Python for Scientists

InSuk Joung (정인석)

KIAS

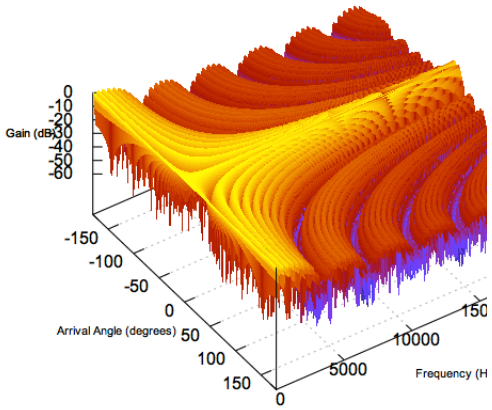
8th Summer School on Parallel and Scientific Computing, 2017

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Scientific Data Processing

Raw data

Processed data: images, tables, documents



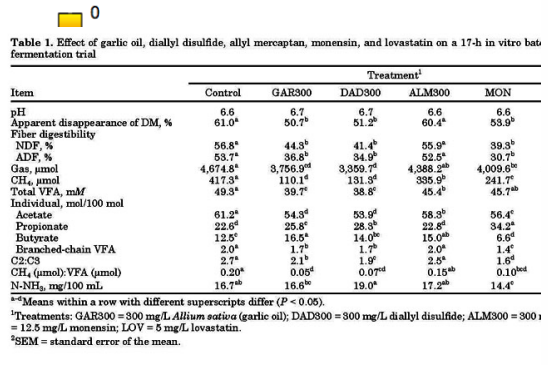


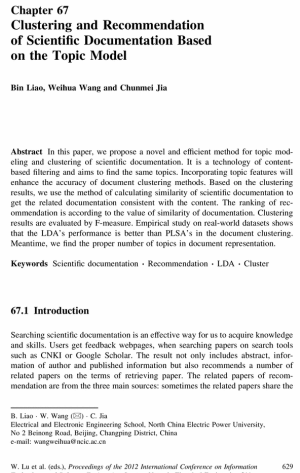
Table 1. Effect of garlic oil, diallyl disulfide, allyl mercaptan, monensin, and lovastatin on a 17-h in vitro bat fermentation trial

	Treatment[1]				
Item	Control	GAR300	DAD300	ALM300	MON
pH	6.6	6.7	6.7	6.6	6.6
Apparent disappearance of DM, %	61.0[a]	50.7[b]	51.2[b]	60.4[a]	53.9[b]
Fiber digestibility					
NDF, %	56.8[a]	44.3[b]	41.4[b]	55.9[a]	39.3[b]
ADF, %	53.7[a]	36.8[b]	34.9[b]	52.5[a]	30.7[b]
Gas, μmol	4,674.8[a]	3,756.9[cd]	3,359.7[d]	4,388.2[ab]	4,009.6[bc]
CH_4, μmol	417.3[a]	110.1[d]	131.5[d]	335.9[b]	241.7[c]
Total VFA, m*M*	49.3[a]	39.7[c]	38.8[c]	45.4[b]	45.7[ab]
Individual, mol/100 mol					
Acetate	61.2[a]	54.3[d]	53.9[d]	58.3[b]	56.4[c]
Propionate	22.6[d]	25.8[c]	28.3[b]	22.6[d]	34.2[a]
Butyrate	12.5[c]	16.5[a]	14.0[bc]	15.0[ab]	6.6[d]
Branched-chain VFA	2.0[a]	1.7[b]	1.7[b]	2.0[a]	1.4[c]
C2:C3	2.7[a]	2.1[b]	1.9[c]	2.5[a]	1.6[d]
CH_4 (μmol):VFA (μmol)	0.20[a]	0.05[d]	0.07[cd]	0.15[ab]	0.10[bcd]
N-NH_3, mg/100 mL	16.7[ab]	16.6[bc]	19.0[a]	17.2[ab]	14.4[c]

[a–d]Means within a row with different superscripts differ ($P < 0.05$).

[1]Treatments: GAR300 = 300 mg/L *Allium sativa* (garlic oil); DAD300 = 300 mg/L diallyl disulfide; ALM300 = 300 = 12.5 mg/L monensin; LOV = 5 mg/L lovastatin.

[2]SEM = standard error of the mean.



Chapter 67
Clustering and Recommendation of Scientific Documentation Based on the Topic Model

Bin Liao, Weihua Wang and Chunmei Jia

Abstract In this paper, we propose a novel and efficient method for topic modeling and clustering of scientific documentation. It is a technology of content-based filtering and aims to find the same topics. Incorporating topic features will enhance the accuracy of document clustering methods. Based on the clustering results, we use the method of calculating similarity of scientific documentation to get the related documentation consistent with the content. The ranking of recommendation is according to the value of similarity of documentation. Clustering results are evaluated by F-measure. Empirical study on real-world datasets shows that the LDA's performance is better than PLSA's in the document clustering. Meantime, we find the proper number of topics in document representation.

Keywords Scientific documentation · Recommendation · LDA · Cluster

67.1 Introduction

Searching scientific documentation is an effective way for us to acquire knowledge and skills. Users get feedback webpages, when searching papers on search tools such as CNKI or Google Scholar. The result not only includes abstract, information of author and published information but also recommends a number of related papers on the terms of retrieving paper. The related papers of recommendation are from the three main sources: sometimes the related papers share the

B. Liao · W. Wang (✉) · C. Jia
Electrical and Electronic Engineering School, North China Electric Power University, No 2 Beinong Road, Beijing, Changping District, China
e-mail: wangweihua@ncic.ac.cn

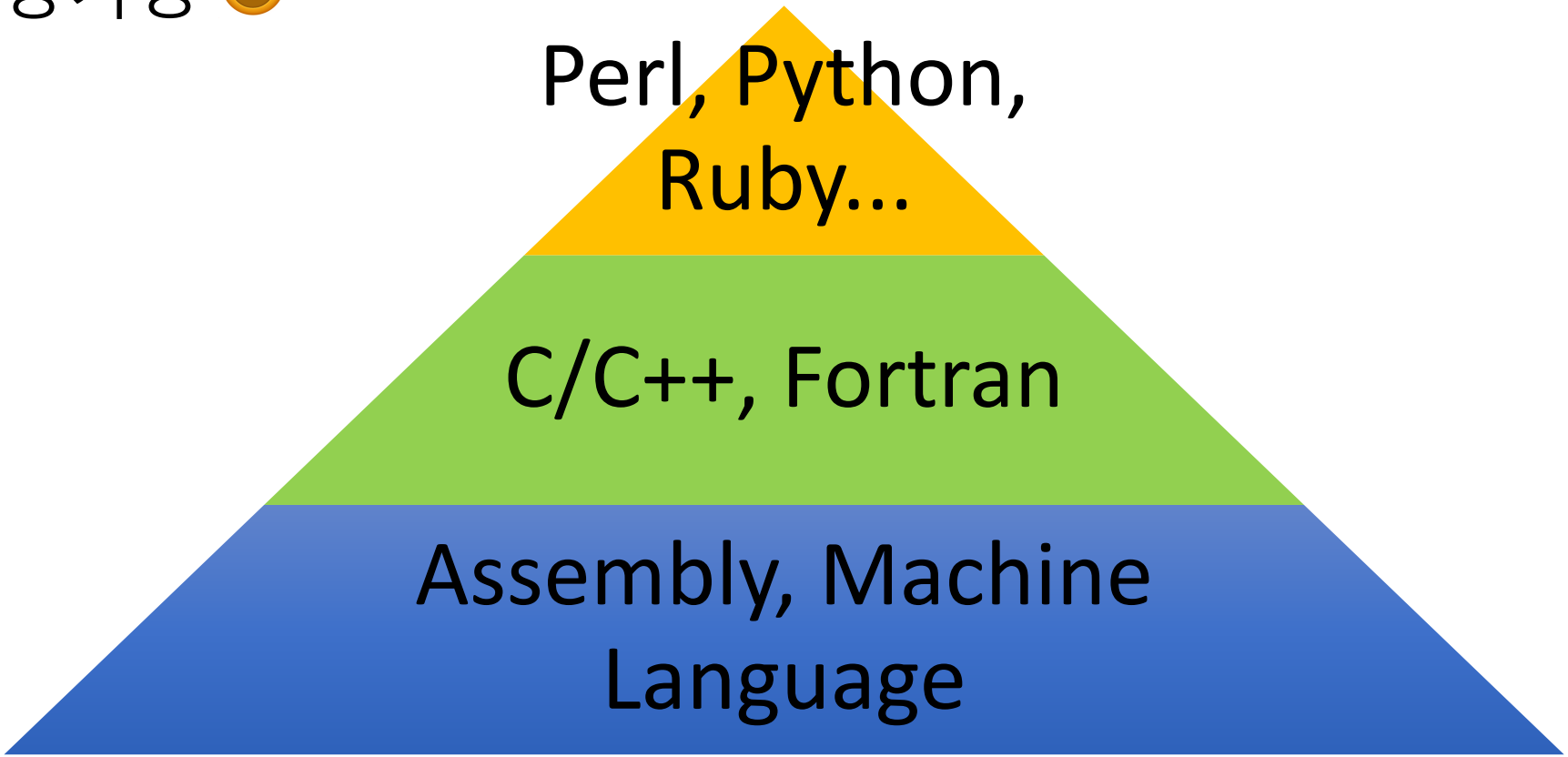
W. Lu et al. (eds.), *Proceedings of the 2012 International Conference on Information Technology and Software Engineering*, Lecture Notes in Electrical Engineering 211,
DOI: 10.1007/978-3-642-34522-7_67, © Springer-Verlag Berlin Heidelberg 2013

629

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Why Python?

