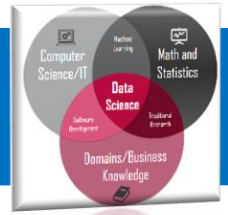
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
Khoa Công Nghệ Thông Tin


Bài giảng

# NHẬP MÔN KHOA HỌC DỮ LIỆU

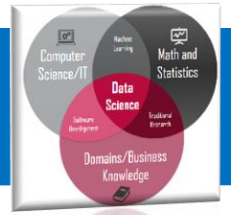
## (Data Science)

ThS. Phạm Đình Tài
0985.73.39.39
pdtai@ntt.edu.vn



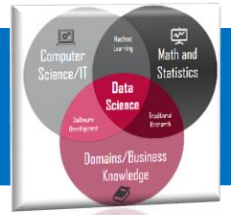

# CHƯƠNG 5

# Thư viện Panda



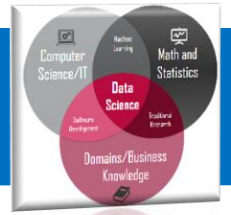

# Nội dung

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. *Bài tập*
6. *Hướng dẫn bài tập buổi trước*
7. *Làm việc với panel*
8. *Chọn và nhóm phần tử*
9. *Sử dụng pandas trong bài toán thực tế*




## Phần 1

# Giới thiệu và cài đặt pandas

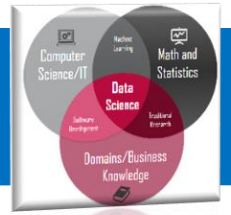



# Cài đặt: "pip install pandas"

- "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"

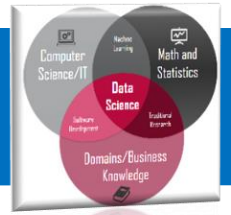
Command Prompt

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\PHATTAI>pip install pandas
Collecting pandas
  Downloading pandas-1.4.0-cp310-cp310-win_amd64.whl (10.6 MB)
     |████████████████████████████████| 10.6 MB 2.2 MB/s
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packa
ges (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.21.0 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packages (from
 pandas) (1.22.1)
Collecting pytz>=2020.1
  Downloading pytz-2021.3-py2.py3-none-any.whl (503 kB)
     |████████████████████████████████| 503 kB 2.2 MB/s
Requirement already satisfied: six>=1.5 in c:\users\phattai\appdata\local\programs\python\python310\lib\site-packages (from pyth
on-dateutil>=2.8.1->pandas) (1.16.0)
Installing collected packages: pytz, pandas
Successfully installed pandas-1.4.0 pytz-2021.3
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\PHATTAI\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgr
ade pip' command.

C:\Users\PHATTAI>
```



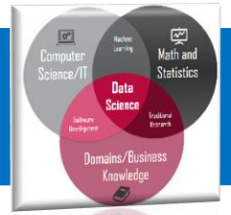

# Đặc điểm nổi bật của pandas

- 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



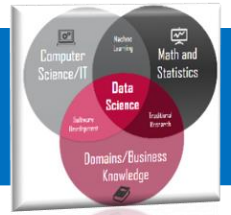

## Phần 2

# Cấu trúc dữ liệu trong pandas




# Cấu trúc dữ liệu trong pandas

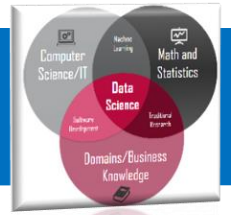
- Dữ liệu của pandas có 3 cấu trúc chính:
  - ✓ Series (loạt): cấu trúc 1 chiều, mảng dữ liệu đồng nhất
  - ✓ Dataframe (khung): cấu trúc 2 chiều, dữ liệu trên các cột là đồng nhất (có phần giống như table trong SQL, nhưng với các dòng được đặt tên)
  - ✓ Panel (bảng): cấu trúc 3 chiều, có thể xem như 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

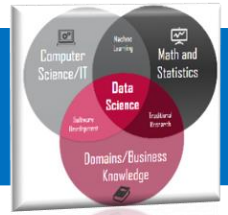
| | country | population | area | capital |
|---|---|---|---|---|
| BR | Brazil | 200 | 8515767 | Brasilia |
| RU | Russia | 144 | 17098242 | Moscow |
| IN | India | 1252 | 3287590 | New Delhi |
| CH | China | 1357 | 9596961 | Beijing |
| SA | South Africa | 55 | 1221037 | Pretoria |




# 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 |



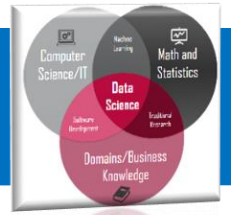

## Phần 3

# 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)
```

```
Command Prompt
Microsoft Windows [Version 10.0.19042.1466]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries1.py"
0    98
1    21
2    74
3    20
dtype: int32
RangeIndex(start=0, stop=4, step=1)
[98 21 74 20]
```




