INTRODUCTION TO DATA MINING

Week01

What is Data Mining?

- Data mining
  - Data mining is the process of discovering patterns, relationships, anomalies, or useful insights from large datasets
  - It involves statistical techniques, database management, and machine learning algorithms to extract meaningful information

- Key objectives of data mining
  - Identify hidden patterns and trends in data
  - Support decision-making through predictive and descriptive analytics
  - Improve business intelligence and automation

What is Data Mining?

Data Mining is

Discovering patterns or extracting information — Action

from data — Resource

to utilize results for further use — Purpose

Are Machine Learning and Data Mining Different?

- Machine learning
  - Machine learning (ML) is a subset of AI that enables computers to learn patterns from data and make predictions without explicit programming.
  - Unlike traditional rule-based systems, ML algorithms improve performance over time as they process more data.

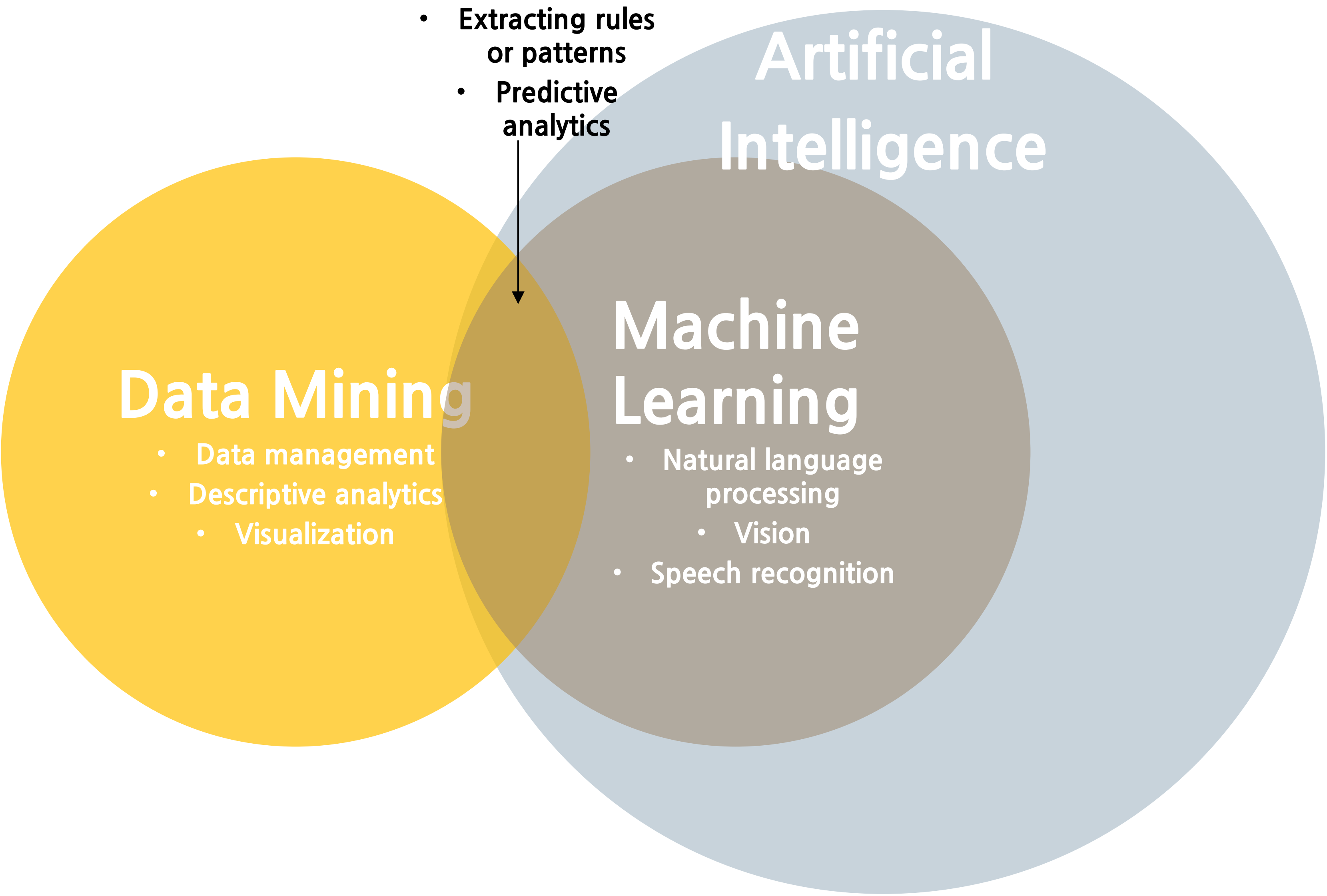
- Key objectives of machine learning
  - Train models to recognize patterns in data
  - Make accurate predictions for unseen data
  - Automate decision-making based on learned patterns

What is Artificial Intelligence?

- Artificial intelligence
  - Artificial intelligence (AI) is a broad field of computer science that aims to create intelligent systems capable of mimicking human cognition, including learning, reasoning, and problem-solving.
  - Machine learning and data mining are part of AI, but AI encompasses more than just these techniques.

- Key objectives of artificial intelligence
  - Develop intelligent systems that can make autonomous decisions
  - Simulate human cognitive abilities such as reasoning, perception, and language processing
  - Enable automation across various industries

What is Data Mining?



- Extracting rules or patterns
- Predictive analytics

Artificial Intelligence

Machine Learning

- Natural language processing
- Vision
- Speech recognition

Data Mining

- Data management
- Descriptive analytics
- Visualization

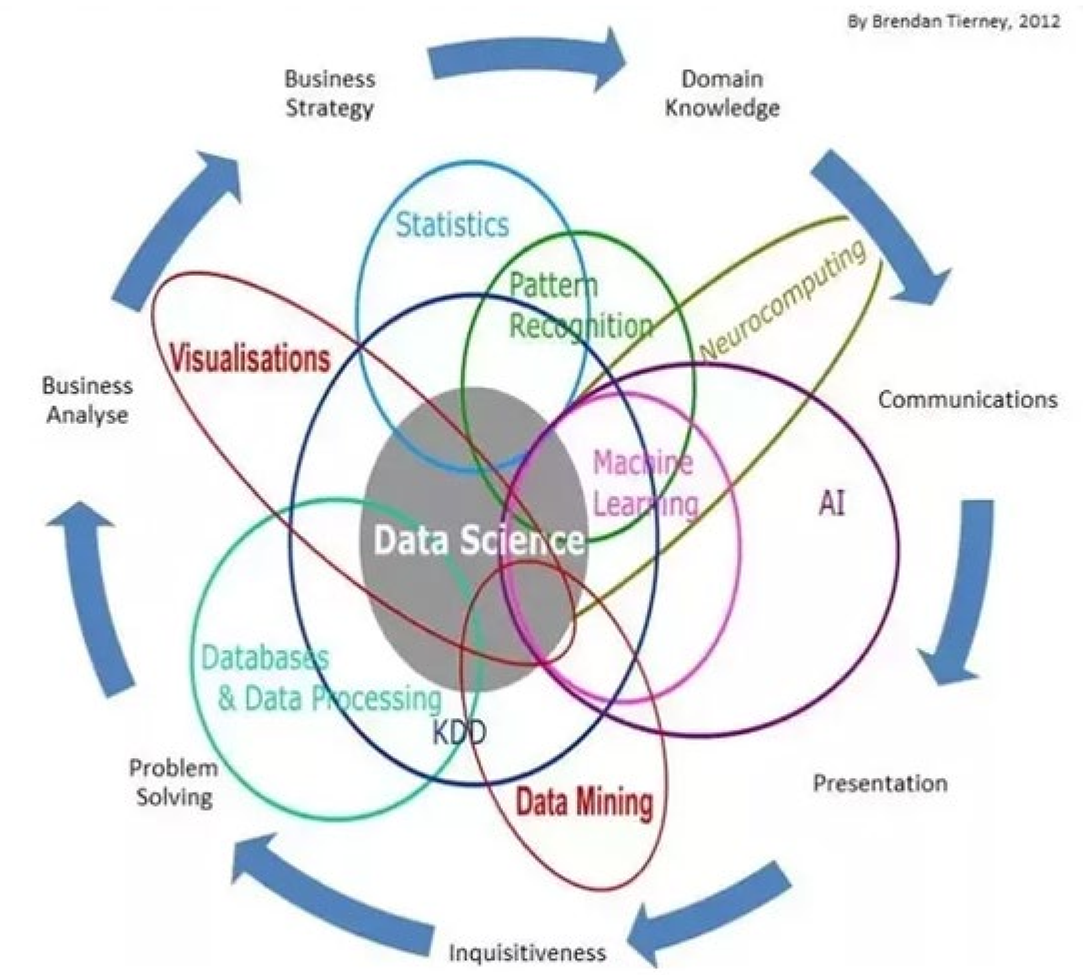
Data Mining vs. Machine Learning vs. Artificial Intelligence