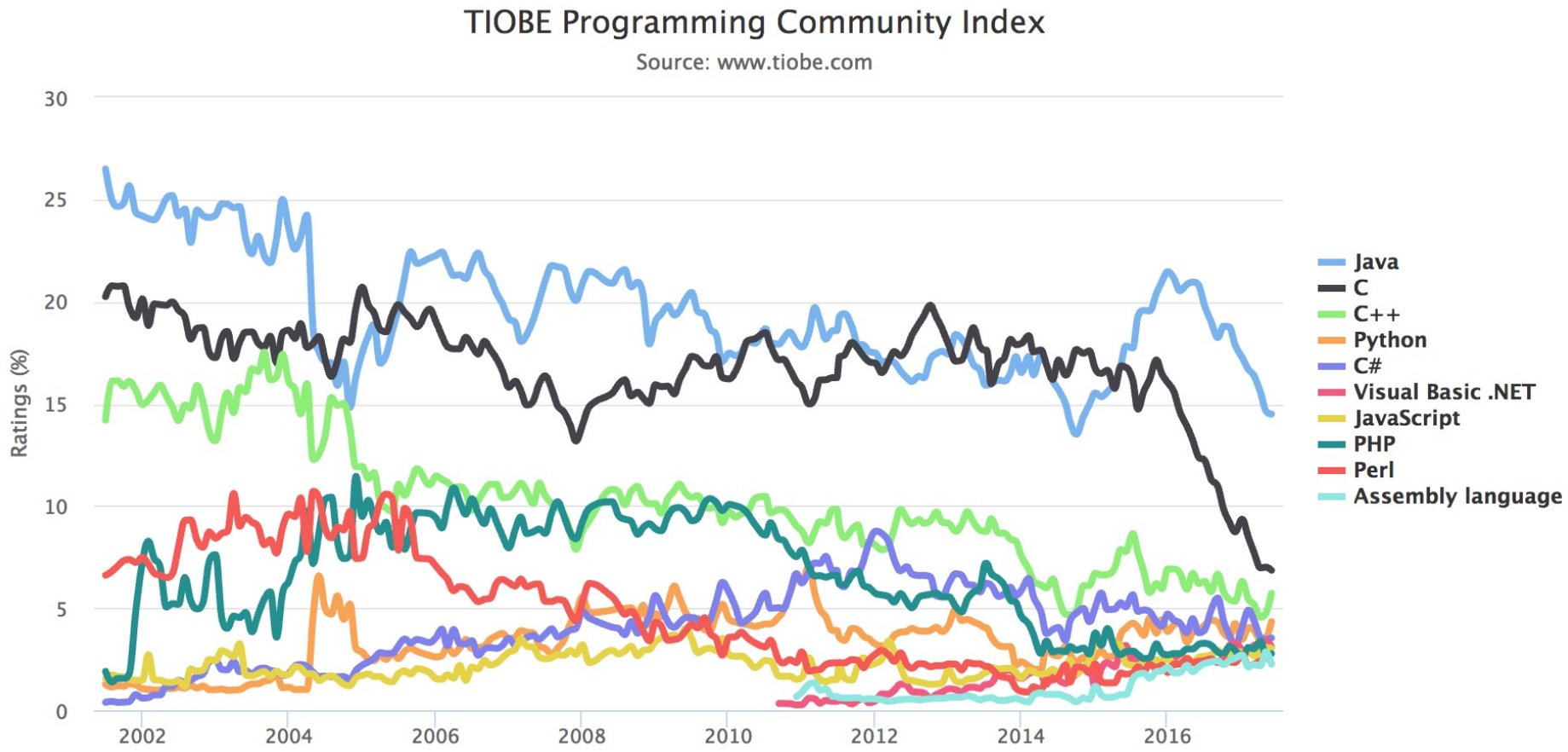
- High-level language (Perl, Python, Ruby, Visual Basic, …)
- 많은 사람들이 사용한다. **특히, 많은 과학자들이 선택하였다.**
- 무료로 사용가능 😄



KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Programing Language Popularity

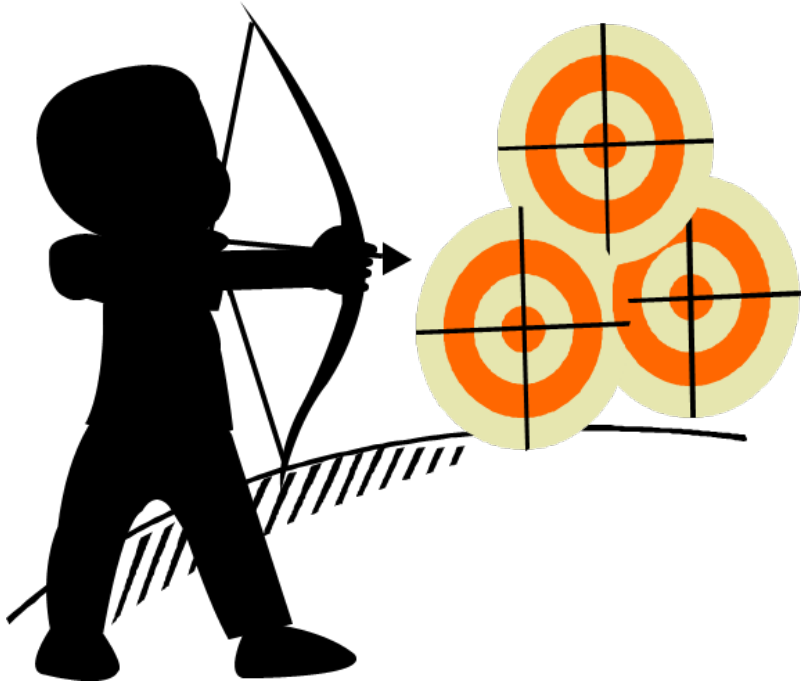
- TIOBE index (Web Search): Java, C, C++, Python, C#
- PYPL index (Google Trends): Java, Python, PHP, C#, Javascript
- RedMonk rating (Github/Stack Overflow): Javascript, Java, Python, PHP, C#



KIAS KOREA INSTITUTE FOR ADVANCED STUDY

About The Lectures...

- 과학자로서 python에 관해 알아야 할 내용 위주로
- ***최소한의 학습***으로 문제를 해결
- 다양한 방법 중에서 ***선택된 몇 가지 방법***만 제시
- 초급 / 중급 / 고급



Selection and Concentration

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Outline

- 초급
  - Data type
  - Flow control
  - Module
  - Class
- 중급
  - numpy
  - mpi4py
- 고급
  - f2py
  - cython

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Python 2 vs Python 3

- Python 2 is still more popular.
- Python 3 is definitely more advanced.

