numpy_pandas

January 30, 2024

1 Numpy

1.1 Ex 1

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
# tim chan le trong khoang 0-9

arr = np.arange(0, 10)
print(arr[arr%2==0])
print(arr[arr%2!=0])
# print(arr)
```

```
[0 2 4 6 8]
[1 3 5 7 9]
```

[3]:
```
# in ra cac phan tu tu 5-10

arr_e = np.array([2,6,1,9,10,3,27,8,6,25,16])

print(arr_e[(arr_e >=5) & (arr_e <=10)])
```

```
[ 6  9 10  8  6]
```

[4]:
```
# In ra thu tu dao nguoc cua mang

arr_h = np.arange(10,25)

print(np.flip(arr_h))
print(arr_h[::-1])
```

```
[24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]
[24 23 22 21 20 19 18 17 16 15 14 13 12 11 10]
```

[5]:
```
# in danh sach cac phan tu != 0

arr_k = np.array([1,2,0,8,2,0,1,3,0,5,0])
```

```
print(arr_k[arr_k!=0])
print(arr_k!=0)

# xoa phan tu
arr_k = np.delete(arr_k, [1, 2])

print(arr_k)
```

```
[1 2 8 2 1 3 5]
[ True  True False  True  True False  True  True False  True False]
[1 8 2 0 1 3 0 5 0]
```

1.2 Ex 2

[6]:
```
# doc file txt va chuyen thanh mang np

with open(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/chapter3_data/
 ↪data/heights_1.txt', 'r') as f:
  heights = f.read().replace("[", "").replace("]", "").split(', ')
np_heights = np.array(heights)
np_heights = np_heights.astype(int)
print(np_heights)
print(np_heights.shape)

with open(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/chapter3_data/
 ↪data/weights_1.txt', 'r') as f:
  weights_1 = f.read().replace("[", "").replace("]", "").split(', ')
np_weights = np.array(weights_1)
np_weights = np_weights.astype(int)
print(np_weights)
print(np_weights.shape)
```

```
[74 74 72 … 75 75 73]
(1015,)
[180 215 210 … 205 190 195]
(1015,)
```

[7]:
```
# nhan arr voi 1 so

arr_heights_m = np_heights * 0.0254
arr_weights_kg = np_weights * 0.453592
print(arr_heights_m)
print(arr_weights_kg)
```

```
[1.8796 1.8796 1.8288 … 1.905  1.905  1.8542]
[81.64656 97.52228 95.25432 … 92.98636 86.18248 88.45044]
```

[8]:
```
# tinh chi so BMI

arr_bmi = arr_weights_kg/(arr_heights_m * arr_heights_m)
```

[9]:
```
# gia tri vi tri can nang index = 50

print(arr_weights_kg[50])
```

```
90.7184
```

[10]:
```
# array_heights_m_100

arr_heights_m_100 = arr_heights_m[100:111]

arr_heights_m_100
```

[10]:
```
array([1.8542, 1.8796, 1.8288, 1.8542, 1.7526, 1.8288, 1.8542, 1.905 ,
       1.905 , 1.8542, 1.8288])
```

[11]:
```
# tao va in ra ket qua bmi < 21

arr_bmi[arr_bmi < 21]
```

[11]:
```
array([20.54255679, 20.54255679, 20.69282047, 20.69282047, 20.34343189,
       20.34343189, 20.69282047, 20.15883472, 19.4984471 , 20.69282047,
       20.9205219 ])
```

[12]:
```
# chieu cao va can nang trung binh

arr_heights_mean = np.mean(arr_heights_m)
arr_weights_mean = arr_weights_kg.mean()
arr_heights_mean
arr_weights_mean
```

[12]:
```
91.33019058916256
```

[13]:
```
# chieu cao va can nang lon nhat (nho nhat) cua cac cau thu

arr_heights_max = max(arr_heights_m)
arr_weights_max = max(arr_weights_kg)

arr_heights_min = min(arr_heights_m)
arr_weights_min = min(arr_weights_kg)

arr_heights_max
arr_weights_max
```

```
arr_heights_min
arr_weights_min
```

```
[13]: 68.0388
```

```
[14]: # cho biet chieu cao lon nhat tai vi tri index

arr_heights_max = max(arr_heights_m)

arr_index_heights_max = np.where(arr_heights_m == arr_heights_max)
print(arr_index_heights_max)
print(arr_heights_max)
```

```
(array([909]),)
2.1082
```

```
[15]: # in ra chieu cao tang dan, giarm dan

print(sorted(arr_heights_m))
print(sorted(arr_heights_m, reverse=True))
```

```
[1.7018, 1.7018, 1.7271999999999998, 1.7271999999999998, 1.7271999999999998,
1.7271999999999998, 1.7271999999999998, 1.7271999999999998, 1.7271999999999998,
1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526,
1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
```

```
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
```

```
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
```

1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.032, 2.032, 2.032, 2.032, 2.032, 2.0574, 2.0574,
2.0827999999999998, 2.0827999999999998, 2.1082]
[2.1082, 2.0827999999999998, 2.0827999999999998, 2.0574, 2.0574, 2.032, 2.032,
2.032, 2.032, 2.032, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 2.0065999999999997,
2.0065999999999997, 2.0065999999999997, 2.0065999999999997, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9811999999999999, 1.9811999999999999,
1.9811999999999999, 1.9811999999999999, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558, 1.9558,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304, 1.9304,
1.9304, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,

1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905, 1.905,
1.905, 1.905, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8796,
1.8796, 1.8796, 1.8796, 1.8796, 1.8796, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,

```
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8541999999999998, 1.8541999999999998, 1.8541999999999998, 1.8541999999999998,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288, 1.8288,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034,
1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.8034, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.778,
1.778, 1.778, 1.778, 1.778, 1.778, 1.778, 1.7526, 1.7526, 1.7526, 1.7526,
1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7526,
1.7526, 1.7526, 1.7526, 1.7526, 1.7526, 1.7271999999999998, 1.7271999999999998,
1.7271999999999998, 1.7271999999999998, 1.7271999999999998, 1.7271999999999998,
1.7271999999999998, 1.7018, 1.7018]
```

1.3 Ex 3

[16]:
```
arr_1D = np.arange(0, 9)
arr_2D = arr_1D.reshape(((3, 3)))
# cột 1 và 3 đổi lại vị trí cho nhau
arr_2D = arr_2D[:, [2,1,0]]
print(arr_2D)
```

```
[[2 1 0]
 [5 4 3]
 [8 7 6]]
```

[17]:
```
# any giống or, nếu có 1 trong các điều kiện là True thì sẽ trả về True, toàn
 ↪bộ là False thì sẽ trả về False
# all ngược lại so với any
# isnan: kiểm tra gái trị rỗng

arr_2D = arr_2D[[1,0,2], :]
arr_2D
```

[17]:
```
array([[5, 4, 3],
       [2, 1, 0],
       [8, 7, 6]])
```

[18]:
```
# dao nguoc cac dong

arr_2D = np.flip(arr_2D, axis = 0)
arr_2D
```

[18]:
```
array([[8, 7, 6],
       [2, 1, 0],
       [5, 4, 3]])
```

[19]:
```
# dao nguoc cac cot

arr_2D = np.flip(arr_2D, axis = 1)
arr_2D
```

[19]:
```
array([[6, 7, 8],
       [0, 1, 2],
       [3, 4, 5]])
```

[20]:
```
# thay the null bang 0
arr_2D_null = np.array([[1,2,3], [np.NaN,5,6], [7, np.NaN, 9], [4, 5, 6]])
print(arr_2D_null)

print(np.where(np.isnan(arr_2D_null).any() == True, 'array rong', 'arr khong
 ↪rong'))
```

```
print(np.where(np.isnan(arr_2D_null) == True, 0, arr_2D_null))
arr_2D_null[np.isnan(arr_2D_null)] = 0
print(arr_2D_null)
```

```
[[ 1.  2.  3.]
 [nan  5.  6.]
 [ 7. nan  9.]
 [ 4.  5.  6.]]
array rong
[[1. 2. 3.]
 [0. 5. 6.]
 [7. 0. 9.]
 [4. 5. 6.]]
[[1. 2. 3.]
 [0. 5. 6.]
 [7. 0. 9.]
 [4. 5. 6.]]
```

1.4 Ex4

[21]:
```
# tinh chieu cao trung binh, can nang trung binh

baseball = [[74, 180], [74, 180], [73, 190], [73, 200], [76, 257], [73, 190],␣
 ↪[75, 220], [70, 165], [77, 205], [72, 200], [77, 208], [74, 185], [75, 215],␣
 ↪[75, 170], [75, 235], [75, 210], [72, 170], [74, 180], [71, 170], [76, 190],␣
 ↪[71, 150], [75, 230], [76, 203], [83, 260], [75, 246], [74, 186], [76, 210],␣
 ↪[72, 198], [72, 210], [75, 215], [75, 180], [72, 200], [77, 245], [73, 200],␣
 ↪[72, 192], [70, 192], [74, 200], [72, 192], [74, 205], [72, 190], [71, 186],␣
 ↪[70, 170], [71, 197], [76, 219], [74, 200], [76, 220], [74, 207], [74, 225],␣
 ↪[74, 207], [75, 212], [75, 225], [71, 170], [71, 190], [74, 210], [77, 230],␣
 ↪[71, 210], [74, 200], [75, 238], [77, 234], [76, 222], [74, 200], [76, 190],␣
 ↪[72, 170], [71, 220], [72, 223], [75, 210], [73, 215], [68, 196], [72, 175],␣
 ↪[69, 175], [73, 189], [73, 205], [75, 210], [70, 180], [70, 180], [74, 197],␣
 ↪[75, 220], [74, 228], [74, 190], [73, 204], [74, 165], [75, 216], [77, 220],␣
 ↪[73, 208], [74, 210], [76, 215], [74, 195], [75, 200], [73, 215], [76, 229],␣
 ↪[78, 240], [75, 207], [73, 205], [77, 208], [74, 185], [72, 190], [74, 170],␣
 ↪[72, 208], [71, 225], [73, 190], [75, 225], [73, 185], [67, 180], [67, 165],␣
 ↪[76, 240], [74, 220], [73, 212], [70, 163], [75, 215], [70, 175], [72, 205],␣
 ↪[77, 210], [79, 205], [78, 208], [74, 215], [75, 180], [75, 200], [78, 230],␣
 ↪[76, 211], [75, 230], [69, 190], [75, 220], [72, 180], [75, 205], [73, 190],␣
 ↪[74, 180], [75, 205], [75, 190], [73, 195]]

np_baseball = np.array(baseball)
print(type(np_baseball))
print(np_baseball.shape)

heights_mean = np.mean(np_baseball, axis = 0)[0]
```

```
weights_mean = np.mean(np_baseball, axis = 0)[1]
heights_mean
weights_mean
```

```
<class 'numpy.ndarray'>
(129, 2)
```

[21]:
```
202.34883720930233
```

[22]:
```
# tinh he so tuong quan

x = np_baseball[:, 0]
y = np_baseball[:, 1]
np.corrcoef(x, y)
```

[22]:
```
array([[1.        , 0.60603766],
       [0.60603766, 1.        ]])
```

1.5 Ex5

[23]:
```
import numpy as np

with open(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/chapter3_data/
 ↪data/heights.txt', 'r') as f:
  heights = f.read().replace("[", "").replace("]", "").split(', ')


with open(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/chapter3_data/
 ↪data/positions.txt', 'r') as f:
  positions = f.read().replace("[", "").replace("]", "").replace("'", "").
 ↪replace(" ", "").split(',')

# print(positions)
np_positions = np.array(positions)
np_heights = np.array(heights)

# print(np_positions)

# convert string to int using astype
np_heights = np_heights.astype(int)
# print(np_heights)

# print(np_heights[np_positions == 'GK'])

GK_heights = np_heights[np_positions == 'GK']
Not_GK_heights = np_heights[np_positions != 'GK']
```

```
M_heights = np_heights[np_positions == 'M']
A_heights = np_heights[np_positions == 'A']
D_heights = np_heights[np_positions == 'D']

print(np.mean(GK_heights))
print(max(GK_heights))
print(min(GK_heights))
print(np.mean(Not_GK_heights))
print(np.mean(M_heights))
print(np.mean(A_heights))
print(np.mean(D_heights))
```