# Tạo dữ liệu series (2)

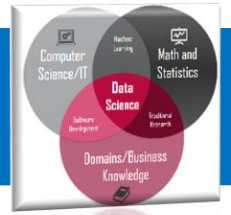
```
import pandas as pd
import numpy as np

chi_so = ["Ke toan", "KT", "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)
```

```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries2.py"
Ke toan    310
KT         360
CNTT       580
Co khi     340
dtype: int64
Index(['Ke toan', 'KT', 'CNTT', 'Co khi'], dtype='object')
[310 360 580 340]

C:\Users\PHATTAI>
```


# Tạo dữ liệu series (3)

```
import pandas as pd
import numpy as np

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)
```

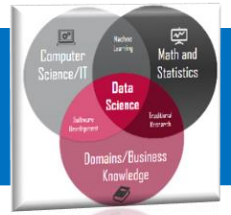
```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\taodulieuseries3.py"
KT        310
KT        360
CNTT      580
Co khi    340
dtype: int64
Index(['KT', 'KT', 'CNTT', 'Co khi'], dtype='object')
[310 360 580 340]
```




# Truy vấn dữ liệu thông qua chỉ số

```
import pandas as pd
import numpy as np

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['Co khi'])
print(S['KT'])
print(S.CNTT)
```

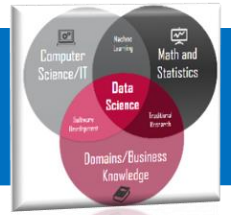
```
C:\Users\PHATTAI>"C:\Users\PHATTAI\Desktop\Python\Chương 5\truyvandulieuthongquachiso.py"
340
KT    310
KT    360
dtype: int64
580
```




# Phép toán trên series

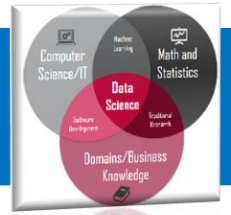
```
import pandas as pd
import numpy as np

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
```



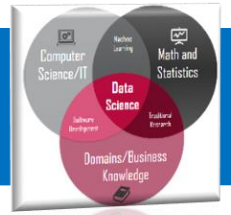

# Phép toán trên series

- 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

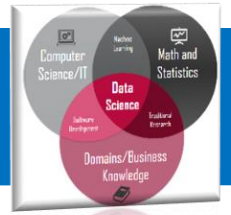
- S.axes: trả về danh sách các chỉ mục của S
- S.dtype: trả về kiểu dữ liệu các phần tử của S
- S.empty: trả về True nếu S rỗng
- S.ndim: trả về số chiều của S (1)
- S.size: trả về số phần tử của S
- S.values: trả về list các phần tử của S
- S.head(n): trả về n phần tử đầu tiên của S
- S.tail(n): trả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




# apply() một hàm khác trên series

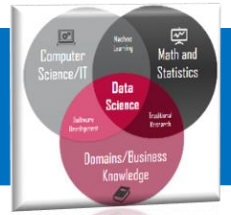
```
import pandas as pd
import numpy as np

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
```




# Phần 4

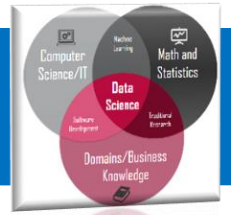
# Làm việc với dataframe




# Khởi tạo dataframe

- Cú pháp chung:

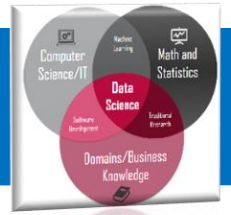
  pandas.DataFrame(data, index, columns, dtype, copy)
- Trong đó:
  - ✓ 'data' sẽ nhận giá trị từ nhiều kiểu khác nhau như list, dictionary, ndarray, series,... và cả các DataFrame khác
  - ✓ 'index' là nhãn chỉ mục hàng của dataframe
  - ✓ 'columns' là nhãn chỉ mục cột của dataframe
  - ✓ 'dtype' là kiểu 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)
```

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

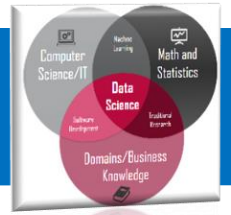



# Tạo dataframe từ list các dictionary

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

| | DHTL | MIT | Stanford |
|---|---|---|---|
| NumOfStudents | 15000 | 5000 | 4500 |
| ranking | 200 | 1 | 2 |

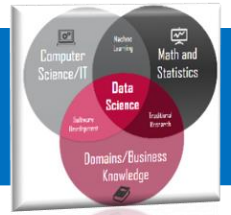
dtype('int64')




# Tạo dataframe từ dictionary serias

