AI VIET NAM
@aivietnam.edu.vn

Data Structure

String & List and Their Applications

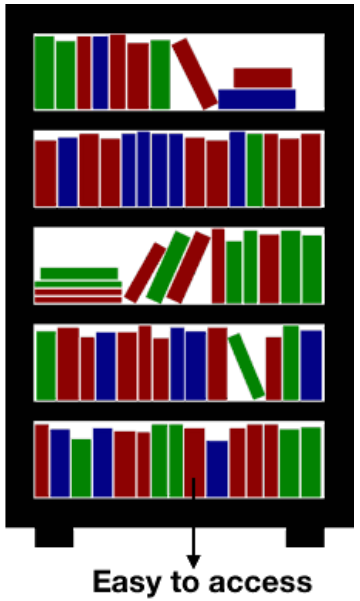
Vinh Dinh Nguyen
PhD in Computer Science

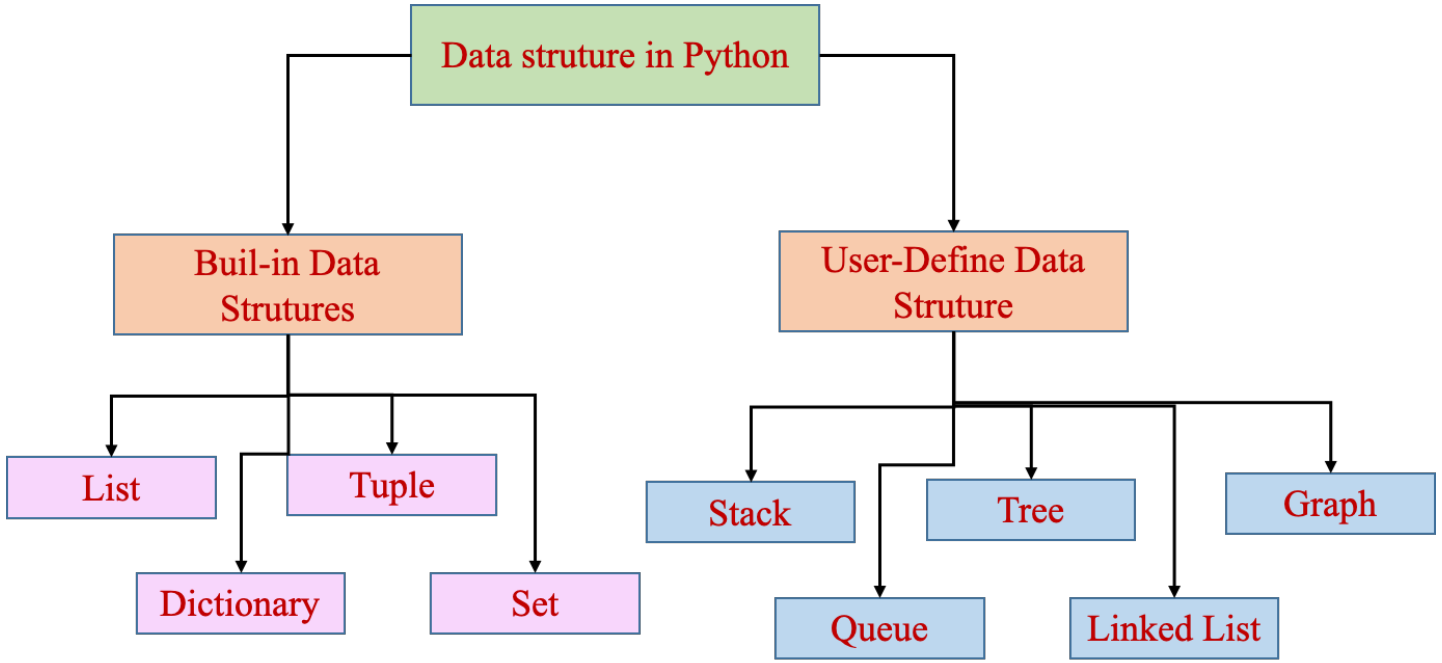
Outline

- Introduction to Data Structure
- String
- List and Algorithms
- 2D List and Example
- Data Pre-processing Using List
- Summary

What is a Data Structure?