```
188.23333333333332
208
173
180.98888467853985
179.0417625780993
180.93852065321806
183.14566929133858
```

[24]:
```
import time
import sys

SIZE = 1000000

L1= range(SIZE)
L2= range(SIZE)
A1= np.arange(SIZE)
A2=np.arange(SIZE)

start= time.time()
result=[(x,y) for x,y in zip(L1,L2)]
print((time.time()-start)*1000)

start=time.time()
result= A1+A2
print((time.time()-start)*1000)
print(result)
```

```
218.9161777496338
55.68242073059082
[      0       2       4 … 1999994 1999996 1999998]
```

1.6 Ex 6

[25]:
```
import numpy as np
import pandas as pd
```

[26]:
```
# numpy doc du lieu tu csv

wines = np.genfromtxt("/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪chapter3_data/data/winequality-red.csv", delimiter= ";", skip_header=1)
print(wines, wines.shape)
wines[:5].tolist()
```

```
[[ 7.4    0.7    0.    …  0.56   9.4    5.   ]
 [ 7.8    0.88   0.    …  0.68   9.8    5.   ]
 [ 7.8    0.76   0.04  …  0.65   9.8    5.   ]
 …
 [ 6.3    0.51   0.13  …  0.75  11.     6.   ]
 [ 5.9    0.645  0.12  …  0.71  10.2    5.   ]
 [ 6.     0.31   0.47  …  0.66  11.     6.   ]] (1599, 12)
```

[26]:
```
[[7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56, 9.4, 5.0],
 [7.8, 0.88, 0.0, 2.6, 0.098, 25.0, 67.0, 0.9968, 3.2, 0.68, 9.8, 5.0],
 [7.8, 0.76, 0.04, 2.3, 0.092, 15.0, 54.0, 0.997, 3.26, 0.65, 9.8, 5.0],
 [11.2, 0.28, 0.56, 1.9, 0.075, 17.0, 60.0, 0.998, 3.16, 0.58, 9.8, 6.0],
 [7.4, 0.7, 0.0, 1.9, 0.076, 11.0, 34.0, 0.9978, 3.51, 0.56, 9.4, 5.0]]
```

[27]:
```
# Tao wines_4 la mang trich ra tu mang wines gom 4 cot "fixed acidity", "ph",␣
 ↪"alcohol", "quality"

wines_4 = wines[:, [0,8,10,11]]
wines_4
```

[27]:
```
array([[ 7.4 ,  3.51,  9.4 ,  5.  ],
       [ 7.8 ,  3.2 ,  9.8 ,  5.  ],
       [ 7.8 ,  3.26,  9.8 ,  5.  ],
       …,
       [ 6.3 ,  3.42, 11.  ,  6.  ],
       [ 5.9 ,  3.57, 10.2 ,  5.  ],
       [ 6.  ,  3.39, 11.  ,  6.  ]])
```

[28]:
```
# tinh trung binh cac cot trong mang wines_4

wines_4.mean(axis = 0)
```

[28]:
```
array([ 8.31963727,  3.3111132 , 10.42298311,  5.63602251])
```

[29]:
```
# tinh trung binh, min, max cuar quality

print(wines[:, [11]].mean(axis = 0))
# np.mean(wines[:, [11]], axis = 0)
print(wines[:, [11]].min())
print(wines[:, [11]].max())
```

```
[5.63602251]
3.0
8.0
```

[30]:
```
# lay tt ruou co diem > 5
wines[wines[:, -1] > 5]
```

[30]:
```
array([[11.2 ,  0.28,  0.56, …,  0.58,  9.8 ,  6.  ],
       [ 7.3 ,  0.65,  0.  , …,  0.47, 10.  ,  7.  ],
       [ 7.8 ,  0.58,  0.02, …,  0.57,  9.5 ,  7.  ],
       …,
       [ 5.9 ,  0.55,  0.1 , …,  0.76, 11.2 ,  6.  ],
       [ 6.3 ,  0.51,  0.13, …,  0.75, 11.  ,  6.  ],
       [ 6.  ,  0.31,  0.47, …,  0.66, 11.  ,  6.  ]])
```

[31]:
```
# cho biet 5 ruou dau tien co diem lon nhat va so luong ruou nhu vay

print(wines[wines[:, -1] == wines[:, -1].max()][:5])

print(wines[wines[:, -1] == wines[:, -1].max()].shape[0])
```

```
[[7.900e+00 3.500e-01 4.600e-01 3.600e+00 7.800e-02 1.500e+01 3.700e+01
  9.973e-01 3.350e+00 8.600e-01 1.280e+01 8.000e+00]
 [1.030e+01 3.200e-01 4.500e-01 6.400e+00 7.300e-02 5.000e+00 1.300e+01
  9.976e-01 3.230e+00 8.200e-01 1.260e+01 8.000e+00]
 [5.600e+00 8.500e-01 5.000e-02 1.400e+00 4.500e-02 1.200e+01 8.800e+01
  9.924e-01 3.560e+00 8.200e-01 1.290e+01 8.000e+00]
 [1.260e+01 3.100e-01 7.200e-01 2.200e+00 7.200e-02 6.000e+00 2.900e+01
  9.987e-01 2.880e+00 8.200e-01 9.800e+00 8.000e+00]
 [1.130e+01 6.200e-01 6.700e-01 5.200e+00 8.600e-02 6.000e+00 1.900e+01
  9.988e-01 3.220e+00 6.900e-01 1.340e+01 8.000e+00]]
18
```

[32]:
```
# cho biet 3 ruou dau tien co diem >7
print(wines[wines[:, -1] >= 7][:3])
```

```
[[7.300e+00 6.500e-01 0.000e+00 1.200e+00 6.500e-02 1.500e+01 2.100e+01
  9.946e-01 3.390e+00 4.700e-01 1.000e+01 7.000e+00]
 [7.800e+00 5.800e-01 2.000e-02 2.000e+00 7.300e-02 9.000e+00 1.800e+01
  9.968e-01 3.360e+00 5.700e-01 9.500e+00 7.000e+00]
 [8.500e+00 2.800e-01 5.600e-01 1.800e+00 9.200e-02 3.500e+01 1.030e+02
```

```
9.969e-01 3.300e+00 7.500e-01 1.050e+01 7.000e+00]]
```

[33]:
```
# cho biet cac loai ruou co diem > 7 va do con > 10

wines[np.where((wines[:, -1] > 7) & (wines[:, -2] > 10))][:, [-2, -1]]
```

[33]:
```
array([[12.8,  8. ],
       [12.6,  8. ],
       [12.9,  8. ],
       [13.4,  8. ],
       [11.7,  8. ],
       [11. ,  8. ],
       [11. ,  8. ],
       [14. ,  8. ],
       [12.7,  8. ],
       [12.5,  8. ],
       [11.8,  8. ],
       [13.1,  8. ],
       [11.7,  8. ],
       [14. ,  8. ],
       [11.3,  8. ],
       [11.4,  8. ]])
```

2 Pandas

2.1 Co ban

2.1.1 Ex1

[34]:
```
import numpy as np
import pandas as pd
```

[35]:
```
# arr_1 va 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)
ser2 = pd.Series(arr_2)

print(ser1)
print(ser2)
```

```
0     2
1     4
2     6
3     8
4    10
```

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

[36]:
```
# ser1 + ser2

print(ser1 + ser2)
```

```
0     3
1     7
2    11
3    15
4    21
dtype: int64
```

[37]:
```
# ser1 -ser2

print(ser1 -ser2)
```

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

[38]:
```
# noi

ser2 = pd.concat([ser2, pd.Series([6, 12])], ignore_index=True)

ser2
```

[38]:
```
0     1
1     3
2     5
3     7
4    11
5     6
6    12
dtype: int64
```

[39]:
```
# ser3 chi chua cac phan tu ser1 ma khong co trong ser2

ser3 = ser1[~ser1.isin(ser2)]
```

```
ser3
```

[39]:
```
0     2
1     4
3     8
4    10
dtype: int64
```

[40]:
```
# truy xuat cac phan tu va thong ke thong tin tren series

np.random.seed(42)
ser6 = pd.Series(np.random.randint(1,10,35))

print(ser6.values)
print(ser6.describe())

from scipy.stats import mode
# phan tu co tan suat xuat hien nhieu nhat
print(ser6.mode())
print(mode(ser6))
```

```
[7 4 8 5 7 3 7 8 5 4 8 8 3 6 5 2 8 6 2 5 1 6 9 1 3 7 4 9 3 5 3 7 5 9 7]
count    35.000000
mean      5.428571
std       2.342519
min       1.000000
25%       3.500000
50%       5.000000
75%       7.000000
max       9.000000
dtype: float64
0    5
1    7
dtype: int64
ModeResult(mode=5, count=6)
```

2.1.2 Ex2

[41]:
```
dic_1 = {'X':[78, 85, 96, 80, 86], 'Y':[86, 94, 89, 83, 86], 'Z': [86, 97, 96,␣
↪72, 83]}

df1=pd.DataFrame(dic_1)
df1
```

[41]:

	X	Y	Z
0	78	86	86

```
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

[42]:
```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily',␣
 ↪'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5,␣
 ↪np.NaN, 9, 20, 14.5, np.NaN, 8, 19], 'attempts':[1,3,2,3,2,3,1,1,2,1],␣
 ↪'quality': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

lables = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df2 = pd.DataFrame(exam_data, index = lables)

df2
```

[42]:

	name	score	attempts	quality
a	Anastasia	12.5	1	yes
b	Dima	9.0	3	no
c	Katherine	16.5	2	yes
d	James	NaN	3	no
e	Emily	9.0	2	no
f	Michael	20.0	3	yes
g	Matthew	14.5	1	yes
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes

[43]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   name      10 non-null     object
 1   score     8 non-null      float64
 2   attempts  10 non-null     int64
 3   quality   10 non-null     object
dtypes: float64(1), int64(1), object(2)
memory usage: 400.0+ bytes
```

[44]:
```
df3 = df2[['name', 'score']]

df3.head()
```

[44]:
```
         name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
```

[45]:
```
df4 = df2.iloc[[1,3,5,6,], [0,1]]

df4
```

[45]:
```
      name  score
b     Dima    9.0
d    James    NaN
f  Michael   20.0
g  Matthew   14.5
```

[46]:
```
df2.isnull().sum()
```

[46]:
```
name        0
score       2
attempts    0
quality     0
dtype: int64
```

[47]:
```
df2[(df2['score'] > 15) & (df2['score'] <20)]
```

[47]:
```
         name  score  attempts quality
c  Katherine   16.5         2     yes
j      Jonas   19.0         1     yes
```

[48]:
```
df2.loc['d', 'score'] = 18

df2.head()
```

[48]:
```
         name  score  attempts quality
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James   18.0         3      no
e      Emily    9.0         2      no
```

[49]:
```
# score co tan suat xuat hien nhieu nhat trong df2

mark = df2['score'].mode()
print(mark[0])
```

```
score = df2[(df2['score'] == mark[0])]
score

df2['score'].value_counts()
```

```
9.0
```

[49]:
```
9.0     2
12.5    1
16.5    1
18.0    1
20.0    1
14.5    1
8.0     1
19.0    1
Name: score, dtype: int64
```

[50]:
```
# them dong k co du lieu
df2.loc['k'] = ['Suresh', 15.5, 1, 'yes']

df2.loc['l'] = ['Janny', 12.5, 2, 'yes']

df2.tail()
```

[50]:

	name	score	attempts	quality
h	Laura	NaN	1	no
i	Kevin	8.0	2	no
j	Jonas	19.0	1	yes
k	Suresh	15.5	1	yes
l	Janny	12.5	2	yes

[51]:
```
df2 = df2.drop(['l'])
```

[52]:
```
df2 = df2.sort_values(by=  'score')
df2
```

[52]:

	name	score	attempts	quality
i	Kevin	8.0	2	no
b	Dima	9.0	3	no
e	Emily	9.0	2	no
a	Anastasia	12.5	1	yes
g	Matthew	14.5	1	yes
k	Suresh	15.5	1	yes
c	Katherine	16.5	2	yes
d	James	18.0	3	no
j	Jonas	19.0	1	yes
f	Michael	20.0	3	yes

