PYTHON

NUMPY AND PANDAS

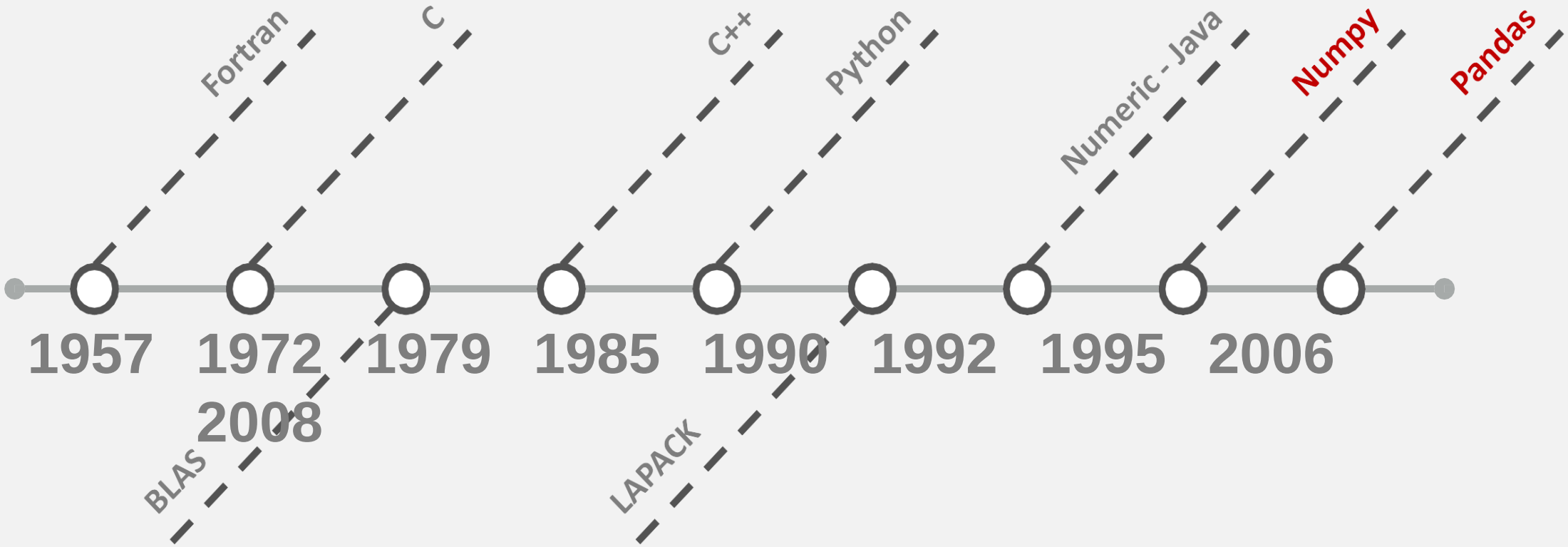
- NUMPY
- PANDAS
- NUMPY & PANDAS APPLICATIONS

HOMEWORK

- Assignment 1
- Assignment 2
- Assignment 3

I

NUMPY


Fortran
C
C++
Python
Numeric - Java
Numpy
Pandas
1957
1972
1979
1985
1990
1992
1995
2006
2008
BLAS
LAPACK

Introduction about NumPy

Numpy, abbreviation of Numerical Python

Introduction

NumPy is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

1. powerful N-dimensional array object,
2. sophisticated functions without loop like DAX
3. also be used as an efficient multi-dimensional container of generic data for reading and writing array data type
4. useful linear algebra, Fourier transform, and random number capabilities
5. integrating C/C++ and Fortran code

1

Numpy basics

NUMPY BASICS

- Multidimentional array - narray
- Functions of arrays
- Read file/ load file
- Random generators
- Case study: Random walk

Multidimentional array - narray

Cấu trúc dữ liệu cho kết quả xử lý nhanh, linh hoạt với tập dữ liệu lớn

3	4	7	8	1	2	3

Columns →

Rows ↓	0	1	2	3	4
0	5	12	17	9	3
1	13	4	8	14	1
2	9	6	3	7	21

2D Array of size 3 x 5

Để sử dụng thư viện NumPy:

```
import numpy as np
```

Khai báo một narray:

```
arr1 = np.array([3, 4, 7, 8, 1, 2, 3])
```

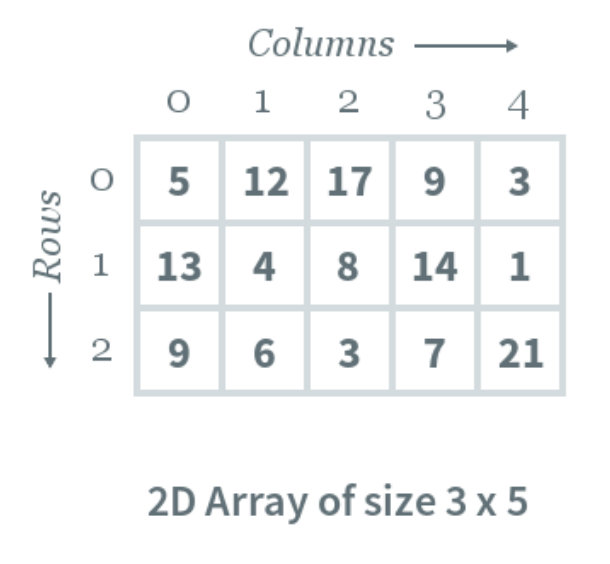
```
arr = np.array([[5, 12, 17, 9, 3], [13, 4, 8, 14, 1], [9, 6, 3, 7, 21]])
```