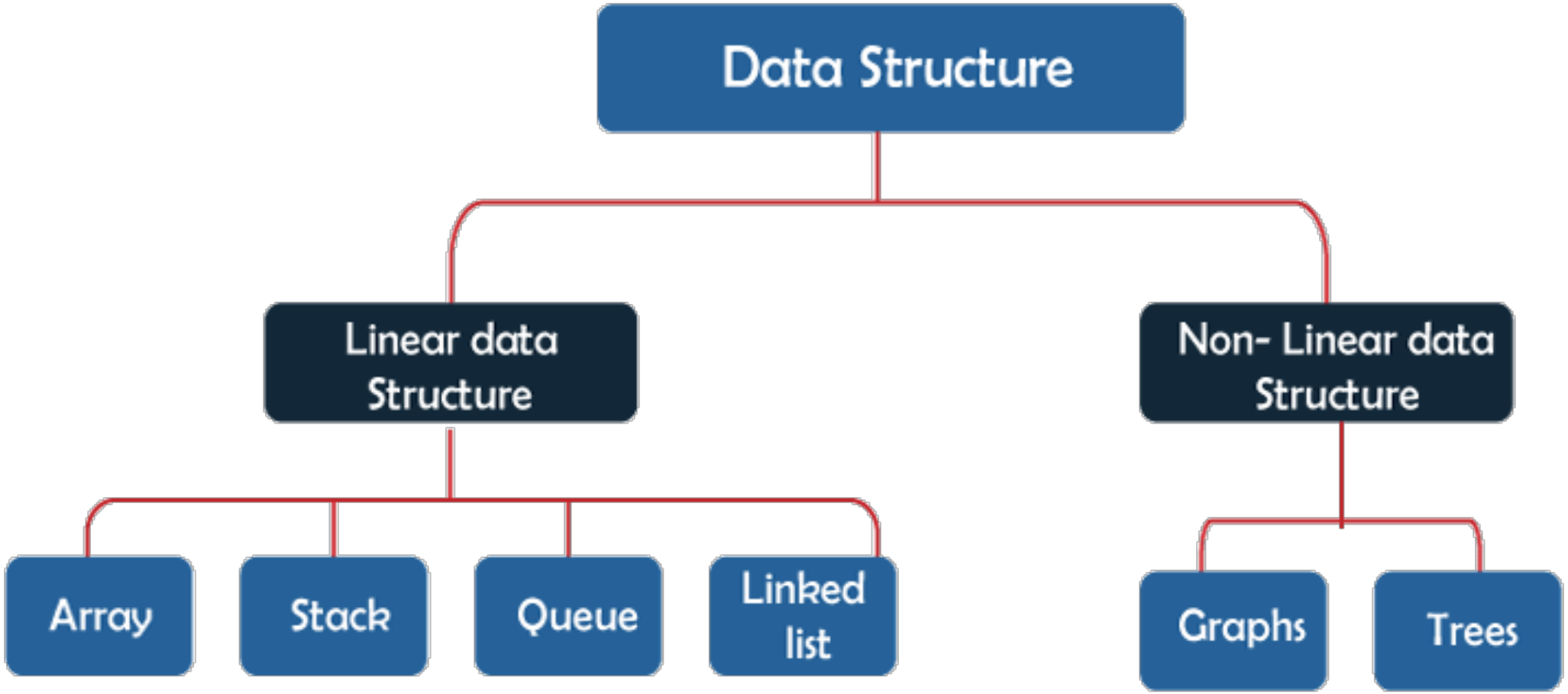
A data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.





What is a Data Structure?

❖ Common Data Structure for Machine Learning



Outline

- Introduction to Data Structure
- String
- List and Algorithms
- 2D List and Example
- Data Pre-processing Using List
- Summary

String Motivation

In Python, numerical values can be stored using different data types, each suitable for specific purposes.

```python
# Integer
a = 10
b = -200

# Floating-point
pi = 3.14159
e = 2.71828

# Complex number
z = 1 + 2j

# Display the values and their types
print(f"a: {a}, type: {type(a)}")
print(f"pi: {pi}, type: {type(pi)}")
print(f"z: {z}, type: {type(z)}")
```

How about the text data?