```
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

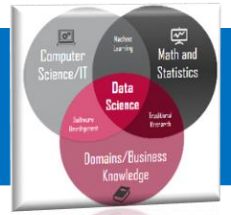
```
import pandas as pd

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

| | Unnamed: 0 | country | population | area | capital |
|---|---|---|---|---|---|
| 0 | BR | Brazil | 200 | 8515767 | Brasilia |
| 1 | RU | Russia | 144 | 17098242 | Moscow |
| 2 | IN | India | 1252 | 3287590 | New Delhi |
| 3 | CH | China | 1357 | 9596961 | Beijing |
| 4 | SA | South Africa | 55 | 1221037 | Pretoria |


# 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 | 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 |

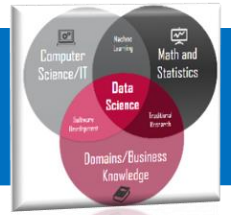



# Truy cập theo từng cột

- 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

```
print(brics["country"])

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

```
print(brics.country)

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

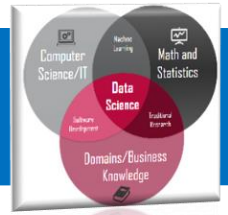



# Thêm một cột (1)

- Bằng cách sử dụng một cột mới chưa có

```
brics["on_earth"] = [True, True, True, True, True]
print(brics)
```

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

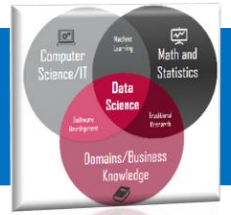



# Thêm một cột (2)

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

```
brics["density"] = brics["population"] / brics["area"] * 1000000
print(brics)
```

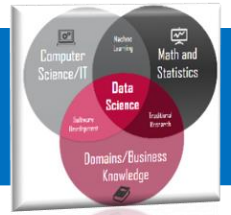
```
        country  population     area    capital  on_earth     density
BR       Brazil         200  8515767   Brasilia      True   23.485847
RU       Russia         144 17098242     Moscow      True    8.421918
IN        India        1252  3287590  New Delhi      True  380.826076
CH        China        1357  9596961    Beijing      True  141.398928
SA South Africa          55  1221037   Pretoria      True   45.043680
```




# Truy cập vào từng ô trên dataframe

- Bằng cách kết hợp chỉ mục dòng và cột

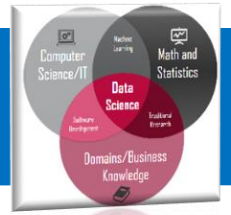
```
print(brics.loc["CH","capital"])

Beijing

print(brics["capital"].loc["CH"])

Beijing

print(brics.loc["CH"]["capital"])

Beijing
```




# 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)
```

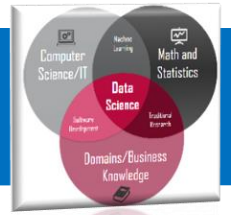
| | country | capital |
|---|---|---|
| BR | Brazil | Brasilia |
| RU | Russia | Moscow |
| IN | India | New Delhi |
| CH | China | Beijing |
| SA | South Africa | Pretoria |