```
h      Laura    NaN          1      no
```

[53]:
```
# them cot result, neu cot >=10 thi co gia tri =1, nguoc lai =0

df2['result'] = df2['score'].map(lambda x: 1 if x >=10 else 0)
df2['result_'] = np.where(df2['score'] >= 10, 1, 0)
df2
```

[53]:

	name	score	attempts	quality	result	result_
i	Kevin	8.0	2	no	0	0
b	Dima	9.0	3	no	0	0
e	Emily	9.0	2	no	0	0
a	Anastasia	12.5	1	yes	1	1
g	Matthew	14.5	1	yes	1	1
k	Suresh	15.5	1	yes	1	1
c	Katherine	16.5	2	yes	1	1
d	James	18.0	3	no	1	1
j	Jonas	19.0	1	yes	1	1
f	Michael	20.0	3	yes	1	1
h	Laura	NaN	1	no	0	0

2.1.3 Ex3

[54]:
```
euro12 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪Chapter4_Data/Euro2012 Data/euro2012.csv', sep = ',', index_col = 0)

print(type(euro12))
print(euro12.shape)
print(euro12.columns)
```

```
<class 'pandas.core.frame.DataFrame'>
(16, 35)
Index(['Team', 'Goals', 'Shots on target', 'Shots off target',
       'Shooting Accuracy', '% Goals-to-shots', 'Total shots (inc. Blocked)',
       'Hit Woodwork', 'Penalty goals', 'Penalties not scored', 'Headed goals',
       'Passes', 'Passes completed', 'Passing Accuracy', 'Touches', 'Crosses',
       'Dribbles', 'Corners Taken', 'Tackles', 'Clearances', 'Interceptions',
       'Clearances off line', 'Clean Sheets', 'Blocks', 'Goals conceded',
       'Saves made', 'Saves-to-shots ratio', 'Fouls Won', 'Fouls Conceded',
       'Offsides', 'Yellow Cards', 'Red Cards', 'Subs on', 'Subs off',
       'Players Used'],
      dtype='object')
```

[55]:
```
euro12[['Team', 'Goals']].sort_values(by = 'Goals', ascending = False).head(5)
```

[55]:

	Team	Goals
13	Spain	12

```
 5    Germany     10
 7      Italy      6
 10  Portugal      6
 3    England      5
```

[56]:
```
euro12.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 16 entries, 0 to 15
Data columns (total 35 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Team                        16 non-null     object
 1   Goals                       16 non-null     int64
 2   Shots on target             16 non-null     int64
 3   Shots off target            16 non-null     int64
 4   Shooting Accuracy           16 non-null     object
 5   % Goals-to-shots            16 non-null     object
 6   Total shots (inc. Blocked)  16 non-null     int64
 7   Hit Woodwork                16 non-null     int64
 8   Penalty goals               16 non-null     int64
 9   Penalties not scored        16 non-null     int64
 10  Headed goals                16 non-null     int64
 11  Passes                      16 non-null     int64
 12  Passes completed            16 non-null     int64
 13  Passing Accuracy            16 non-null     object
 14  Touches                     16 non-null     int64
 15  Crosses                     16 non-null     int64
 16  Dribbles                    16 non-null     int64
 17  Corners Taken               16 non-null     int64
 18  Tackles                     16 non-null     int64
 19  Clearances                  16 non-null     int64
 20  Interceptions               16 non-null     int64
 21  Clearances off line         15 non-null     float64
 22  Clean Sheets                16 non-null     int64
 23  Blocks                      16 non-null     int64
 24  Goals conceded              16 non-null     int64
 25  Saves made                  16 non-null     int64
 26  Saves-to-shots ratio        16 non-null     object
 27  Fouls Won                   16 non-null     int64
 28  Fouls Conceded              16 non-null     int64
 29  Offsides                    16 non-null     int64
 30  Yellow Cards                16 non-null     int64
 31  Red Cards                   16 non-null     int64
 32  Subs on                     16 non-null     int64
 33  Subs off                    16 non-null     int64
 34  Players Used                16 non-null     int64
```

```
dtypes: float64(1), int64(29), object(5)
memory usage: 4.5+ KB
```

[57]:
```
a = euro12[['Team', 'Goals']].iloc[0:3, 1]
print(a)
```

```
0    4
1    4
2    4
Name: Goals, dtype: int64
```

[58]:
```
euro12[euro12.Goals > 6]
```

[58]:

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals	Penalties not scored	…	Saves made	Saves-to-shots ratio	Fouls Won	Fouls Conceded	Offsides	Yellow Cards	Red Cards	Subs on	Subs off	Players Used
5	Germany	10	32	32	47.8%	15.6%	80	2	1	0	…	10	62.6%	63	49	12	4	0	15	15	17
13	Spain	12	42	33	55.9%	16.0%	100	0	1	0	…	15	93.8%	102	83	19	11	0	17	17	18

[2 rows x 35 columns]

[59]:
```
# in cac doi ma ten bat dau bang 'G'

euro12[euro12.Team.str.startswith('G')]
```

[59]:

	Team	Goals	Shots on target	Shots off target	Shooting Accuracy	% Goals-to-shots	Total shots (inc. Blocked)	Hit Woodwork	Penalty goals	Penalties not scored	…	Saves made	Saves-to-shots ratio	Fouls Won
5	Germany	10	32	32	47.8%	15.6%	80	2	1	0	…	10	62.6%	63
6	Greece	5	8	18	30.7%	19.2%	32	1	1					

```
5                    0  …          10                 62.6%         63
6                    1  …          13                 65.1%         67

  Fouls Conceded  Offsides  Yellow Cards  Red Cards  Subs on  Subs off  \
5             49        12             4          0       15        15
6             48        12             9          1       12        12

   Players Used
5            17
6            20

[2 rows x 35 columns]
```

[60]:
```
# in 7 cot dau cua euro12

euro12.iloc[:, 0:7]
```

[60]:
```
                  Team  Goals  Shots on target  Shots off target  \
0              Croatia      4               13                12
1       Czech Republic      4               13                18
2              Denmark      4               10                10
3              England      5               11                18
4               France      3               22                24
5              Germany     10               32                32
6               Greece      5                8                18
7                Italy      6               34                45
8          Netherlands      2               12                36
9               Poland      2               15                23
10            Portugal      6               22                42
11  Republic of Ireland      1                7                12
12              Russia      5                9                31
13               Spain     12               42                33
14              Sweden      5               17                19
15             Ukraine      2                7                26

   Shooting Accuracy % Goals-to-shots  Total shots (inc. Blocked)
0                51.9%            16.0%                          32
1                41.9%            12.9%                          39
2                50.0%            20.0%                          27
3                50.0%            17.2%                          40
4                37.9%             6.5%                          65
5                47.8%            15.6%                          80
6                30.7%            19.2%                          32
7                43.0%             7.5%                         110
8                25.0%             4.1%                          60
9                39.4%             5.2%                          48
10               34.3%             9.3%                          82
```

```
11              36.8%              5.2%                        28
12              22.5%             12.5%                        59
13              55.9%             16.0%                       100
14              47.2%             13.8%                        39
15              21.2%              6.0%                        38
```

[61]:
```
# in tat ca cac cot, tru 3 cot cuoi

euro12.iloc[:, :-3]
```

[61]:
```
                   Team  Goals  Shots on target  Shots off target  \
0               Croatia      4               13                12
1        Czech Republic      4               13                18
2               Denmark      4               10                10
3               England      5               11                18
4                France      3               22                24
5               Germany     10               32                32
6                Greece      5                8                18
7                 Italy      6               34                45
8           Netherlands      2               12                36
9                Poland      2               15                23
10             Portugal      6               22                42
11  Republic of Ireland      1                7                12
12               Russia      5                9                31
13                Spain     12               42                33
14               Sweden      5               17                19
15              Ukraine      2                7                26

   Shooting Accuracy % Goals-to-shots  Total shots (inc. Blocked)  \
0              51.9%            16.0%                          32
1              41.9%            12.9%                          39
2              50.0%            20.0%                          27
3              50.0%            17.2%                          40
4              37.9%             6.5%                          65
5              47.8%            15.6%                          80
6              30.7%            19.2%                          32
7              43.0%             7.5%                         110
8              25.0%             4.1%                          60
9              39.4%             5.2%                          48
10             34.3%             9.3%                          82
11             36.8%             5.2%                          28
12             22.5%            12.5%                          59
13             55.9%            16.0%                         100
14             47.2%            13.8%                          39
15             21.2%             6.0%                          38

    Hit Woodwork  Penalty goals  Penalties not scored  …  Clean Sheets  \
```

0	0	0	0	…	0
1	0	0	0	…	1
2	1	0	0	…	1
3	0	0	0	…	2
4	1	0	0	…	1
5	2	1	0	…	1
6	1	1	1	…	1
7	2	0	0	…	2
8	2	0	0	…	0
9	0	0	0	…	0
10	6	0	0	…	2
11	0	0	0	…	0
12	2	0	0	…	0
13	0	1	0	…	5
14	3	0	0	…	1
15	0	0	0	…	0

	Blocks	Goals conceded	Saves made	Saves-to-shots ratio	Fouls Won	\
0	10	3	13	81.3%	41	
1	10	6	9	60.1%	53	
2	10	5	10	66.7%	25	
3	29	3	22	88.1%	43	
4	7	5	6	54.6%	36	
5	11	6	10	62.6%	63	
6	23	7	13	65.1%	67	
7	18	7	20	74.1%	101	
8	9	5	12	70.6%	35	
9	8	3	6	66.7%	48	
10	11	4	10	71.5%	73	
11	23	9	17	65.4%	43	
12	8	3	10	77.0%	34	
13	8	1	15	93.8%	102	
14	12	5	8	61.6%	35	
15	4	4	13	76.5%	48	

	Fouls Conceded	Offsides	Yellow Cards	Red Cards
0	62	2	9	0
1	73	8	7	0
2	38	8	4	0
3	45	6	5	0
4	51	5	6	0
5	49	12	4	0
6	48	12	9	1
7	89	16	16	0
8	30	3	5	0
9	56	3	7	1
10	90	10	12	0

```
11              51        11             6          1
12              43         4             6          0
13              83        19            11          0
14              51         7             7          0
15              31         4             5          0

[16 rows x 32 columns]
```

[62]:
```
# in cac cot chi hien thi 'Team', 'Shooting Accuracy' từ 'England', 'Italy',␣
↪'Russia'

euro12.loc[euro12.Team.isin(['England', 'Italy', 'Russia']), ['Team', 'Shooting␣
↪Accuracy']]
```

[62]:
```
       Team Shooting Accuracy
3   England             50.0%
7     Italy             43.0%
12   Russia             22.5%
```

[63]:
```
# chuyen kieu Shooting Accuracy sang float

euro12['Shooting Accuracy'].dtype
euro12['Shooting Accuracy'] = euro12['Shooting Accuracy'].str.replace('%', '')
euro12['Shooting Accuracy'] = euro12['Shooting Accuracy'].astype(float)

euro12
```

[63]:
```
                   Team  Goals  Shots on target  Shots off target  \
0               Croatia      4               13                12
1        Czech Republic      4               13                18
2               Denmark      4               10                10
3               England      5               11                18
4                France      3               22                24
5               Germany     10               32                32
6                Greece      5                8                18
7                 Italy      6               34                45
8           Netherlands      2               12                36
9                Poland      2               15                23
10             Portugal      6               22                42
11  Republic of Ireland      1                7                12
12               Russia      5                9                31
13                Spain     12               42                33
14               Sweden      5               17                19
15              Ukraine      2                7                26