In Python, Text data can used stored using strings in Python. You can create strings using single quotes ('), double quotes ("), triple single quotes ('''), or triple double quotes (""").

```python
# Single quotes
single_quoted_string = 'Hello, World!'

# Double quotes
double_quoted_string = "Hello, World!"

# Triple single quotes (useful for multi-line strings)
triple_single_quoted_string = '''Hello,
World!'''

# Triple double quotes (useful for multi-line strings)
triple_double_quoted_string = """Hello,
World!"""
```

A string is a sequence of characters

Strings in Python are a versatile and powerful data type for handling and manipulating text data

String

❖ Create and iterate a string

name = 'AI'

name =	A	I
index	0	1

```python
# create a string
name = 'AI'
print(name)
```

```
AI
```

```python
# iterate a string
name = 'AI'
for character in name:
    print(character)
```

```
A
I
```

```python
# iterate a string
name = 'AI'
length = 2

for index in range(length):
    print(name[index])
```

```
A
I
```

String

❖Can you guess the output?

```
str = 'From aivnresearch@eiu.edu.vn Sat Jan  5 09:14:16
    2023'
atpos = str.find('@')
sppos = str.find(' ',atpos)
host = str[atpos+1 : sppos]
print(host)
```

String

❖Application of String Data Type in Python

Machine Learning: In machine learning applications, strings are used to represent text data

```
import pandas as pd
df = pd.read_csv('/content/student-mat.csv', sep=";")
df.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6

Student Performance Dataset from UCI Machine Learning Repository

Outline

- Introduction to Data Structure
- String
- List and Algorithms
- 2D List and Example
- Data Pre-processing Using List
- Summary

List Motivation

Develop a program to store hundred usernames entered by the user

```
#please enter hundred names
name1 = input("Enter name 1: ")
name2 = input("Enter name 2: ")
name3 = input("Enter name 3: ")
name4 = input("Enter name 4: ")
name5 = input("Enter name 5: ")
name6 = input("Enter name 6: ")
name7 = input("Enter name 7: ")
name8 = input("Enter name 8: ")
name9 = input("Enter name 9: ")
name10 = input("Enter name 10: ")
...
```

PROBLEM

List Solution

```
name_list = []
for i in range(100):
  name = input("Enter name " + str(i+1) + ": ")
  name_list.append(name)
print(name_list)
```

List

❖ A container that can contain elements

list_name = [element-1, …, element-n]

```
// create a list
data = [6, 5, 7, 1, 9, 2]
```

data =	6	5	7	1	9	2
index	0	1	2	3	4	5

```
# danh sách trống
emty_list = []

# danh sách số tự nhiên nhỏ hơn 10
my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# danh sách kết hợp nhiều kiểu dữ liệu
mixedList = [True, 5, 'some string', 123.45]
n_list = ["Happy", [2,0,1,5]]

#danh sách các loại hoa quả
shoppingList = ['táo', 'chuối', 'cherries', 'dâu', 'mận']
```

AI VIET NAM
@aivietnam.edu.vn

List

❖ Index

data = [4, 5, 6, 7, 8, 9]

Forward index	0	1	2	3	4	5
	4	5	6	7	8	9
Backward index	-6	-5	-4	-3	-2	-1

data[0]
4

data[3]
7

data[-1]
9

data[-3]
7

❖ Slicing

list[start:end:step]

data = [4, 5, 6, 7, 8, 9]

	4	5	6	7	8	9
Forward index	0	1	2	3	4	5

data[:3]

4	5	6

data[2:4]

6	7

data[3:]

7	8	9

Giá trị mặc định của start là 0, của end là len(list), và của step là 1

List

❖ Add an element

data =

6	5	7	1	9	2

data.append(4) # thêm 4 vào vị trị cuối list

data =

6	5	7	1	9	2	4

data =

6	5	7	1	9	2

data.insert(0, 4) # thêm 4 vào vị trị có
index = 0

data =

4	6	5	7	1	9	2

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.append(4)
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[6, 5, 7, 1, 9, 2, 4]
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.insert(0, 4)
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[4, 6, 5, 7, 1, 9, 2]
```

List

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data[1] = 4
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[6, 4, 7, 1, 9, 2]
```

```python
data = [6, 5, 7, 1]
print(data)
data.extend([9, 2])
print(data)
```

```
[6, 5, 7, 1]
[6, 5, 7, 1, 9, 2]
```

❖ Updating an element

data =

6	5	7	1	9	2

thay đổi phần tử thứ 1
data[1] = 4

data =

6	4	7	1	9	2

❖ Add a list of elements

data =

6	5	7	1

data.extend([9, 2]**)** # thêm 9 và 2 vào vị trí cuối list

data =

6	5	7	1	9	2

List

❖ + and * operators

data1 =

6	5	7

data2 =

1	9	2

nối 2 list
data = data1 + data2

data =

6	5	7	1	9	2

data =

6	5

nhân list với một số nguyên
data_m = data * 3

data_m =

6	5	6	5	6	5