# Tính tổng và tổng tích lũy

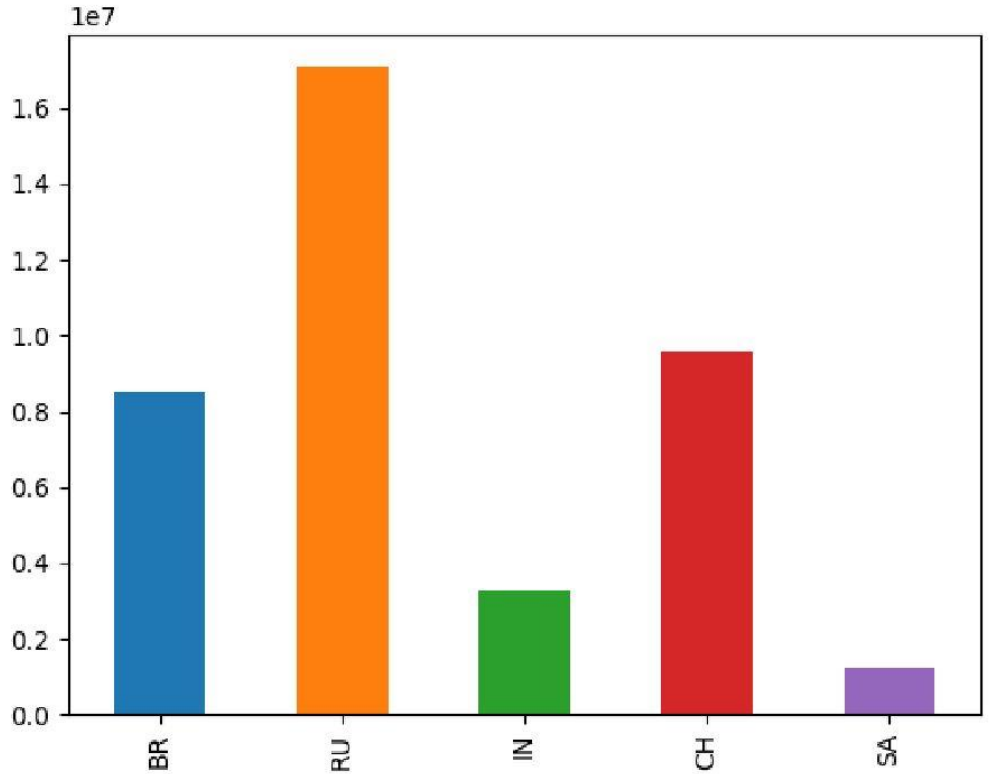
```
# tính tổng của cột population, trả về tổng
print(d.population.sum())
# tính tổng của cột population, trả về các tổng trong quá trình cộng
print(d.population.cumsum())
```

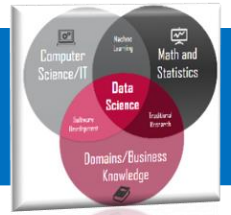
```
3008
BR      200
RU      344
IN     1596
CH     2953
SA     3008
Name: population, dtype: int64
```




# Kết hợp giữa pandas và matplotlib

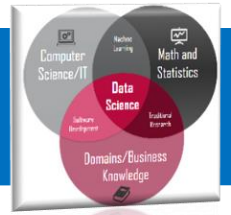
```
import pandas as pd
import matplotlib.pyplot as plt
d = pd.read_csv("brics.csv", index_col = 0)
d.area.plot(kind='bar')
plt.show()
```






## Phần 7

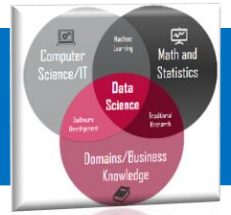
# Làm việc với panel




# 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 |

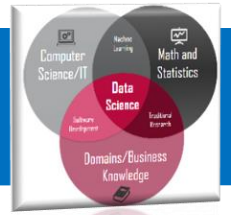



# Tạo panel

- Cú pháp:

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

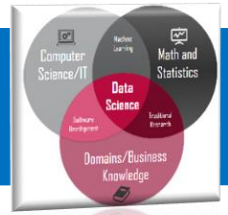
- Trong đó:
  - 'data' có thể nhận các kiểu dữ liệu sau: ndarray, series, map, lists, dict, hằng số và cả dataframe khác
  - 'items' là axis = 0
  - 'major_axis' là axis = 1
  - 'minor_axis' là axis = 2
  - 'dtype' là kiểu dữ liệu mỗi cột
  - 'copy' nhận giá trị True/False để khởi tạo dữ liệu có chia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

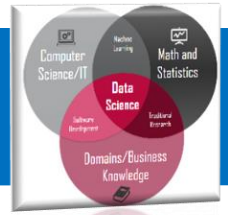
```
<class 'pandas.core.panel.Panel'>
Dimensions:   2   (items)   x   3   (major_axis)   x   4
(minor_axis)
Items axis: 0 to 1
Major_axis axis: 0 to 2
Minor_axis axis: 0 to 3
```




