series筆記

```
# series的建立
import pandas as pd

s = pd.Series([10, 2, 3, 4, 5])
# 自定義索引
s = pd.Series([10, 2, 3, 4, 5], index=['A', 'B', 'C', 'D', 'E'])
# s = pd.Series([10,2,3,4,5], index=[1,2,3,4,5])
# 定義name
s = pd.Series([10, 2, 3, 4, 5], index=['A', 'B', 'C', 'D', 'E'], name='月份')
print(s)
```

```
A   10
B    2
C    3
D    4
E    5
Name: 月份, dtype: int64
```

```
# 透過字典來建立
s = pd.Series({"a": 1, "b": 2, "c": 3, "d": 4, "e": 5})
# print(s)
s2 = pd.Series([10, 2, 3, 4, 5], index=['A', 'B', 'C', 'D', 'E'], name='月份')
s1 = pd.Series(s2, index=["A", "C"])
print(s1)
```

```
A   10
C    3
Name: 月份, dtype: int64
```

```
# series的屬性
'''
index:Series的索引物件
```

values:Series的值
dtype或dtypes"Series的元素型別
shape:Series的形狀
ndim:Series的維度
size:Series的元素個數
name:Series的名稱
loc[] 顯式索引，按標籤索引或切片
iloc[] 隱式索引，按位置索引或切片
at[] 使用標籤訪問單個元素
iat[] 使用位置訪問單個元素
'''

```
# print(s.index)
# print(s.values)
# print(s.shape,s.ndim,s.size)
# s.name = 'test'
# print(s.dtype,s.name)
print(s.loc['a']) #顯式索引
print(s.iloc[0]) #隱式索引
print(s.at['a'])
print(s.iat[0])
```

```
1
1
1
1
```

```
# 訪問資料
# print(s[1])
# print(s['c'])
# print(s)
# print(s[s<3])
s['f'] = 6
print(s.head(2))
print(s.tail(1))
```

```
a  1
b  2
```

```
dtype: int64
f   6
dtype: int64
```

```
# 常見函式
s = pd.Series([10, 2, np.nan, None, 3, 4, 5], index=['A', 'B', 'C', 'D', 'E', 'F', 'G'],
name='data')
print(s)
```

```
---------------------------------------------------------------------------

NameError                 Traceback (most recent call last)

Cell In[5], line 2
      1 # 常見函式
----> 2 s = pd.Series([10,2,np.nan,None,3,4,5], index=['A', 'B', 'C', 'D', 'E','F','G'], name=
'data')
      3 print(s)


NameError: name 'np' is not defined
```

```
s.head(3)  # 預設取前5行的資料
s.tail(2)  #預設取後5行的資料
```

```
# 檢視所有的描述性資訊
s.describe()
```

```
# 獲取元素個數(忽略缺失值)
print(s.count())
```

```
# 獲取索引
print(s.keys())  # 方法
print(s.index)  # 屬性
```

```
print(s.isna()) #檢查Series裡的每一個元素是否為缺失值
s.isna()
```

```
s.isin([4, 5, 6]) # 檢查每個元素是否在引數集合中
```

```
s.describe()
```

```
print(s.mean()) #平均值
print(s.sum()) #總和
print(s.std()) #標準差
print(s.var()) #方差
print(s.min()) #最小值
print(s.max()) #最大值
print(s.median()) #中位數
```

```
print(s)
```

```
# print(s.sort_values())
print(s.quantile(0.8)) #分位數
#————————————————————
#2 3 4 5 10
#位置 4*0.8=3.2
#值的計算 5 + （10-5）*0.2 = 6
```

```
#眾數
s['H'] = 4
print(s.mode())
```

```
print(s.value_counts()) # 每個元素的計數
```

```
s.drop_duplicates() #去重
s.unique()
print(s.nunique()) #去重後的元素個數
```

```
# 排序 值、索引
s.sort_index() # 按索引排序
s.sort_values() #按值排序
```

```
'''建立一個包含10名學生數學成績的Series，成績範圍在50-100之間。
計算平均分、最高分、最低分，並找出高於平均分的學生人數。'''

import pandas as pd
import numpy as np

np.random.seed(42)
values = np.random.randint(50, 101, 10)
indexes = []
for i in range(1, 11):
    indexes.append('學生' + str(i))
scores = pd.Series(values, indexes)
# print(scores)
print('平均分：', scores.mean())
print('最高分：', scores.max())
print('最低分：', scores.min())
