

LẬP TRÌNH TRÍ TUỆ NHÂN TẠO

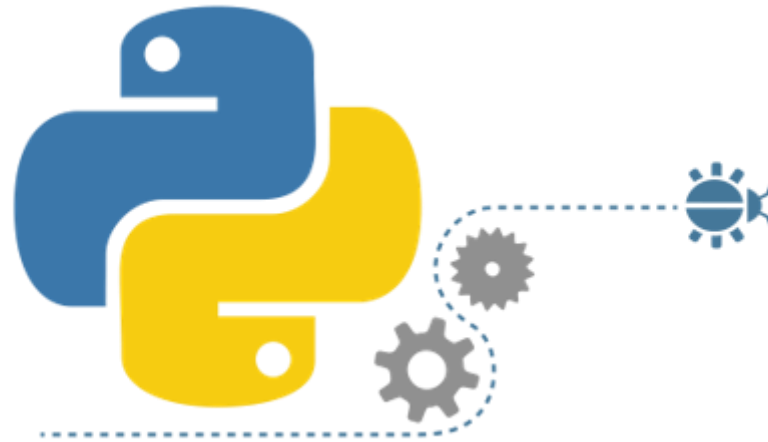


THS. VƯƠNG XUÂN CHÍ

VXCHI@NTT.EDU.VN

0903 270 567

CHƯƠNG 9

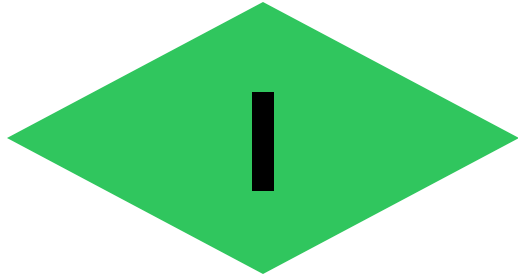


PANDAS



Nội dung

- 1. Giới thiệu và cài đặt pandas*
- 2. Cấu trúc dữ liệu trong pandas*
- 3. Làm việc với series*
- 4. Làm việc với dataframe*
- 5. Làm việc với panel*
- 6. Chọn và nhóm phần tử*



Giới thiệu và cài đặt pandas



Cài đặt: “pip install pandas”

- “pandas” là thư viện mở rộng từ numpy, chuyên để xử lý dữ liệu cấu trúc dạng bảng
- Tên “pandas” là dạng số nhiều của “panel data”

```
pip install pandas
```



Đặc điểm nổi bật của pandas

- Đọc dữ liệu từ nhiều định dạng
- Liên kết dữ liệu và tích hợp xử lý dữ liệu bị thiếu
- Xoay và chuyển đổi chiều của dữ liệu dễ dàng
- Tách, đánh chỉ mục và chia nhỏ các tập dữ liệu lớn dựa trên nhãn
- Có thể nhóm dữ liệu cho các mục đích hợp nhất và chuyển đổi
- Lọc dữ liệu và thực hiện query trên dữ liệu
- Xử lý dữ liệu chuỗi thời gian và lấy mẫu

2

Cấu trúc dữ liệu trong pandas

Cấu trúc dữ liệu trong pandas

- **Dữ liệu của pandas có 3 cấu trúc chính:**
 - ✓ **Series** (loạt): cấu trúc 1 chiều, mảng dữ liệu đồng nhất
 - ✓ **Dataframe** (khung): cấu trúc 2 chiều, dữ liệu trên các cột là đồng nhất (có phần giống như table trong SQL, nhưng với các dòng được đặt tên)
 - ✓ **Panel** (bảng): cấu trúc 3 chiều, có thể xem như một tập các dataframe với thông tin bổ sung
- **Dữ liệu series gần giống kiểu array trong numpy, nhưng có 2 điểm khác biệt quan trọng:**
 - ✓ Chấp nhận dữ liệu thiếu (NaN – không xác định)
 - ✓ Hệ thống chỉ mục phong phú (giống dictionary?)