```python
data1 = [6, 5, 7]
data2 = [1, 9, 2]

# concatenate
data = data1 + data2
print(data)
```

```
[6, 5, 7, 1, 9, 2]
```

```python
data = [6, 5]

# multiply with a number
data_m = data*3
print(data_m)
```

```
[6, 5, 6, 5, 6, 5]
```

List

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.sort()
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[1, 2, 5, 6, 7, 9]
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.sort(reverse = True)
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

❖ sort() – Sắp xếp các phần tử

data =

6	5	7	1	9	2

data.sort()

data =

1	2	5	6	7	9

data =

6	5	7	1	9	2

data.sort(reverse = True)

data =

9	7	6	5	2	1

List

❖ Deleting an element

data =

6	5	7	1	9	2

data.pop(2) # tại vị trí index = 2

data =

6	5	1	9	2

data =

6	5	7	1	9	2

data.remove(5) # xóa phần tử đầu tiên
có giá trị là 5

data =

6	7	1	9	2

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.pop(2)  # by index
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[6, 5, 1, 9, 2]
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.remove(2) # by value
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[6, 5, 7, 1, 9]
```

```python
data = [6, 5, 2, 1, 9, 2]
print(data)
data.remove(2) # by value
print(data)
```

```
[6, 5, 2, 1, 9, 2]
[6, 5, 1, 9, 2]
```

List

```python
data = [6, 5, 7, 1, 9, 2]
print(data)

del data[1:3]
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[6, 1, 9, 2]
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)

data.clear()
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[]
```

❖ Delete elements

data =

6	5	7	1	9	2

xóa phần tử thứ 1 và 2
del data[1:3]

data =

6	1	9	2

data =

6	5	7	1	9	2

data.clear()

data = []

List

index() – Trả về vị trí đầu tiên

data =

6	5	7	1	9	2

trả về vị trí của phần tử đầu tiên có giá trị là 9
data.index(9) = 4

```
data = [6, 5, 7, 1, 9, 2]
print(data)

indexOf9 = data.index(9)
print(indexOf9)
```

```
[6, 5, 7, 1, 9, 2]
4
```

reverse() – Đảo ngược vị trí các phần tử

data =

6	5	7	1	9	2

data.reserse()

data =

2	9	1	7	5	6

```
data = [6, 5, 7, 1, 9, 2]
print(data)

data.reverse()
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[2, 9, 1, 7, 5, 6]
```

List

```python
data = [6, 5, 7, 1, 9, 2]
print(data)

numOf7 = data.count(7)
print(numOf7)
```

```
[6, 5, 7, 1, 9, 2]
1
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)

aCopy = data.copy()
print(aCopy)
```

```
[6, 5, 7, 1, 9, 2]
[6, 5, 7, 1, 9, 2]
```

count() – Trả về số lần xuất hiện của một phần tử

data =

6	5	7	1	9	2

trả về số lần phần tử 7 xuất hiện trong list

data.count(7) = 1

copy() – copy một list

data =

6	5	7	1	9	2

data_copy = data.copy()

data_copy =

6	5	7	1	9	2

List

❖Built in Functions for List

len(), min(), and max()

