18 21 28 17 12]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #9. Tạo mảng arr5 kích thước 7x7 với giá trị bên trong là 0 và các giá trị biên là 1
arr5 = np.zeros([7,7],dtype=int)
arr5[[0,6],:]=1
arr5[:,[0,6]]=1
print(arr5)
```

```
[[1 1 1 1 1 1 1]
 [1 0 0 0 0 0 1]
 [1 0 0 0 0 0 1]
 [1 0 0 0 0 0 1]
 [1 0 0 0 0 0 1]
 [1 0 0 0 0 0 1]
 [1 1 1 1 1 1 1]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #10. Cộng thêm 3 vào các phần tử của mảng arr5
arr5 += 3
arr5
```

```
array([[4, 4, 4, 4, 4, 4, 4],
       [4, 3, 3, 3, 3, 3, 4],
       [4, 3, 3, 3, 3, 3, 4],
       [4, 3, 3, 3, 3, 3, 4],
       [4, 3, 3, 3, 3, 3, 4],
       [4, 3, 3, 3, 3, 3, 4],
       [4, 4, 4, 4, 4, 4, 4]])
```

► Nhấn vào đây để xem kết quả !

```
[ ] #11. Tạo mảng arr6 là mảng 1 chiều của mảng arr5
import itertools
# arr6 = list(arr5)
arr6 = list(itertools.chain.from_iterable(arr5))
arr6 = np.array(arr6)
arr6
```

```
array([4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4,
       3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 3, 3, 3, 3, 3, 4, 4, 4,
       4, 4, 4, 4, 4])
```

► Nhấn vào đây để xem kết quả !

```
[ ] #12. Tạo mảng arr7 là 5 phần tử ngẫu nhiên từ arr3
np.random.seed(8)
arr7 = np.random.choice(arr3,5)
arr7
```

```
array([15, 34, 18, 21, 25])
```

► Nhấn vào đây để xem kết quả !

▼ Part 2 - Đọc dữ liệu và tính toán thống kê

```
[ ] #1. Đọc dữ liệu từ tập tin shark.tsv (dữ liệu số vụ cá mập tấn công các tháng của 3 quốc gia USA,Australia,South Africa)
# và đưa vào mảng shark (Gợi ý: sử dụng hàm np.genfromtxt)
from google.colab import drive
drive.mount('/content/drive')

# data = np.genfromtxt('/content/drive/MyDrive/aTest/shark.tsv',delimiter=',',skip_header=1)
data = np.genfromtxt('/content/drive/MyDrive/aTest/shark.tsv',dtype=int,delimiter=None,skip_header=1)
data
```

```
Mounted at /content/drive
array([[ -1,  28,  94,  68],
       [ -1,  27,  78,  32],
       [ -1,  66,  63,  34],
       [ -1, 103,  54,  25],
       [ -1, 106,  21,  25],
       [ -1, 173,  25,  22],
       [ -1, 232,  25,  28],
       [ -1, 208,  28,  15],
       [ -1, 182,  35,  17],
       [ -1, 141,  56,  16],
       [ -1,  79,  65,  20],
       [ -1,  37,  94,  46]])
```

► Nhấn vào đây để xem kết quả !

```
[ ] #2. Cho biết số phần tử mỗi chiều của mảng
data.shape
```

```
(12, 4)
```

► Nhấn vào đây để xem kết quả !


```
[ ] #3. Thay cột 1 (giá trị -1) bằng giá trị tương ứng các tháng trong năm 1-12
data[:,0]=np.arange(1,13)
data

array([[ 1, 28, 94, 68],
       [ 2, 27, 78, 32],
       [ 3, 66, 63, 34],
       [ 4, 103, 54, 25],
       [ 5, 106, 21, 25],
       [ 6, 173, 25, 22],
       [ 7, 232, 25, 28],
       [ 8, 208, 28, 15],
       [ 9, 182, 35, 17],
       [10, 141, 56, 16],
       [11, 79, 65, 20],
       [12, 37, 94, 46]])
```

► Nhấn vào đây để xem kết quả !