- Comparison

Feature	Data Mining	Machine Learning	Artificial Intelligence
Definition	Extracting patterns and knowledge from data	Training models to learn from data and make predictions	Creating intelligent systems that mimic human cognition
Main Goal	Discover hidden relationships in data	Enable computers to learn and adapt without explicit programming	Automate reasoning, learning, and problem-solving
Methods Used	Statistical analysis, rule-based systems, machine learning	Supervised, unsupervised, and reinforcement learning	Machine learning, Neural networks (deep learning), expert systems
Relationship	Uses ML techniques to analyze data	A subset of AI focused on learning patterns	The broadest concept, encompassing ML and data mining

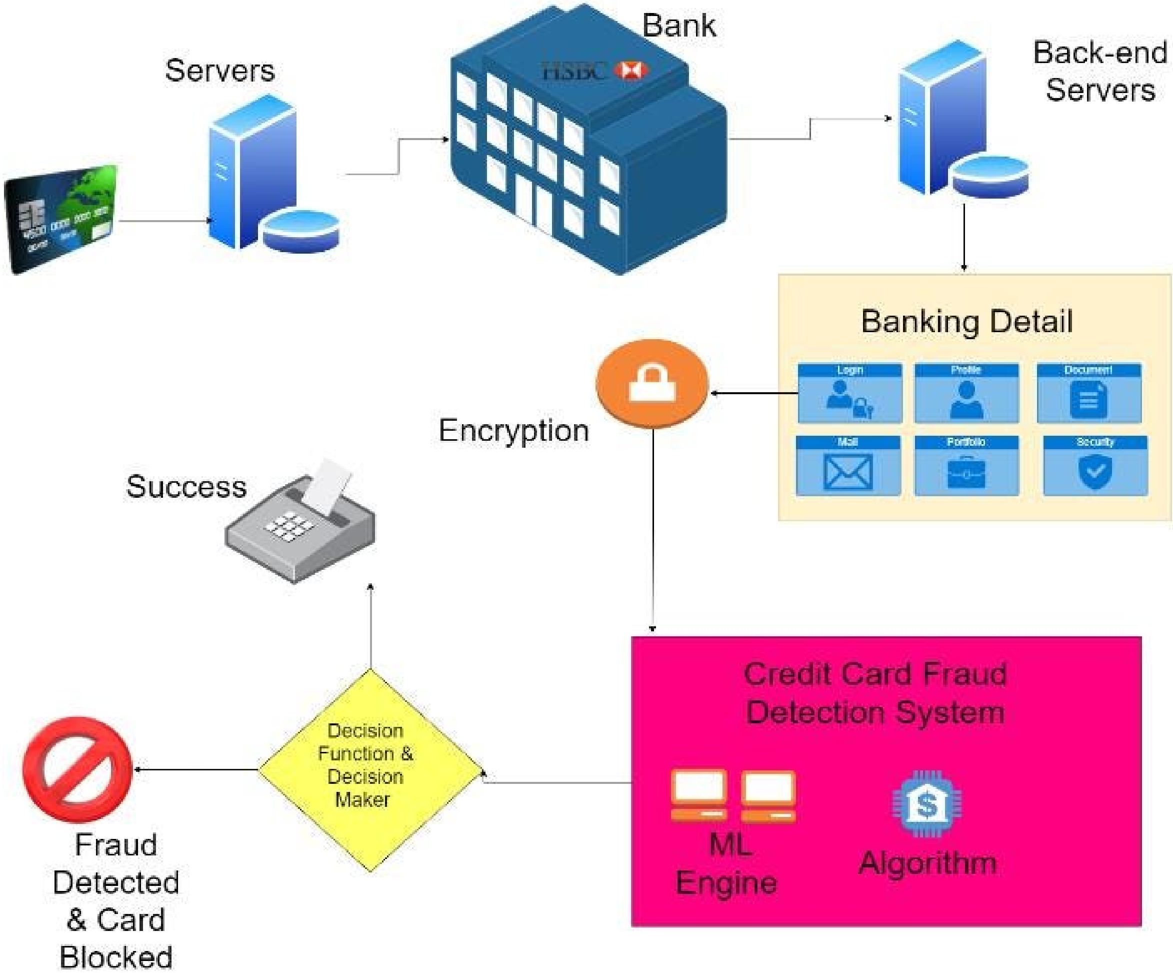
Data Mining, Part of Data Science

- Data Science if Multidisciplinary



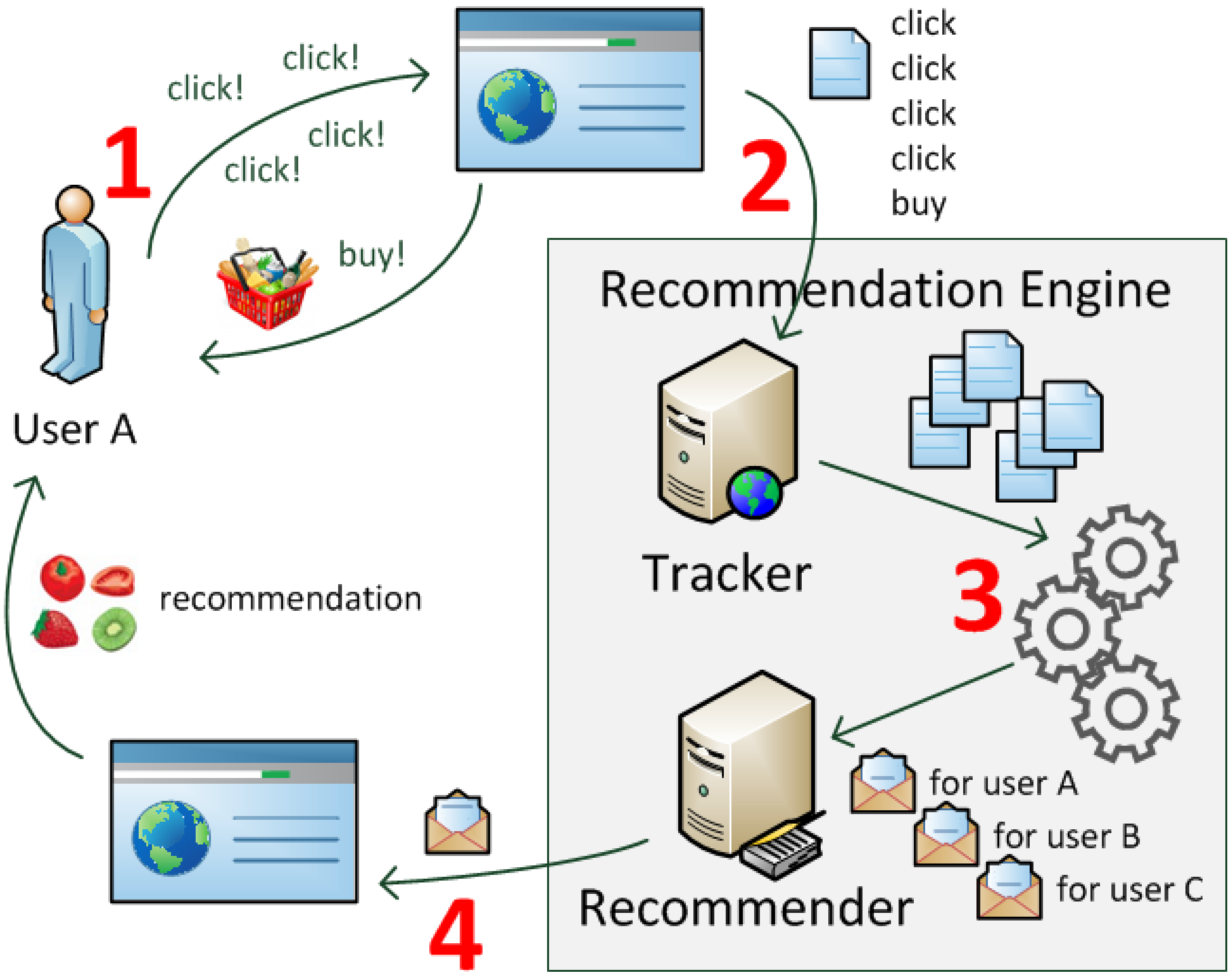
Application Areas of Data Mining

- Fraud detection systems



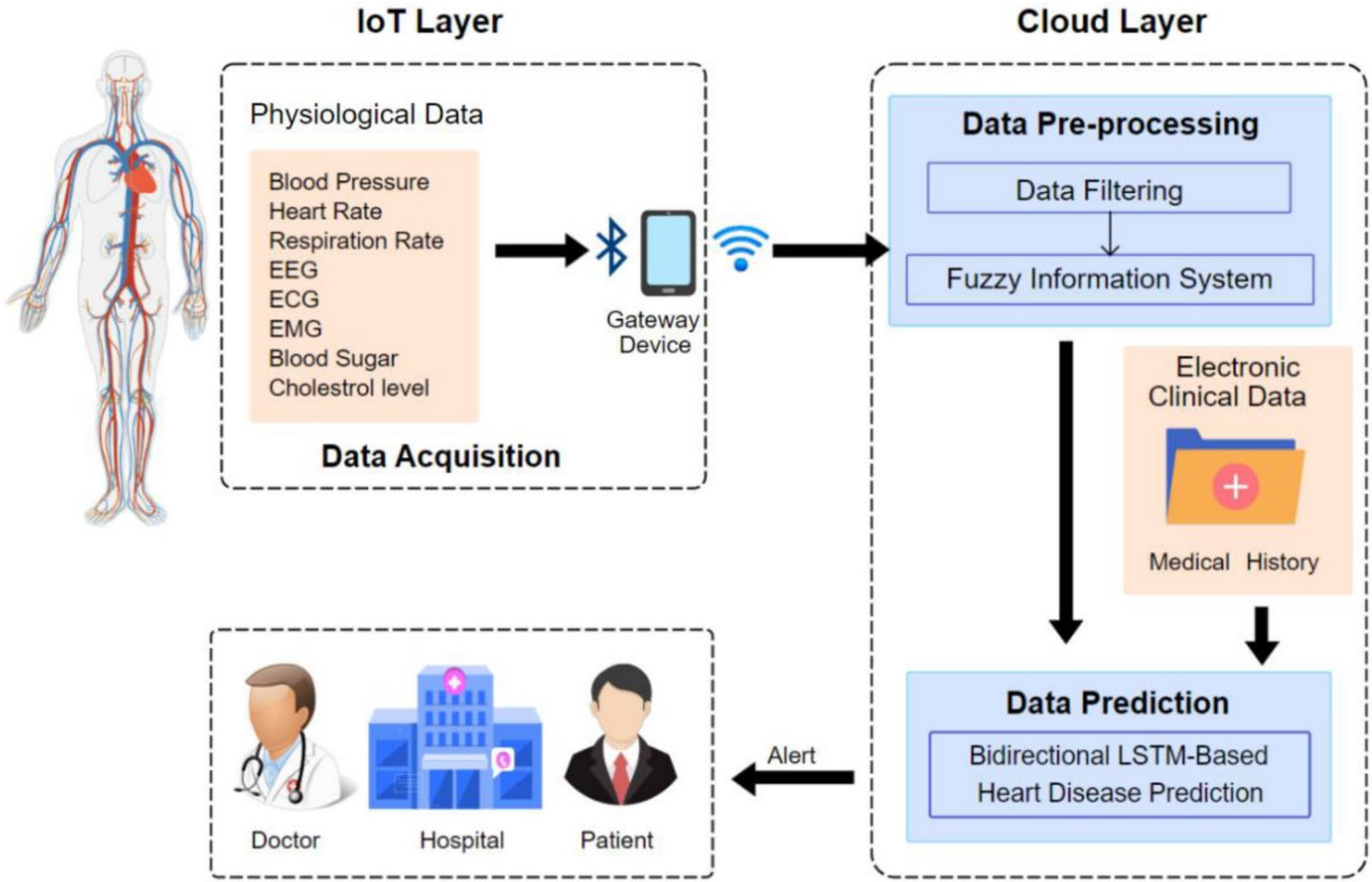
Application Areas of Data Mining

- Recommendation systems



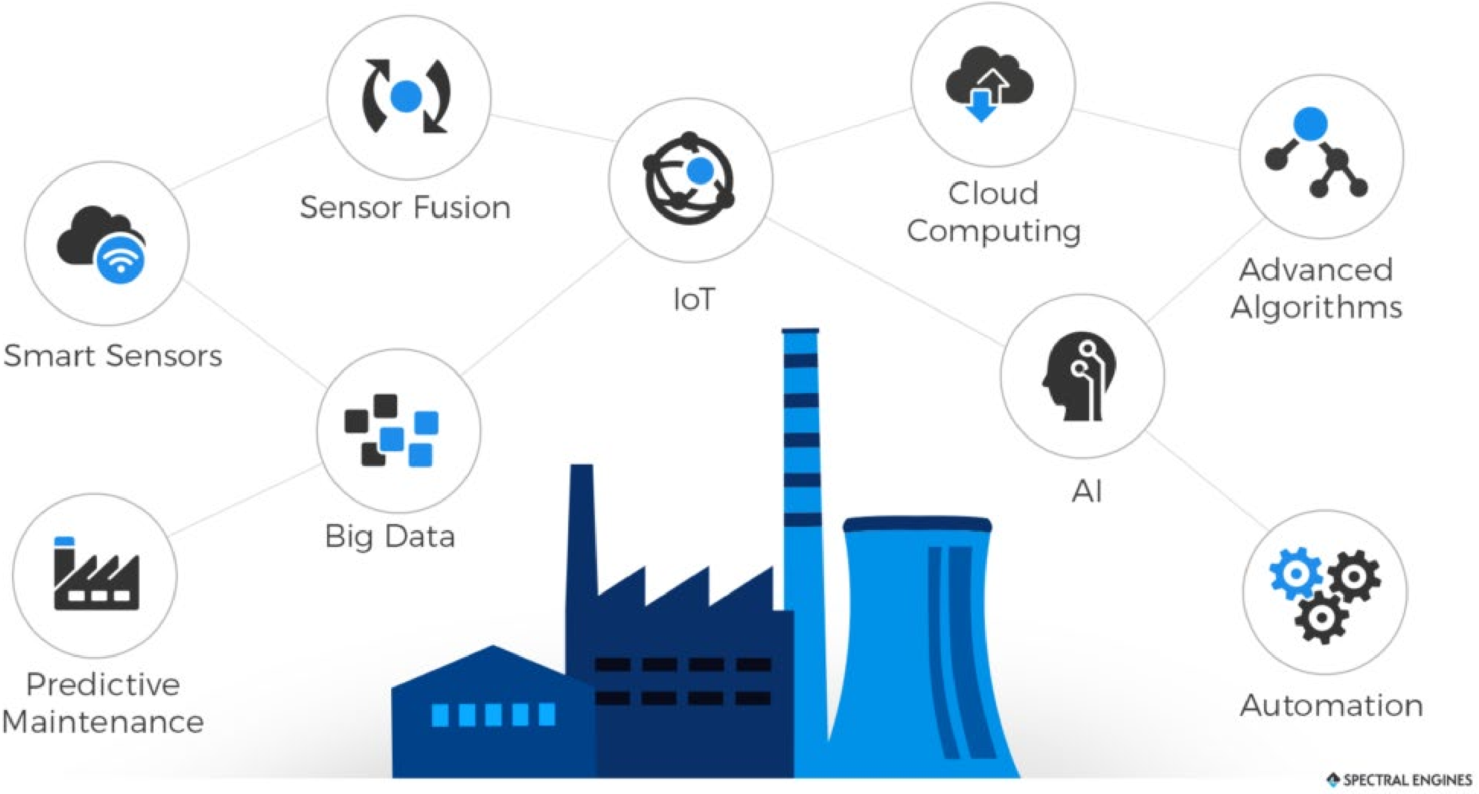
Application Areas of Data Mining

- Healthcare monitoring systems