data =

6	5	7	1	9	2

trả về số phần tử
len(data) = 6

trả về số phần tử có giá trị nhỏ nhất
min(data) = 1

trả về số phần tử có giá trị lớn nhất
max(data) = 9

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
```

```
[6, 5, 7, 1, 9, 2]
```

```python
# get a number of elements
length = len(data)
print(length)
```

```
6
```

```python
# get the min and max values
print(min(data))
print(max(data))
```

```
1
9
```

List

❖Built in Functions for List

❖ sorted(aList) – Sắp xếp các phần tử

sorted(iterable, reverse=reverse)

data =

6	5	7	1	9	2

sorted_data = sorted(data)

sorted_data =

1	2	5	6	7	9

data =

6	5	7	1	9	2

sorted_data = sorted(data, reverse=True)

sorted_data =

9	7	6	5	2	1

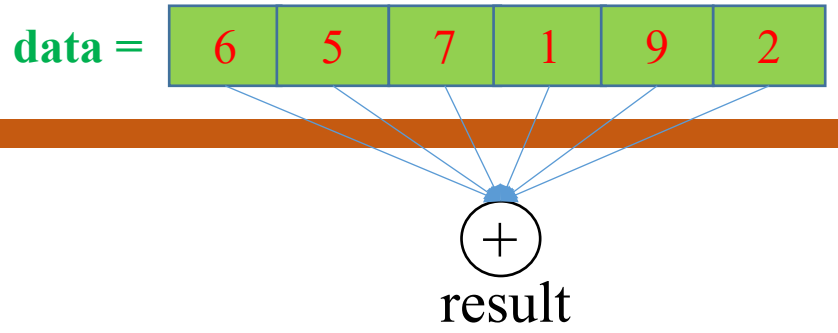
```python
# sorted
data = [6, 5, 7, 1, 9, 2]
print(data)

sorted_data = sorted(data)
print(sorted_data)
```

```
[6, 5, 7, 1, 9, 2]
[1, 2, 5, 6, 7, 9]
```

```python
# sorted
data = [6, 5, 7, 1, 9, 2]
print(data)

sorted_data = sorted(data, reverse=True)
print(sorted_data)
```

```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

List



❖Built in Functions for List

sum()

$$summation = \sum_{i=0}^{n} data_i$$

data =					
6	5	7	1	9	2

tính tổng
sum(data) = 30

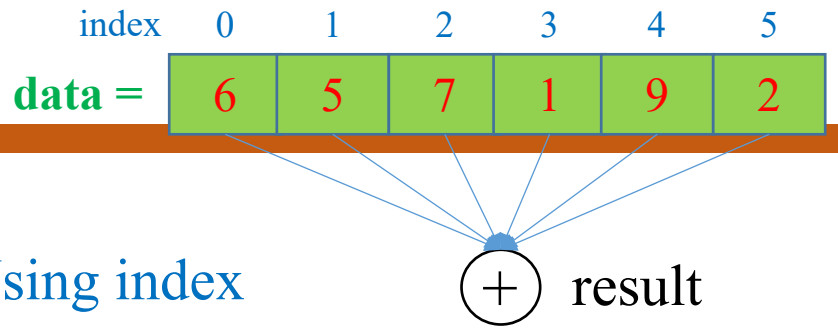
```python
data = [6, 5, 7, 1, 9, 2]
print(data)

summation = sum(data)
print(summation)
```

```
[6, 5, 7, 1, 9, 2]
30
```

```python
# custom summation - way 1
def computeSummation(data):
    result = 0

    for value in data:
        result = result + value

    return result

# test
data = [6, 5, 7, 1, 9, 2]
summation = computeSummation(data)
print(summation)
```

```
30
```

List



❖Built in Functions for List

sum()

$$summation = \sum_{i=0}^{n} data_i$$

data =					
6	5	7	1	9	2

tính tổng
sum(data) = 30

```
data = [6, 5, 7, 1, 9, 2]
print(data)

summation = sum(data)
print(summation)
```

```
[6, 5, 7, 1, 9, 2]
30
```

Using index

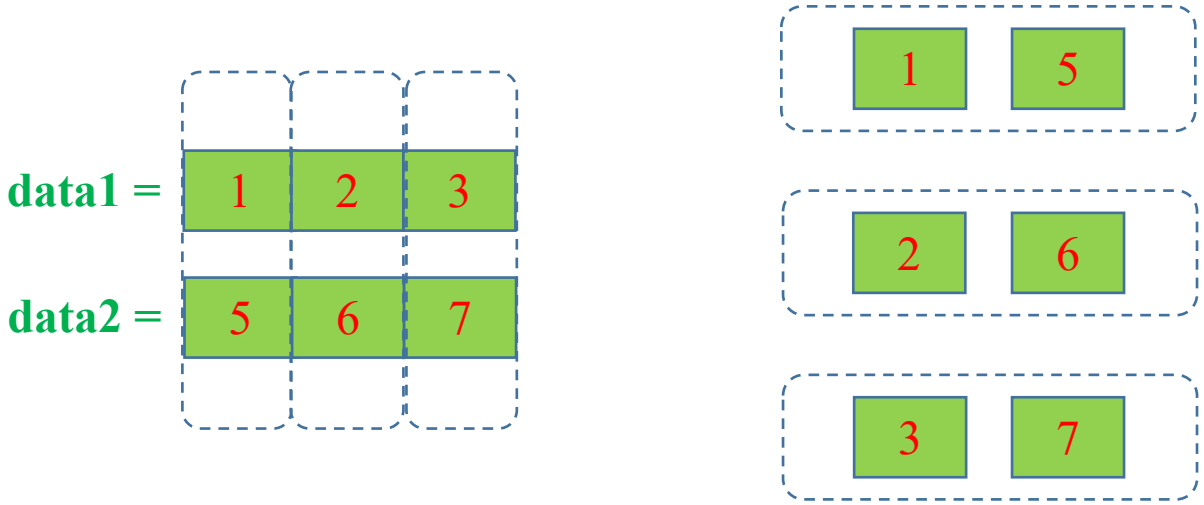
```
# custom summation - way 2
def computeSummation(data):
    result = 0

    length = len(data)
    for index in range(length):
        result = result + data[index]

    return result

# test
data = [6, 5, 7, 1, 9, 2]
summation = computeSummation(data)
print(summation)
```

```
30
```