    Shooting Accuracy % Goals-to-shots  Total shots (inc. Blocked)  \
0                  51.9           16.0%                          32
```

1	41.9	12.9%	39	
2	50.0	20.0%	27	
3	50.0	17.2%	40	
4	37.9	6.5%	65	
5	47.8	15.6%	80	
6	30.7	19.2%	32	
7	43.0	7.5%	110	
8	25.0	4.1%	60	
9	39.4	5.2%	48	
10	34.3	9.3%	82	
11	36.8	5.2%	28	
12	22.5	12.5%	59	
13	55.9	16.0%	100	
14	47.2	13.8%	39	
15	21.2	6.0%	38	

	Hit Woodwork	Penalty goals	Penalties not scored	…	Saves made	\
0	0	0	0	…	13	
1	0	0	0	…	9	
2	1	0	0	…	10	
3	0	0	0	…	22	
4	1	0	0	…	6	
5	2	1	0	…	10	
6	1	1	1	…	13	
7	2	0	0	…	20	
8	2	0	0	…	12	
9	0	0	0	…	6	
10	6	0	0	…	10	
11	0	0	0	…	17	
12	2	0	0	…	10	
13	0	1	0	…	15	
14	3	0	0	…	8	
15	0	0	0	…	13	

	Saves-to-shots ratio	Fouls Won	Fouls Conceded	Offsides	Yellow Cards	\
0	81.3%	41	62	2	9	
1	60.1%	53	73	8	7	
2	66.7%	25	38	8	4	
3	88.1%	43	45	6	5	
4	54.6%	36	51	5	6	
5	62.6%	63	49	12	4	
6	65.1%	67	48	12	9	
7	74.1%	101	89	16	16	
8	70.6%	35	30	3	5	
9	66.7%	48	56	3	7	
10	71.5%	73	90	10	12	
11	65.4%	43	51	11	6	

```
12                77.0%         34             43          4             6
13                93.8%        102             83         19            11
14                61.6%         35             51          7             7
15                76.5%         48             31          4             5

    Red Cards  Subs on  Subs off  Players Used
0           0        9         9            16
1           0       11        11            19
2           0        7         7            15
3           0       11        11            16
4           0       11        11            19
5           0       15        15            17
6           1       12        12            20
7           0       18        18            19
8           0        7         7            15
9           1        7         7            17
10          0       14        14            16
11          1       10        10            17
12          0        7         7            16
13          0       17        17            18
14          0        9         9            18
15          0        9         9            18

[16 rows x 35 columns]
```

2.2 Tien xu ly

2.2.1 Ex1

[64]:
```
# group by

drinks = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪Chapter5_Data/data/drinks.csv', index_col= 0)
drinks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 193 entries, 0 to 192
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   country                       193 non-null    object
 1   beer_servings                 193 non-null    int64
 2   spirit_servings               193 non-null    int64
 3   wine_servings                 193 non-null    int64
 4   total_litres_of_pure_alcohol  193 non-null    float64
 5   continent                     170 non-null    object
dtypes: float64(1), int64(3), object(2)
memory usage: 10.6+ KB
```

[65]:
```
# dem du lieu NULL

drinks.isnull().sum()
```

[65]:
```
country                         0
beer_servings                   0
spirit_servings                 0
wine_servings                   0
total_litres_of_pure_alcohol    0
continent                      23
dtype: int64
```

[66]:
```
# so luong quoc gia cua moi chau luc

drinks.groupby('continent')['continent'].count()
```

[66]:
```
continent
AF    53
AS    44
EU    45
OC    16
SA    12
Name: continent, dtype: int64
```

[67]:
```
# so luong bia tieu thu trung binh o moi chau luc

drinks.groupby('continent')['beer_servings'].mean()
```

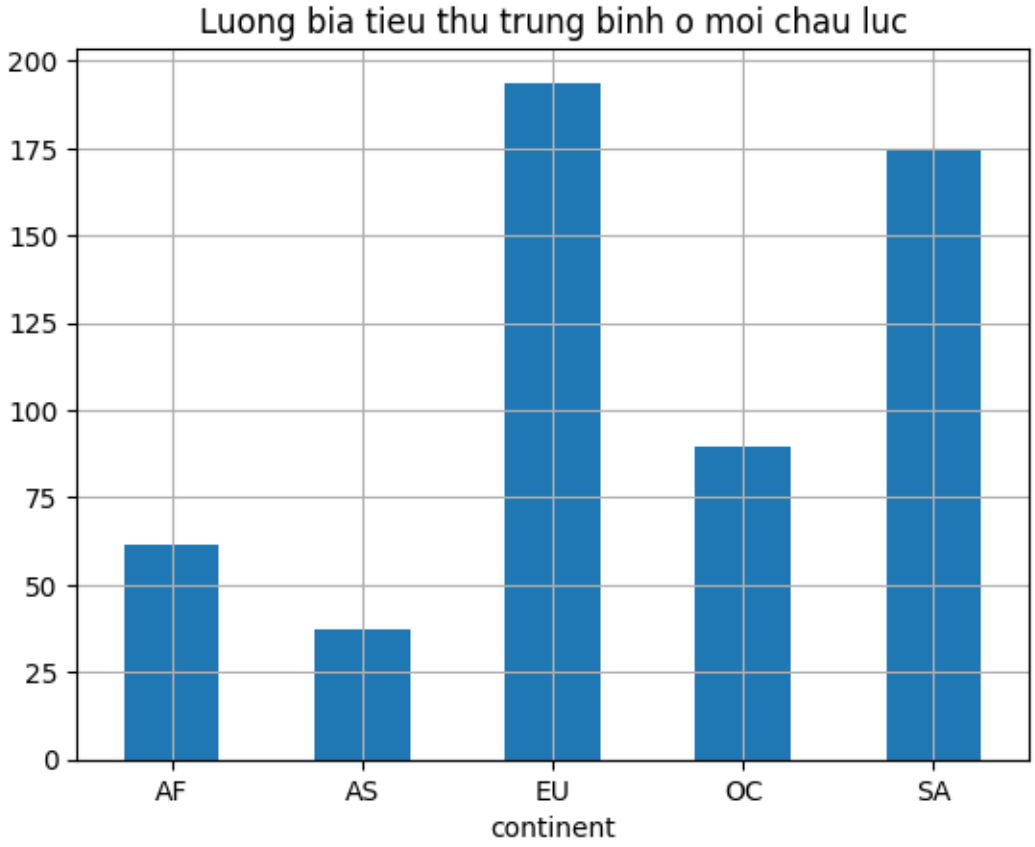
[67]:
```
continent
AF     61.471698
AS     37.045455
EU    193.777778
OC     89.687500
SA    175.083333
Name: beer_servings, dtype: float64
```

[68]:
```
# so luong bia tieu thu trung binh o moi chau luc

drinks.groupby('continent')['beer_servings'].mean().sort_values(ascending=False)
```

[68]:
```
continent
EU    193.777778
SA    175.083333
OC     89.687500
AF     61.471698
AS     37.045455
Name: beer_servings, dtype: float64
```

[69]:
```
# ve bieu do

drinks.groupby('continent').beer_servings.mean().plot.bar(title='Luong bia tieu␣
↪thu trung binh o moi chau luc', rot = 0, grid = True)
```

[69]:
```
<Axes: title={'center': 'Luong bia tieu thu trung binh o moi chau luc'},
xlabel='continent'>
```



[70]:
```
# thong tin thong ke tong quat (describe) so luong ruou vang duoc tieu thu o␣
↪moi chau luc

drinks.groupby('continent')['wine_servings'].describe()
```

[70]:

continent	count	mean	std	min	25%	50%	75%	max
AF	53.0	16.264151	38.846419	0.0	1.0	2.0	13.00	233.0
AS	44.0	9.068182	21.667034	0.0	0.0	1.0	8.00	123.0
EU	45.0	142.222222	97.421738	0.0	59.0	128.0	195.00	370.0
OC	16.0	35.625000	64.555790	0.0	1.0	8.5	23.25	212.0
SA	12.0	62.416667	88.620189	1.0	3.0	12.0	98.50	221.0

[71]:
```
# so luong bia va ruou tieu thu trung binh o moi chau luc

drinks.groupby('continent')['beer_servings', 'wine_servings',␣
 ↪'spirit_servings', 'total_litres_of_pure_alcohol'].mean()
drinks.groupby('continent')['beer_servings', 'wine_servings',␣
 ↪'spirit_servings', 'total_litres_of_pure_alcohol'].mean()
```

```
<ipython-input-71-f62f48e7f1cb>:3: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  drinks.groupby('continent')['beer_servings', 'wine_servings',
'spirit_servings', 'total_litres_of_pure_alcohol'].mean()
<ipython-input-71-f62f48e7f1cb>:4: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  drinks.groupby('continent')['beer_servings', 'wine_servings',
'spirit_servings', 'total_litres_of_pure_alcohol'].mean()
```

[71]:

continent	beer_servings	wine_servings	spirit_servings	total_litres_of_pure_alcohol
AF	61.471698	16.264151	16.339623	3.007547
AS	37.045455	9.068182	60.840909	2.170455
EU	193.777778	142.222222	132.555556	8.617778
OC	89.687500	35.625000	58.437500	3.381250
SA	175.083333	62.416667	114.750000	6.308333

[72]:
```
# gia tri trung vi (median) cua cac loai bia va ruou tieu thu o moi chau luc

drinks.groupby('continent')['beer_servings', 'wine_servings'].median()
```

```
<ipython-input-72-6147c7830a54>:3: FutureWarning: Indexing with multiple keys
(implicitly converted to a tuple of keys) will be deprecated, use a list
instead.
  drinks.groupby('continent')['beer_servings', 'wine_servings'].median()
```

[72]:

continent	beer_servings	wine_servings
AF	32.0	2.0
AS	17.5	1.0

```
EU                219.0          128.0
OC                 52.5            8.5
SA                162.5           12.0
```

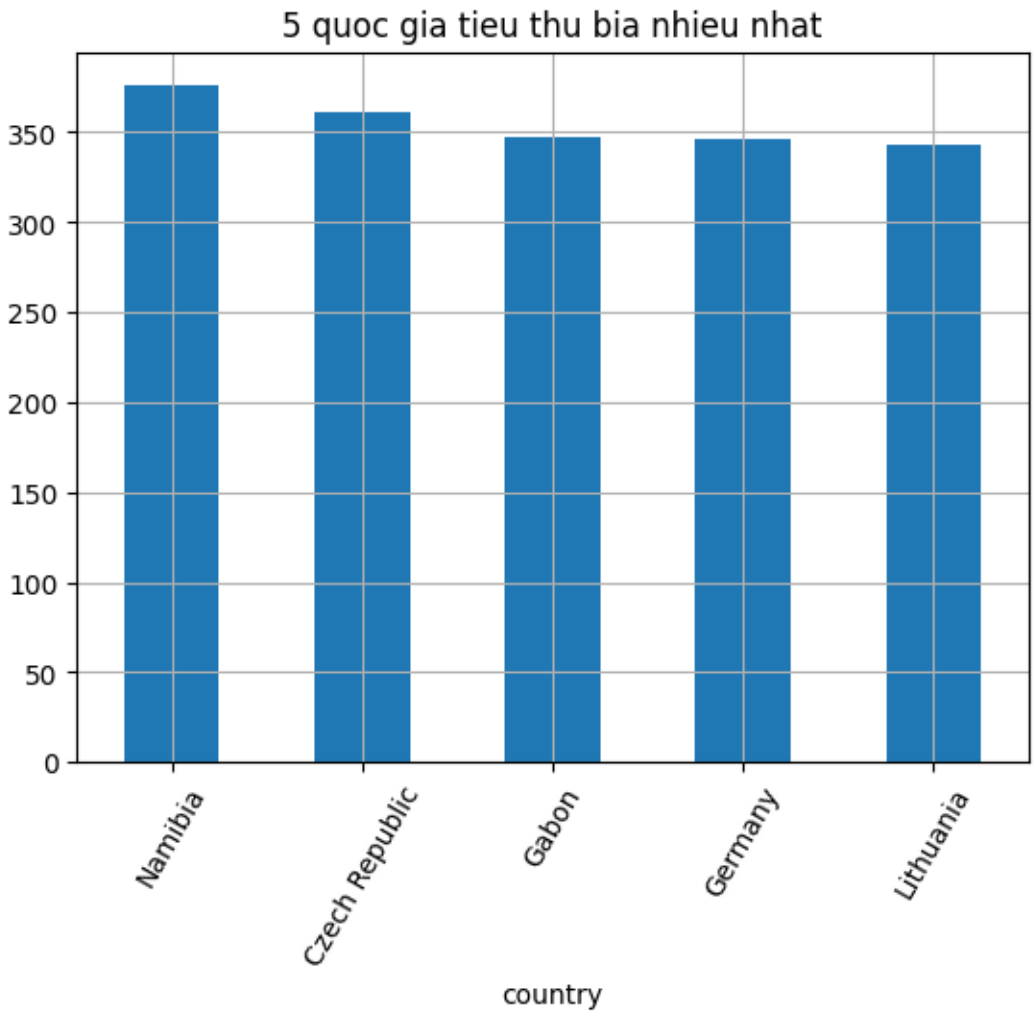
[73]:
```
drinks.groupby('continent').spirit_servings.agg(['mean', 'min', 'max'])
```

[73]:
```
                mean  min  max
continent
AF         16.339623    0  152
AS         60.840909    0  326
EU        132.555556    0  373
OC         58.437500    0  254
SA        114.750000   25  302
```

[74]:
```
# 5 quoc gia co luong tieu thu bia lon nhat