Hiển thị:

```
arr
```

```
array([[ 5, 12, 17,  9,  3],
       [13,  4,  8, 14,  1],
       [ 9,  6,  3,  7, 21]])
```

Multidimentional array - narray



Kiểm tra các thuộc tính của mảng:

Kích thước:

```
arr.shape
```

```
(3, 5)
```

Kiểu dữ liệu của các phần tử

```
arr.dtype
```

```
dtype('int32')
```

- **Lưu ý:** Các phần tử của mảng phải cùng một kiểu dữ liệu

Multidimentional array - narray

Thay đổi các thuộc tính của mảng:

Thay đổi kích thước

```
arr.reshape([5, 3])
```

```
array([[ 5, 12, 17],
       [ 9,  3, 13],
       [ 4,  8, 14],
       [ 1,  9,  6],
       [ 3,  7, 21]])
```

Thay đổi kiểu dữ liệu

```
arr.astype(np.float32)
```

```
array([[ 5., 12., 17.,  9.,  3.],
       [13.,  4.,  8., 14.,  1.],
       [ 9.,  6.,  3.,  7., 21.]], dtype=float32)
```

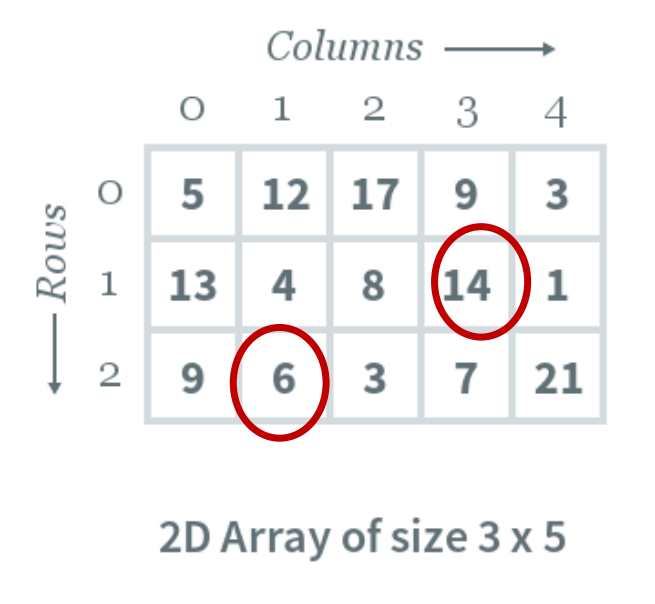
Multidimentional array - narray

Truy cập phần tử của mảng

3	4	7	8	1	2	3

```
arr1[3]
```

```
8
```




```
arr[1, 3]
```

```
14
```

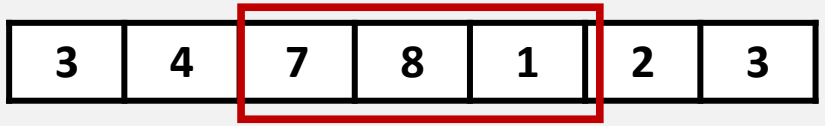
```
arr[2][1]
```

```
6
```

1

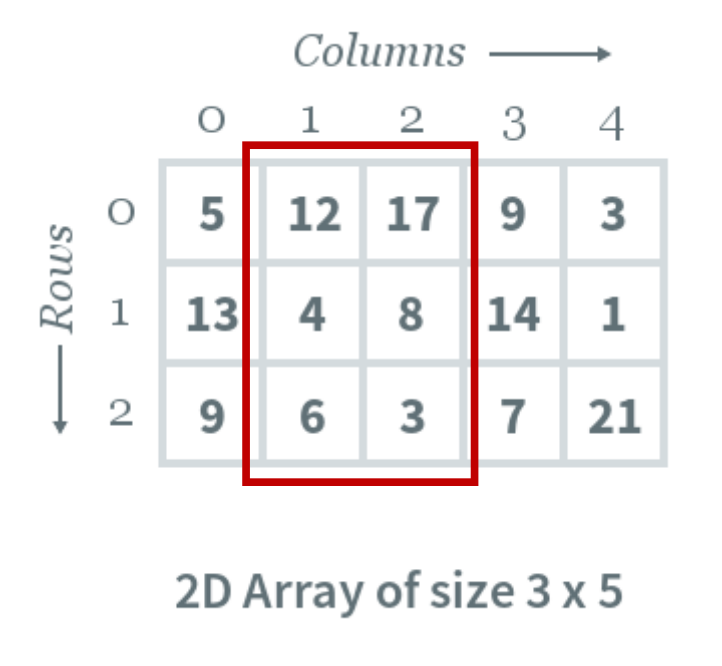
Multidimentional array - narray

Truy cập phần tử của mảng




```
arr1[2:5]
```

```
array([7, 8, 1])
```




```
arr[:, 1:3]
```

```
array([[12, 17],
       [ 4,  8],
       [ 6,  3]])
```

Multidimentional array - narray

List

- Các phần tử có thể có nhiều kiểu khác nhau.
- Thao tác trên từng biến.
- Các lát cắt thay đổi trên dữ liệu copy
- Có thể thay đổi số lượng phần tử