- We will use python 2.
- *But it is OK to use python 3 if you want.*

- Python 2 코드를 3 으로 변환툴: 2to3

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Useful Methods for Interactive Mode

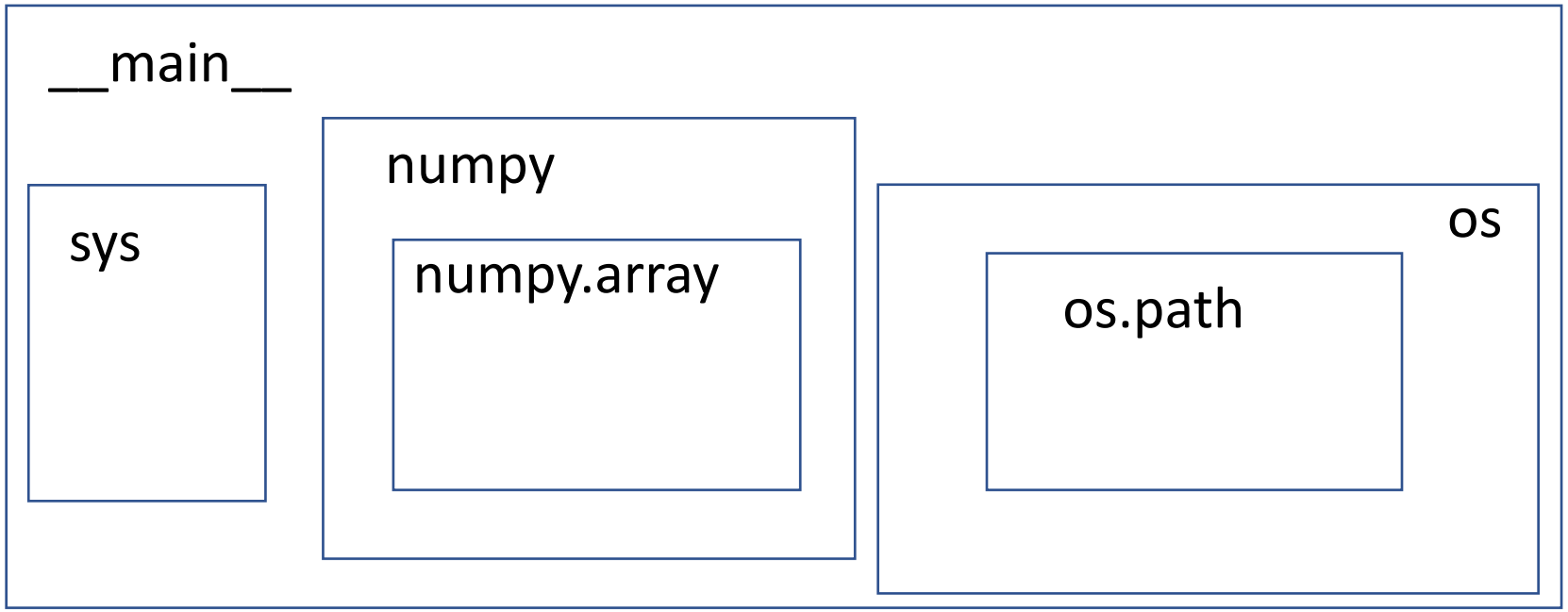
- **help**(*object*): shows the help message of the *object*
- **dir**(*object*): shows the namespace of the *object*
- **type**(*object*): shows the type of the *object*

Python Namespace

- *dot(.) separator*
  - sys.stdout (module.object)
  - numpy.array.mean (module.class.object)
  - os.path.basename (module.module.object)
  - ‘a b c’.split (object.object)

Dot의 의미에 대해서 정리하면…
1. 해당 모듈의 namespace로 접근
2. Class 또는 Class instance 의 멤버 변수 또는 method로 접근

Namespace



KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Basic Types

- **int**: 1, 3
- **bool**: True/False
- **float**: 3.14, 1.414, …
- **complex**: 1.0+3.0j
- **str**: “hello”, ‘abc’
- **None**

Basic containers

- **list**
  - a = [1, 3.0, ‘abc’]
- **dict**: named list
  - a = { ‘name’: ‘John’, ‘age’: 22, ‘height’: 179. }
- **set**
  - a= set([1,2,3,2,1,7]) (*중복된 value는 제거된다*)
- **tuple**: immutable list
  - a = (1,2,3,4)
  - Use tuple if possible (because it is *faster* than list).
  - A tuple can be a ‘key’ of a dict.
- **str**: string, immutable
  - a = ‘abcdefg’

Python Data Structure Quiz

- 다음 a 데이터의 구조는?
- a[0]['x'][(1,3)]

- 추론
  - a는 0을 인덱스로 가지므로 list, tuple, dict 와 같은 구조일 것이다.
  - a[0] 는 'x'를 인덱스(키)로 가지므로 dict 와 같은 구조일 것이다.
  - a[0]['x']는 (1,3) tuple을 인덱스(키)로 가지므로 dict와 같은 구조일 것이다.

- 답은 여러개…
  - 그 중 하나가 될 수 있는 것은 a = [{ "x": { (1,3): *value* } }]

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Flow Control

- **if/elif**
  - if a == b: ← compares equal
  - elif a in b: ← ‘b’ is a container (list/dict/…)
  - elif a is True: ← ‘is’ is identity comparison. Use ‘is’ only for True/False/None if confusing
- **for**
  - for a in b: ← ‘b’ is a container (list/dict/…)
  - for i in range(10): ← ‘i’ gets from 0 to 9
- **while**
  - while True:
  - while a < 10:
- **break/continue**
- **exit**(*exit_code*)

에러검출

try:

…

except:

…

finally:

…

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Indentation

- 블럭을 형성할 때 들여쓰기는 필수
- 공백 4개로 하자. (**PEP8**)
- tab 키를 사용할 경우 에디터에서 공백 4개로 자동 변환되도록 설정
  - Vi 에서 set tabstop=4 / set expandtab

```
for i in range(10):
    if i < 5:
        print ‘less than 5’
    else:
        print ‘equal to or greater than 5’
```

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Method vs Subroutine

def

- **Subroutine**: can be called anywhere
- **Method**: Only the class or class instance can call

Import module

- import *module_name*
- import *module_name* as *different_name*
- from *module_name* import *module_member1*, *module_member2*, …

```
import os
import numpy as np
from sys import exit
```

Module Importing Error and Installation

module

```
>>> import some_module
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named some_module
```

- 현재 디렉토리에서 해당 모듈을 찾을 수 없고
- $PYTHONPATH 라는 환경 변수에서 해당 모듈을 찾을 수 없고
- sys.path 에서도 해당 모듈을 찾을 수 없을 때 발생하는 에러

- pip
  - pip search *search_keyword*
  - pip install *module_name* ← system-wide install ('sudo' is usually required)
  - pip install *module_name* --user ← installs on the home directory

- Manual installation
  - python setup.py install
  - python setup.py install --user

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Class

```
class Fruit(object):
    del_counter = 0

    def __init__(self,name,taste):
       self.name = name
       self.taste = taste
       self._price = 0

    def squeeze(self):
        print 'You got a cup of', self.name, 'juice! It tastes', self.taste + '.'

    @property
    def price(self):
        return self._price

    @price.setter
    def price(self,value):
       self._price = value

    def __del__(self):
       Fruit.del_counter += 1
```

Class member variable

Class 초기화 method

Class method

Function decorator

getter

setter

Class 소멸 method

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Class Inheritance (클래스 상속)

class

- 클래스 상속 왜 하나?