drinks_des = drinks[['country', 'beer_servings']].sort_values('beer_servings',␣
↪ascending=False).head(5)
```

[75]:
```
# ve bieu do

drinks_des.plot.bar(x= 'country', y = 'beer_servings', rot = 60, legend =␣
↪False, title = '5 quoc gia tieu thu bia nhieu nhat', grid = True)
```

[75]:
```
<Axes: title={'center': '5 quoc gia tieu thu bia nhieu nhat'}, xlabel='country'>
```



[76]:
```
# 5 quoc gia co luong bia tieu thu it nhat o chau a

drinks.where((drinks['continent'] == 'AS') & (drinks['beer_servings'] !=␣
↪0))[['country', 'beer_servings']].sort_values('beer_servings').head()
```

[76]:

	country	beer_servings
171	Timor-Leste	1.0
137	Qatar	1.0
168	Tajikistan	2.0
116	Myanmar	5.0
167	Syria	5.0

2.2.2 Ex2

[77]:
```
stocks1 = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪Chapter5_Data/stock_trading_data/stocks1.csv')
stocks2 = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪Chapter5_Data/stock_trading_data/stocks2.csv')

stocks1.head()
stocks2.head()
```

[77]:

	date	symbol	open	high	low	close	volume
0	01-03-19	FB	162.60	163.132	161.69	162.28	11097770
1	04-03-19	FB	163.90	167.500	163.83	167.37	18894689
2	05-03-19	FB	167.37	171.880	166.55	171.26	28187890
3	06-03-19	FB	172.90	173.570	171.27	172.51	21531723
4	07-03-19	FB	171.50	171.740	167.61	169.13	18306504

[78]:
```
# cho biet luong stocks1 co du lieu Null

stocks1.isnull().any()
stocks1.isnull().sum()
```

[78]:
```
date      0
symbol    0
open      0
high      2
low       2
close     0
volume    0
dtype: int64
```

[79]:
```
stocks1[stocks1['high'].isnull()]
```

[79]:

	date	symbol	open	high	low	close	volume
3	06-03-19	AMZN	1695.97	NaN	NaN	1668.95	3996001
12	05-03-19	GOOG	1150.06	NaN	NaN	1162.03	1443174

[80]:
```
# thay the nhung gia tri null

stocks1['high'].fillna(stocks1.groupby(['symbol'])['high'].transform(max),␣
 ↪inplace =True)
stocks1['low'].fillna(stocks1.groupby(['symbol'])['low'].transform(max),␣
 ↪inplace =True)
```

[81]:
```
stocks1.iloc[[3,12]]
```

[81]:
```
        date symbol     open     high      low    close   volume
3   06-03-19   AMZN  1695.97  1709.43  1689.01  1668.95  3996001
12  05-03-19   GOOG  1150.06  1167.57  1155.49  1162.03  1443174
```

[82]:
```
# gop stocks1 va stocks2

stocks = pd.concat([stocks1, stocks2], ignore_index=True)
stocks.tail(15)
```

[82]:
```
        date symbol     open       high      low    close    volume
10  01-03-19   GOOG  1124.90  1142.9700  1124.75  1140.99   1450316
11  04-03-19   GOOG  1146.99  1158.2800  1130.69  1147.80   1446047
12  05-03-19   GOOG  1150.06  1167.5700  1155.49  1162.03   1443174
13  06-03-19   GOOG  1162.49  1167.5700  1155.49  1157.86   1099289
14  07-03-19   GOOG  1155.72  1156.7600  1134.91  1143.30   1166559
15  01-03-19     FB   162.60   163.1320   161.69   162.28  11097770
16  04-03-19     FB   163.90   167.5000   163.83   167.37  18894689
17  05-03-19     FB   167.37   171.8800   166.55   171.26  28187890
18  06-03-19     FB   172.90   173.5700   171.27   172.51  21531723
19  07-03-19     FB   171.50   171.7400   167.61   169.13  18306504
20  01-03-19   TSLA   306.94   307.1300   291.90   294.79  22911375
21  04-03-19   TSLA   298.12   299.0000   282.78   285.36  17096818
22  05-03-19   TSLA   282.00   284.0000   270.10   276.54  18764740
23  06-03-19   TSLA   276.48   281.5058   274.39   276.24  10335485
24  07-03-19   TSLA   278.84   284.7000   274.25   276.59   9442483
```

[83]:
```
# gop stocks va companies

companies = stocks1 = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/
 ↪Numpy_Pandas/Chapter5_Data/stock_trading_data/companies.csv')

stocks_companies = stocks.merge(companies, left_on = 'symbol', right_on =␣
 ↪'name', how = 'inner')

companies
```

[83]:
```
   name  employees headquarters_city headquarters_state
0  AMZN     613300           Seattle                 WA
1  GOOG      98771     Mountain View                 CA
2  AAPL     132000         Cupertino                 CA
3    FB      48268        Menlo Park                 CA
4  TSLA      48016         Palo Alto                 CA
```

[84]:
```
stocks_companies.head()
```

[84]:
```
        date symbol     open     high      low    close   volume  name  \
0  01-03-19   AMZN  1655.13  1674.26  1651.00  1671.73  4974877  AMZN
```

```
1  04-03-19   AMZN  1685.00  1709.43  1674.36  1696.17  6167358  AMZN
2  05-03-19   AMZN  1702.95  1707.80  1689.01  1692.43  3681522  AMZN
3  06-03-19   AMZN  1695.97  1709.43  1689.01  1668.95  3996001  AMZN
4  07-03-19   AMZN  1667.37  1669.75  1620.51  1625.95  4957017  AMZN

   employees headquarters_city headquarters_state
0     613300           Seattle                 WA
1     613300           Seattle                 WA
2     613300           Seattle                 WA
3     613300           Seattle                 WA
4     613300           Seattle                 WA
```

[85]:
```
# cho biet gia (open, high, low, close) trung binh va volume trung binh cua moi cong ty


cols = ['symbol', 'open', 'high', 'low', 'close', 'volume']

stocks_companies[cols].groupby('symbol').mean()
```

[85]:

symbol	open	high	low	close	volume
AAPL	174.890	175.76600	173.472	174.674	23733309.4
AMZN	1681.284	1694.13400	1664.778	1671.046	4755355.0
FB	167.654	169.56440	166.190	168.510	19603715.2
GOOG	1148.032	1158.63000	1140.266	1150.396	1321077.0
TSLA	288.476	291.26716	278.684	281.904	15710180.2

[86]:
```
# cho biet gia dong cua trung binh, lon nhat, nho  nhat o moi cong ty

stocks_companies.groupby('symbol').close.agg(['mean', 'min', 'max'])
```

[86]:

symbol	mean	min	max
AAPL	174.674	172.50	175.85
AMZN	1671.046	1625.95	1696.17
FB	168.510	162.28	172.51
GOOG	1150.396	1140.99	1162.03
TSLA	281.904	276.24	294.79

[87]:
```
# tao cot parsed_time trong stocks_companies bnag cach doi thoi gian sang dinh dang DateTime

stocks_companies['parsed_time'] = pd.to_datetime(stocks_companies['date'])

print(stocks_companies['parsed_time'].dtype)
stocks_companies.head()
```

datetime64[ns]

[87]:
```
        date symbol     open     high      low    close   volume  name  \
0  01-03-19   AMZN  1655.13  1674.26  1651.00  1671.73  4974877  AMZN
1  04-03-19   AMZN  1685.00  1709.43  1674.36  1696.17  6167358  AMZN
2  05-03-19   AMZN  1702.95  1707.80  1689.01  1692.43  3681522  AMZN
3  06-03-19   AMZN  1695.97  1709.43  1689.01  1668.95  3996001  AMZN
4  07-03-19   AMZN  1667.37  1669.75  1620.51  1625.95  4957017  AMZN

   employees headquarters_city headquarters_state parsed_time
0     613300           Seattle                 WA  2019-01-03
1     613300           Seattle                 WA  2019-04-03
2     613300           Seattle                 WA  2019-05-03
3     613300           Seattle                 WA  2019-06-03
4     613300           Seattle                 WA  2019-07-03
```

[88]:
```
# them cot result, neu gia 'close' > 'open' thi result cos gia tri 'up', nguoc␣
↪lai la 'down'

stocks_companies.loc[stocks_companies['close'] > stocks_companies['open'],␣
↪'result' ] = 'up'
stocks_companies.loc[stocks_companies['close'] < stocks_companies['open'],␣
↪'result' ] = 'down'

stocks_companies.head()
```

[88]:
```
        date symbol     open     high      low    close   volume  name  \
0  01-03-19   AMZN  1655.13  1674.26  1651.00  1671.73  4974877  AMZN
1  04-03-19   AMZN  1685.00  1709.43  1674.36  1696.17  6167358  AMZN
2  05-03-19   AMZN  1702.95  1707.80  1689.01  1692.43  3681522  AMZN
3  06-03-19   AMZN  1695.97  1709.43  1689.01  1668.95  3996001  AMZN
4  07-03-19   AMZN  1667.37  1669.75  1620.51  1625.95  4957017  AMZN

   employees headquarters_city headquarters_state parsed_time result
0     613300           Seattle                 WA  2019-01-03     up
1     613300           Seattle                 WA  2019-04-03     up
2     613300           Seattle                 WA  2019-05-03   down
3     613300           Seattle                 WA  2019-06-03   down
4     613300           Seattle                 WA  2019-07-03   down
```

2.2.3 Ex3

[89]:
```
movies = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
↪Chapter5_Data/movies_data/movies.csv', sep =',')
tags = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
↪Chapter5_Data/movies_data/tags.csv', sep =',')
```

```
ratings = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
 ↪Chapter5_Data/movies_data/ratings.csv', sep =',')

movies.head()
```

[89]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                        Comedy|Drama|Romance
4                                      Comedy
```

[90]:
```
# kiem ra du lieu null

movies.isnull().sum()
```

[90]:
```
movieId    0
title      0
genres     1
dtype: int64
```

[91]:
```
# loai bo dong co du lieu null

movies.dropna(subset=['genres'], axis=0, inplace=True)
movies.isnull().any()
```

[91]:
```
movieId    False
title      False
genres     False
dtype: bool
```

[92]:
```
tags.dropna(subset=['timestamp'], axis=0, inplace=True)
tags.isnull().any()
```

[92]:
```
userId       False
movieId      False
tag          False
timestamp    False
dtype: bool
```

[93]:
```
# kiem tra xem co du lieu rating nao khong hop le hay khong (> 5 hoac < 0), neu co thay the bang gia tri xuat hien nhieu nhat

filter_rating = np.logical_or(ratings['rating'] > 5, ratings['rating'] < 0)

filter_rating.any()
```

[93]:
```
True
```

[94]:
```
ratings[filter_rating]
```

[94]:
```
    userId  movieId  rating  timestamp
56       2      350     6.0  835355697
```

[95]:
```
ratings.loc[filter_rating, 'rating'] = ratings['rating'].mode()[0]

ratings[filter_rating]
```

[95]:
```
    userId  movieId  rating  timestamp
56       2      350     4.0  835355697
```

[96]:
```
# gop df

movies_tags = movies.merge(tags, on = 'movieId', how = 'inner')

movies_tags.head()
```

[96]:
```
   movieId                              title  \
0        1                   Toy Story (1995)
1        5  Father of the Bride Part II (1995)
2       47         Seven (a.k.a. Se7en) (1995)
3       47         Seven (a.k.a. Se7en) (1995)
4       47         Seven (a.k.a. Se7en) (1995)

                                       genres  userId           tag  \
0  Adventure|Animation|Children|Comedy|Fantasy     501         Pixar
1                                      Comedy     431  steve martin
2                           Mystery|Thriller     364      biblical
3                           Mystery|Thriller     364         crime
4                           Mystery|Thriller     364          dark

      timestamp
0  1.292956e+09
1  1.140455e+09
2  1.444535e+09
3  1.444535e+09
4  1.444535e+09
```