Mảng 1 chiều

- Các phần tử cùng kiểu
- Thao tác trên các lát cắt
- Các lát cắt thay đổi trên dữ liệu ban đầu
- Không thể thay đổi số lượng phần tử

Arithmetic Operators in arrays

Các phép toán biến đổi từng phần tử của ma trận

Function	Description
`abs, fabs`	Compute the absolute value element-wise for integer, floating point, or complex values. Use `fabs` as a faster alternative for non-complex-valued data
`sqrt`	Compute the square root of each element. Equivalent to `arr ** 0.5`
`square`	Compute the square of each element. Equivalent to `arr ** 2`
`exp`	Compute the exponent e^x of each element
`log, log10, log2, log1p`	Natural logarithm (base *e*), log base 10, log base 2, and log(1 + x), respectively
`sign`	Compute the sign of each element: 1 (positive), 0 (zero), or -1 (negative)
`ceil`	Compute the ceiling of each element, i.e. the smallest integer greater than or equal to each element
`floor`	Compute the floor of each element, i.e. the largest integer less than or equal to each element

Arithmetic Operators arrays

Các phép toán hai ngôi

Function	Description
add	Add corresponding elements in arrays
subtract	Subtract elements in second array from first array
multiply	Multiply array elements
divide, floor_divide	Divide or floor divide (truncating the remainder)
power	Raise elements in first array to powers indicated in second array
maximum, fmax	Element-wise maximum. fmax ignores NaN
minimum, fmin	Element-wise minimum. fmin ignores NaN
mod	Element-wise modulus (remainder of division)
copysign	Copy sign of values in second argument to values in first argument

Operators in arrays

Biểu diễn điều kiện logic

Đề bài: Cho ma trận a và b như sau:

```
a_array = np.array([1, 3, 4, 2, 5])
```

```
b_array = np.array([4, 5, 7, 2, 1])
```

Trả ra một ma trận thỏa mãn điều kiện sau: nếu phần tử của b lớn hơn phần tử của a (tương ứng theo từng index) thì trả ra kết quả True, nếu không trả ra False.

np.where(cond, True_value, False_value)

Sorting

array.sort()

Read & Loading file

Lưu ma trận vào file

```
np.save('arr_save.npy', arr)
```

Đọc ma trận từ file

```
arr = np.load('arr_save.npy')
```

Đọc và ghi ma trận từ file text

```
arr = np.loadtxt('array_text.txt', delimiter=',')
```

```
np.savetxt('array_text.txt', arr, delimiter=',')
```

Random Generators

np.random module

Function	Description
seed	Seed the random number generator
permutation	Return a random permutation of a sequence, or return a permuted range
shuffle	Randomly permute a sequence in place
rand	Draw samples from a uniform distribution
randint	Draw random integers from a given low-to-high range
randn	Draw samples from a normal distribution with mean 0 and standard deviation 1 (MATLAB-like interface)
binomial	Draw samples a binomial distribution
normal	Draw samples from a normal (Gaussian) distribution
beta	Draw samples from a beta distribution
chisquare	Draw samples from a chi-square distribution
gamma	Draw samples from a gamma distribution
uniform	Draw samples from a uniform [0, 1) distribution

Random walk

Đề bài: Một người đi trên đường dựa vào quy tắc như sau: Nếu tung đồng xu ra mặt ngửa, anh ta sẽ tiến 1 bước. Nếu tung đồng xu ra mặt sấp, anh ta sẽ lùi 1 bước. Giả sử xác suất ra mặt sấp và mặt ngửa là như nhau. Mỗi lần chơi anh ta được tung đồng xu 100 lần. Tính xác suất anh ta tiến được 30 bước.

II

PANDAS

Introduction about Pandas

Introduction

pandas is a high-level package supporting analytical tools to analyze data in a very fast way in Python.

To use this package, you need to declare as below:

```
import pandas as pd
```

1. Providing a dataframe with column name & index
2. Integrated functions and timeseries
3. Easy to manage and handle missing data
4. Perform operators as database
5. Math operators and decrease data size

Introduction about Pandas

- used for high-performance data manipulation and data analysis using its powerful data structures.
- Popular in Finance, Economics, Statistics, Advertising, Web Analytics, and more.
- accomplish five typical steps in the processing and analysis of data, regardless of the origin of data:
  - Load
  - Organize
  - Manipulate
  - Model
  - Analyse the data

Key features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

Data structure - Series

Series là bao gồm một ma trận giá trị 1 chiều và một ma trận nhãn tương ứng với nó.

Index	Data
1	'A'
2	'B'
3	'C'
4	'D'
5	'E'

Khai báo một DataFrame

```
In [40]: ser = pd.Series(['A', 'B', 'C', 'D', 'E'], index=[1, 2, 3, 4, 5])
```

```
In [41]: ser

Out[41]: 1    A
         2    B
         3    C
         4    D
         5    E
         dtype: object
```

Data Type - Series

Series là bao gồm một ma trận giá trị 1 chiều và một ma trận nhãn tương ứng với nó.

Khai báo một DataFrame thông qua một dictionary

```
In [55]: sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}
```

```
In [56]: ser2 = pd.Series(sdata)
```

```
In [57]: ser2

Out[57]: Ohio      35000
         Texas     71000
         Oregon    16000
         Utah       5000
         dtype: int64
```

Data structure - Series

Series là bao gồm một ma trận giá trị 1 chiều và một ma trận nhãn tương ứng với nó.