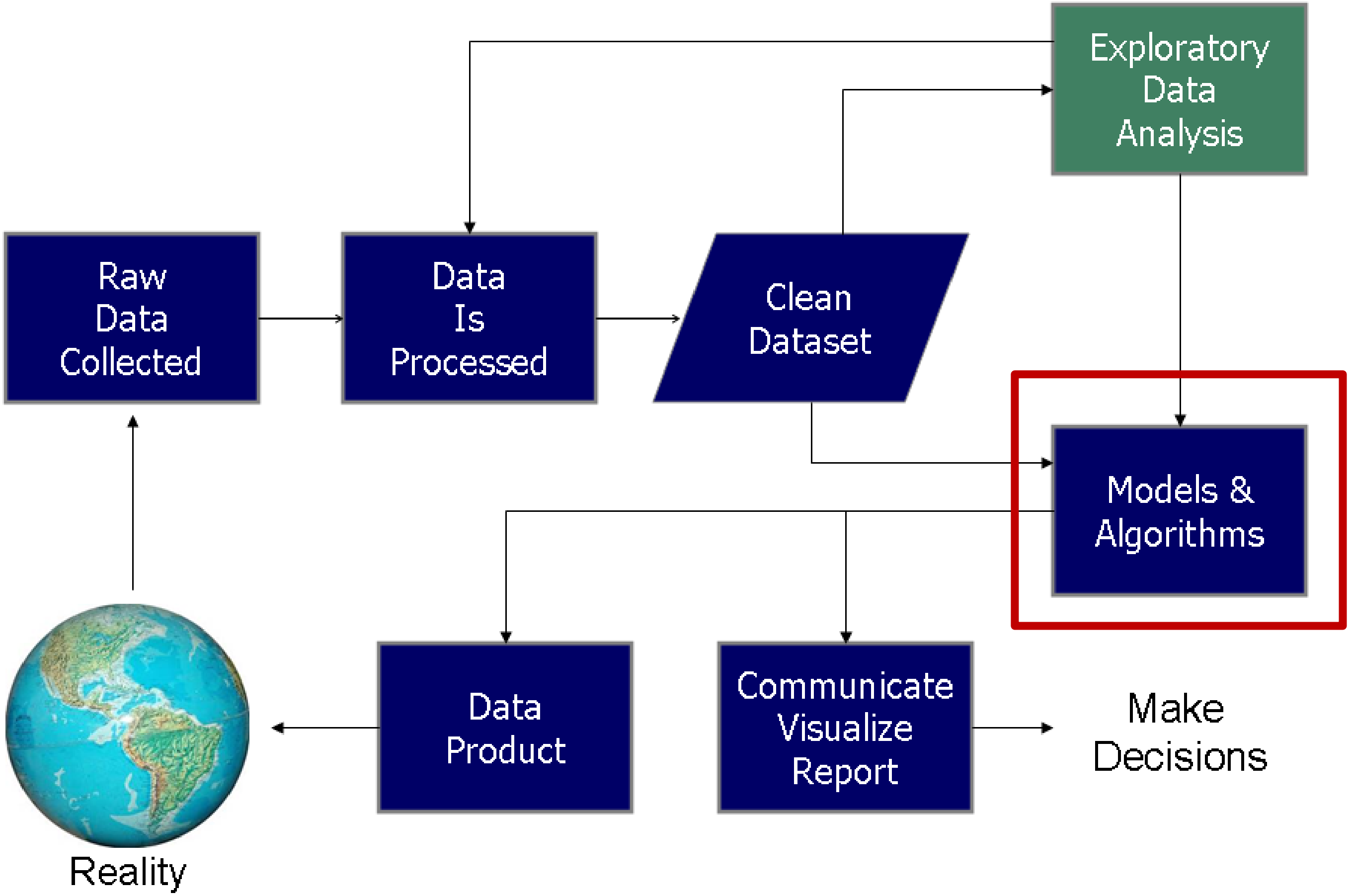


Application Areas of Data Mining

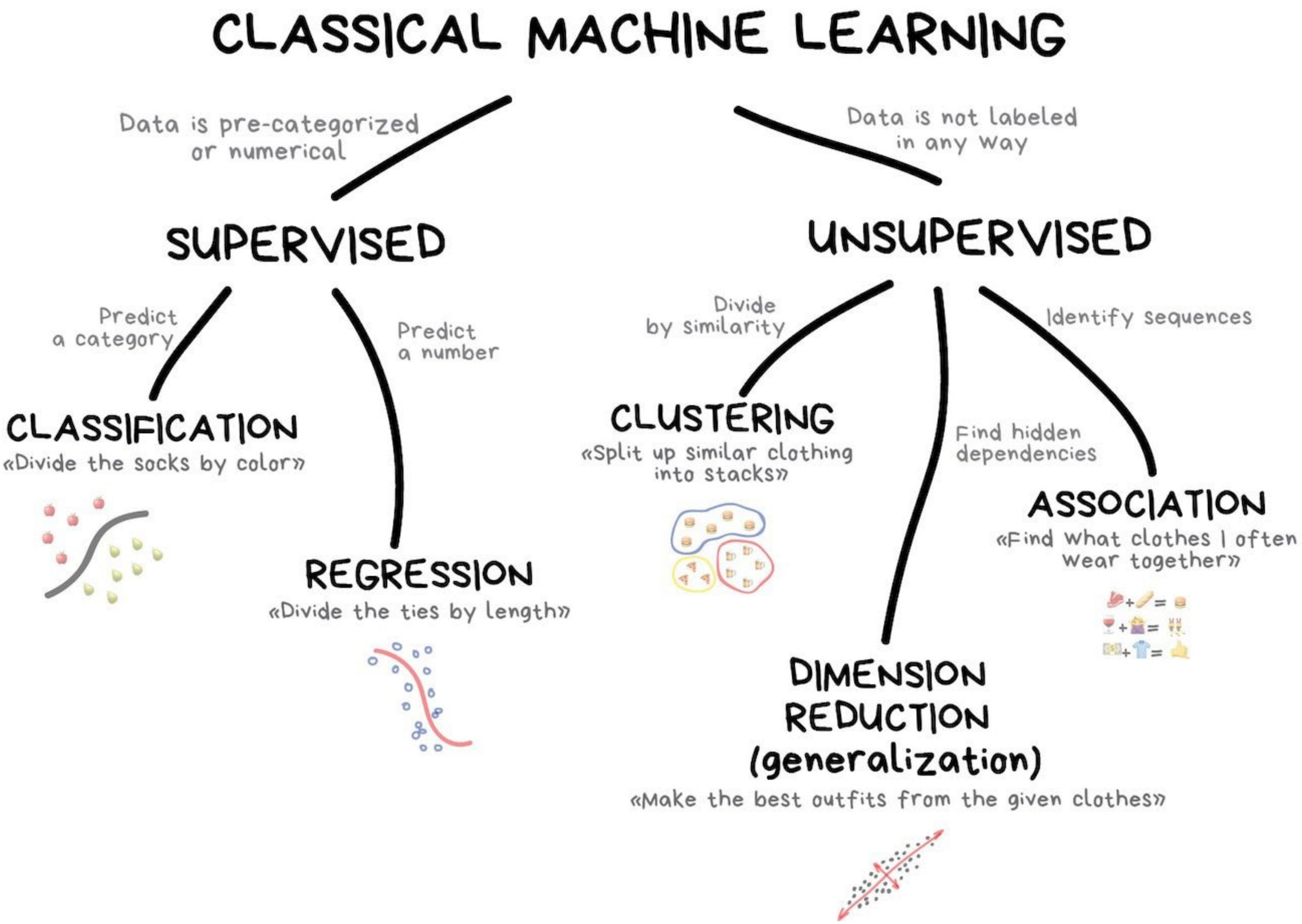
- Smart factory



What We Will Learn in This Class



Topics Covered in This Class

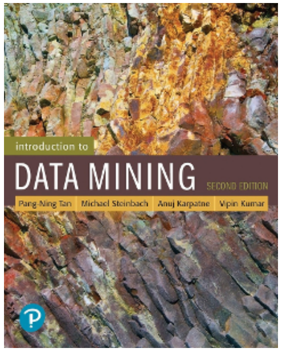

CLASSICAL MACHINE LEARNING
Data is pre-categorized or numerical
Data is not labeled in any way
SUPERVISED
UNSUPERVISED
Predict a category
Predict a number
CLASSIFICATION
«Divide the socks by color»
REGRESSION
«Divide the ties by length»
Divide by similarity
Identify sequences
Find hidden dependencies
CLUSTERING
«Split up similar clothing into stacks»
ASSOCIATION
«Find what clothes I often wear together»
DIMENSION REDUCTION (generalization)
«Make the best outfits from the given clothes»

Topics Covered in This Class

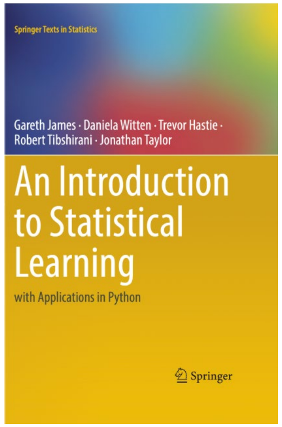
- Supervised learning
  - Regression
    - Linear regression
    - Nearest neighbor methods
    - Decision tree
  - Classification
    - Logistic regression
    - Naïve Bayes
    - Nearest neighbor methods
    - Decision tree
- Unsupervised learning
  - Clustering
    - k-means
    - Hierarchical clustering
  - Dimension reduction
    - Principal component analysis (PCA)
  - Association rule mining