[97]:
```
movies_ratings = movies.merge(ratings, on = 'movieId', how = 'inner')

movies_ratings.head()
```

[97]:
```
   movieId             title                                       genres  \
0        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
1        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
2        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
3        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
4        1  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy

   userId  rating   timestamp
0       7     3.0   851866703
1       9     4.0   938629179
2      13     5.0  1331380058
3      15     2.0   997938310
4      19     3.0   855190091
```

[98]:
```
# loc du lieu theo yeu cau

tag_counts = tags['tag'].value_counts().to_frame()

tag_counts.head()
```

[98]:
```
            tag
getdvd       33
Ei muista    29
toplist07    26
tivo         26
toplist11    20
```

[99]:
```
# tao is_highly_rated co rating > 4.0 cua df ratings

is_hightly_rated = ratings['rating'] >= 4

is_hightly_rated.head()
```

[99]:
```
0    False
1    False
2    False
3    False
4     True
Name: rating, dtype: bool
```

[100]:
```
ratings[is_hightly_rated][['movieId', 'rating']].head()
```

[100]:
```
     movieId  rating
4       1172     4.0
12      1953     4.0
13      2105     4.0
20        10     4.0
21        17     5.0
```

[101]:
```
# tao is_animation theo dk genres cua movies co chu chuoi 'Animation'

is_animation = movies['genres'].str.contains('Animation')

is_animation.head()
```

[101]:
```
0     True
1    False
2    False
3    False
4    False
Name: genres, dtype: bool
```

[102]:
```
movies[is_animation].head(10)
```

[102]:
```
     movieId                                       title  \
0          1                            Toy Story (1995)
12        13                                Balto (1995)
46        48                            Pocahontas (1995)
211      239                       Goofy Movie, A (1995)
216      244                     Gumby: The Movie (1995)
279      313                   Swan Princess, The (1994)
328      364                       Lion King, The (1994)
354      392  Secret Adventures of Tom Thumb, The (1993)
494      551      Nightmare Before Christmas, The (1993)
500      558                       Pagemaster, The (1994)

                                              genres
0        Adventure|Animation|Children|Comedy|Fantasy
12                     Adventure|Animation|Children
46          Animation|Children|Drama|Musical|Romance
211               Animation|Children|Comedy|Romance
216                             Animation|Children
279                             Animation|Children
328  Adventure|Animation|Children|Drama|Musical|IMAX
354                           Adventure|Animation
494              Animation|Children|Fantasy|Musical
500      Action|Adventure|Animation|Children|Fantasy
```

[103]:
```
# tach cot tu genres bang |

movie_genres = movies['genres'].str.split('|', expand= True)

movie_genres.tail()
```

[103]:

	0	1	2	3	4	5	6	7	8	9
9120	Adventure	Drama	Romance	None	None	None	None	None	None	None
9121	Action	Adventure	Fantasy	Sci-Fi	None	None	None	None	None	None
9122	Documentary	None	None	None	None	None	None	None	None	None
9123	Comedy	None	None	None	None	None	None	None	None	None
9124	Documentary	None	None	None	None	None	None	None	None	None

[104]:
```
# them cot moi

movie_genres['isComedy'] = movies['genres'].str.contains('Comedy')

movie_genres.head()
```

[104]:

	0	1	2	3	4	5	6	7	8	9	isComedy
0	Adventure	Animation	Children	Comedy	Fantasy	None	None	None	None	None	True
1	Adventure	Children	Fantasy	None	None	None	None	None	None	None	False
2	Comedy	Romance	None	None	None	None	None	None	None	None	True
3	Comedy	Drama	Romance	None	None	None	None	None	None	None	True
4	Comedy	None	None	None	None	None	None	None	None	None	True

[105]:
```
# them cot tu year bang regex

movies['year'] = movies['title'].str.extract('.*\((.*)\).*', expand = True)

movies.head()
```

[105]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres  year
0  Adventure|Animation|Children|Comedy|Fantasy  1995
1                   Adventure|Children|Fantasy  1995
2                              Comedy|Romance  1995
3                        Comedy|Drama|Romance  1995
4                                      Comedy  1995
```

[106]:
```
# thong ke du lieu
ratings.rating.describe()
```

[106]:
```
count    100004.000000
mean          3.543608
std           1.058064
min           0.500000
25%           3.000000
50%           4.000000
75%           4.000000
max           5.000000
Name: rating, dtype: float64
```

[107]:
```
ratings['rating'].value_counts()
```

[107]:
```
4.0    28750
3.0    20064
5.0    15095
3.5    10538
4.5     7723
2.0     7271
2.5     4449
1.0     3326
1.5     1687
0.5     1101
Name: rating, dtype: int64
```

[108]:
```
# thong ke dem so luong phim theo rating

count_of_films = movies_ratings.groupby(['rating'], as_index = False).movieId.
 ↪count()
count_of_films.rename(columns = {'movieId' : 'Count of films'}, inplace = True)
```

```
count_of_films
```

[108]:

	rating	Count of films
0	0.5	1101
1	1.0	3326
2	1.5	1687
3	2.0	7268
4	2.5	4449
5	3.0	20058
6	3.5	10535
7	4.0	28743
8	4.5	7723
9	5.0	15094

[109]:
```
# dem so luong rating theo phim va luu vao bien 'movie_count'

lst = ['movieId', 'title', 'rating', 'genres']

movie_count = movies_ratings[lst].groupby(['movieId', 'title', 'genres'],␣
 ↪as_index = False).rating.count()
movie_count.rename(columns = {'rating': 'Total ratings'}, inplace = True)

movie_count.head()
```

[109]:

	movieId	title	genres	Total ratings
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	247
1	2	Jumanji (1995)	Adventure|Children|Fantasy	107
2	3	Grumpier Old Men (1995)	Comedy|Romance	59
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	13
4	5	Father of the Bride Part II (1995)	Comedy	56

[110]:
```
# tinh rating trung binh theo moi phim va luu vao bien avg_ratings

avg_ratings = movies_ratings[lst].groupby(['movieId', 'title', 'genres'],␣
 ↪as_index=False).mean()

avg_ratings.rename(columns={'rating' : 'Average ratings'}, inplace=True)
avg_ratings.head()
```

[110]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
1        2                      Jumanji (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)

                                        genres  Average ratings
0  Adventure|Animation|Children|Comedy|Fantasy         3.872470
1                   Adventure|Children|Fantasy         3.401869
2                              Comedy|Romance         3.161017
3                        Comedy|Drama|Romance         2.384615
4                                       Comedy         3.267857
```

[111]:
```
# hien thi rating trung binh cua cac phim 'Comedy'

is_comedy = avg_ratings['genres'].str.contains('Comedy')
avg_ratings[is_comedy].head()
```

[111]:
```
   movieId                               title  \
0        1                    Toy Story (1995)
2        3             Grumpier Old Men (1995)
3        4            Waiting to Exhale (1995)
4        5  Father of the Bride Part II (1995)
6        7                      Sabrina (1995)

                                        genres  Average ratings
0  Adventure|Animation|Children|Comedy|Fantasy         3.872470
2                              Comedy|Romance         3.161017
3                        Comedy|Drama|Romance         2.384615
4                                       Comedy         3.267857
6                              Comedy|Romance         3.283019
```

[114]:
```
# hien thi rating trung binh cua cac phim la 'Comdy' va co rating >= 4

rating4 = avg_ratings['Average ratings'] >= 4.0
avg_ratings[rating4 & is_comedy][-5:]
```

[114]:
```
      movieId                                  title  \
9018   152081                  Zootopia (2016)
9022   153584      The Last Days of Emma Blank (2009)
9026   156025   Ice Age: The Great Egg-Scapade (2016)
9036   158314  Daniel Tosh: Completely Serious (2007)
9051   160567   Mike & Dave Need Wedding Dates (2016)

                                       genres  Average ratings
9018  Action|Adventure|Animation|Children|Comedy              4.0
```

```
9022                                        Comedy              5.0
9026         Adventure|Animation|Children|Comedy              5.0
9036                                        Comedy              4.5
9051                                        Comedy              4.0
```

[115]:
```
print(movies.groupby('year').size())
```

```
year
1902      1
1915      1
1916      2
1917      1
1918      1
       ...
2012    222
2013    224
2014    221
2015    182
2016     65
Length: 105, dtype: int64
```

[116]:
```
# tinh trung binh rating theo year va luu vao yearly_average
joined = movies.merge(ratings, how = 'inner')
yearly_average = joined[['year', 'rating']].groupby('year', as_index=False).
  ↪mean()
print(yearly_average.shape)
yearly_average.head()
```

```
(105, 2)
```

[116]:

	year	rating
0	1902	4.333333
1	1915	3.000000
2	1916	3.500000
3	1917	4.250000
4	1918	4.250000

[117]:
```
joined.head()
```

[117]:

	movieId	title	genres	year	userId	rating	timestamp
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy				
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy				
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy				
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy				
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy				

```
0  1995       7     3.0   851866703
1  1995       9     4.0   938629179
2  1995      13     5.0  1331380058
3  1995      15     2.0   997938310
4  1995      19     3.0   855190091
```

[118]:
```
# sap xep tang dan

yearly_average_asc = yearly_average.sort_values(by = 'year', ascending = True)
yearly_average_asc.head()
```

[118]:
```
   year    rating
0  1902  4.333333
1  1915  3.000000
2  1916  3.500000
3  1917  4.250000
4  1918  4.250000
```

[119]:
```
# parsing timestamps

tags['parsed_time'] = pd.to_datetime(tags['timestamp'], unit = 's')

print(tags['parsed_time'].dtype)
tags.head()
```

```
datetime64[ns]
```

[119]:
```
   userId  movieId                     tag     timestamp         parsed_time
0      15      339  sandra 'boring' bullock  1.138538e+09 2006-01-29 12:29:30
1      15     1955                 dentist  1.193435e+09 2007-10-26 21:44:21
2      15     7478                Cambodia  1.170561e+09 2007-02-04 03:49:57
3      15    32892                 Russian  1.170626e+09 2007-02-04 21:59:26
4      15    34162             forgettable  1.141392e+09 2006-03-03 13:16:05
```

[120]:
```
# tao selected_rows chua cac dong co tag['parsed_time'] > '2015-02-01'

t = tags['parsed_time'] > '2015-02-01'
selected_rows = tags[t]
selected_rows.head()
```

[120]:
```
    userId  movieId          tag     timestamp         parsed_time
8       15   100365     activist  1.425876e+09 2015-03-09 04:43:40
9       15   100365  documentary  1.425876e+09 2015-03-09 04:43:40
10      15   100365       uganda  1.425876e+09 2015-03-09 04:43:40
15      73   107999       action  1.430799e+09 2015-05-05 04:13:04
16      73   107999        anime  1.430799e+09 2015-05-05 04:13:04
```