Khai báo một DataFrame thông qua một dictionary

```
In [58]: ser2 = pd.Series(sdata, index=["Ohio", "Texas", "Cali"])

In [59]: ser2

Out[59]: Ohio     35000.0
         Texas    71000.0
         Cali         NaN
         dtype: float64
```

Data structure - Series

Cách truy cập phần tử trong Series

Truy cập đến một phần tử

```
In [51]: ser[1]

Out[51]: 'A'
```

Truy cập nhiều phần tử

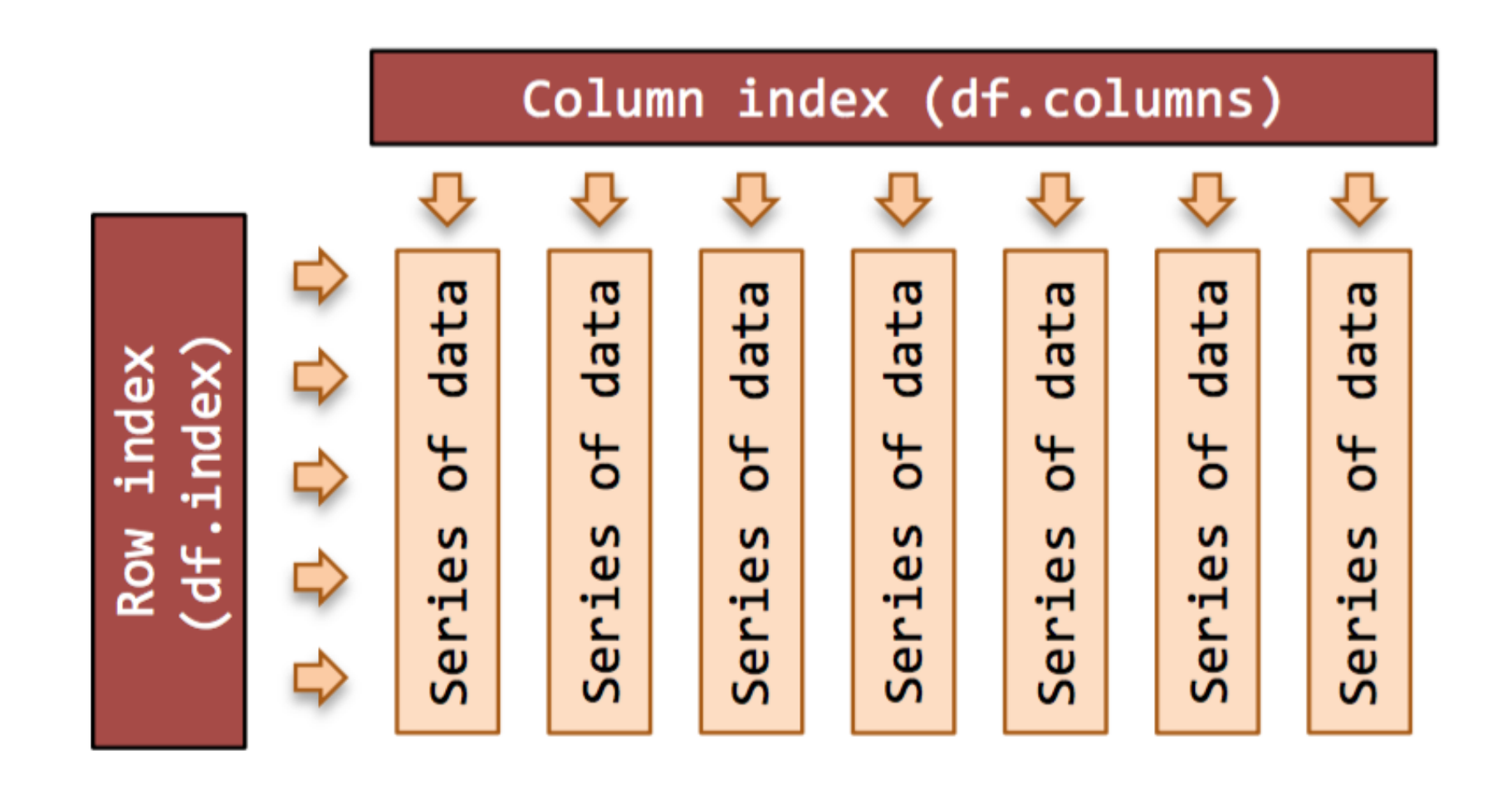
```
In [52]: ser[[1, 2, 3]]

Out[52]: 1    A
         2    B
         3    C
         dtype: object
```

Truy cập theo điều kiện

```
In [54]: ser[ser == 'A']

Out[54]: 1    A
         dtype: object
```

Data structure - DataFrame

DataFrame biểu diễn dữ liệu dạng bảng, nó bao gồm các cột được sắp thứ tự, mỗi cột có một kiểu giá trị khác nhau.