Cấu trúc dataframe

- Dữ liệu **2 chiều**
- Các **cột** có tên
- Dữ liệu trên cột là đồng nhất (series?)
- Các **dòng** có thể có tên
- Có thể có ô thiếu dữ liệu

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria



Cấu trúc panel

- Dữ liệu **3 chiều**
- Một tập các **dataframe**
- Các dataframe có cấu trúc tương đồng
- Có thể có các thông tin bổ sung cho từng dataframe

		Open	Close
Major	Minor		
3/31/2015	IBM	23.602	132.903
	APPL	421.412	212.665
	CVX	568.055	409.201
	BHP	487.414	515.413
4/30/2015	IBM	150.868	457.895
	APPL	204.729	957.179
	CVX	90.679	888.687
	BHP	831.527	714.202
5/31/2015	IBM	788.582	922.422
	APPL	329.716	304.964
	CVX	36.578	981.508
	BHP	313.848	882.293

3

Làm việc với series

Tạo dữ liệu series (1)

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

```
S = pd.Series(np.random.randint(100, size = 4))  
print(S)  
print(S.index)  
print(S.values)
```

```
0      62  
1      68  
2      91  
3      99  
dtype: int32  
RangeIndex(start=0, stop=4, step=1)  
[62 68 91 99]
```

Tạo dữ liệu series (2)

```
import pandas as pd
```

```
chi_so = ["Ke toan", "Kinh Te", "CNTT", "Co khi"]
```

```
gia_tri = [310, 360, 580, 340]
```

```
S = pd.Series(gia_tri, index=chi_so)
```

```
print(S)
```

```
print(S.index)
```

```
print(S.values)
```

```
KT          310
KT          360
CNTT        580
Co khi      340
dtype: int64
Index(['KT', 'KT', 'CNTT', 'Co khi'],
      dtype='object')
[310 360 580 340]
```



Tạo dữ liệu series (3)

```
import pandas as pd
```

```
chi_so = ["KT", "KT", "CNTT", "Co khi"] # trùng nhau
```

```
gia_tri = [310, 360, 580, 340]
```

```
S = pd.Series(gia_tri, index=chi_so)
```

```
print(S)
```

```
print(S.index)
```

```
print(S.values)
```

```
KT          310
KT          360
CNTT        580
Co khi      340
dtype: int64
Index(['KT', 'KT', 'CNTT', 'Co khi'],
      dtype='object')
[310 360 580 340]
```

Truy vấn dữ liệu thông qua chỉ số

```
import pandas as pd
```

```
chi_so= ["KinhTe", "KinhTe", "CNTT", "Co khi"] #trùng nhau
```

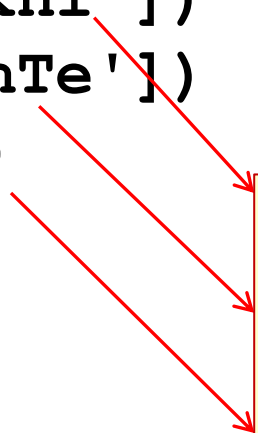
```
gia_tri = [310, 360, 580, 340]
```

```
S = pd.Series(gia_tri, index=chi_so)
```

```
print(S['Co khi'])
```

```
print(S['KinhTe'])
```

```
print(S.CNTT)
```



```
340  
KinhTe      310  
KinhTe      360  
dtype: int64  
580
```

Phép toán trên series

```
import pandas as pd
```

```
chi_so = ["Ke toan", "KT", "CNTT", "Co khi"]
```

```
gia_tri = [310, 360, 580, 340]
```

```
# chỉ số giống nhau thì tính gộp, nếu không thì NaN
```

```
S = pd.Series(gia_tri, index=chi_so)
```

```
P = pd.Series([100, 100], ['CNTT', 'PM'])
```

```
Y = S + P
```

```
print(Y)
```

CNTT	680.0
Co khi	NaN
KT	NaN
Ke toan	NaN
PM	NaN
dtype:	float64

Phép toán trên series

- Nguyên tắc chung của việc thực hiện phép toán trên series như sau:
 - ✓ Nếu là phép toán giữa 2 series, thì các giá trị cùng chỉ số sẽ thực hiện phép toán với nhau, trường hợp không có giá trị ở cả 2 series thì trả về NaN
 - ✓ Nếu là phép toán giữa series và 1 số, thì thực hiện phép toán trên số đó với tất cả các giá trị trong series