```
[ ] #4. Hiển thị số vụ cá mập tấn công trung bình của 3 quốc gia
qg1 = np.mean(data[:,1])
qg2 = np.mean(data[:,2])
qg3 = np.mean(data[:,3])
np.array([qg1,qg2,qg3])

array([115.16666667, 53.16666667, 29.        ])
```

► Nhấn vào đây để xem kết quả !

```
[ ] #5. Hiển thị số vụ cá mập tấn công trung bình của mỗi tháng
th1 = np.mean(data[0,1:])
th2 = np.mean(data[1,1:])
th3 = np.mean(data[2,1:])
th4 = np.mean(data[3,1:])
th5 = np.mean(data[4,1:])
th6 = np.mean(data[5,1:])
th7 = np.mean(data[6,1:])
th8 = np.mean(data[7,1:])
th9 = np.mean(data[8,1:])
th10 = np.mean(data[9,1:])
th11 = np.mean(data[10,1:])
th12 = np.mean(data[11,1:])
np.array([th1,th2,th3,th4,th5,th6,th7,th8,th9,th10,th11,th12])
```

```
array([63.33333333, 45.66666667, 54.33333333, 60.66666667, 50.66666667,
       73.33333333, 95.        , 83.66666667, 78.        , 71.        ,
       54.66666667, 59.        ])
```

► Nhấn vào đây để xem kết quả !


```
[ ] #6. Tạo thêm cột tổng số vụ cá mập tấn công theo mỗi tháng
# print(data)
TongTh1 = np.sum(data[0,1:])
TongTh2 = np.sum(data[1,1:])
TongTh3 = np.sum(data[2,1:])
TongTh4 = np.sum(data[3,1:])
TongTh5 = np.sum(data[4,1:])
TongTh6 = np.sum(data[5,1:])
TongTh7 = np.sum(data[6,1:])
TongTh8 = np.sum(data[7,1:])
TongTh9 = np.sum(data[8,1:])
TongTh10 = np.sum(data[9,1:])
TongTh11 = np.sum(data[10,1:])
TongTh12 = np.sum(data[11,1:])
data = np.insert(data,[4],[TongTh1],[TongTh2],[TongTh3],[TongTh4],[TongTh5],[TongTh6],[TongTh7],[TongTh8],[TongTh9],[TongTh10],[TongTh11],[TongTh12])
print(data)
```

```
[[ 1  28  94  68 190]
 [ 2  27  78  32 137]
 [ 3  66  63  34 163]
 [ 4 103  54  25 182]
 [ 5 106  21  25 152]
 [ 6 173  25  22 220]
 [ 7 232  25  28 285]
 [ 8 208  28  15 251]
 [ 9 182  35  17 234]
 [10 141  56  16 213]
 [11  79  65  20 164]
 [12  37  94  46 177]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #7. Cho biết những tháng có tổng số vụ cá mập tấn công >200
# print(data)
# print(np.where(data[:,4]>200))
vitri = np.where(data[:,4]>200)
print(data[vitri,:])
```

```
[[[ 6 173 25 22 220]
   [ 7 232 25 28 285]
   [ 8 208 28 15 251]
   [ 9 182 35 17 234]
   [10 141 56 16 213]]]
```

► Nhấn vào đây để xem kết quả !

```
[ ] #8. Cho biết tổng số vụ cá mập tấn công nhiều nhất trong tháng là bao nhiêu
print(np.max(data[:,4]))
```

285

► Nhấn vào đây để xem kết quả !