List

❖Built in Functions for List

zip()




```python
l1 = [1, 2, 3]
l2 = [5, 6, 7]

# print in pairs
length = len(l1)
for i in range(length):
    print(l1[i], l2[i])
```

```
1 5
2 6
3 7
```

```python
l1 = [1, 2, 3]
l2 = [5, 6, 7]

# print in pairs
for v1, v2 in zip(l1, l2):
    print(v1, v2)
```

```
1 5
2 6
3 7
```

List

❖Built in Functions for List

reversed()

data =

6	1	7

reversed(data) =

7	1	6

```python
# for and list
data = [6, 1, 7]
for value in data:
    print(value)
```

```
6
1
7
```

```python
# reversed
data = [6, 1, 7]
for value in reversed(data):
    print(value)
```

```
7
1
6
```

List

❖Built in Functions for List

enumerate()

data =

6	1	7

enumerate(data) =

6	1	7
0	1	2

index

```python
# get index and value
data = [6, 1, 7]

length = len(data)
for index in range(length):
    print(index, data[index])
```

```
0 6
1 1
2 7
```

```python
# enumerate
data = [6, 1, 7]
for index, value in enumerate(data):
    print(index, value)
```

```
0 6
1 1
2 7
```

Examples

Sum of even numbers

data =

6	5	7	1	9	2

```python
# sum of even number
def sum1(data):
    result = 0

    for value in data:
        if value%2 == 0:
            result = result + value

    return result

# test
data = [6, 5, 7, 1, 9, 2]
summation = sum1(data)
print(summation)
```

```
8
```

Sum of elements with even indices

data =

6	5	7	1	9	2

```python
# sum of numbers with even indices
def sum2(data):
    result = 0

    length = len(data)
    for index in range(length):
        if index%2 == 0:
            result = result + data[index]

    return result

# test
data = [6, 5, 7, 1, 9, 2]
summation = sum2(data)
print(summation)
```

```
22
```

List Comprehension

```python
# square function
def square(data):
    result = []

    for value in data:
        result.append(value*value)

    return result
```

omitted

```python
# using list comprehension
def square(data):
    result = [value*value for value in data]

    return result
```

added

[expression for item in iterable]

```python
# using list comprehension
def square(data):
    result = [value*value for value in data]

    return result

# test
data = [6, 5, 7, 1, 9, 2]
print(data)
data_s = square(data)
print(data_s)
```

```
[6, 5, 7, 1, 9, 2]
[36, 25, 49, 1, 81, 4]
```

List Comprehension

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases}$$

$$\mathrm{ReLU}'(x) = \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0 \end{cases}$$