Một số phương thức hữu ích

- **S.axes**: trả về danh sách các chỉ mục của S
- **S.dtype**: trả về kiểu dữ liệu các phần tử của S
- **S.empty**: trả về True nếu S rỗng
- **S.ndim**: trả về số chiều của S (1)
- **S.size**: trả về số phần tử của S
- **S.values**: trả về list các phần tử của S
- **S.head(n)**: trả về n phần tử đầu tiên của S
- **S.tail(n)**: trả về n phần tử cuối cùng của S



apply() một hàm khác trên series

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

```
def Tang(x) :
    return x if x > 500 else x + 1000
chi_so = ["Ke toan", "KT", "CNTT", "Co khi"]
gia_tri = [310, 360, 580, 340]
S = pd.Series(gia_tri, chi_so)
# áp dụng Tang trên S (không thay đổi S)
print(S.apply(Tang))
```

Ke toan	1310
KT	1360
CNTT	580
Co khi	1340
dtype:	int64

4

Làm việc với dataframe

Khởi tạo dataframe

Cú pháp:

pandas.DataFrame(data, index, columns, dtype, copy)

■ Trong đó:

- ✓ **'data'** sẽ nhận giá trị từ nhiều kiểu khác nhau như list, dictionary, ndarray, series,... và cả các DataFrame khác
- ✓ **'index'** là nhãn chỉ mục hàng của dataframe
- ✓ **'columns'** là nhãn chỉ mục cột của dataframe
- ✓ **'dtype'** là kiểu dữ liệu cho mỗi cột
- ✓ **'copy'** nhận giá trị True/False để chỉ rõ dữ liệu có được copy sang vùng nhớ mới không, mặc định là False

Tạo dataframe từ list

```
crimes_rates = {  
    "Year": [1960, 1961, 1962, 1963, 1964],  
    "Population": [179323175, 182992000, 185771000, 188483000,  
191141000],  
    "Total": [3384200, 3488000, 3752200, 4109500, 4564600],  
    "Violent": [288460, 289390, 301510, 316970, 364220]  
}  
crimes_dataframe = pd.DataFrame(crimes_rates)  
print(crimes_dataframe)
```

	Year	Population	Total	Violent
0	1960	179323175	3384200	288460
1	1961	182992000	3488000	289390
2	1962	185771000	3752200	301510
3	1963	188483000	4109500	316970
4	1964	191141000	4564600	364220

Tạo dataframe từ list các dictionary

```
data = [  
    {'MIT': 5000, 'Stanford': 4500, "NTT":15000},  
    {'MIT': 1, 'Stanford': 2, "NTT":200}  
]  
df = pd.DataFrame(data, index=['NumOfStudents', "ranking"])  
print(df)  
print(df.NTT.dtype)
```

	MIT	Stanford	NTT
NumOfStudents	5000	4500	15000
ranking	1	2	200
int64			

Tạo dataframe từ dictionary series

```
data = {  
    "one": pd.Series([1,23,45], index = [1,2,3]),  
    "two": pd.Series([1000,2400,1132,3434],index= [1,2,3,4])  
}  
df = pd.DataFrame(data)  
print(df)
```

	one	two
1	1.0	1000
2	23.0	2400
3	45.0	1132
4	NaN	3434

Đọc dữ liệu từ file .csv

■ Nội dung của file brics.csv:

- ✓ Số liệu về các quốc gia thuộc khối BRICS
- ✓ Sử dụng dấu phẩy để ngăn giữa các dữ liệu
- ✓ Mỗi dữ liệu trên 1 dòng
- ✓ Dòng đầu tiên là tên các cột

```
,country,population,area,capital  
BR,Brazil,200,8515767,Brasilia  
RU,Russia,144,17098242,Moscow  
IN,India,1252,3287590,New Delhi  
CH,China,1357,9596961,Beijing  
SA,South Africa,55,1221037,Pretoria
```



Đọc dữ liệu từ file .csv

```
import pandas as pd
```

```
d = pd.read_csv("brics.csv")  
print(d)
```

	Unnamed: 0	country	capital	area	population
0	BR	Brazil	Brasilia	8.516	200.40
1	RU	Russia	Moscow	17.100	143.50
2	IN	India	New Delhi	3.286	1252.00
3	CH	China	Beijing	9.597	1357.00
4	SA	South Africa	Pretoria	1.221	52.98