Data structure - DataFrame

Khai báo một DataFrame

Khai báo thông qua một dictionary, với keys là tên cột và values là list các giá trị của cột đó.

```
In [2]: data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
                'year': [2000, 2001, 2002, 2001, 2002],
                'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
```

```
In [3]: frame = pd.DataFrame(data)
```

```
In [4]: frame
```

Out[4]:

	state	year	pop
0	Ohio	2000	1.5
1	Ohio	2001	1.7
2	Ohio	2002	3.6
3	Nevada	2001	2.4
4	Nevada	2002	2.9

Data structure - DataFrame

Khai báo một DataFrame

Có thể khai báo cụ thể các cột và index của DataFrame bằng cách bổ sung các tham số. Trong trường hợp cột được thêm vào không có dữ liệu, giá trị của cột đó bằng NaN.

```
In [2]: data = {'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
                'year': [2000, 2001, 2002, 2001, 2002],
                'pop': [1.5, 1.7, 3.6, 2.4, 2.9]}
```

```
In [5]: frame_1 = pd. DataFrame(data, columns=['year', 'state', 'pop', 'debt'],
                                index=['one', 'two', 'three', 'four', 'five'])
```

```
In [6]: frame_1
```

Out[6]:

	year	state	pop	debt
one	2000	Ohio	1.5	NaN
two	2001	Ohio	1.7	NaN
three	2002	Ohio	3.6	NaN
four	2001	Nevada	2.4	NaN
five	2002	Nevada	2.9	NaN

Data structure - DataFrame

Khai báo một DataFrame

Type	Notes
2D ndarray	A matrix of data, passing optional row and column labels
dict of arrays, lists, or tuples	Each sequence becomes a column in the DataFrame. All sequences must be the same length.
NumPy structured/record array	Treated as the "dict of arrays" case
dict of Series	Each value becomes a column. Indexes from each Series are unioned together to form the result's row index if no explicit index is passed.
dict of dicts	Each inner dict becomes a column. Keys are unioned to form the row index as in the "dict of Series" case.
list of dicts or Series	Each item becomes a row in the DataFrame. Union of dict keys or Series indexes become the DataFrame's column labels
List of lists or tuples	Treated as the "2D ndarray" case
Another DataFrame	The DataFrame's indexes are used unless different ones are passed
NumPy MaskedArray	Like the "2D ndarray" case except masked values become NA/missing in the DataFrame result

Data structure - DataFrame

Truy cập phần tử trong DataFrame

Chọn một cột trong DataFrame

```
In [11]: frame_1['year']
```

```
Out[11]: one      2000
         two      2001
         three    2002
         four     2001
         five     2002
         Name: year, dtype: int64
```

Chọn một hàng trong DataFrame

```
In [21]: frame_1.loc["three"]
```

```
Out[21]: year      2002
         state     Ohio
         pop        3.6
         debt       NaN
         Name: three, dtype: object
```

Chọn nhiều cột cùng lúc

```
In [18]: frame_1[['year', 'pop']]
```

Out[18]:

	year	pop
one	2000	1.5
two	2001	1.7
three	2002	3.6

Chọn nhiều hàng trong DataFrame

```
In [22]: frame_1.loc[["three", "two"]]
```

Out[22]:

	year	state	pop	debt
three	2002	Ohio	3.6	NaN
two	2001	Ohio	1.7	NaN

Data structure - DataFrame

Khai báo giá trị trong hàng hoặc cột

- Bằng một giá trị (cả cột/ hàng đều bằng giá trị đó)

```
In [27]: frame['year'] = 1996
```

- Bằng một list có độ dài bằng số dòng

```
In [29]: frame['year'] = [1996, 1997, 1998, 1999, 2000]
```

- Bằng một Series, không nhất thiết cùng độ dài, những index nào có trong Series sẽ được gán giá trị mới, những index nào không được gán sẽ nhận giá trị NaN.

```
In [37]: frame['year'] = pd.Series([1996, 1998, 2001], index=[0, 1, 3])
```

	state	year	pop
0	Ohio	1996.0	1.5
1	Ohio	1998.0	1.7
2	Ohio	NaN	3.6
3	Nevada	2001.0	2.4
4	Nevada	NaN	2.9

Data structure - Index Object

Index Object kiểm soát nhãn của các trục và một số thông tin khác (tên trục).

```
In [60]: ser2.index
Out[60]: Index(['Ohio', 'Texas', 'Cali'], dtype='object')
```

```
In [61]: frame_1.index
Out[61]: Index(['one', 'two', 'three', 'four', 'five'], dtype='object')
```

Lưu ý: Các giá trị trong index object không thay đổi được

Data structure - Index Object

Table 5-4. Indexing options with DataFrame

Type	Notes
`df[val]`	Select single column or sequence of columns from the DataFrame; special case conveniences: boolean array (filter rows), slice (slice rows), or boolean DataFrame (set values based on some criterion)
`df.loc[val]`	Selects single row or subset of rows from the DataFrame by label
`df.loc[:, val]`	Selects single column or subset of columns by label
`df.loc[val1, val2]`	Select both rows and columns by label
`df.iloc[where]`	Selects single row or subset of rows from the DataFrame by integer position

Type	Notes
`df.iloc[:, where]`	Selects single column or subset of columns by integer position
`df.iloc[where_i, where_j]`	Select both rows and columns by integer position
`df.at[label_i, label_j]`	Select a single scalar value by row and column label
`df.iat[i, j]`	Select a single scalar value by row and column position (integers)
`reindex` method	Select either rows or columns by labels
`get_value`, `set_value` methods	Select single value by row and column label

Some basic functions

Reindexing

frame.reindexing()

Delete one row or column in DataFrame

frame.drop()

Handling missing data

frame.dropna()

frame.fillna()