# 高於平均分的學生人數
mean = scores.mean()
print('高於平均分的學生人數:', len(scores[scores > mean]))
print('高於平均分的學生人數:', scores[scores > mean].count())
```

```
'''溫度資料統計
給定某城市一週每天的最高溫度Series，完成以下任務：
找出溫度超過30度的天數
計算平均溫度
將溫度從高到低排序
找出溫度變化最大的兩天
'''
import pandas as pd
import numpy as np
```

```
temperatures = pd.Series([28, 31, 29, 32, 30, 27, 33],
                         index=['週一', '週二', '週三', '週四', '週五', '週六', '週日'])
```

```
# 找出溫度超過30度的天數
n = temperatures[temperatures > 30].count()
print('超過30度的天數：', n)
```

```
# 計算平均溫度
print('平均溫度：', temperatures.mean())
```

```
# 將溫度從高到低排序
t2 = temperatures.sort_values(ascending=False)
print('從高到低排序：', t2)
```

```
# 找出溫度變化最大的兩天
# 28 31 29 32 30 27 33
# none 3 -2 3 -2 -3 6
t3 = temperatures.diff().abs()  #計算series的變化值

print('溫度變化最大的兩天', *(t3.sort_values(ascending=False).keys()[:2].tolist()))
```

```
'''
股票價格分析
給定某股票連續10個交易日的收盤價Series：
計算每日收益率（當日收盤價/前日收盤價 - 1）
找出收益率最高和最低的日期
計算波動率（收益率的標準差）


prices = pd.Series([102.3, 103.5, 105.1, 104.8, 106.2, 107.0, 106.5, 108.1, 109.3, 110.2],
index=pd.date_range('2023-01-01', periods=10))
'''
```

```
import pandas as pd
import numpy as np
```

```
# 日期序列
date = pd.date_range('2000-06-1', periods=60)
print(list(date))
```

```
prices = pd.Series([102.3, 103.5, 105.1, 104.8, 106.2, 107.0, 106.5, 108.1, 109.3, 110.2],
                   index=pd.date_range('2023-01-01', periods=10))
```

```
prices
```

```
'''計算每日收益率（當日收盤價/前日收盤價 - 1）
找出收益率最高和最低的日期
計算波動率（收益率的標準差）'''
# 計算每日收益率
a = prices.pct_change()  #percent  103.5/102.3 - 1
```

```
# 收益率最高的日期
print(a.idxmax())
# 收益率最低的日期
print(a.idxmin())
```

```
# 波動率
print(a.std())
```

```
'''銷售資料分析
某產品過去12個月的銷售量Series：
計算季度平均銷量（每3個月為一個季度）
找出銷量最高的月份
計算月環比增長率
找出連續增長超過2個月的月份

sales = pd.Series([120, 135, 145, 160, 155, 170, 180, 175, 190, 200, 210,
220],index=pd.date_range('2022-01-01', periods=12, freq='MS'))'''
```

```
a = pd.date_range('2022-01-01', periods=12, freq='MS')
```

```
sales = pd.Series([120, 135, 145, 160, 155, 170, 180, 175, 190, 200, 210, 220],
                  index=pd.date_range('2022-01-01', periods=12, freq='MS'))
```

```
sales
```

```
# 季度的平均銷量
# (120+135+145)/3 = 400/3
sales.resample('QS').mean()  #重新取樣
```

```
print('銷量最高的月份', sales.idxmax())
```

```
print('月環比的增長率')
sales.pct_change()
```

```
# 找出連續增長超過2個月的月份
sales
```

```
a = sales.pct_change()
b = a > 0
b[b.rolling(3).sum() == 3].keys().tolist()
```

```
'''每小時銷售資料分析
某商店每小時銷售額Series：
按天重取樣計算每日總銷售額
計算每天營業時間（8:00-22:00）和非營業時間的銷售額比例
找出銷售額最高的3個小時'''
```

```
import pandas as pd
import numpy as np

np.random.seed(42)
```

```
h = pd.Series(np.random.randint(0, 100, 24),
        index=pd.date_range('2025-01-01', periods=24, freq='h'))
# 按天重取樣計算每日總銷售額
day_sales = h.resample('D').sum()
# hours_sales.sum()
# 計算每天營業時間（8:00-22:00）和非營業時間的銷售額比例
mask = (h.index.hour >= 8) & ((h.index.hour <= 22))
b = h[mask]
n_b = h[~mask]
print(b.sum() / n_b.sum())
# 找出銷售額最高的3個小時
print(h.nlargest(3).keys())
```