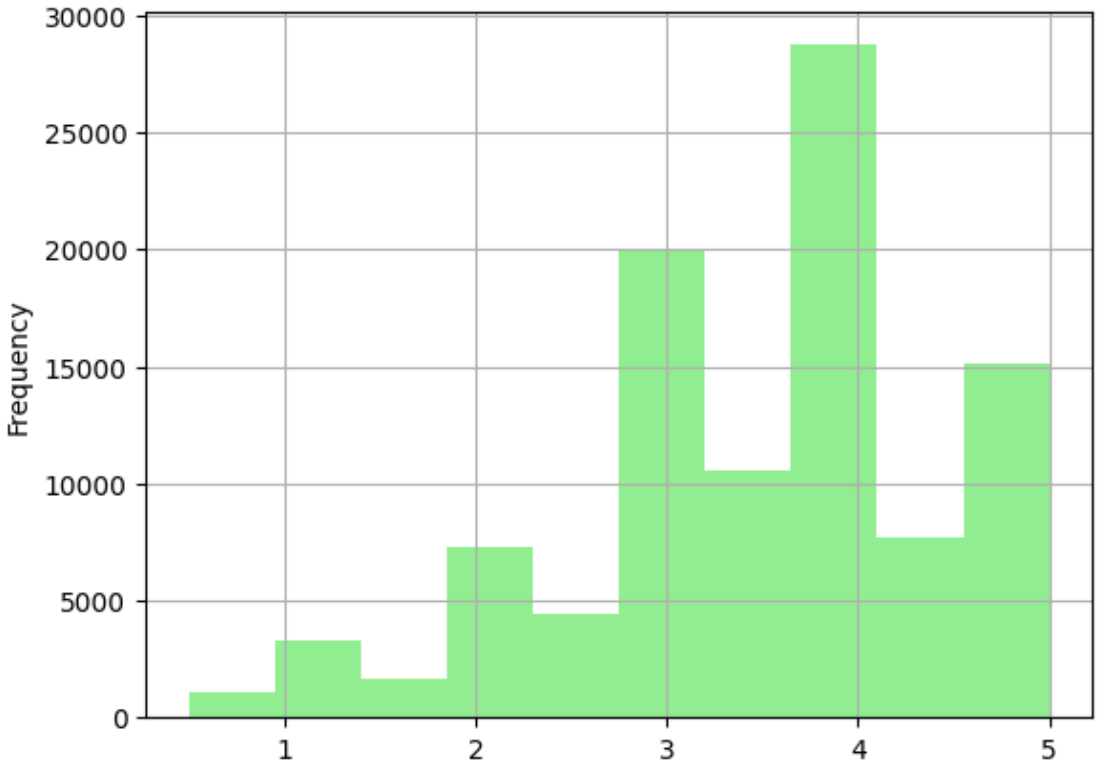
[121]:
```
# sap xep du lieu tags tang dan theo parsed_time

tags.sort_values(by = 'parsed_time', ascending= True)[:10]
```

[121]:

	userId	movieId	tag	timestamp	parsed_time
338	353	35836	dumb	1.137217e+09	2006-01-14 05:44:00
0	15	339	sandra 'boring' bullock	1.138538e+09	2006-01-29 12:29:30
232	294	36401	fairy tales	1.138983e+09	2006-02-03 16:11:04
229	294	6754	vampire	1.138983e+09	2006-02-03 16:17:49
333	353	4721	As historicaly correct as Germany winning WW2	1.140389e+09	2006-02-19 22:44:16
334	353	4721	but still a fun movie.	1.140389e+09	2006-02-19 22:44:16
335	353	7376	The Rocks "finest" work need I say more?	1.140390e+09	2006-02-19 22:51:51
336	353	31221	Try not to mistake this for an episode of Alias	1.140390e+09	2006-02-19 22:53:15
230	294	8865	1940's feel	1.140396e+09	2006-02-20 00:38:50
231	294	8865	unique look	1.140396e+09	2006-02-20 00:38:50

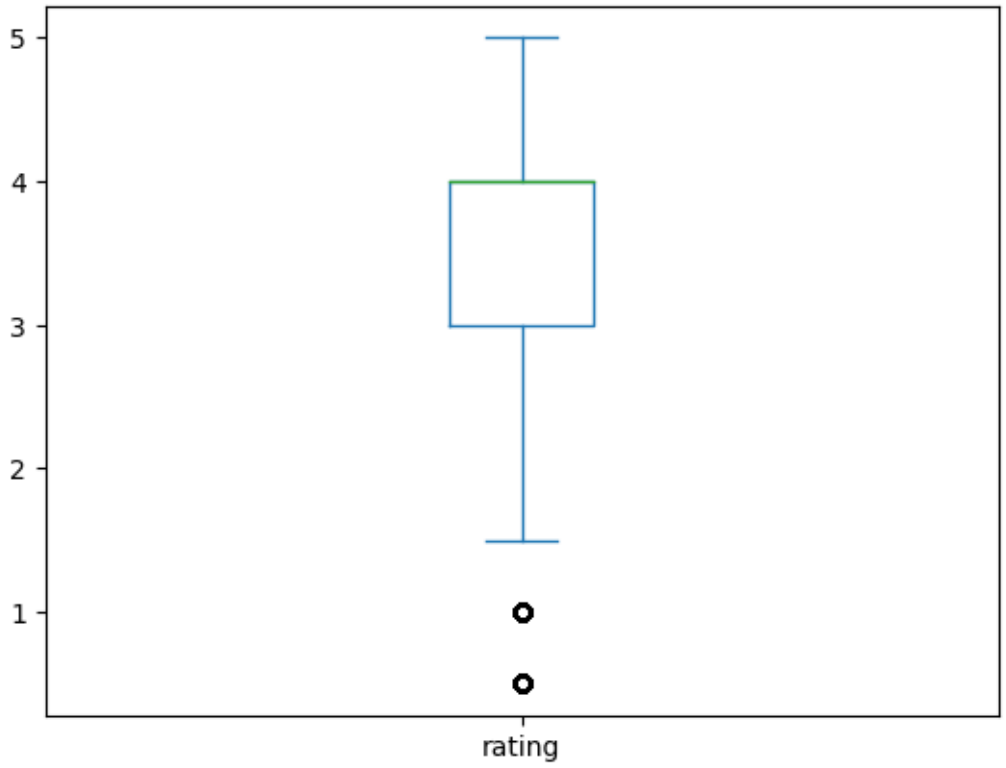
[122]:
```
# truc quan hoa

ratings['rating'].plot.hist(grid = True, color = 'lightgreen')
```

[122]:
```
<Axes: ylabel='Frequency'>
```



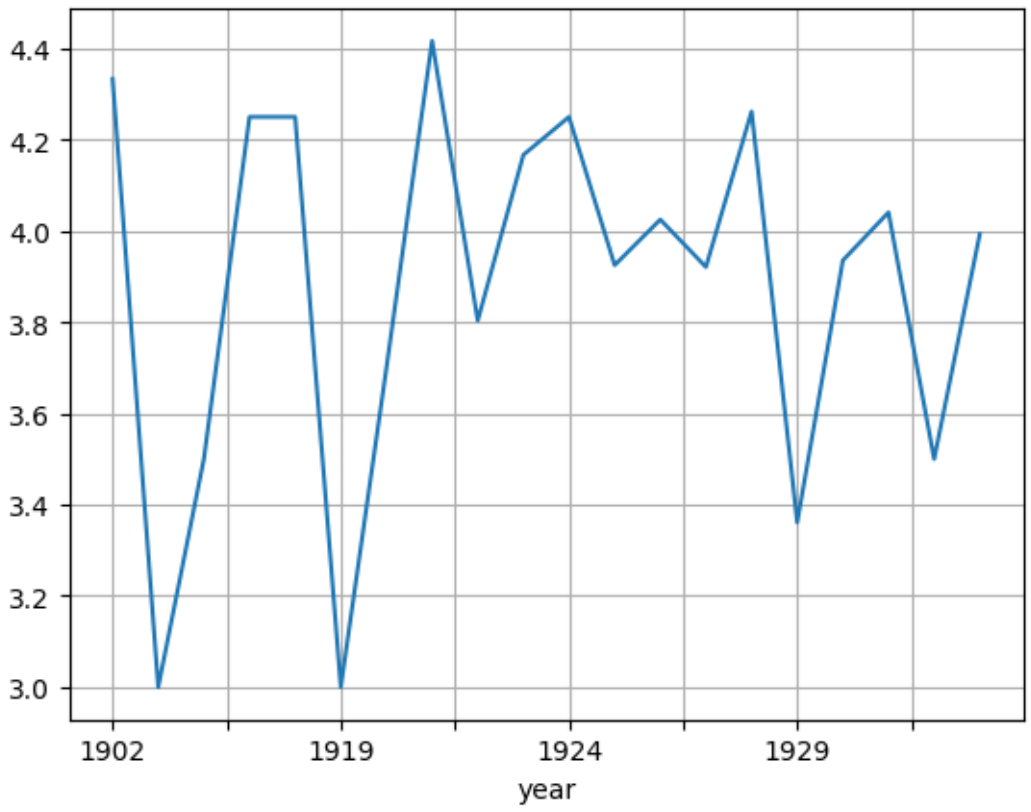
[123]:
```
ratings['rating'].plot.box()
```

[123]: <Axes: >



[125]:
```
yearly_average_asc.head(20).plot(x= 'year', y = 'rating', legend = None, grid =␣
↪True)
```

[125]: <Axes: xlabel='year'>



2.2.4 Ex4

[126]:
```
df = pd.read_csv(r'/content/drive/MyDrive/Colab Notebooks/Numpy_Pandas/
  ↪Chapter5_Data/data/chipotle.tsv', sep = '\t')

print(df.shape)
print(df.columns)
```

```
(4622, 5)
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

[127]:
```
df.head()
```

[127]:

	order_id	quantity	item_name
0	1	1	Chips and Fresh Tomato Salsa
1	1	1	Izze
2	1	1	Nantucket Nectar
3	1	1	Chips and Tomatillo-Green Chili Salsa
4	2	2	Chicken Bowl

```
                                choice_description item_price
0                                              NaN     $2.39
1                                    [Clementine]     $3.39
2                                         [Apple]     $3.39
3                                              NaN     $2.39
4  [Tomatillo-Red Chili Salsa (Hot), [Black Beans…    $16.98
```

[128]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4622 entries, 0 to 4621
Data columns (total 5 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   order_id            4622 non-null   int64 
 1   quantity            4622 non-null   int64 
 2   item_name           4622 non-null   object
 3   choice_description  3376 non-null   object
 4   item_price          4622 non-null   object
dtypes: int64(2), object(3)
memory usage: 180.7+ KB
```

[129]:
```
# xu ly null va duplicate
# so luong null

df.isnull().sum()
```

[129]:
```
order_id                0
quantity                0
item_name               0
choice_description    1246
item_price               0
dtype: int64
```

[130]:
```
# so luong duplicate

df.duplicated().sum()
```

[130]:
```
59
```

[131]:
```
# xoa du lieu trung, giu lai du lieu dau tien

df.drop_duplicates(keep = 'first', inplace = True)
df.shape
```

[131]:
```
(4563, 5)
```

[132]:
```
# xu ly du lieu

df['item_price'] = df['item_price'].str.replace('$', '').astype(float)
df['item_price'].dtype
```

```
<ipython-input-132-dab456b5d983>:3: FutureWarning: The default value of regex
will change from True to False in a future version. In addition, single
character regular expressions will *not* be treated as literal strings when
regex=True.
  df['item_price'] = df['item_price'].str.replace('$', '').astype(float)
```

[132]:
```
dtype('float64')
```

[133]:
```
df.head()
```

[133]:

	order_id	quantity	item_name	choice_description	item_price
0	1	1	Chips and Fresh Tomato Salsa	NaN	2.39
1	1	1	Izze	[Clementine]	3.39
2	1	1	Nantucket Nectar	[Apple]	3.39
3	1	1	Chips and Tomatillo-Green Chili Salsa	NaN	2.39
4	2	2	Chicken Bowl	[Tomatillo-Red Chili Salsa (Hot), [Black Beans…	16.98

[134]:
```
# co bao nhieu du lieu co item_price=0

df[df['item_price'] == 0].size
```

[134]:
```
0
```

[135]:
```
# tao df chi gom cac cot: order_id, item_name, quantity, item_price

df = df[['order_id', 'item_name', 'quantity', 'item_price']]
df.head()
```

[135]:

	order_id	item_name	quantity	item_price
0	1	Chips and Fresh Tomato Salsa	1	2.39
1	1	Izze	1	3.39
2	1	Nantucket Nectar	1	3.39
3	1	Chips and Tomatillo-Green Chili Salsa	1	2.39
4	2	Chicken Bowl	2	16.98

[136]:
```
# tao them cot revenue, voi revenue = quantityy * item_price

df['revenue'] = df['quantity'] * df['item_price']
df.head()
```

[136]:
```
   order_id                              item_name  quantity  item_price  \
0         1           Chips and Fresh Tomato Salsa         1        2.39
1         1                                   Izze         1        3.39
2         1                       Nantucket Nectar         1        3.39
3         1  Chips and Tomatillo-Green Chili Salsa         1        2.39
4         2                           Chicken Bowl         2       16.98

   revenue
0     2.39
1     3.39
2     3.39
3     2.39
4    33.96
```

[137]:
```
# thong ke
# thong ke dem so mon an torng moi don dat hang, sap giam dan theo dem

df['order_id'].value_counts().head()
```

[137]:
```
926     21
1483    14
1786    11
759     11
691     11
Name: order_id, dtype: int64
```