```
[ ] #9. Cho biết tháng nào có tổng số vụ cá mập tấn công nhiều nhất
LstThang = [TongTh1,TongTh2,TongTh3,TongTh4,TongTh5,TongTh6,TongTh7,TongTh8,TongTh9,TongTh10,TongTh11,TongTh12]
# print(LstThang.index(max(LstThang)))
print('Thang co tong so vu ca map tan cong nhieu nhat la thang:',LstThang.index(max(LstThang))+1)
```

Thang co tong so vu ca map tan cong nhieu nhat la thang: 7

► Nhấn vào đây để xem kết quả !

Chapter 4 - Exercise 1: Thực hiện những yêu cầu liên quan đến series, và đối chiếu với kết quả cho trước

Part 1: Thực hiện các phép toán trên series

```
[ ] import numpy as np
import pandas as pd
```

```
[ ] # Câu 1a: Cho arr_1 là mảng số nguyên chẵn [2, 4, 6, 8, 10], arr_2 là mảng số nguyên lẻ [1, 3, 5, 7, 11]
# Tạo biến kiểu Series ser1 từ arr_1, ser2 từ arr_2
arr_1 = np.array([2, 4, 6, 8, 10])
arr_2 = np.array([1, 3, 5, 7, 11])
ser1 = pd.Series(arr_1, dtype='int32')
ser2 = pd.Series(arr_2, dtype='int32')
# In danh sách các phần tử của ser1 và ser2
print(ser1)
print(ser2)
```

```
0    2
1    4
2    6
3    8
4   10
dtype: int32
0    1
1    3
2    5
3    7
4   11
dtype: int32
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1b: Thực hiện phép toán và thể hiện kết quả của: ser1 + ser2
print(ser1 + ser2)
```

```
0      3
1      7
2     11
3     15
4     21
dtype: int32
```

► Nhấn vào đây để xem kết quả!

```
[ ]
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1d: Thực hiện phép toán và thể hiện kết quả của: ser1 * ser2
print(ser1*ser2)
```

```
0      2
1     12
2     30
3     56
4    110
dtype: int32
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1e: Thực hiện phép toán và thể hiện kết quả của: ser1 / ser2
print (ser1/ser2)
```

```
0    2.000000
1    1.333333
2    1.200000
3    1.142857
4    0.909091
dtype: float64
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 2a: Kiểm tra xem các phần tử của ser1 có > các phần tử của ser2 không?
print(ser1>ser2)
```


```
0    True
1    True
2    True
3    True
4    False
dtype: bool
```


► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 2b: Kiểm tra xem các phần tử của ser1 có < các phần tử của ser2 không?
print(ser1<ser2)
```

```
0    False
1    False
2    False
3    False
4    True
dtype: bool
```

► Nhấn vào đây để xem kết quả!

 # Câu 2c: Kiểm tra xem các phần tử của ser1 có = các phần tử của ser2 không?
`print(ser1==ser2)`

 0 False
1 False
2 False
3 False
4 False
dtype: bool

► Nhấn vào đây để xem kết quả!

[] # Câu 3a: Thêm 2 phần tử [6, 12] vào ser2 => sử dụng append

```
# In lại danh sách các phần tử của ser2.  
ser2 = ser2.append(pd.Series([6,12]))  
print(ser2)
```

```
0    1  
1    3  
2    5  
3    7  
4   11  
0    6  
1   12  
dtype: int64
```