Đọc dữ liệu từ file .csv

```
import pandas as pd
```

```
# đọc dữ liệu và quy định cột 0 dùng làm chỉ số dòng  
d = pd.read_csv("brics.csv", index_col = 0)  
print(d)
```

	country	capital	area	population
BR	Brazil	Brasilia	8.516	200.40
RU	Russia	Moscow	17.100	143.50
IN	India	New Delhi	3.286	1252.00
CH	China	Beijing	9.597	1357.00
SA	South Africa	Pretoria	1.221	52.98

Truy cập theo từng cột

- Sử dụng tên cột làm chỉ số hoặc dùng luôn tên cột
- Việc truy cập này trả về tham chiếu đến dữ liệu, vì vậy có thể sử dụng phép gán để cập nhật dữ liệu theo cột

```
d = pd.read_csv("brics.csv")  
print(d["country"]) #hoặc print(d.country)
```

```
BR          Brazil  
RU          Russia  
IN          India  
CH          China  
SA          South Africa  
Name: country, dtype: object
```

Thêm một cột (1)

- Bằng cách sử dụng một cột mới chưa có

```
d = pd.read_csv("brics.csv", index_col=0)
d["on_earth"]=[True, True, True, True, True]
print(d)
```

	country	capital	area	population	on_earth
BR	Brazil	Brasilia	8.516	200.40	True
RU	Russia	Moscow	17.100	143.50	True
IN	India	New Delhi	3.286	1252.00	True
CH	China	Beijing	9.597	1357.00	True
SA	South Africa	Pretoria	1.221	52.98	True

Thêm một cột (2)

- Bằng cách sử dụng một cột mới chưa có và thiết lập công thức phù hợp

```
d = pd.read_csv("brics.csv", index_col=0)
d["density"] = d["population"] / d["area"] * 1000
print(d)
```

	country	capital	area	population	density
BR	Brazil	Brasilia	8.516	200.40	23532.174730
RU	Russia	Moscow	17.100	143.50	8391.812865
IN	India	New Delhi	3.286	1252.00	381010.346926
CH	China	Beijing	9.597	1357.00	141398.353652
SA	South Africa	Pretoria	1.221	52.98	43390.663391

Truy cập vào từng ô trên dataframe

- Bằng cách kết hợp chỉ mục dòng và cột

```
d = pd.read_csv("brics.csv", index_col=0)
```

```
print(d.loc["BR", "capital"]) # Brasilia
```

```
print(d["capital"].loc["BR"]) # Brasilia
```

```
print(d.loc["BR"]["capital"]) # Brasilia
```

Xóa dòng hoặc cột bằng drop

tạo ra dataframe mới bằng cách xóa 2 cột

```
print(d.drop(["area", "population"], axis=1))
```

#trường hợp muốn xóa trên d, thêm tham số inplace=True

```
d.drop(["area", "population"], axis=1, inplace=True)  
print(d)
```

	country	capital
BR	Brazil	Brasilia
RU	Russia	Moscow
IN	India	New Delhi
CH	China	Beijing
SA	South Africa	Pretoria

Tính tổng và tổng tích lũy

tính tổng của cột population, trả về tổng

```
print(d.population.sum())
```

tính tổng của cột population, trả về các tổng trong quá trình cộng

```
print(d.population.cumsum())
```

```
3005.88
```

```
BR      200.40
```

```
RU      343.90
```

```
IN     1595.90
```