Recommended Data Mining Textbooks

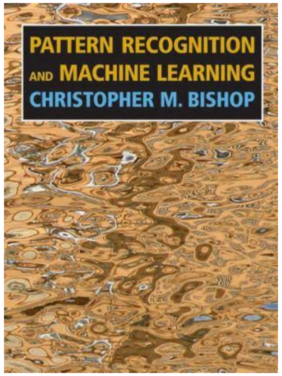
1. Introduction to Data Mining, 2nd edition
   - https://www-users.cse.umn.edu/~kumar001/dmbook/index.php



2. An Introduction to Statistical Learning: with Applications in Python
   - https://www.statlearning.com/

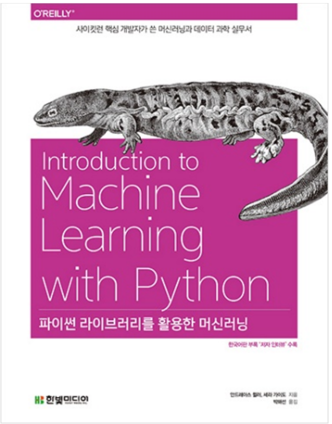

Springer Texts in Statistics
Gareth James · Daniela Witten · Trevor Hastie ·
Robert Tibshirani · Jonathan Taylor
An Introduction to Statistical Learning
with Applications in Python
Springer

3. Pattern Recognition and Machine Learning
   - https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/

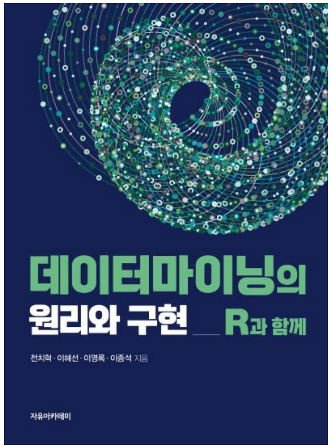


Recommended Data Mining Textbooks

4. 파이썬 라이브러리를 활용한 머신러닝
   - https://www.hanbit.co.kr/store/books/look.php?p_code=B6119391002


O'REILLY
Introduction to Machine Learning with Python
파이썬 라이브러리를 활용한 머신러닝
한빛미디어

5. 데이터마이닝의 원리와 구현 : R과 함께
   - http://www.freeaca.com/new/book/MainBookView.aspx?bookid=31311&page=1&ca1=3&ca2=54&sword=&stype=



Principals of Lecture

- Understand main goal and basic principles of each data mining techniques
  - Why is an algorithm proposed?
  - What is a key point?