Hàm apply và mapping

frame.apply()

Sorting and ranking

frame.sort_index()

frame.rank()

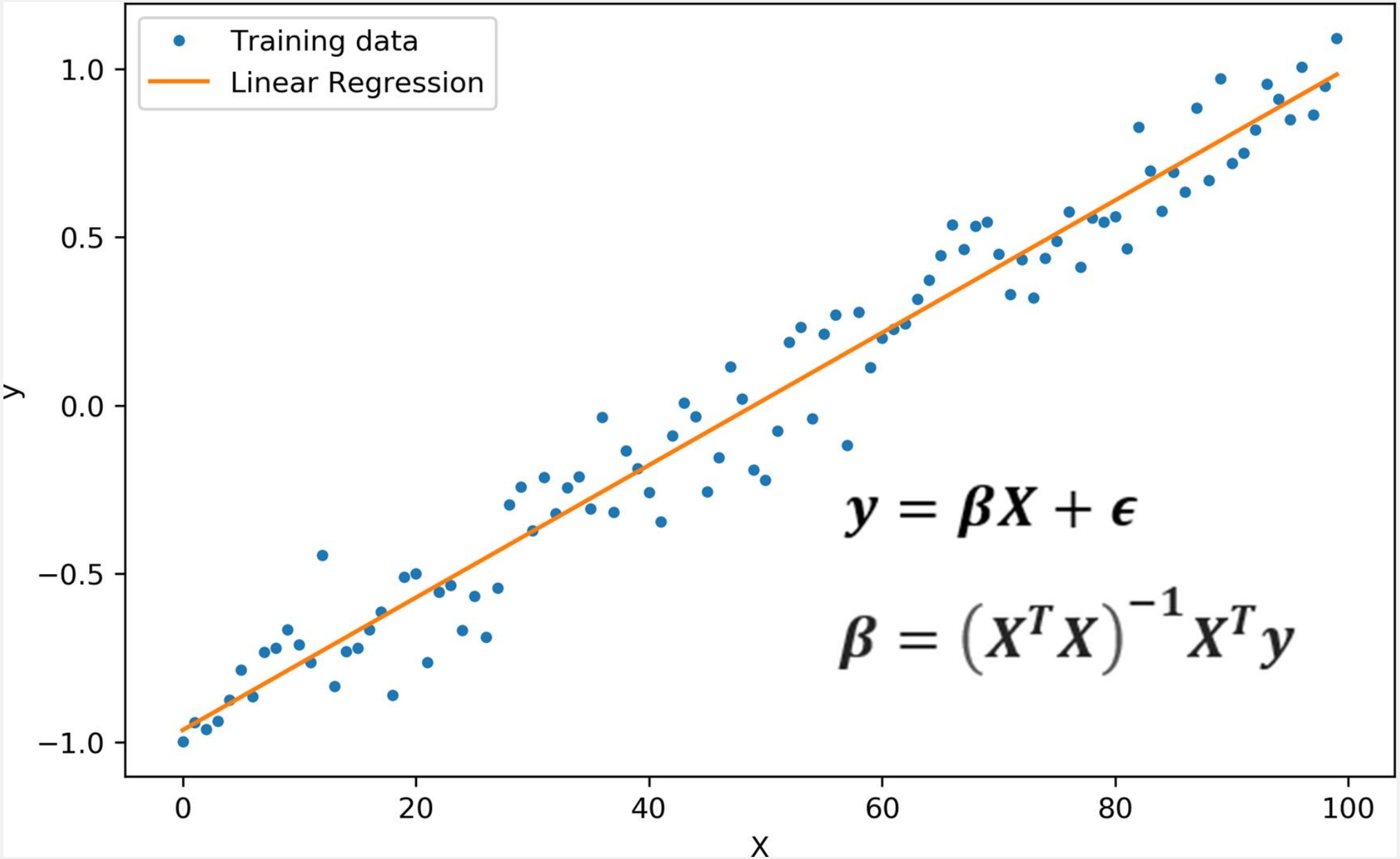
III

NUMPY & PANDAS APPLICATION

Multiple Linear Regression with Numpy

3

NUMPY & PANDAS APPLICATION



$$y = \beta X + \epsilon$$

$$\beta = \left(X^T X\right)^{-1} X^T y$$

NUMPY & PANDAS APPLICATION

```
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

```
boston = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv')
boston.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

Basic API

```
estimator.fit(X, [y])
```

estimator.predict	estimator.transform
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

3

NUMPY & PANDAS APPLICATION

```
X = boston.drop('medv', axis=1).values
y = boston['medv'].values
```

```
model = OrdinaryLeastSquares()
```

```
model.fit(X, y)
```

```
model.coefficients
```

```
array([ 3.64594884e+01, -1.08011358e-01,  4.64204584e-02,  2.05586264e-02,
        2.68673382e+00, -1.77666112e+01,  3.80986521e+00,  6.92224640e-04,
       -1.47556685e+00,  3.06049479e-01, -1.23345939e-02, -9.52747232e-01,
        9.31168327e-03, -5.24758378e-01])
```

```
model.predict(X[0])
```

```
30.003843377016945
```

```
y_preds = []

for row in X: y_preds.append(model.predict(row))
```

```
pd.DataFrame({
    'Actual': y,
    'Predicted': np.ravel(y_preds)
})
```