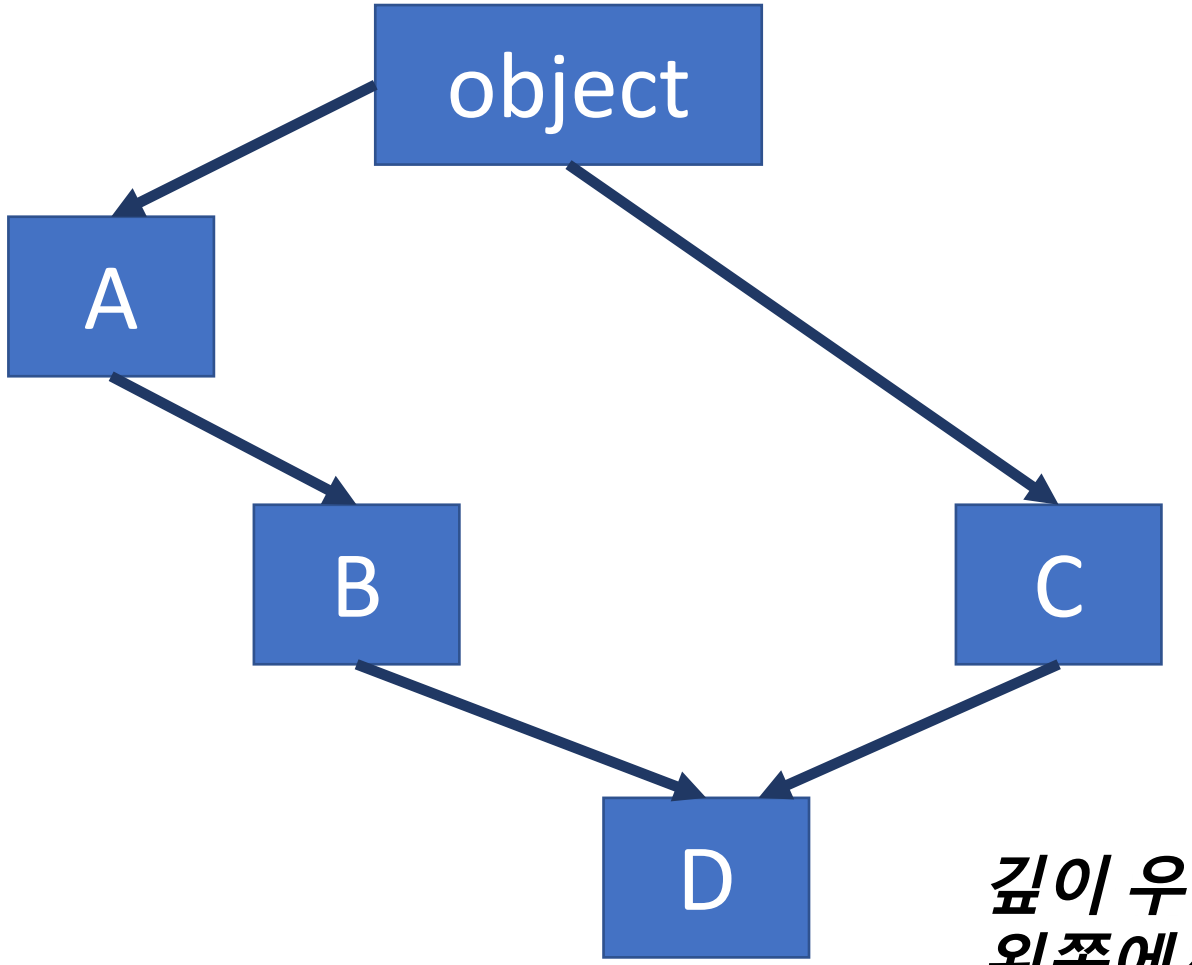
- 기존 클래스의 기능을 **확장**하기 위해서 (메소드 및 변수 추가)
- 기존 클래스의 기능을 **변경**하기 위해서 (메소드 override)

- 이렇게 하면 원래 클래스를 건드리지 않고도 코드를 수정할 수 있다.

Method Resolution Order (MRO) and **super**()

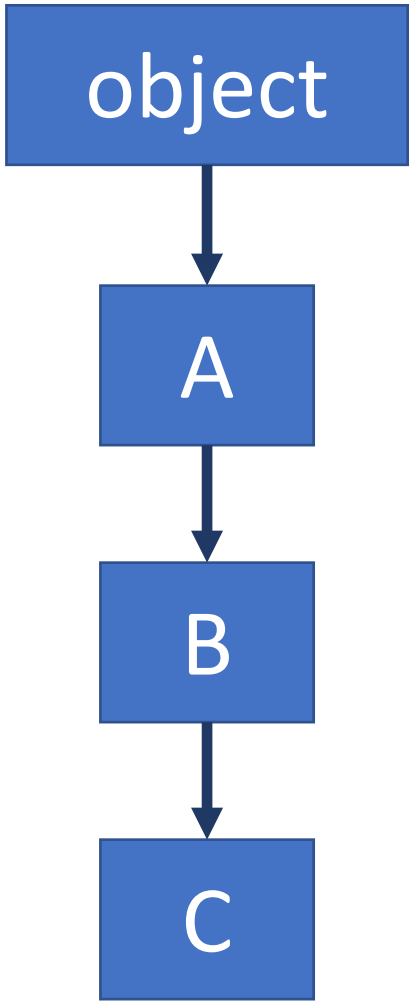
class

- super(*current_class*, *self*).*parent_method*()



깊이 우선,
왼쪽에서 오른쪽으로

MRO: D → B → A → C



MRO: C → B → A

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

File I/O

file

- f = **open**(filename,mode)
  - mode: ‘r’, ‘w’, ‘a’ (read/write/append)
  - f.write() ← 쓰기
  - f.read() ← 한번에 다 읽기
  - f.readline() ← 한줄 읽기
  - f.readlines() ← 줄별로 다 읽기
- f.close()

- with open(*filename*,*mode*) as *filehandler*: ← No need to close explicitly

- String format (% operator)
  - %3d ← 3자리 정수 (부호 포함)
  - %12.6E ← 12자리 실수 (부호 포함) 소수점 이하 6자리 (예: 1.234567E+02)
  - %7.3f ← 7자리 실수 (부호포함) 소수점 이하 3자리 (예: 123.456)
  - %4s ← 4자리 문자
  - Example: ‘output %5s %3d %7.3f’%(title, number, real_num)

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Argument Parsing

argparse

- sys.argv ← 인풋 argument가 list 형태로 저장되어 있다

```
import argparse

parser = argparse.ArgumentParser(description='argument process.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer')
parser.add_argument('-a', dest='a_option', type=int, help='a option')
parser.add_argument('-b', dest='b_option', help='b option')
parser.add_argument('-c', dest='c_option', action='store_true',
                    help='c option')

args = parser.parse_args()
```

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

(Text) File Parsing

parsing word_count

- **str** methods
  - Indexing: [*i*:*j*:*k*]
  - str.split ← 문자열을 리스트로
  - str.strip ← 좌우 공백 제거
  - str.join ← 리스트를 문자열로

- Regular Expression
  - import re
  - re.search (re.finditer, re.findall)
  - re.sub
  - re.split

- Regular expression (more)
  - Special characters
    - \s: space (\S: non-space)
    - \w: word ([a-zA-Z]) (\W: non-word)
    - \d: digit ([0-9]) (\D: non-digit)
    - . : 아무 문자
    - ^: 문자열 시작
    - $: 문자열 끝
  - Modifier
    - ?: 0개 혹은 1개
    - *: 0개 이상
    - +: 1개 이상
    - {*m*,*n*}: m 개에서 n개 사이
  - OR:
    - [abcd]: a 또는 b 또는 c 또는 d
    - (abc|def): abc 또는 def
  - NOT
    - [^a]: a 가 아닌 문자

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Path Handling

path

- **os.path.dirname**(*path*)
- **os.path.basename**(*path*)
- **os.path.abspath**(*rel_path*)

- Globbing (from glob import glob) ← *Wildcard*
  - glob('*'): all the files
  - glob('b??'): files starting with ‘b’ and ending with two characters
  - glob('[abc]*'): all the files starting with ‘a’, ‘b’, or ‘c’

- Path splitting
  - full_path.**split**(**os.sep**)

Subprocess

tmalign

- **os.system**(*command*)
  - os.system('ls')
  - os.system('cp src target')

- Capture Standard Ouput (STDOUT/STDERR)
  - p = **subprocess.Popen**(['cmd',arg1,arg2],stdout=subprocess.PIPE)
  - stdout_lines = p.stdout.readlines()
  - stderr_lines = p.stderr.readlines()

- Changing directory
  - ~~**os.system**('cd subdir')~~ ← This does not work. *Why*?
  - **os.chdir**('subdir') ← Correct way

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Temporary File

```
import tempfile
import shutil
temp_dir = tempfile.mkdtemp()
# work
shutil.rmtree(temp_dir)
```

Temporary directory

```
tf = tempfile.NamedTemporaryFile(delete=True)
tempfile_name = tf.name
# work
tf.close()
os.remove(tempfile_name) # This is only for ‘delete=False’
```

Temporary File

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Plotting

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
line1, = ax.plot(x1,y1,options)
line2, = ax.plot(x2,y2,options)
plt.show()
```

matplotlib
http://matplotlib.org