# Tạo panel

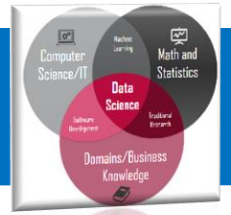
p.to_frame()

| | | 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 |




# Phần 8

# 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

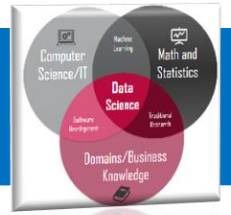

# Nhóm phần tử

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

| X | Y |
|---|---|
| A | 7 |
| B | 3 |

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

| X | Y |
|---|---|
| B | 3 |
| A | 7 |



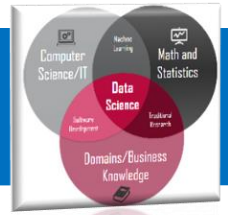

# Nhóm phần tử

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

| | X | Y |
|---|---|---|
| 0 | A | 1 |
| 2 | A | 3 |

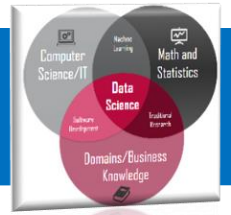
```
df3.groupby(['X']).get_group('B')
```

| | X | Y |
|---|---|---|
| 1 | B | 4 |
| 3 | B | 2 |



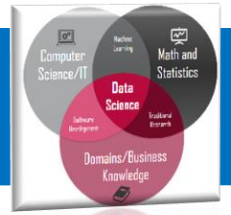

## Phần 9

# Sử dụng pandas trong bài toán thực tế



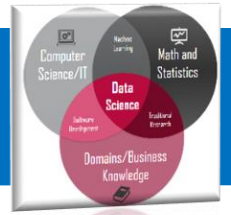

# Dữ liệu kết quả xổ số

- Dữ liệu kết quả xổ số (độc đắc) từ ngày 1-1-2000 đến ngày 21-5-2018 (hôm qua)
- 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 (vui + khoa học): phân tích các chiến lược chơi số đề mà người dân hay theo



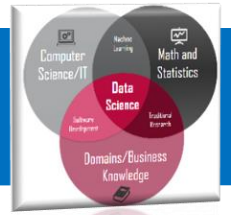

# Đọc và tiền xử lý dữ liệu

```
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 đắc
df['Cuoi'] = df.So % 100
```



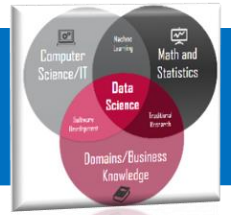

# Khảo sát dữ liệu

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

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

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




# Viết hàm tính số tiền thu về

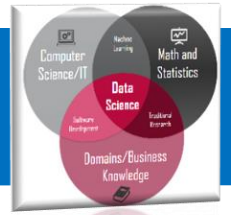
```
# thử bộ số myNums, kết quả về là result, số tiền chơi là money
def one_day(myNums, result, money):
pay = len(myNums) * money
get = money * 70 if result in myNums else 0
return get-pay

# chơi nhiều ngày bộ số myNums, kết quả về là results
def many_day(myNums, results, money):
total = 0
for x in results:
total += one_day(myNums, x, money)
return total
```




# Chiến lược: nuôi một số

money = 1000

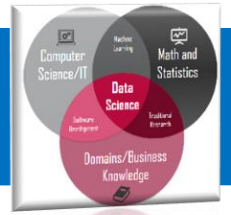
```
# thử chiến lược chơi: nuôi một con
print("Chơi con 76 toàn năm 2000:", many_day([76], s[0:367], money))
print("Chơi con 76 toàn bộ các năm:", many_day([76], s, money))
# thử chiến lược chơi: nuôi nhiều con
print("Nuôi nhiều số toàn năm 2000:", many_day([76, 92, 3, 10, 51, 45], s[0:367], money))
print("Nuôi nhiều số toàn bộ các năm:", many_day([76, 92, 3, 10, 51,45], s, money))
```




# Chiến lược: thống kê

# thống kê con ra nhiều nhất rồi chơi

```
x = s[0:362].value_counts().idxmax()
y = s.value_counts().idxmax()
print("Chơi theo số ra nhiều nhất năm 2000:", x,
many_day([x], s, money))
print("Chơi theo số ra nhiều nhất các năm:", y,
many_day([y], s, money))
```




# Chiến lược: ngẫu nhiên

# chơi ngẫu nhiên, mỗi ngày một con

```
total = 0
for d in s:
total -= money
m = np.random.randint(100)
if (m == d): total += 70 * money
print("Chơi ngẫu nhiên:", total)
```

# Thank You!

ThS. Phạm Đình Tài
0985.73.39.39
pdtai@ntt.edu.vn