	Actual	Predicted
0	24.0	30.003843
1	21.6	25.025562
2	34.7	30.567597
3	33.4	28.607036
4	36.2	27.943524
...	...	...
501	22.4	23.533341
502	20.6	22.375719
503	23.9	27.627426
504	22.0	26.127967
505	11.9	22.344212

506 rows × 2 columns

```python
class OrdinaryLeastSquares(object):

    def __init__(self):
        self.coefficients = []

    def fit(self, X, y):
        if len(X.shape) == 1: X = self._reshape_x(X)

        X = self._concatenate_ones(X)
        self.coefficients = np.linalg.inv(X.transpose().dot(X)).dot(X.transpose()).dot(y)

    def predict(self, entry):
        b0 = self.coefficients[0]
        other_betas = self.coefficients[1:]
        prediction = b0

        for xi, bi in zip(entry, other_betas): prediction += (bi * xi)
        return prediction

    def _reshape_x(self, X):
        return X.reshape(-1, 1)

    def _concatenate_ones(self, X):
        ones = np.ones(shape=X.shape[0]).reshape(-1, 1)
        return np.concatenate((ones, X), 1)
```

Data Processing with Pandas

3

Using Breast Cancer Dataset:

Original Wisconsin Breast Cancer Database

NUMPY & PANDAS APPLICATION

- **Data exploration** — columns, unique values in a column, describe, duplicates
- **Dealing with missing values** — quantifying missing values per column, filling & dropping missing values
- **Reshaping data** — one hot encoding, pivot tables, joins, grouping and aggregating
- **Filtering data**
- **Other** — Making descriptive columns, element-wise conditional operations

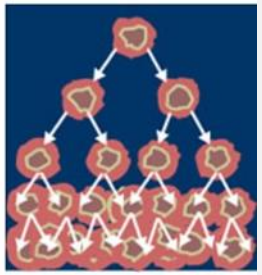
3 Overview:

```
import pandas as pd

filename = 'data/breast_cancer_data.csv'
df = pd.read_csv(filename)
df.dtypes

patient_id                int64
clump_thickness         float64
cell_size_uniformity    float64
cell_shape_uniformity     int64
marginal_adhesion         int64
single_ep_cell_size       int64
bare_nuclei              object
bland_chromatin         float64
normal_nucleoli         float64
mitoses                   int64
class                    object
doctor_name              object
dtype: object
```

Breast Cancer Wisconsin (Original) Data Set

Download: Data Folder, Data Set Description

Abstract: Original Wisconsin Breast Cancer Database



Data Set Characteristics:	Multivariate	**Number of Instances:**	699	**Area:**	Life
Attribute Characteristics:	Integer	**Number of Attributes:**	10	**Date Donated**	1992-07-15
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	573481

- Patient ID: *id number*
- Clump Thickness: *1–10*
- Uniformity of Cell Size: *1–10*
- Uniformity of Cell Shape: *1–10*
- Marginal Adhesion: *1–10*
- Single Epithelial Cell Size: *1–10*
- Bare Nuclei: *1–10*
- Bland Chromatin: *1–10*
- Normal Nucleoli: *1–10*
- Mitoses: 1–10
- Class: *malignant or benign*
- Doctor name: *4 different doctors.*

Overview:


```
df.describe()
```

	patient_id	clump_thickness	cell_size_uniformity	cell_shape_uniformity
count	6.990000e+02	698.000000	698.000000	699.000000
mean	1.071704e+06	4.416905	3.137536	3.207439
std	6.170957e+05	2.817673	3.052575	2.971913
min	6.163400e+04	1.000000	1.000000	1.000000
25%	8.706885e+05	2.000000	1.000000	1.000000
50%	1.171710e+06	4.000000	1.000000	1.000000
75%	1.238298e+06	6.000000	5.000000	5.000000
max	1.345435e+07	10.000000	10.000000	10.000000

3

Group by Categorical Data:

```
df.groupby(by =['class', 'doctor_name']).size()
```

```
class      doctor_name
benign     Dr. Doe        127
           Dr. Lee        121
           Dr. Smith      102
           Dr. Wong       108
malignant  Dr. Doe         58
           Dr. Lee         60
           Dr. Smith       74
           Dr. Wong        49
dtype: int64
```

Dealing with missing values:

```
df.isna().sum()
```

```
patient_id                 0
clump_thickness            1
cell_size_uniformity       1
cell_shape_uniformity      0
marginal_adhesion          0
single_ep_cell_size        0
bare_nuclei                2
bland_chromatin            4
normal_nucleoli            1
mitoses                    0
class                      0
doctor_name                0
dtype: int64
```

```
df = df.dropna(axis = 0, how = 'any    #Drop rows with any missing values
```

3 Inspecting Duplicates:

```
df.nunique()
```

Some patients appear more than once

```
patient_id                 645
clump_thickness             10
cell_size_uniformity        10
cell_shape_uniformity       10
marginal_adhesion           10
single_ep_cell_size         10
bare_nuclei                 11
bland_chromatin             10
normal_nucleoli             10
mitoses                      9
class                        2
doctor_name                  4
dtype: int64
```

3 Inspecting Duplicates:

```
df[df.duplicated(subset = 'patient_id', keep
=False)].sort_values('patient_id')
```

	patient_id	clump_thickness	cell_size_uniformity	cell_shape_uniformity	marginal_adhesion	single_ep_cell_size	bare_nuclei	bland_chromatin	normal_nucleoli	mitoses	class	doctor_name
267	320675	3.0	3.0	5	2	3	10	7.0	1.0	1	malignant	Dr. Wong
272	320675	3.0	3.0	5	2	3	10	7.0	1.0	1	malignant	Dr. Smith
575	385103	5.0	1.0	2	1	2	1	3.0	1.0	1	benign	Dr. Smith
269	385103	1.0	1.0	1	1	2	1	3.0	1.0	1	benign	Dr. Doe

pandas.DataFrame.duplicated

`DataFrame.duplicated`(*self, subset: Union[Hashable, Sequence[Hashable], NoneType] = None, keep: Union[str, bool] = 'first'*) → 'Series' [source]

Return boolean Series denoting duplicate rows.