→ Deliver principles as easy as possible without mathematics

- Understand detailed process of each data mining techniques
  - How are an algorithm working?

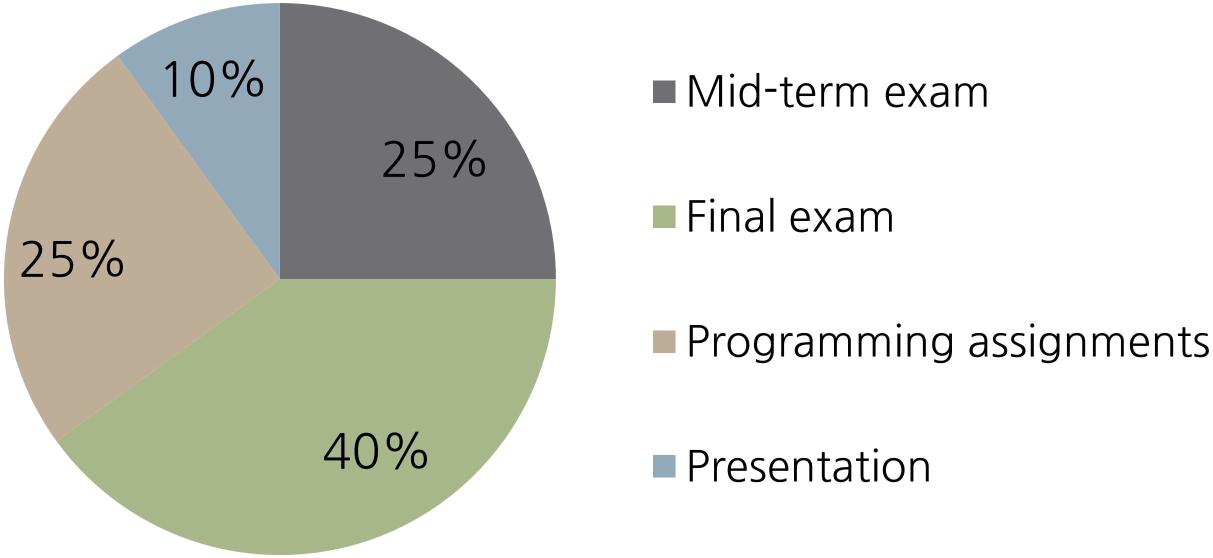
→ Explain process step by step

※ Some equations will be introduced for explanation

- Exercise what you learned during lectures
  - Main programming language: Python
  - Confirm algorithms studied during lectures through programming exercises

Principals of Lecture: Assessment

- Course assessment
  - Exams will be held two times: mid-term and final exams
    - Final exam will cover the whole lectures
  - Programming assignments related with lectures
  - Team presentation: Case study
    - Topic proposal will be presented on the **9th** week
    - The final result will be presented on the **15th** week
    - Each team consist of 2~3 students (random)



Principals of Lecture: Assessment

- Exams
  - Assess the theoretical knowledge learned in class
    - Must understand principles and process of the data mining algorithms covered in class
  - No multiple choice questions
  - Can use a scientific calculator
  - Schedule
    - Mid-term exam: 8th Week, 4/24 (in the evening)
    - Final exam: 14th Week, 6/5 (in the evening)

Principals of Lecture: Assessment

- Team presentation
  - Case study using data mining
    - The purpose of data analysis
      - What is the problem?
    - Method
      - How did they solve the problem through data mining?
    - Result
      - What kinds of implication could be derived from the results of data analysis?

Schedule

Week	Date	Contents	Remarks
1	3/6	Introduction	
2	3/13	Background of data mining	
3	3/20	Explanatory data analysis (EDA)	
4	3/28	Linear regression: Theory Part 1 & Exercise	
5	4/3	Linear regression: Theory Part 2 & Exercise	
6	4/10	Logistic regression: Theory & Exercise	
7	4/17	Naïve Bayes classifier: Theory & Exercise	
8	**4/24**	**Mid-term exam (in the evening)**	
9	5/1	Nearest neighbor algorithm: Theory & Exercise **Presentation: Case study topic proposal**	
10	5/8	Decision tree: Theory & Exercise	
11	5/15	Clustering: Theory & Exercise	
12	5/22	Dimensionality reduction: Theory & Exercise	
13	5/29	Association rule mining: Theory & Exercise	
14	**6/5**	**Final exam (in the evening)**	
15	6/12	**Presentation: Case study**	

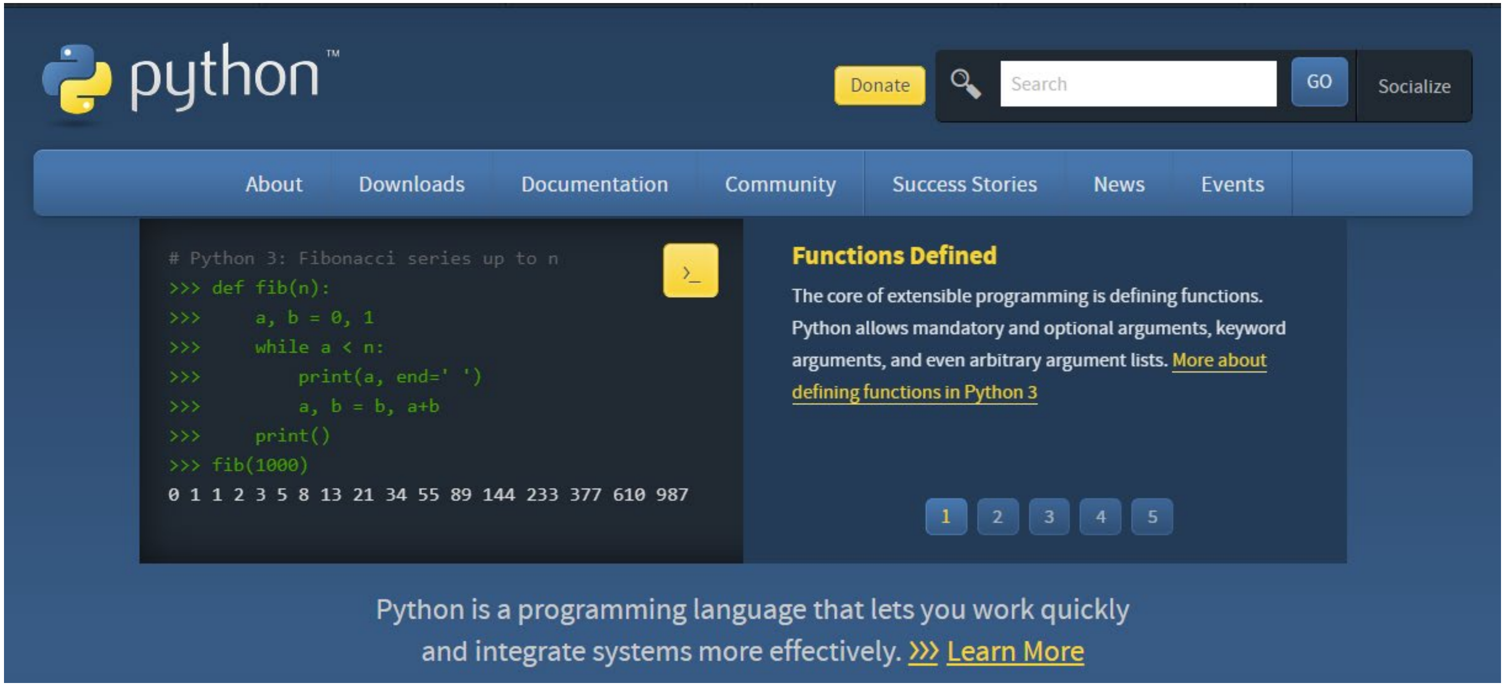
Q & A

- If you want to ask a question related with lectures for data mining algorithms outside of class, please use the Q&A board of the e-class
  - Your question may be helpful to other students
    - Share your questions with other students
  - Do not ask individual questions by e-mail