ReLU Function

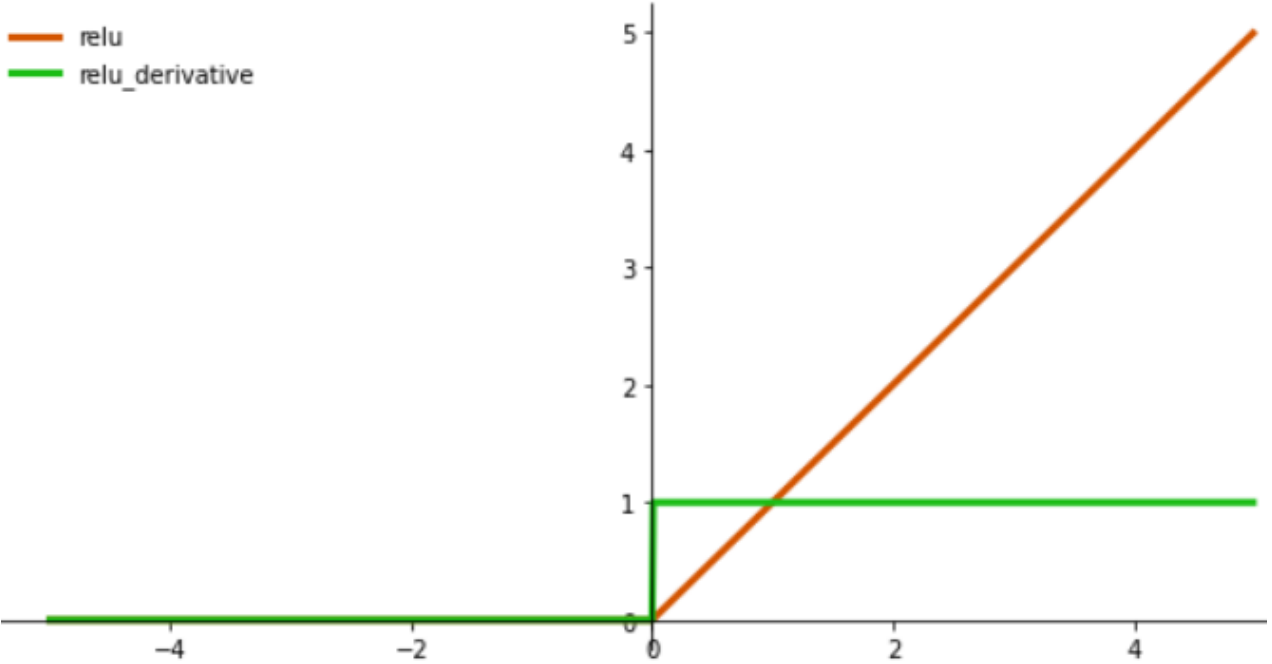
data =

1	5	-4	3	-2

data_a = ReLU(data)

data_a =

1	5	0	3	0




```python
def relu(x):
    result = 0
    if x > 0:
        result = x

    return result

def reluForList(data):
    result = [relu(x) for x in data]
    return result

# test
data = [1, 5, -4, 3, -2]
print(data)
data_a = reluForList(data)
print(data_a)
```

```
[1, 5, -4, 3, -2]
[1, 5, 0, 3, 0]
```

List Comprehension

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if } x \le 0 \\ x & \text{if } x > 0 \end{cases}$$

$$\mathrm{ReLU}'(x) = \begin{cases} 0 & \text{if } x \le 0 \\ 1 & \text{if } x > 0 \end{cases}$$

