


# IMD0033 - Probabilidade

## Aula 10 - Explorando dados com pandas

Ivanovitch Silva  
Setembro, 2018



# Agenda

---

- Select columns, rows and individual items using their integer location.
- Work with integer axis labels.
- How to use pandas methods to produce boolean arrays.
- Use boolean operators to combine boolean comparisons to perform more complex analysis.
- Use index labels to align data.
- Use aggregation to perform advanced analysis using loops.

# Atualizar o repositório

---

```
git clone https://github.com/ivanovitchm/imd0033_2018_2.git
```

Ou ....

```
git pull
```

# Introduction (Pandas vs Numpy)


```
ndarray[2,0]
```

*located at row 2, column 0*


```
ndarray[1]
```

*located at row 1*

Numpy

	A	B	C
x			
y			
z			

```
df.loc["z","A"]
```

*located at row with label z,  
column with label A*

	A	B	C
x			
y			
z			

```
df.loc["y"]
```

*located at row with label y*

Pandas

# Using iloc to select by integer position

	A	B	C
x			
y			
z			

```
df.iloc[2,0]
```

	A	B	C
x			
y			
z			

```
df.iloc[1]
```

```
first_column = f500.iloc[:,0]  
print(first_column)
```

```
0           Walmart  
1       State Grid  
2       Sinopec Group  
...  
497  Wm. Morrison Supermarkets  
498           TUI  
499       AutoNation  
Name: company, dtype: object
```

# Slicing with iloc

With `loc[]`, the ending slice is **included**.

With `iloc[]`, the ending slice is **not included**.

```
1 | f500[1:4].
```

	rank	revenues
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573

```
1 | f500.iloc[1:4].
```

	rank	revenues
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573

# Loc vs iLoc

`df.iloc[1]`

`iloc[1]` uses the integer position of the row to select the second row

	A	B	C
0			
1			
2			

`df.iloc[1]`

`iloc[1]` uses the integer position of the row to select the second row

	A	B	C
0			
2			
1			

`df.loc[1]`

`loc[1]` uses the label of the row to select the row with an axis label of `1`.

	A	B	C
0			
1			
2			

`df.loc[1]`

`loc[1]` uses the label of the row to select the row with an axis label of `1`.

	A	B	C
0			
2			
1			

# Lesson\_10\_Introduction\_to\_pandas.ipynb

## Up to section 1.4





# Using pandas methods to create boolean masks

---

```
>>> is_california = usa["hq_location"].str.endswith("CA")
```

```
>>> print(is_california.head())
```

```
0      False
```

```
7      False
```

```
8       True
```

```
9      False
```

```
10     True
```

```
Name: hq_location, dtype: bool
```

```
0      Bentonville, AR
```

```
7      Omaha, NE
```

```
8      Cupertino, CA
```

```
9      Irving, TX
```

```
10     San Francisco, CA
```

```
Name: hq_location, dtype: object
```

# Using boolean operators to select items

	company	revenues	country
0	Walmart	485873	USA
1	State Grid	315199	China
2	Sinopec Group	267518	China
3	China Nation...	262573	China
4	Toyota Motor	254694	Japan

f500\_sel

```
over_265 = f500_sel["revenues"] > 265000
china = f500_sel["country"] == "China"
```

0	True
1	True
2	True
3	False
4	False

over\_265

0	False
1	True
2	True
3	True
4	False

china

```
combined = over_265 & china
```

0	True	&	0	False	=	0	False
1	True	&	1	True	=	1	True
2	True	&	2	True	=	2	True
3	False	&	3	True	=	3	False
4	False	&	4	False	=	4	False

over\_265      china      combined

# Using boolean operators to select items

```
final_cols = ["company", "revenues"]  
result = f500_sel.loc[combined, final_cols]
```

		company	revenues	country			company	revenues
0	False	0	Walmart	485873	USA			
1	True →	1	State Grid	315199	China	→	1	State Grid
2	True →	2	Sinopec Group	267518	China	→	2	Sinopec Group
3	False	3	China Nation...	262573	China			
4	False	4	Toyota Motor	254694	Japan			

**combined**                      **f500\_sel**                      **result**

# Lesson\_10\_Introduction\_to\_pandas.ipynb

## Sections 1.5, 1.6, 1.7



# Pandas Index Alignment

	fruit_veg	qty		
tomato	fruit	4	corn	yellow
carrot	veg	2	carrot	orange
lime	fruit	4	tomato	red
corn	veg	1	lime	green
eggplant	veg	2	eggplant	purple
food			colors	

	fruit_veg	qty	color
tomato	fruit	4	red
carrot	veg	2	orange
lime	fruit	4	green
corn	veg	1	yellow
eggplant	veg	2	purple
food			

```
food["color"] = colors
```

	fruit_veg	qty		
tomato	fruit	4	corn	yellow
carrot	veg	2	carrot	orange
lime	fruit	4	tomato	red
corn	veg	1	lime	green
eggplant	veg	2	eggplant	purple
food			colors	

# Pandas Index Alignment

---

arugula	rocket
eggplant	aubergine
corn	maize

**alt\_name**

```
food["alt_name"] = alt_name
```

	fruit_veg	qty	color	alt_name
tomato	fruit	4	red	NaN
carrot	veg	2	orange	NaN
lime	fruit	4	green	NaN
corn	veg	1	yellow	maize
eggplant	veg	2	purple	aubergine

**food**

# Using Loops in Pandas

---

```
>>> print(df)
```

	A	B	C
x	6	1	0
y	1	8	8
z	3	8	7

```
>>> for i in df:  
    print(i)
```

A  
B  
C

Because one of the key benefits of pandas is that it has vectorized methods to work with data more efficiently, **we want to avoid using loops wherever we can**

# Challenge: calculating return on assets by sector

---

```
{'Aerospace & Defense': 'Lockheed Martin',
'Apparel': 'Nike',
'Business Services': 'Adecco Group',
'Chemicals': 'LyondellBasell Industries',
'Energy': 'National Grid',
'Engineering & Construction': 'Pacific Construction Group',
'Financials': 'Berkshire Hathaway',
'Food & Drug Stores': 'Publix Super Markets',
'Food, Beverages & Tobacco': 'Philip Morris International',
'Health Care': 'Gilead Sciences',
'Hotels, Restaurants & Leisure': 'McDonald\xe2\x80\x99s',
'Household Products': 'Unilever',
'Industrials': '3M',
'Materials': 'CRH',
'Media': 'Disney',
'Motor Vehicles & Parts': 'Subaru',
'Retailing': 'H & M Hennes & Mauritz',
'Technology': 'Accenture',
'Telecommunications': 'KDDI',
'Transportation': 'Delta Air Lines',
'Wholesalers': 'McKesson'}
```

$$\text{return on assets} = \frac{\text{profits}}{\text{assets}}$$



# Lesson\_10\_Introduction\_to\_pandas.ipynb

## Sections 1.8, 1.9, 1.10