► Nhấn vào đây để xem kết quả!


```
[ ] # Câu 3b: Tạo series ser3 chỉ chứa các phần tử có trong ser1 mà không có trong ser2. => gọi y dung: ham isin
#      In danh sách các phần tử của ser3
print("Ser1:")
print(ser1)
print("Ser2:")
print(ser2)
print("Ser3:")
# ~ser1.isin(ser2)
ser3 = ser1[~ser1.isin(ser2)]
print(ser3)
```

```
Ser1:
0      2
1      4
2      6
3      8
4     10
dtype: int32
Ser2:
0      1
1      3
2      5
3      7
4     11
0      6
1     12
dtype: int64
Ser3:
0      2
1      4
3      8
4     10
dtype: int32
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 3c: Tạo series ser4 chỉ chứa các phần tử có trong ser2 mà không có trong ser1. => gọi y dung: ham isin
#      In danh sách các phần tử của ser4
print("Ser1:")
print(ser1)
print("Ser2:")
print(ser2)
print("Ser4:")
ser4 = ser2[~ser2.isin(ser1)]
print(ser4)
```

```
Ser1:
0      2
1      4
2      6
3      8
4     10
dtype: int32
Ser2:
0      1
1      3
2      5
3      7
4     11
0      6
1     12
dtype: int64
Ser4:
0      1
1      3
2      5
3      7
4     11
1     12
dtype: int64
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 4: Tạo series ser5 chứa các phần tử chỉ có trong ser1 và chỉ có trong ser2
#     In danh sách các phần tử của ser5
ser5 = ser3.append(pd.Series(ser4))
print('ser5:')
print(ser5)
```

```
ser5:
0      2
1      4
3      8
4     10
0      1
1      3
2      5
3      7
4     11
1     12
dtype: int64
```

► Nhấn vào đây để xem kết quả!

▼ Part 2: Truy xuất các phần tử, và thống kê thông tin trên series

```
[ ] # Câu 1a: Tạo series ser6 có 35 phần tử số nguyên ngẫu nhiên có giá trị trong khoảng từ 1 đến 9.  
    np.random.seed(1)  
    ser6 = pd.Series(np.random.randint(1,10,35),dtype='int32')  
    # print(ser6)  
    # Cho biết kích thước (shape) của ser6  
    print('kích thước shape của ser6:',ser6.shape)  
  
    # Xem 5 dòng dữ liệu đầu tiên (head) và 5 dòng dữ liệu cuối cùng (tail) có trong ser6  
    print(ser6.head())  
    print(ser6.tail())
```

kích thước shape của ser6: (35,)

```
0    6  
1    9  
2    6  
3    1  
4    1  
dtype: int32  
30   8  
31   4  
32   7  
33   6  
34   2  
dtype: int32
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1b: In danh sách các phần tử của ser6 theo dạng array  
print(np.array(ser6))
```

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

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1c: Cho biết thông tin thống kê chung (describe()) của ser6  
print(ser6.describe())
```

```
count      35.000000  
mean       5.200000  
std        2.763204  
min        1.000000  
25%        2.500000  
50%        6.000000  
75%        8.000000  
max        9.000000  
dtype: float64
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1d: Cho biết tổng của các phần tử có trong ser6  
print(ser6.sum())
```

```
182
```

► Nhấn vào đây để xem kết quả!



+ Code + Text

```
[ ] # Câu 1e: Cho biết phần tử có tần suất xuất hiện nhiều nhất trong ser6 => sử dụng mode
print(ser6.mode())
```

```
0      8
dtype: int32
```

▶ Nhấn vào đây để xem kết quả!

```
[ ] # Câu 2: Liệt kê các dòng trong ser6 mà giá trị chia hết cho 2 và cho 3
print(ser6[(ser6%2==0) & (ser6%3==0)])
```

```
0      6
2      6
10     6
33     6
dtype: int32
```

▶ Nhấn vào đây để xem kết quả!

```
[ ] # Câu 3: In các phần tử ở vị trí 0, 5, 10, 15 trong ser6
# print(ser6.index[[0,5,10,15]].values)
print(pd.Series(ser6,index=[0,5,10,15]))
```

```
0      6
5      2
10     6
15     8
dtype: int32
```

▶ Nhấn vào đây để xem kết quả!