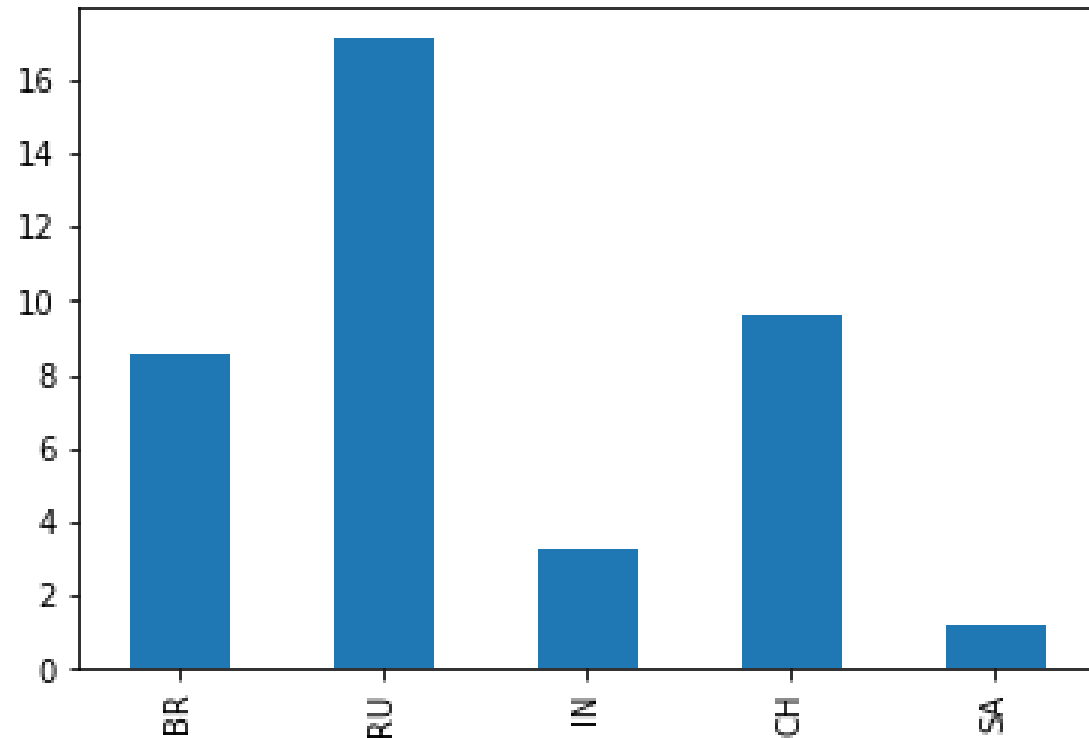
```
CH     2952.90
```

```
SA     3005.88
```

```
Name: population, dtype: float64
```

Kết hợp giữa pandas và matplotlib

```
import pandas as pd
import matplotlib.pyplot as plt
d = pd.read_csv("brics.csv", index_col = 0)
d.area.plot(kind="bar")
plt.show()
```



5

Làm việc với panel

Cấu trúc panel

- **Panel** được sử dụng nhiều trong kinh tế lượng
- Dữ liệu có **3 trục**:
- **Items** (trục 0): mỗi item là một dataframe bên trong
- **Major axis** (trục 1 – trục chính): các dòng
- **Minor axis** (trục 2 – trục phụ): các cột
- Không được phát triển tiếp (thay bởi MultiIndex)

		Open	Close
Major	Minor		
3/31/2015	IBM	23.602	132.903
	APPL	421.412	212.665
	CVX	568.055	409.201
	BHP	487.414	515.413
4/30/2015	IBM	150.868	457.895
	APPL	204.729	957.179
	CVX	90.679	888.687
	BHP	831.527	714.202
5/31/2015	IBM	788.582	922.422
	APPL	329.716	304.964
	CVX	36.578	981.508
	BHP	313.848	882.293

Tạo panel

Cú pháp:

`pandas.Panel(data, items, major_axis, minor_axis, dtype, copy)`

- Trong đó:
 - `'data'` có thể nhận các kiểu dữ liệu sau: ndarray, series, map, lists, dict, hằng số và cả dataframe khác
 - `'items'` là axis = 0
 - `'major_axis'` là axis = 1
 - `'minor_axis'` là axis = 2
 - `'dtype'` là kiểu dữ liệu mỗi cột
 - `'copy'` nhận giá trị True/False để khởi tạo dữ liệu có chia sẻ memory hay không

Tạo panel

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

```
data = np.random.rand(2, 3, 4)  
p = pd.Panel(data)  
print(p)
```

```
<class 'pandas.core.panel.Panel'>  
Dimensions: 2 (items) x 3 (major_axis) x 4 (minor_axis)  
Items axis: 0 to 1  
Major_axis axis: 0 to 2  
Minor_axis axis: 0 to 3
```



Tạo panel

p.to_frame()