Python: Installation

Installation

- Python
  - Visit https://www.python.org/downloads/ and download Python installation file depending on your OS(Windows, Linux/UNIX, Mac OS X) and which version you want to install
    - This slide assumes that OS is Windows
  - There are two stable versions of Python: 3.X, 2.X
    - Two versions are a little bit different, but different features do not matter in this course



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

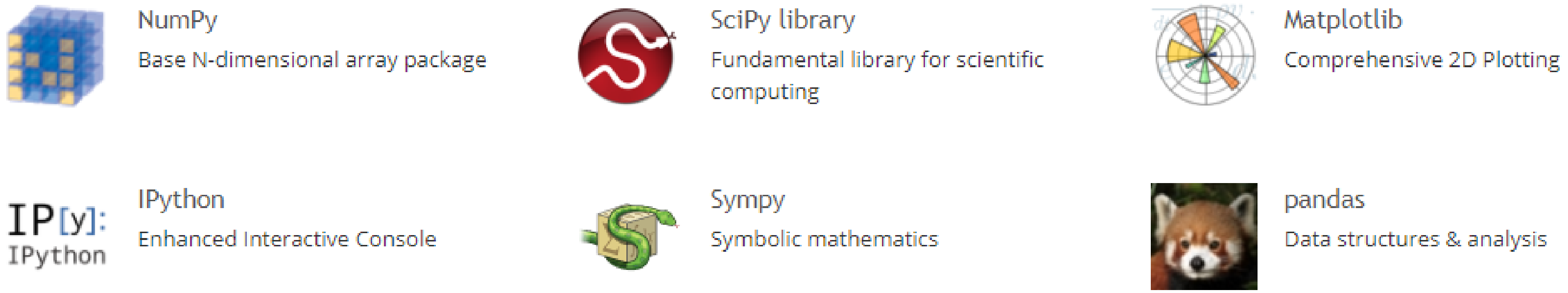
Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Installation Useful Packages

- SciPy
  - Python-based ecosystem of open-source software for mathematics, science, and engineering
  - http://www.scipy.org/



NumPy
Base N-dimensional array package

SciPy library
Fundamental library for scientific computing

Matplotlib
Comprehensive 2D Plotting

IPython
Enhanced Interactive Console

Sympy
Symbolic mathematics

pandas
Data structures & analysis

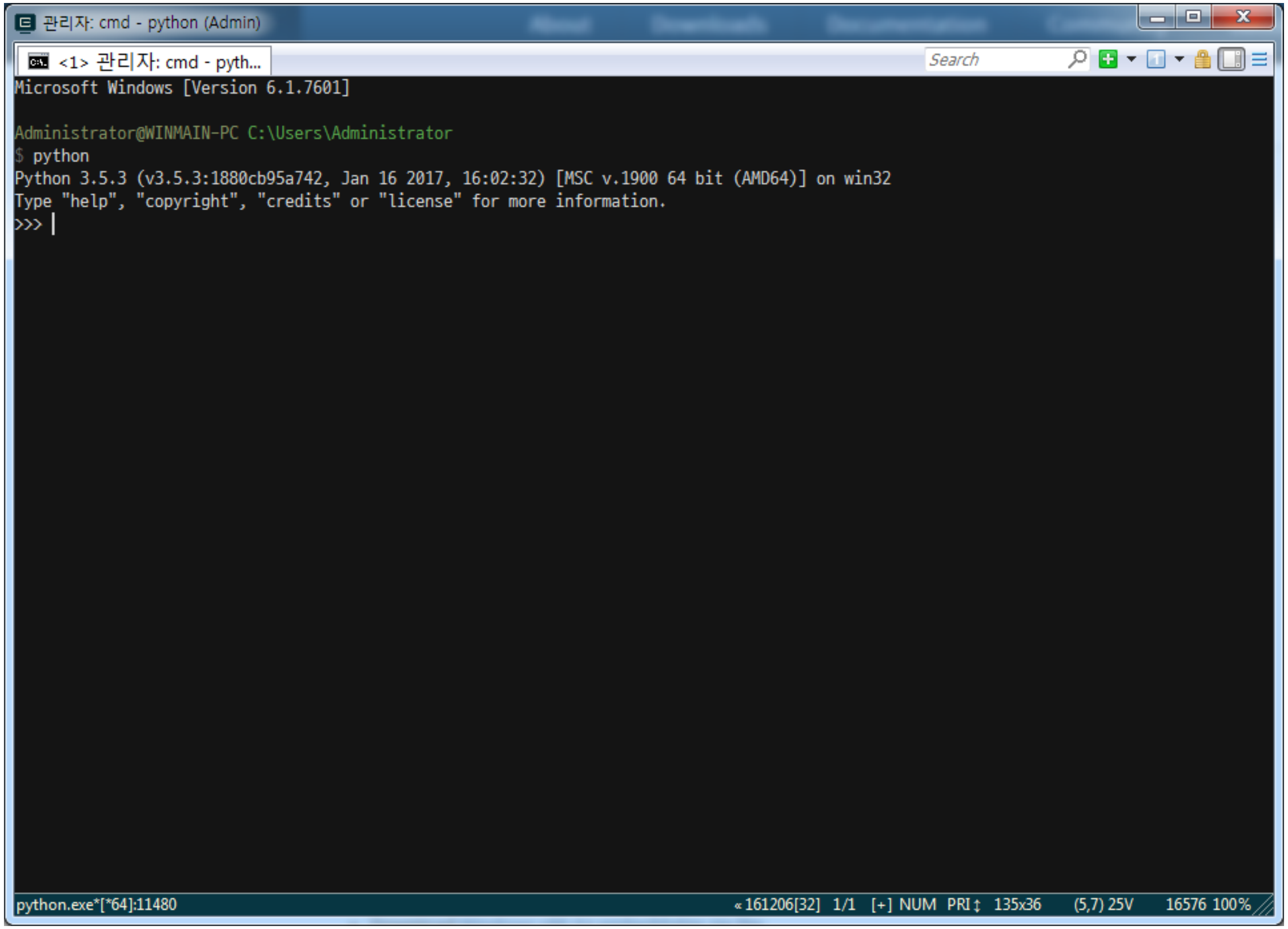
[Core packages]

Installation Useful Packages

- sci-kit learn
  - Free software machine learning library for the Python programming language
    - Simple and efficient tools for predictive data analysis
    - Built on Numpy, Scipy, and matplotlib
  - https://scikit-learn.org/stable/index.html

Start Python

- To start python, just type python at cmd prompt
  - Python is script language



관리자: cmd - python (Admin)
<1> 관리자: cmd - pyth...
Search

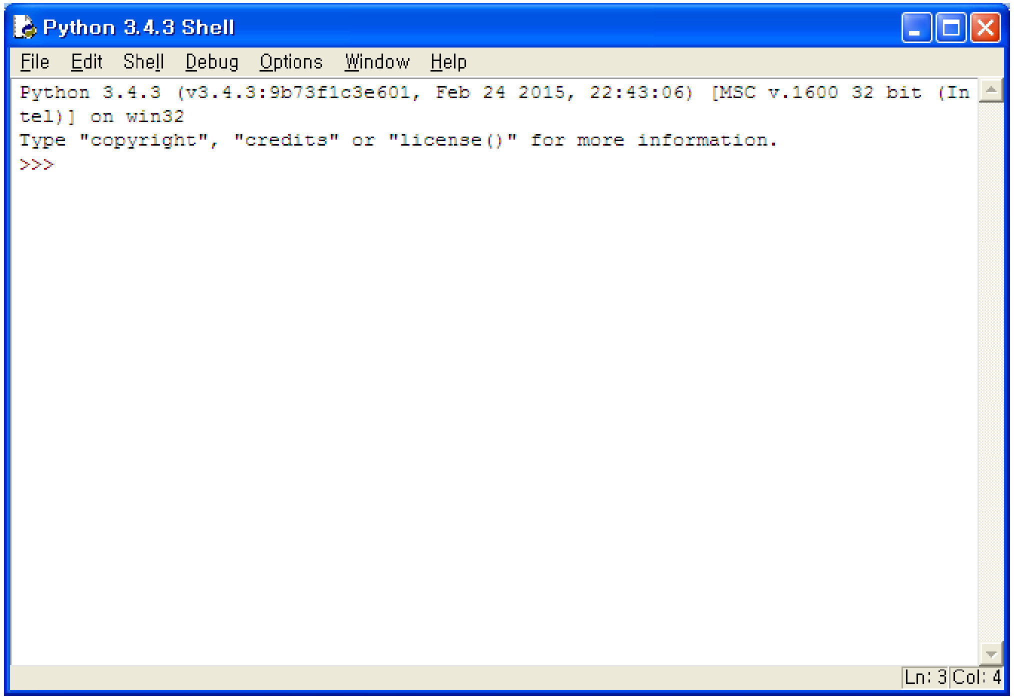
```
Microsoft Windows [Version 6.1.7601]

Administrator@WINMAIN-PC C:\Users\Administrator
$ python
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> |
```