[138]:
```
# thong ke chung cua 3 cot

df[['quantity', 'item_price', 'revenue']].describe()
```

[138]:
```
          quantity   item_price      revenue
count  4563.000000  4563.000000  4563.000000
mean      1.076704     7.490083     8.528185
std       0.412739     4.244155    12.701196
min       1.000000     1.090000     1.090000
25%       1.000000     3.750000     3.990000
50%       1.000000     8.750000     8.750000
75%       1.000000     9.250000    10.980000
max      15.000000    44.250000   663.750000
```

[139]:
```
# thong ke chung cot revenue theo nhoom item_name
```

```
df.groupby('item_name')['revenue'].describe().head()
```

[139]:
```
                        count       mean       std   min   25%    50%    75%  \
item_name
6 Pack Soft Drink        54.0   6.850556  2.649531  6.49  6.49   6.49   6.49
Barbacoa Bowl            65.0  10.201692  1.265312  8.69  9.25   9.25  11.75
Barbacoa Burrito         90.0   9.838889  1.144220  8.69  9.25   9.25  11.38
Barbacoa Crispy Tacos    11.0  12.610000  8.183734  8.99  9.25   9.25  11.75
Barbacoa Salad Bowl       9.0  10.778889  1.317616  9.39  9.39  11.89  11.89

                          max
item_name
6 Pack Soft Drink       25.96
Barbacoa Bowl           11.75
Barbacoa Burrito        11.75
Barbacoa Crispy Tacos   37.00
Barbacoa Salad Bowl     11.89
```

[141]:
```
# cho biet tong thanh tien cua moi hoa don

df.groupby('order_id')['revenue'].sum().head()
```

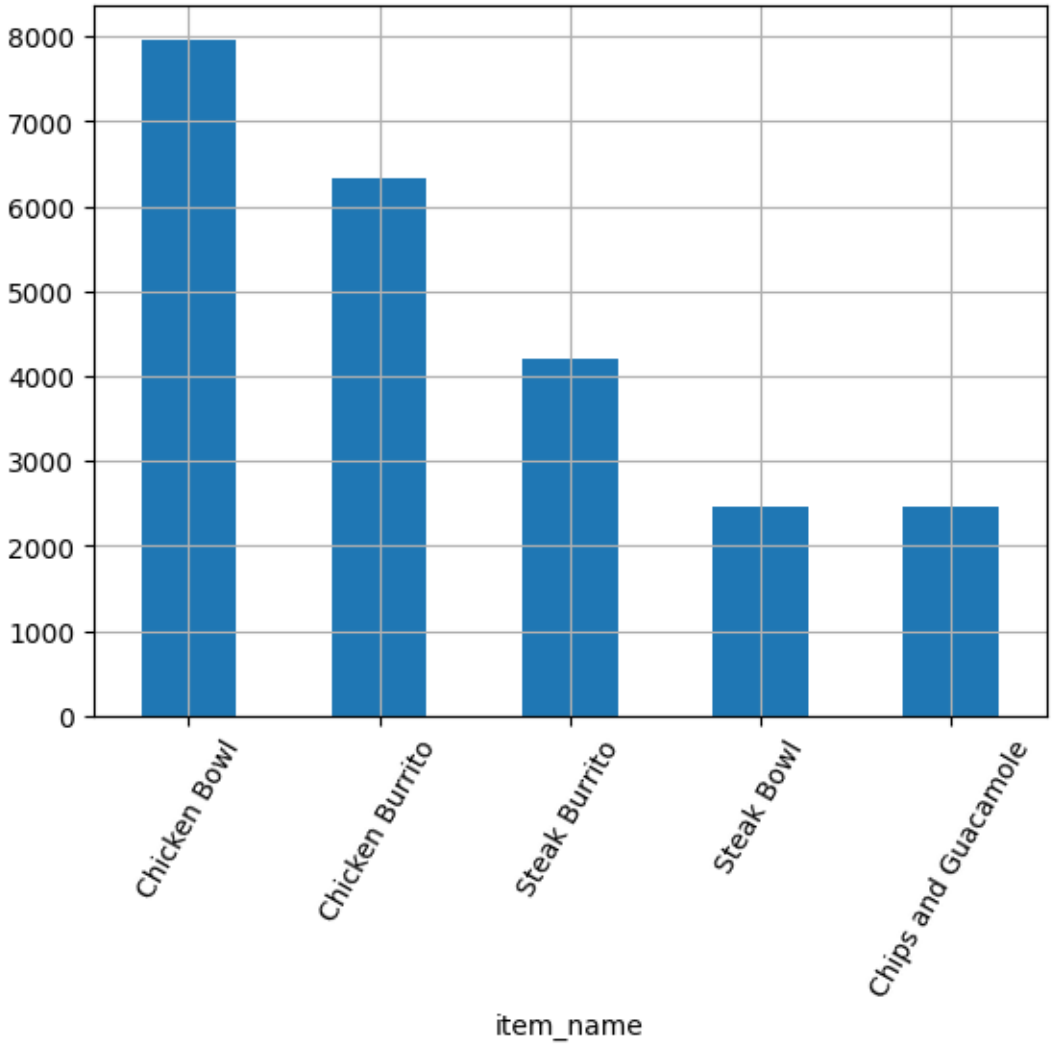
[141]:
```
order_id
1    11.56
2    33.96
3    12.67
4    21.00
5    13.70
Name: revenue, dtype: float64
```

[142]:
```
# cho biet 5 hoa don co tong thanh tien lon nhat

df.groupby('order_id')['revenue'].sum().sort_values(ascending = False).head()
```

[142]:
```
order_id
1443    1074.24
511      315.29
1559     246.00
1660     218.30
1786     197.70
Name: revenue, dtype: float64
```

[144]:
```
# cho biet 5 mon an co tong thanh tien lon nhat

df_ = df.groupby('item_name')['revenue'].sum().sort_values(ascending = False).
 ↪head()
df_.plot.bar(rot = 60, grid = True)
```

[144]: <Axes: xlabel='item_name'>



[145]:
```
# cho biet 5 mon an duoc dat nhieu nhat

df.groupby('item_name').size().sort_values(ascending = False).head()
```

[145]:
```
item_name
Chicken Bowl           717
Chicken Burrito        546
Chips and Guacamole    474
Steak Burrito          365
Canned Soft Drink      290
dtype: int64
```

[146]:
```
# cho biet mon an nao co so luong dat nhieu nhat

df[df['quantity'] == df['quantity'].max()][['order_id', 'item_name',␣
 ↪'quantity']]
```

[146]:

	order_id	item_name	quantity
3598	1443	Chips and Fresh Tomato Salsa	15

[147]:
```
# cho biet don dat hang nao dat nhieu mon an nhat

df_ = df['order_id'].value_counts().head(1)
df_
```

[147]:
```
926    21
Name: order_id, dtype: int64
```

[150]:
```
# liet ke cac mon an duoc dat

df[df['order_id']==df_.index[0]]
```

[150]:

	order_id	item_name	quantity	item_price	revenue
2304	926	Steak Burrito	1	9.25	9.25
2305	926	Chicken Bowl	1	8.75	8.75
2306	926	Chicken Bowl	1	8.75	8.75
2308	926	Steak Bowl	1	9.25	9.25
2309	926	Chicken Bowl	1	8.75	8.75
2310	926	Steak Burrito	1	9.25	9.25
2311	926	Chicken Burrito	1	8.75	8.75
2312	926	Chicken Bowl	1	8.75	8.75
2313	926	Chicken Bowl	1	8.75	8.75
2314	926	Chicken Salad Bowl	1	8.75	8.75
2315	926	Steak Bowl	1	9.25	9.25
2316	926	Chicken Burrito	1	8.75	8.75
2317	926	Steak Bowl	1	9.25	9.25
2319	926	Steak Bowl	1	9.25	9.25
2320	926	Chicken Burrito	1	8.75	8.75
2321	926	Chicken Bowl	1	8.75	8.75
2322	926	Chicken Bowl	1	8.75	8.75
2323	926	Barbacoa Burrito	1	9.25	9.25
2324	926	Chicken Burrito	1	8.75	8.75
2325	926	Steak Bowl	1	9.25	9.25
2326	926	Veggie Bowl	1	8.75	8.75

[152]:
```
# cho biet don dat hang nao co tong thanh tien lon nhat

df_ = df.groupby('order_id')['revenue'].sum()
max_revenue = df_.max()
```

```
df_[df_== max_revenue]
```

[152]:
```
order_id
1443    1074.24
Name: revenue, dtype: float64
```

[153]:
```
# cho biet doanh thu theo ngay

df['revenue'].sum()
```

[153]:
```
38914.11
```

[154]:
```
# cho biet mon an nao duoc dat nhieu nhat

df['item_name'].value_counts().head(1)
```

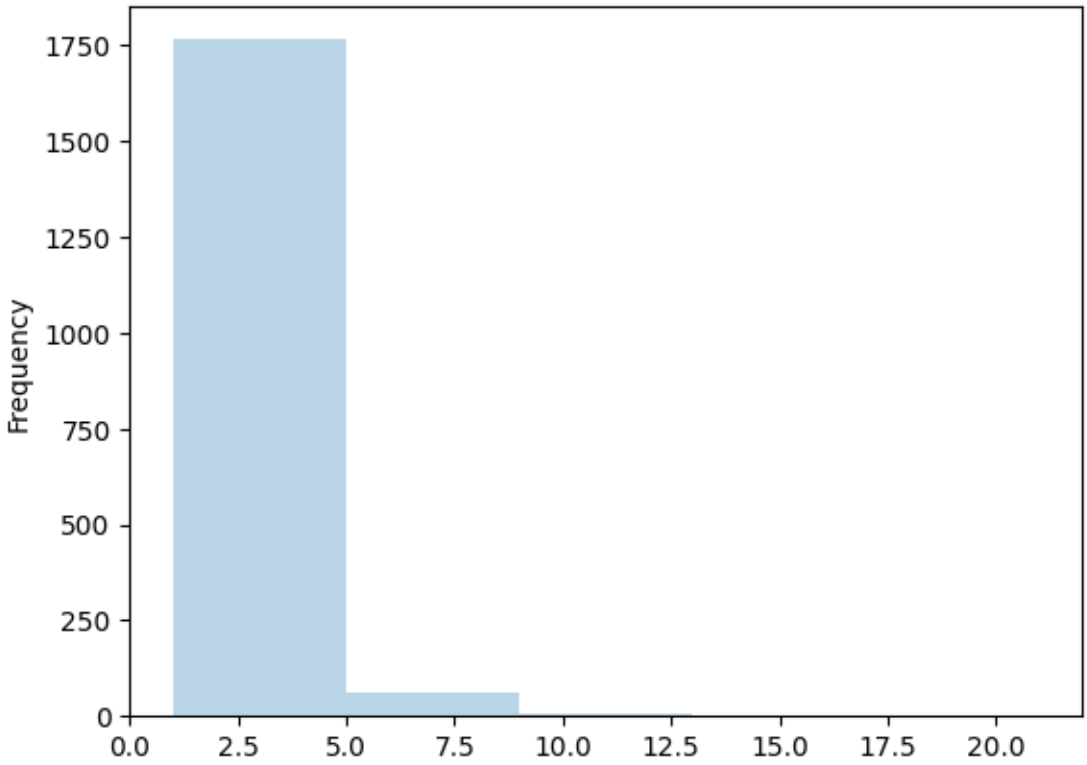
[154]:
```
Chicken Bowl    717
Name: item_name, dtype: int64
```

[158]:
```
# cho biet thanh tien nho nhat, lon nhat va trung binh cua cac hoa don

df_ = df.groupby('order_id')['revenue'].sum()
df_.agg(['min', 'max', 'mean'])
```

[158]:
```
min        8.750000
max     1074.240000
mean      21.218162
Name: revenue, dtype: float64
```

[159]:
```
df_ = df.groupby('order_id').size()
df_.plot.hist(bins = 5, alpha = 0.3)
```

[159]:
```
<Axes: ylabel='Frequency'>
```



[ ]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
↪texlive-plain-generic
```

[ ]:
```
!jupyter nbconvert --to pdf '/content/drive/MyDrive/Colab Notebooks/
↪Numpy_Pandas/numpy_pandas.ipynb'
```

[113]:
```

```