		0	1
major	minor		
0	0	0.335571	0.010409
	1	0.267106	0.843688
	2	0.840885	0.211749
	3	0.049653	0.722182
1	0	0.755207	0.282777
	1	0.674844	0.543207
	2	0.634314	0.433802
	3	0.290120	0.613040
2	0	0.322059	0.263548
	1	0.341035	0.702612
	2	0.634411	0.917126
	3	0.281678	0.809592

6

Chọn và nhóm phần tử

Chọn với iloc, loc và ix

- **Pandas có 3 phương pháp chọn phần tử**
 1. **Dùng iloc**: chọn theo chỉ số hàng và cột
 - Cú pháp: `data.iloc[<row selection>, <column selection>]`
 - Tham số có thể là số nguyên, list các số nguyên, slice object với các số nguyên (ví dụ 2:7), mảng boolean,...
 2. **Dùng loc**: chọn theo nhãn hàng hoặc nhãn cột
 - Cú pháp: `data.loc[<row selection>, <column selection>]`
 - Tham số là nhãn (chứ không phải chỉ số)
 3. **Dùng ix**: lai giữa 2 cách trên, nếu truyền tham số là số nguyên thì nó làm việc như iloc, truyền kiểu giá trị khác thì nó làm việc như loc.

Nhóm phần tử

```
df2= pd.DataFrame({'X': ['B', 'B', 'A', 'A'], 'Y': [1, 2, 3, 4]})  
df2.groupby(['X']).sum()  
print(df2)
```

```
df2.groupby(['X'], sort=False).sum()
```

	X	Y
0	B	1
1	B	2
2	A	3
3	A	4

	Y
X	
A	7
B	3

Nhóm phần tử

```
df3= pd.DataFrame({'X': ['A', 'B', 'A', 'B'], 'Y': [1, 4, 3, 2]})  
df3.groupby(['X']).get_group('A')  
print(df3)
```

	X	Y
0	A	1
1	B	4
2	A	3
3	B	2

```
df3.groupby(['X']).get_group('B')
```

	X	Y
0	A	1
2	A	3

7

Ứng dụng sử dụng pandas

Các bài toán phân tích dữ liệu

- Dữ liệu kết quả xổ số (độc đắc) từ ngày 1-1-2000 đến ngày 1-21-2022
Lưu ở định dạng csv, 2 cột:
 - ✓ Cột 1: ngày ra số
 - ✓ Cột 2: số độc đắc
 - Dạng số (nếu không đủ 5 chữ số thì có nghĩa là đã bị xóa các chữ số 0 ở đầu)
 - Có thể không có dữ liệu (mỗi năm có 4 ngày không quay xổ số)
- Bài toán phân tích các chiến lược kinh doanh
- Bài toán khoa học
- ...

Đọc và tiền xử lý dữ liệu

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

# đọc dữ liệu từ file csv, chuyển dữ
# liệu cột 1 sang date
df = pd.read_csv("kqxs.csv", index_col
=0, parse_dates=True)

# xóa bỏ các dòng không có dữ liệu
df.dropna(inplace= True)
# thêm cột mới là 2 số cuối của giải
# độc đặc
df['Cuoi'] = df.So % 100
```

	So	Cuoi
Ngày		
2000-01-01	22434.0	34.0
2000-01-02	59434.0	34.0
2000-01-03	27277.0	77.0
2000-01-04	33585.0	85.0
2000-01-05	69095.0	95.0

2022-01-17	87623.0	23.0
2022-01-18	38740.0	40.0
2022-01-19	15245.0	45.0
2022-01-20	32370.0	70.0
2022-01-21	96511.0	11.0
[6649 rows x 2 columns]		



Khảo sát dữ liệu

```
# trích xuất cột mới thành dữ liệu series để dễ xử lý  
s = pd.Series(df.Cuoi, dtype='int64')
```

```
# xem phân bố dữ liệu: biểu đồ histogram, 100 nhóm  
plt.plot('hist', bins=100)  
plt.show()
```

```
# một dạng phân bố dữ liệu khác: biểu đồ bar, đếm tần suất  
plt.value_counts().sort_index().plot('bar')  
plt.show()
```



THANK YOU

THS. VƯƠNG XUÂN CHÍ

VXCHI@NTT.EDU.VN

0903 270 567