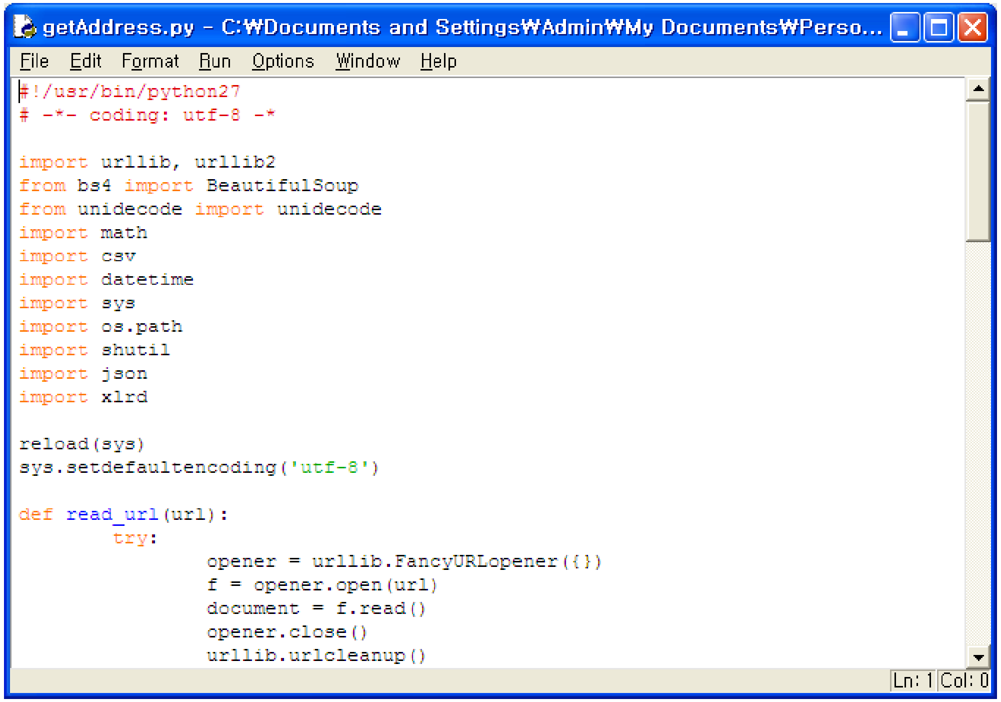
python.exe*[*64]:11480
« 161206[32] 1/1 [+] NUM PRI↕ 135x36 (5,7) 25V 16576 100%

Start Python

- There are many Python IDEs(Integrated Development Environment)
  - However, notepad is also used for writing Python scripts
  - If you want to use better IDE than notepad
    - http://pedrokroger.net/choosing-best-python-ide/
  - There is also default IDE installed with Python



```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```


getAddress.py - C:₩Documents and Settings₩Admin₩My Documents₩Perso...
File
Edit
Format
Run
Options
Window
Help
Ln: 1
Col: 0

```
#!/usr/bin/python27
# -*- coding: utf-8 -*

import urllib, urllib2
from bs4 import BeautifulSoup
from unidecode import unidecode
import math
import csv
import datetime
import sys
import os.path
import shutil
import json
import xlrd

reload(sys)
sys.setdefaultencoding('utf-8')

def read_url(url):
        try:
                opener = urllib.FancyURLopener({})
                f = opener.open(url)
                document = f.read()
                opener.close()
                urllib.urlcleanup()
```

Python: Easy Installation

Scientific Python distributions

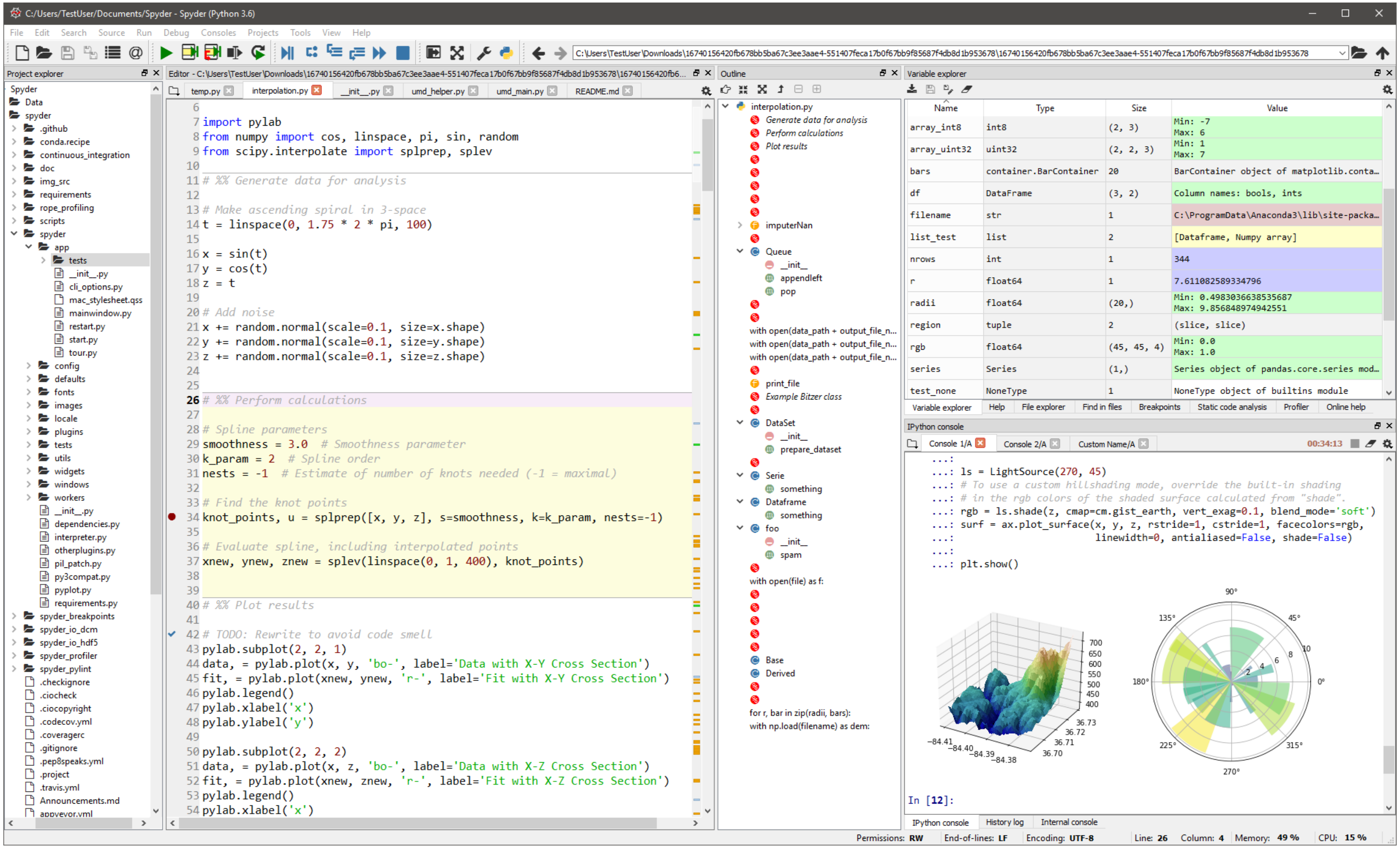
- The easiest way to install the packages of the SciPy stack is to download one of these Python distributions, which includes all the key packages
  - Anaconda: A free distribution for the SciPy stack. Supports Linux, Windows and Mac.
  - Enthought Canopy: The free and commercial versions include the core SciPy stack packages. Supports Linux, Windows and Mac.
  - Python(x,y): A free distribution including the SciPy stack, based around the Spyder IDE. Windows only.
  - WinPython: A free distribution including the SciPy stack. Windows only.
  - Pyzo: A free distribution based on Anaconda and the IEP interactive development environment. Supports Linux, Windows and Mac.

Scientific Python distributions

- Anaconda
  - URL: https://www.anaconda.com/distribution/
- WinPython
  - URL : http://winpython.github.io/
    https://sourceforge.net/projects/winpython/

Scientific Python distributions

- Spyder
  - The Scientific PYthon Development EnviRonment



```

import pylab
from numpy import cos, linspace, pi, sin, random
from scipy.interpolate import splprep, splev

# %% Generate data for analysis

# Make ascending spiral in 3-space
t = linspace(0, 1.75 * 2 * pi, 100)

x = sin(t)
y = cos(t)
z = t

# Add noise
x += random.normal(scale=0.1, size=x.shape)
y += random.normal(scale=0.1, size=y.shape)
z += random.normal(scale=0.1, size=z.shape)


# %% Perform calculations

# Spline parameters
smoothness = 3.0  # Smoothness parameter
k_param = 2  # Spline order
nests = -1  # Estimate of number of knots needed (-1 = maximal)