```
# Câu 4: In ra các giá trị unique (array) trong ser6  
pd.unique(pd.Series(ser6))
```



```
array([6, 9, 1, 2, 8, 7, 3, 5, 4], dtype=int32)
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 5: Tạo series ser7 với mỗi phần tử có giá trị = lập phương của phần tử trong ser6.  
ser7 = ser6.map(lambda x: x ** 3.0)
```

```
# Xem 5 dòng dữ liệu đầu tiên (head) của ser7  
print(ser7.head())
```

```
0    216.0  
1    729.0  
2    216.0  
3      1.0  
4      1.0  
dtype: float64
```

► Nhấn vào đây để xem kết quả!

▼ Part 3: Tạo series từ list, chuỗi và biểu thức điều kiện

```
[ ] # Câu 1: Cho list sau:  
lst = ["abc", "defg", "htmlmj", "dfg", "ljsac"]
```

```
[ ] # Câu 1a: Tạo series ser_chuoi từ lst  
ser_chuoi = pd.Series(lst)  
print(ser_chuoi)
```

```
0    abc  
1   defg  
2  htmlmj  
3    dfg  
4   ljsac  
dtype: object
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 1b: Tạo series ser_dodai với mỗi phần tử có giá trị là chiều dài của mỗi phần tử trong ser_chuoi  
ser_dodai = ser_chuoi.str.len()  
print(ser_dodai)
```

```
0    3  
1    4  
2    5  
3    3  
4    5  
dtype: int64
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 2: Cho ser = pd.Series(np.array([1, 2, 4, 5, 8, 7, 6, 9])).
# Sử dụng biểu thức điều kiện thích hợp để in ra các dòng trong ser có giá trị là số nguyên tố
import math

ser = pd.Series(np.array([1, 2, 4, 5, 8, 7, 6, 9]))
print(ser)

def PrimeNum(n):
    if (n<=1):
        return False
    for i in range (2,int(math.sqrt(n))+1):
        if (n%i==0):
            return False
    return True

print(ser[ser.map(lambda x : PrimeNum(x))])
```

```
0    1
1    2
2    4
3    5
4    8
5    7
6    6
7    9
dtype: int64
1    2
3    5
5    7
dtype: int64
```

► Nhấn vào đây để xem kết quả!

```
[ ] import re
# Câu 3: Cho mẫu email như sau: => có thể đợi thầy hướng dẫn
pattern = '[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}'

# Tạo một series ser_ch, với mỗi phần tử trong ser_ch là một chuỗi
# Gợi ý: 'reading newspaper from tuoitre.vn', 'tubirona@gmail.com', 'nguyen.nn@yahoo.com', 'tran_2014@hotmail.com.vn'
ser_ch = pd.Series(['reading newspaper from tuoitre.vn', 'tubirona@gmail.com', 'nguyen.nn@yahoo.com', 'tran_2014@hotmail.com.vn'])
is_email = ser_ch.map(lambda x : bool(re.match(pattern,x)))
is_email
# In ra những dòng trong ser_ch thỏa điều kiện chuỗi là email
ser_ch[is_email]
```

```
1      tubirona@gmail.com
2      nguyen.nn@yahoo.com
3      tran_2014@hotmail.com.vn
dtype: object
```

► Nhấn vào đây để xem kết quả!

```
[ ] # Câu 4: Cho series:
ser_names = pd.Series(['Manufacturer', 'Model', 'CarType', 'Min_Price', 'Price', 'Max_Price',
                        'MPG_city', 'MPG_highway', 'AirBags', 'DriveTrain', 'Cylinders',
                        'EngineSize', 'Horsepower', 'RPM', 'Rev_per_mile', 'Man_trans_avail',
                        'Fuel_tank_capacity', 'Passengers', 'Length', 'Wheelbase', 'Width',
                        'Turn_circle', 'Rear_seat_room', 'Luggage_room', 'Weight', 'Origin',
                        'Make'])
# Sử dụng biểu thức điều kiện thích hợp để in ra các dòng của ser_names thỏa điều kiện trong chuỗi có chữ 'Price'
is_hasPrice = ser_names.map(lambda x: "Price" in x)
is_hasPrice
ser_names[is_hasPrice]
```

```
3      Min_Price
4      Price
5      Max_Price
dtype: object
```

► Nhấn vào đây để xem kết quả!