Considering certain columns is optional.

Parameters: **subset** : *column label or sequence of labels, optional*

Only consider certain columns for identifying duplicates, by default use all of the columns.

keep : *{'first', 'last', False}, default 'first'*

Determines which duplicates (if any) to mark.

- `first` : Mark duplicates as `True` except for the first occurrence.
- `last` : Mark duplicates as `True` except for the last occurrence.
- False : Mark all duplicates as `True`.

Returns: Series

Inspecting Duplicates:


```
repeat_patients = df.groupby(by =
'patient_id').size().sort_values(ascending =False)
```

```
patient_id
1182404    6
1276091    5
1105524    2
1299596    2
385103     2
734111     2
411453     2
1143978    2
1218860    2
822829     2
1240603    2
```

Without the tilde ("~") we would get all individuals that repeat more than twice. By adding a tilde the pandas boolean series is reversed and thus the resulting data frame is of those that do NOT repeat more than twice.

```
filtered_patients = repeat_patients[repeat_patients >
2].to_frame().reset_index()

filtered_df = df[~df.patient_id.isin(filtered_patients.patient_id)]
```

pandas.pivot_table

`pandas.pivot_table`(*data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False*) → 'DataFrame' [source]

Create a spreadsheet-style pivot table as a DataFrame.

The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.

pandas.DataFrame.merge

`DataFrame.merge`(*self, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True, indicator=False, validate=None*) → 'DataFrame'

Merge DataFrame or named Series objects with a database-style join. [source]

The join is done on columns or indexes. If joining columns on columns, the DataFrame indexes *will be ignored*. Otherwise if joining indexes on indexes or indexes on a column or columns, the index will be passed on.

3 Reshaping Data – Pivoting & Merging:

creating a new dataframe with the categorical data.

```
categorical_df = df[['patient_id','doctor_name']]
categorical_df['doctor_count'] = 1
```

Pivot this table so that we only have numerical values in the cells and the columns become the doctors' name. Then fill in the empty cells with 0.

```
doctors_one_hot_encoded = pd.pivot_table( categorical_df,
                                          index = categorical_df.index,
                                          columns = ['doctor_name'],
                                           values = ['doctor_count'] )

doctors_one_hot_encoded = doctors_one_hot_encoded.fillna(0)
```

Reshaping Data – Pivoting & Merging:

drop the multiIndex columns

```
doctors_one_hot_encoded.columns =
doctors_one_hot_encoded.columns.droplevel()
```

doctor_name	Dr. Doe	Dr. Lee	Dr. Smith	Dr. Wong
0	1.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0
2	0.0	1.0	0.0	0.0
3	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0
5	0.0	0.0	1.0	0.0
6	1.0	0.0	0.0	0.0
7	0.0	0.0	1.0	0.0
8	0.0	0.0	1.0	0.0
9	1.0	0.0	0.0	0.0

Reshaping Data – Pivoting & Merging:

join this back to our main table

```
combined_df = pd.merge(df, one_hot_encoded, left_index =
True,right_index =True, how ='left')
```

single_ep_cell_size	bare_nuclei	bland_chromatin	normal_nucleoli	mitoses	class	doctor_name	Dr. Doe	Dr. Lee	Dr. Smith	Dr. Wong
2	1	3.0	1.0	1	benign	Dr. Doe	1.0	0.0	0.0	0.0
7	10	3.0	2.0	1	benign	Dr. Smith	0.0	0.0	1.0	0.0
2	2	3.0	1.0	1	benign	Dr. Lee	0.0	1.0	0.0	0.0
3	4	3.0	7.0	1	benign	Dr. Smith	0.0	0.0	1.0	0.0
2	1	3.0	1.0	1	benign	Dr. Wong	0.0	0.0	0.0	1.0
7	10	9.0	7.0	1	malignant	Dr. Smith	0.0	0.0	1.0	0.0
2	10	3.0	1.0	1	benign	Dr. Doe	1.0	0.0	0.0	0.0

```
combined_df = combined_df.drop(columns=['doctor_name']
```

Row-wise Operations - Example:

create a new column that categorizes a patients cell as normal or abnormal based on its attributes

```
def celltypelabel(x):

    if ((x['cell_size_uniformity'] > 5) &
(x['cell_shape_uniformity'] > 5)):

        return('normal')

    else:
        return('abnormal')

combined_df['cell_type_label'] = combined_df.apply(lambda x:
celltypelabel(x), axis=1)
```

III

HOMEWORK

1

Assignment 01

HOMEWORK

1

Assignment 02

HOMEWORK

1

Assignment 03

HOMEWORK

THANKS FOR LISTENING!!!