```
import Gnuplot
p = Gnuplot.Gnuplot()
p('set term X11')
p('plot "textfile" u 1:2 w lines')
```

gnuplot
http://www.gnuplot.info

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

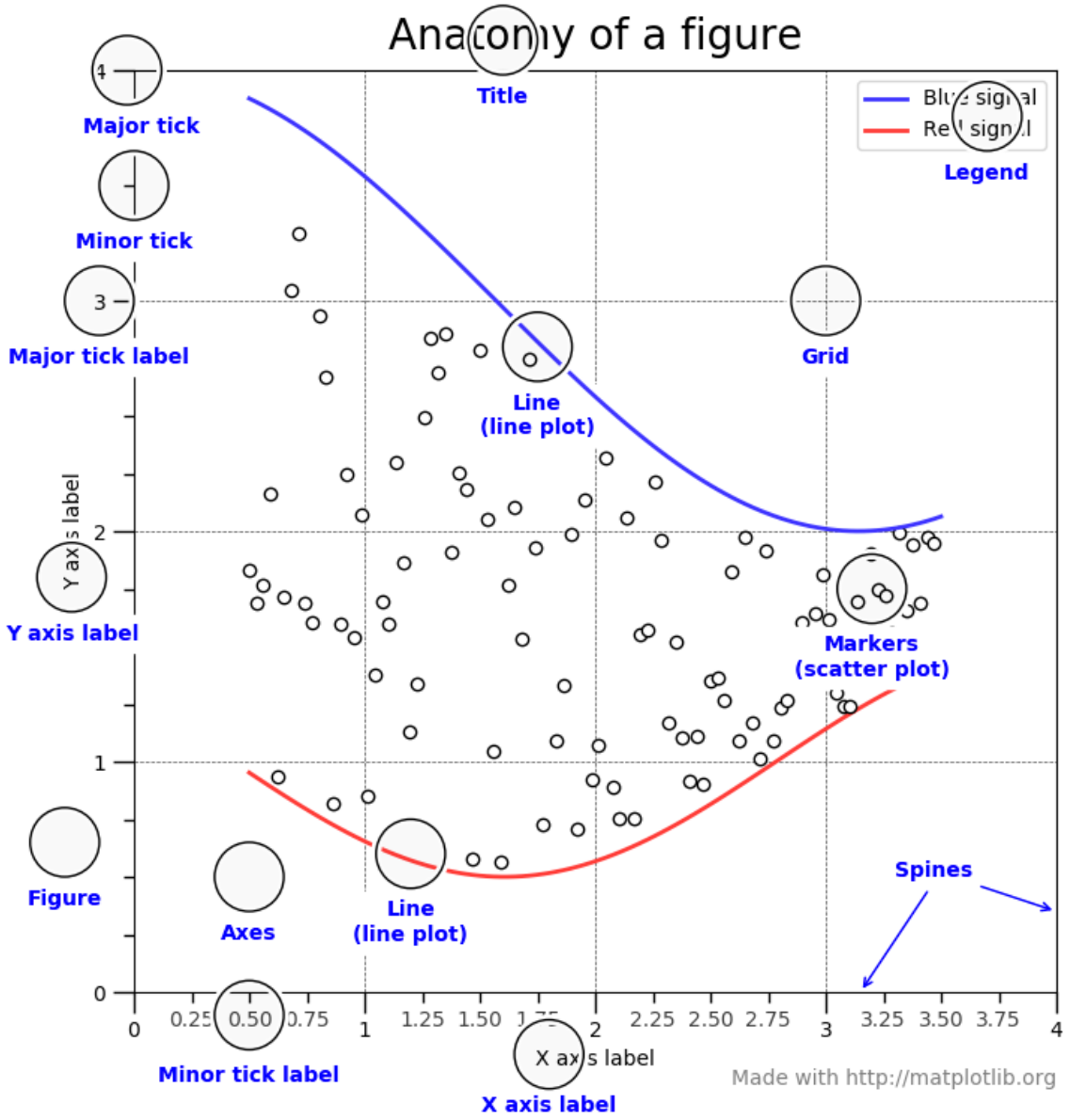
matplotlib.pyplot

plot_rama

1. figure, axes object 생성
   - figure: 한 페이지에 나타나는 이미지
   - axes: 개개의 plot

   - fig, ax = plt.subplots(*m*,*n*) (*m*행 *n*열)

   - fig = plt.figure()
   - ax = plt.subplot(*m*,*n*,*o*) (*m*행 *n*열 *o*번째)
2. plot, hist, fill, boxplot …
   - plt.plot(x,y)
   - plt.hist(x,bins=50)
3. plt.show(), fig.savefig()



Numpy (*is not a poor man's MATLAB*)

- The fundamental module for scientific computing
  - Many scientific modules are dependent on the numpy module.
- An efficient data container
  - *N*-dimensional matrix
  - Easy to communicate with C/C++ and Fortran
- Fast looping
  - Loop calculation is as fast as C/Fortran
- Many built-in functions for data handling and linear algebra

Numpy Data Types

numpy_basic

- Integer (np.int64)
- Float (np.float64) ← DEFAULT
- Complex (np.complex128)
- Bool (np.bool, but np.uint8 is preferred! 왜냐하면 bool이 C의 표준이 아님)

- Popular ways to create numpy data
  - **np.array**([1.0, 2.0, 3.0])
  - **np.empty**((2,4),dtype=np.int64)
  - **np.zeros**(10,dtype=np.uint8)
  - **np.ones**((3,3,3),dtype=np.float64)

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Numpy Data Handling

numpy_handling

- Slicing: ***i:j:k***
- Adding a dummy dimension: **newaxis** (*alias* of **None**)
- **reshape**: change into an arbitrary dimension
- **vstack**: stack vertically
- **hstack**: stack horizontally
- **concatenate**: stack in arbitrary dimension
- **ravel**: convert to one dimension
- **T**: transpose
- **copy**: copy the data
- Masking: a[a<0.5], a[[0,3,6]]

WARN: The stored data in the memory are reused as much as possible *unless it is copied*!

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Profiling

```
import time
t = time.time() # the time in seconds since the epoch
# run something
print ‘runtime is’, time.time() – t
```

- import cProfile
- python –m cProfile –s cumtime script.py

Broadcasting (implicit looping)

broadcast | distance/numpy

- 1차원 이상의 두 numpy array 를 연산(operation)할 때,
- 두 array의 차원이 같을 때는 한 array의 크기가 1 또는 다른 array의 크기와 동일해야 한다.
- 두 array의 차원이 다를 때는 차원이 낮은 array는 높은 array와 동일한 차원이 되도록 앞 쪽에 크기가 1인 차원이 추가되어 있다고 가정한다.
- 결과 array는 동일한 차원의 array가 되며 각 차원의 크기는 두 array 중 최대값에 해당한다.

- (256,100,3) *op* (1, 1, 3) → (256,100,3)
- (3,3) *op* (3) → (3,3) 1차원 (3) 은 (1,3) 으로 간주
- (m,n,1) *op* (n,p) → (m,n,p) (n,p) 는 (1,n,p)로 간주
- (m,1,n) *op* (1,p,n) → (m,p,n)

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Save/Load Data

save_load

- np.loadtxt(*fname, dtype=np.float, usecols=(0,2,4,6)*)
- np.savetxt(*fname, numpy_obj, fmt='%.4f %.4f', header='data1 data2'*)

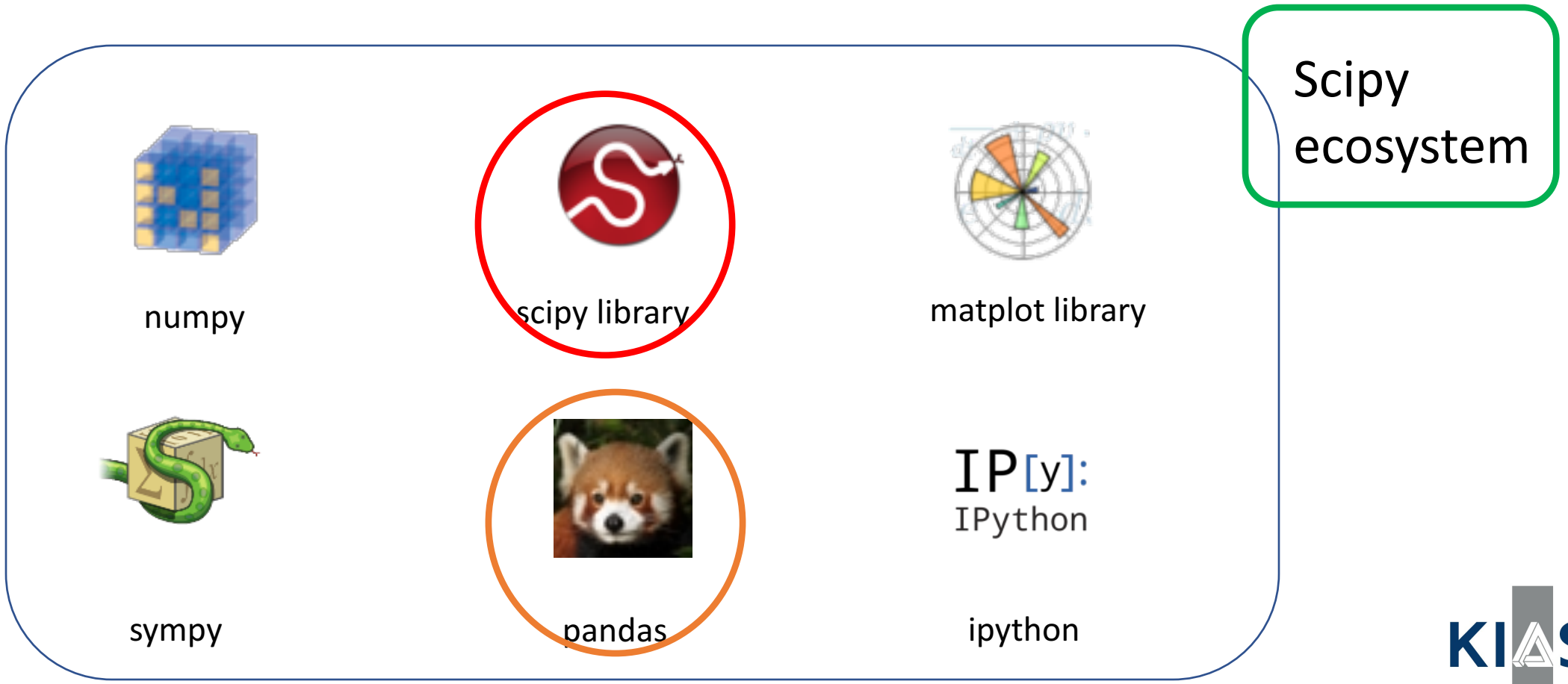
- np.save(fname,numpy_obj)
- np.load(fname)

- cPickle.dump(*data,filehandler,2*)
- data = cPickle.load(*filehandler*)

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Scipy

- SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
- 좁은 의미에서는 scipy library 를 의미



KIAS KOREA INSTITUTE FOR ADVANCED STUDY

MPI