# Find the knot points
knot_points, u = splprep([x, y, z], s=smoothness, k=k_param, nests=-1)

# Evaluate spline, including interpolated points
xnew, ynew, znew = splev(linspace(0, 1, 400), knot_points)


# %% Plot results

# TODO: Rewrite to avoid code smell
pylab.subplot(2, 2, 1)
data, = pylab.plot(x, y, 'bo-', label='Data with X-Y Cross Section')
fit, = pylab.plot(xnew, ynew, 'r-', label='Fit with X-Y Cross Section')
pylab.legend()
pylab.xlabel('x')
pylab.ylabel('y')

pylab.subplot(2, 2, 2)
data, = pylab.plot(x, z, 'bo-', label='Data with X-Z Cross Section')
fit, = pylab.plot(xnew, znew, 'r-', label='Fit with X-Z Cross Section')
pylab.legend()
pylab.xlabel('x')
```

Name	Type	Size	Value
array_int8	int8	(2, 3)	Min: -7 Max: 6
array_uint32	uint32	(2, 2, 3)	Min: 1 Max: 7
bars	container.BarContainer	20	BarContainer object of matplotlib.conta…
df	DataFrame	(3, 2)	Column names: bools, ints
filename	str	1	C:\ProgramData\Anaconda3\lib\site-packa…
list_test	list	2	[Dataframe, Numpy array]
nrows	int	1	344
r	float64	1	7.611082589334796
radii	float64	(20,)	Min: 0.4983036638535687 Max: 9.856848974942551
region	tuple	2	(slice, slice)
rgb	float64	(45, 45, 4)	Min: 0.0 Max: 1.0
series	Series	(1,)	Series object of pandas.core.series mod…
test_none	NoneType	1	NoneType object of builtins module

```
   ...:
   ...: ls = LightSource(270, 45)
   ...: # To use a custom hillshading mode, override the built-in shading
   ...: # in the rgb colors of the shaded surface calculated from "shade".
   ...: rgb = ls.shade(z, cmap=cm.gist_earth, vert_exag=0.1, blend_mode='soft')
   ...: surf = ax.plot_surface(x, y, z, rstride=1, cstride=1, facecolors=rgb,
   ...:                        linewidth=0, antialiased=False, shade=False)
   ...:
   ...: plt.show()

In [12]:
```

Python: Short Tutorial

Variable Types

- List
  - A list contains items separated by commas and enclosed within square brackets ([])

```
list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']

print list          # Prints complete list
print list[0]       # Prints first element of the list
print list[1:3]     # Prints elements starting from 2nd till 3rd
print list[2:]      # Prints elements starting from 3rd element
print tinylist * 2  # Prints list two times
print list + tinylist # Prints concatenated lists
```

Variable Types

- Tuples
  - A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses
  - The main differences between lists and tuples are
    - Lists are enclosed in brackets ([ ]) and their elements and size can be changed
    - Tuples are enclosed in parentheses (( )) and cannot be updated (read-only)

```
tuple = ( 'abcd', 786 , 2.23, 'john', 70.2  )
tinytuple = (123, 'john')

print tuple           # Prints complete list
print tuple[0]        # Prints first element of the list
print tuple[1:3]      # Prints elements starting from 2nd till 3rd
print tuple[2:]       # Prints elements starting from 3rd element
print tinytuple * 2   # Prints list two times
print tuple + tinytuple # Prints concatenated lists
```

Variable Types

- Dictionary
  - They work like associative arrays or hashes found in Perl and consist of key-value pairs
  - Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([])

```
dict = {}
dict['one'] = "This is one"
dict[2]     = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print dict['one']       # Prints value for 'one' key
print dict[2]           # Prints value for 2 key
print tinydict          # Prints complete dictionary
print tinydict.keys()   # Prints all the keys
print tinydict.values() # Prints all the values
```

Data Conversion

Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
long(x [,base])	Converts x to a long integer. base specifies the base if x is a string
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
repr(x)	Converts object x to an expression string.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.
frozenset(s)	Converts s to a frozen set.
chr(x)	Converts an integer to a character.
unichr(x)	Converts an integer to a Unicode character.
ord(x)	Converts a single character to its integer value.
hex(x)	Converts an integer to a hexadecimal string.
oct(x)	Converts an integer to an octal string.

Basic Operation

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a - b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	9//2 = 4 and 9.0//2.0 = 4.0

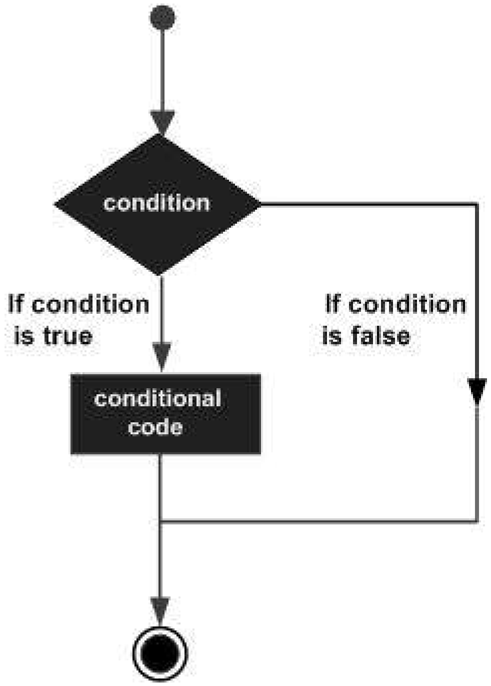
Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the v alue of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the valu e of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Decision Making

□ Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions

```
var = 100
if ( var  == 100 ):
    print("Value of expression is 100")
else:
    print("Value of expression is not 100")
```



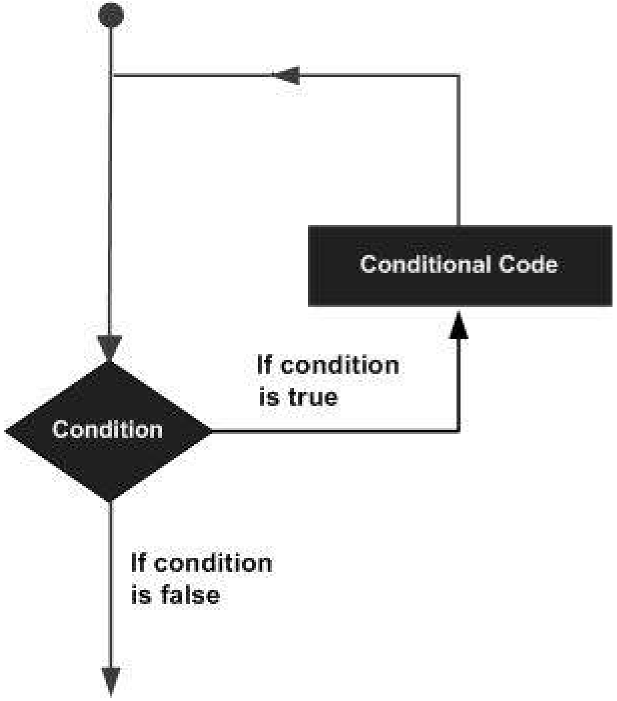
Loop

- A loop statement allows us to execute a statement or group of statements multiple times

```
primes = [2, 3, 5, 7]
for prime in primes:
    print(prime)

for x in range(5): # or range(5)
    print(x)

count = 0
while count < 5:
    print(count)
    count += 1  # This is the same as count = count + 1
```



Loop

□ Loop control statements change execution from its normal sequence

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
pass statement	The pass statement in Python is used when a statement is required s yntactically but you do not want any command or code to execute.

```
count = 0
while True:
    print(count)
    count += 1
    if count >= 5:
        break
```

List comprehensions

- Python supports a concept called “list comprehensions” used to construct lists in a very natural, easy way

```
>>> A=[x**2 for x in range(10)]
>>> print(A)
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>> B = [x for x in S if x % 2 == 0]
>>> print(B)
[0, 4, 16, 36, 64]
>>> C = [x+3 for x in A]
>>> print(C)
[3, 4, 7, 12, 19, 28, 39, 52, 67, 84]
>>> D = [x+3 if x%2==0 else x for x in A]
>>> print(D)
[3, 1, 7, 9, 19, 25, 39, 49, 67, 81]
```

List comprehensions

- A=[x**2 for x in range(10)]
  - range(10) creates list whose elements are from zero to nine

    [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  - for x in range(10): loop for elements in range(10)
    - x represents each element in range(10)
  - x**2 for x in range(10): for every element in range(10), calculate x^2
    - Results are stored in A as list

Index of Python

□ Python list

index	0	1	2	3	4	5	6
negative index	-7	-6	-5	-4	-3	-2	-1
	8	7	5	13	75	65	11

```
>>>A=[8,7,5,13,75,65,11]
>>>A[0]
8
>>>A[3]
13
>>>A[-1]
11
>>>A[1:4]
[7,5,13]
>>>A[:3]
[8,7,5]
>>>A[4:]
[75,65,11]
```