```
from mpi4py import MPI
mpicomm = MPI.COMM_WORLD
mpisize = mpicomm.Get_size()
mpirank = mpicomm.Get_rank()
```

- mpicomm.bcast(*data,root=num*)
- mpicomm.send(*data,dest=num*)
- mpicomm.recv(*data,source=num*)
- mpicomm.scatter(*data,root=num*)
- mpicomm.gather(*data,root=num*)

- 대문자 명령어 (Bcast,Send,Recv..)와 소문자 명령어 (bcast,send,recv..)

- Python subroutine 자체의 overhead가 있기 때문에 fine-grain의 parallelization은 일의 규모가 작을 때 오히려 느려질 수도 있다. (profiling 필요)

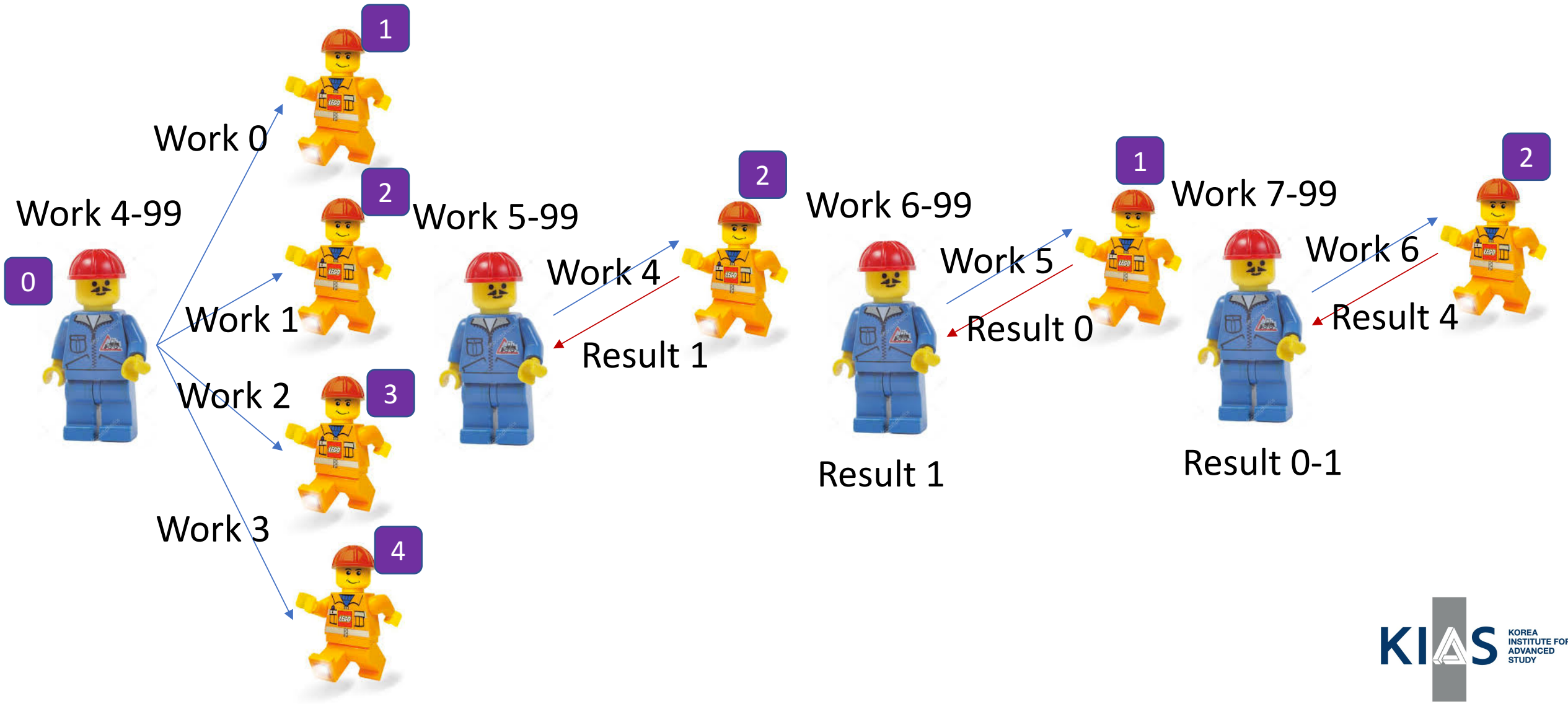
- 좀 더 추상적인 개념의 work를 설정하고 그 work를 나누어서 하는 것이 python level에서는 더 적절할 수 있다.

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Master / Slave Parallelization

mpi

나눠서 수행할 **일의 계산량이 일정하지 않을 때** 적합하다.

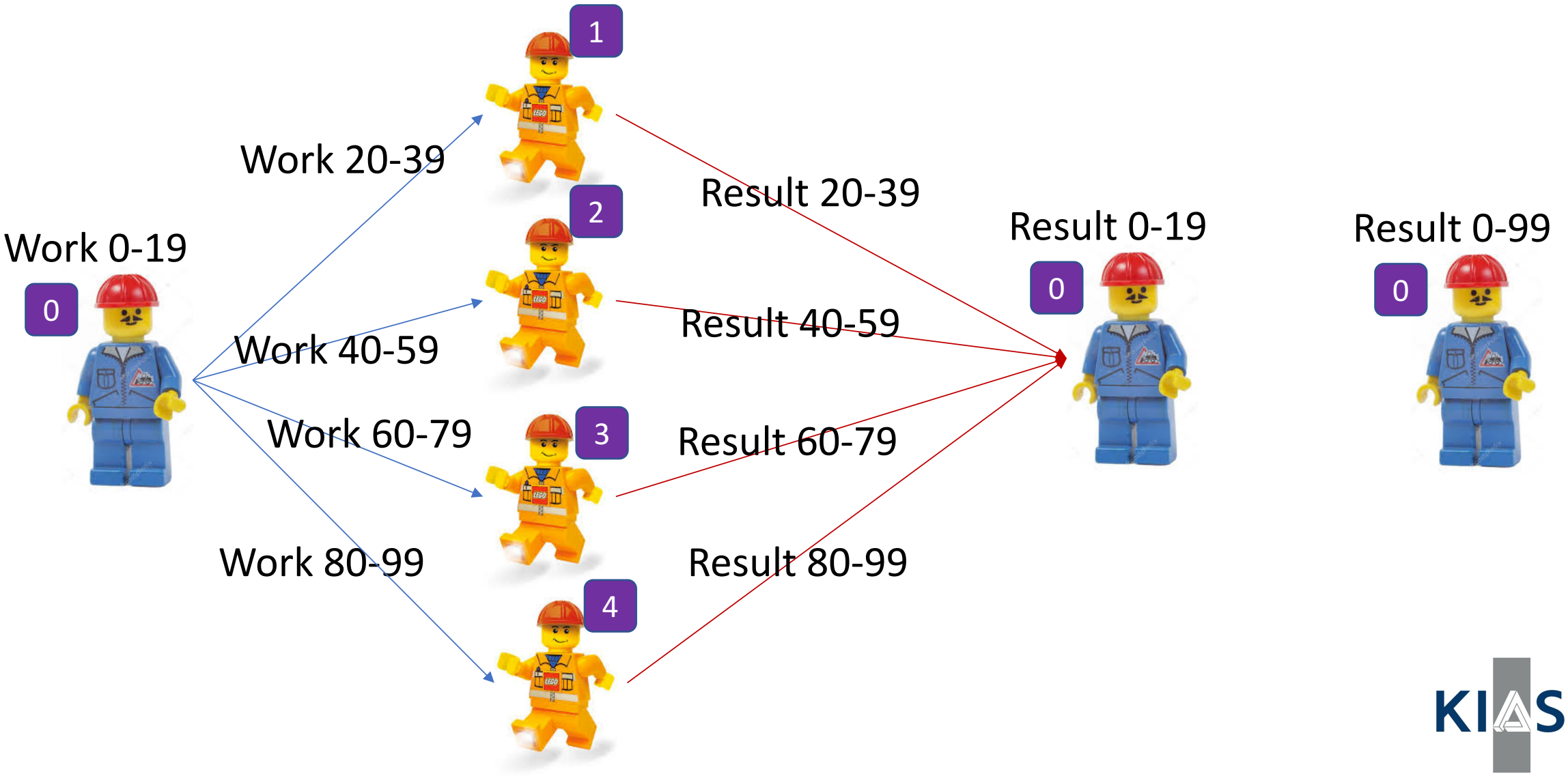


KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Equally Distributed Works

mpi

나눠서 수행할 **일의 계산량이 일정할 때** 적합하다.

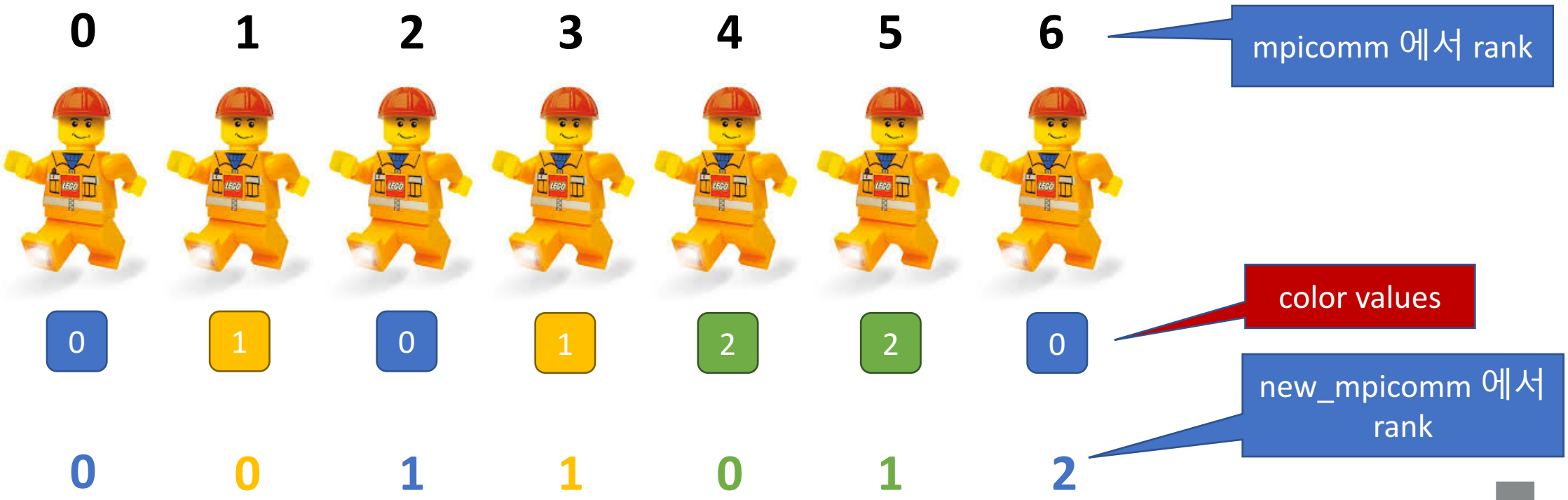


KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Splitting Communcator

mpi

- new_mpicomm = mpicomm.Split(*color*,*key*)
  - color: 같은 communicator에 들어갈 프로세서들은 같은 값을 준다.
  - key: 같은 communicator에서 rank 순서를 정하기 위한 값 (보통 mpirank를 쓰면 됨)



KIAS KOREA INSTITUTE FOR ADVANCED STUDY

MPI Tips

- 소문자 MPI 명령어 (bcast, send, recv, …)등으로 작동하는 코드를 우선 만든다.
- 처음부터 대문자 명령 (Fortran/C 스타일)으로 하는 것은 좋지 않다. Fortran/C 에서와는 달리 **Python에서는 임의의 object를 자유롭게 주고 받는 것이 쉽다는 점을 활용하자!**
- 필요에 따라 프로파일링을 해 보고 대문자 명령으로 바꾸는 것을 고려할 수 있다. (하지만 coarse-grained parallization에서는 속도 차이가 거의 없을 수도 있다.)
- Block send/recv를 **immediate (non-blocking) send/recv** 등으로 바꾸는 것을 고려 할 수 있다. (의도하지 않은 결과가 될 수도 있다. 디버깅 필요!)

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Boost Speed

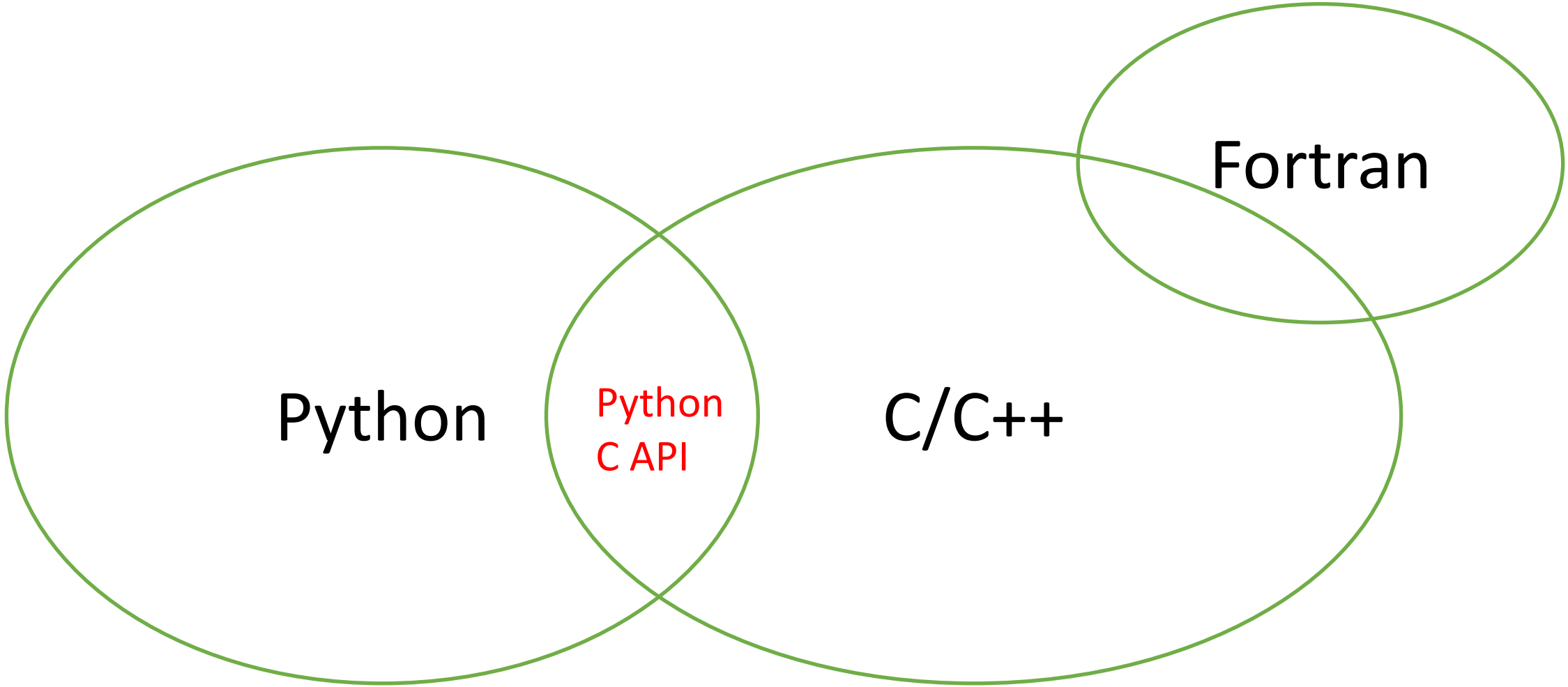
boost

distance/numba

- map / filter / reduce
- Built-in functions
- Numpy (broadcasting / masking)
- Numba: just-in-time compilation by LLVM compiler
- Glue Fortran or C/C++

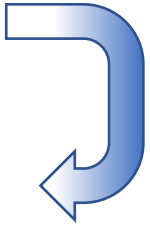
KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Glue Other Languages



Gluing the C Language

- Manual coding
- Cython
- Swig
- F2py



`c_sub.py`

Python

```
c_sub_wrap()
```

`c_sub_wrap.c`

Python C API

```
def c_sub_wrap():
    return c_sub()
```

`c_sub.c`

C

```
void c_sub() {
    …
}
```

gcc -c -o c_module.so c_sub_wrap.c c_sub.c

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Input Formats of the Wrappers

- **F2py**: (*C API code를 생성하는 input 이*) Fortran과 유사하다.
- **Cython**: (*C API code를 생성하는 input 이*) Python과 유사하다.
- **Swig**: (*C API code를 생성하는 input 이*) C와 유사하다.

- Input file 특징
  - F2py: 비교적 쉽고 단순하다. (기능이 한정되어 있다.)
  - Cython: cython 언어 (python의 상위 호환성을 지향)
  - Swig: Python C API를 써야 할 경우가 많다.

- 각 input이 *사실상* 새로운 언어다!
- 그냥 f2py와 cython만 배우자!

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Usage of f2py

- Compile in one step (*비추천*)
  - f2py -c -m module_name fort01.f90 fort.02.f90 …
- Generate a signature file
  - f2py -h mysig.pyf -m module_name fort01.f90 fort02.f90 …
- Compile a signature file and link fortran object files
  - f2py -c mysig.pyf fort01.o fort02.o …
- Generate a C wrapper using a signature file
  - f2py mysig.pyf

Recommended Procedure for f2py

1. Signature file을 만든다.
   - f2py -h mysig.pyf -m module_name fort01.f90 fort02.f90 …
2. Signature file을 열어서 편집
   - Python에서 호출할 필요가 없는 subroutine 및 변수를 지운다.
   - Python에서 호출할 필요가 있는 subroutine일 경우 indent(in), intent(out)을 적절히 고친다. 입출력과 관계 없는 변수는 지운다.
   - (numpy) array를 주고 받을 때는 각 array의 dimension을 shape 명령을 통해서 알 수 있으므로 intent(hide)할 수 있다.
3. 컴파일은 setup.py를 이용한다.

Example of a .pyf File

distance/f2py

```
python module distance_fortran
    interface
        subroutine calc_distance(c1_size,n1,c2_size,n2,c1,c2,distance)
            integer, optional,intent(in),check(shape(c1,0)==c1_size),depend(c1) :: c1_size=shape(c1,0)
            integer, optional,intent(in),check(shape(c1,1)==n1),depend(c1) :: n1=shape(c1,1)
            integer, optional,intent(in),check(shape(c2,0)==c2_size),depend(c2) :: c2_size=shape(c2,0)
            integer, optional,intent(in),check(shape(c2,1)==n2),depend(c2) :: n2=shape(c2,1)
            real*8 dimension(c1_size,n1),intent(in) :: c1
            real*8 dimension(c2_size,n2),intent(in) :: c2
            real*8 dimension(c1_size,c2_size),intent(out),depend(c1_size,c2_size) :: distance
        end subroutine calc_distance
    end interface
end python module distance_fortran
```

optional: python side에서 optional argument로 처리된다. Argument의 위치가 뒤로 이동한다.
intent(hide): python side에서 해당 변수에 대한 argument를 받지 않는다 (다른 변수로부터 추정)
intent(out): python side에서 return value로 처리된다. 즉, 입력 argument에서는 사라진다.
check: assert와 유사하다. 변수의 입력값이 올바른지 확인한다.
depend: 해당 변수의 값이 다른 변수의 값에 의존적일 때 표시한다.

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

COMMON / MODULE

tmscore

```
python module fortran_wrap
   interface
      real*8 :: v1, v2 ! common variables
      common /common_grp/ v1, v2 ! Grouping common variables
      module module_name
         integer :: mv1, mv2 ! module variables
         subroutine aaa() ! module subroutine
            …
         subroutine
      end module
   end interface
end python module fortran_wrap
```

COMMON은 F77 스타일로 메모리 관리 등에서 문제점이 있으므로 전역 변수가 필요한 경우에는 F90 스타일의 module을 이용한다. COMMON은 old code를 binding할 때만 사용하도록 하자.

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Cython

- Cython is a language.
- Cython is not CPython. (CPython 은 Python 언어를 C로 구현한 것)
- Cython is a (supposedly) superset of Python. (Python에서 유효한 문법은 Cython에서도 유효하다.)
- Cython is a converter of Python code to C/C++ code

- Cython으로 할 수 있는 일
  - 기존의 python 코드를 수정하여 C 수준의 속도를 얻기
  - C로 작성된 서브루틴을 wrapping하여 python에서 사용하기

Converting Python Code to Cython Code

cython/fibç cython/dsum

- Python code를 그대로 사용해도 된다. 하지만 성능을 더 향상시키기 위해서는 아래와 같은 작업을 수행한다.
- 서브루틴의 argument의 type이 있다면 명시한다.
  - def test(**list** data, **int** n1, **float** parm):
- 서브루틴에서 사용하는 변수의 type이 있다면 미리 정의한다.

```
def test():
    cdef int i, j, k
    cdef double v1, v2, v3
    cdef list result = []
```

- Numpy data type 선언
  - import numpy as np
  - cimport numpy as np
  - np.ndarray [np.double_t,ndim=2] data1, data2
  - np.ndarray [np.int_t,ndim=1] input
- Cython annotation (cython –a) 을 이용하여 의도대로 optimize가 되었는지 확인한다.

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

def / cdef / cpdef

distance/cython

distance/c_cython_wrap

- def
  - Python functions
  - Returns a python object
  - Not accessible from C-side
- cdef
  - C functions
  - Returns c types
  - Not accessible from Python-side
- cpdef
  - C functions
  - Returns c types
  - Accessible from both sides, Python and C

Pointers

```
from cython.operator cimport dereference as deref

cdef int *a, *b
cdef int c
a = NULL # NULL pointer
c = 3
a = &c # address

# dereference
print a[0]
print deref(a)
```

KIAS KOREA INSTITUTE FOR ADVANCED STUDY

Type Extension vs C++ Class

cython/type_ext | cython/cpp

```
from libc.stdlib cimport malloc, free

cdef extern from "shape.h":
    cdef struct _Circle:
       double x,y
       double r

    ctypedef _Circle Circle

    cdef double area(Circle *c)
    cdef double circumference(Circle *c)
    cdef void center(Circle *c, double *x, double *y)

cdef class circle:
    cdef Circle *_pt
    def __cinit__(self,x,y,r):
       self._pt = <Circle*> malloc(sizeof(Circle))
       self._pt.x = x
       self._pt.y = y
       self._pt.r = r

    cpdef double area(self):
        return area(self._pt)

    cpdef double circumference(self):
        return circumference(self._pt)

    cpdef tuple center(self):
       cdef double x,y
       center(self._pt, &x, &y)
       return (x,y)

    def __dealloc__(self):
       free(self._pt)
```

```
# distutils: language = c++

cdef extern from "shape.h" namespace "shape":
    cdef cppclass Circle:
        Circle(double xin, double yin, double rin)
       double area()
        double circumference()
        void center(double *xout, double *yout)

cdef class circle:
    cdef Circle *_pt

    def __cinit__(self,x,y,r):
       self._pt = new Circle(x,y,r)

    def area(self):
        return self._pt.area()

    def circumference(self):
        return self._pt.circumference()

    def center(self):
       cdef double x,y
       self._pt.center(&x, &y)
       return (x,y)

    def __dealloc__(self):
       del self._pt
```

KIAS KOREA INSTITUTE FOR ADVANCED STUDY