ReLU Function

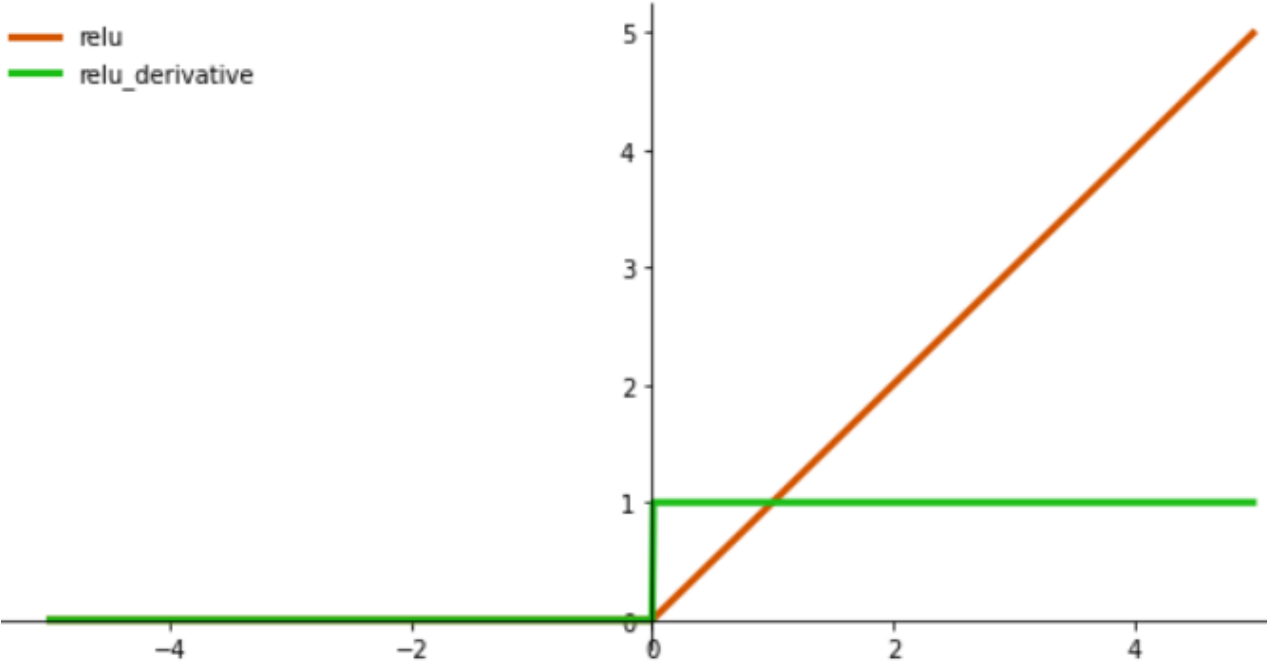
data =

1	5	-4	3	-2

data_a = ReLU(data)

data_a =

1	5	0	3	0




```
result = 0
if x > 0:
    result = x
```

```
# relu function
def relu(data):
    result = [x if x>0 else 0 for x in data]
    return result

# test
data = [1, 5, -4, 3, -2]
print(data)
data_a = relu(data)
print(data_a)
```

```
[1, 5, -4, 3, -2]
[1, 5, 0, 3, 0]
```

List Comprehension

[expression for x in data if condition]

```python
# quiz 1
data = [1, 5, -4, 3, -2]
print(data)

data_a = [x if x>0 else 0 for x in data]
print(data_a)
```

[1, 5, 0, 3, 0]

```python
# quiz 2
data = [1, 5, -4, 3, -2]
print(data)

data_a = [x if x>0 for x in data]
print(data_a)
```

invalid syntax

```python
# quiz 3
data = [1, 5, -4, 3, -2]
print(data)

data_a = [x for x in data if x>0]
print(data_a)
```

[1, 5,3]

```python
# quiz 4
data = [1, 5, -4, 3, -2]
print(data)

data_a = [x for x in data if x>0 else 0]
print(data_a)
```

invalid syntax

List Sorting

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.sort()
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[1, 2, 5, 6, 7, 9]
```

```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.sort(reverse = True)
print(data)
```

```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

```python
# sorted
data = [6, 5, 7, 1, 9, 2]
print(data)

sorted_data = sorted(data)
print(sorted_data)
```

```
[6, 5, 7, 1, 9, 2]
[1, 2, 5, 6, 7, 9]
```

```python
# sorted
data = [6, 5, 7, 1, 9, 2]
print(data)

sorted_data = sorted(data, reverse=True)
print(sorted_data)
```

```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

Algorithms on List

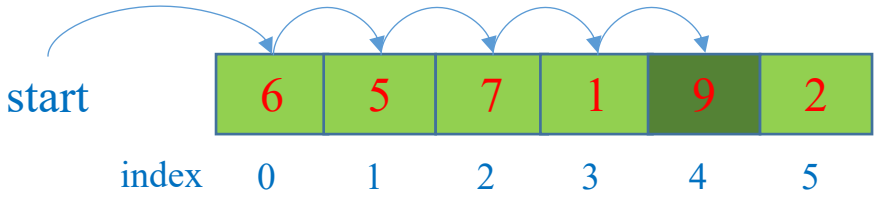
❖ Linear searching

data =

6	5	7	1	9	2
0	1	2	3	4	5

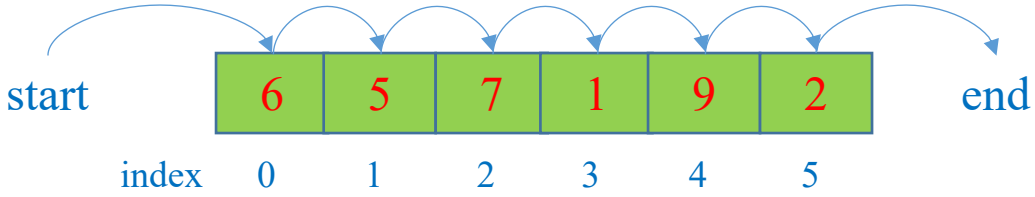
index

Searching for 9

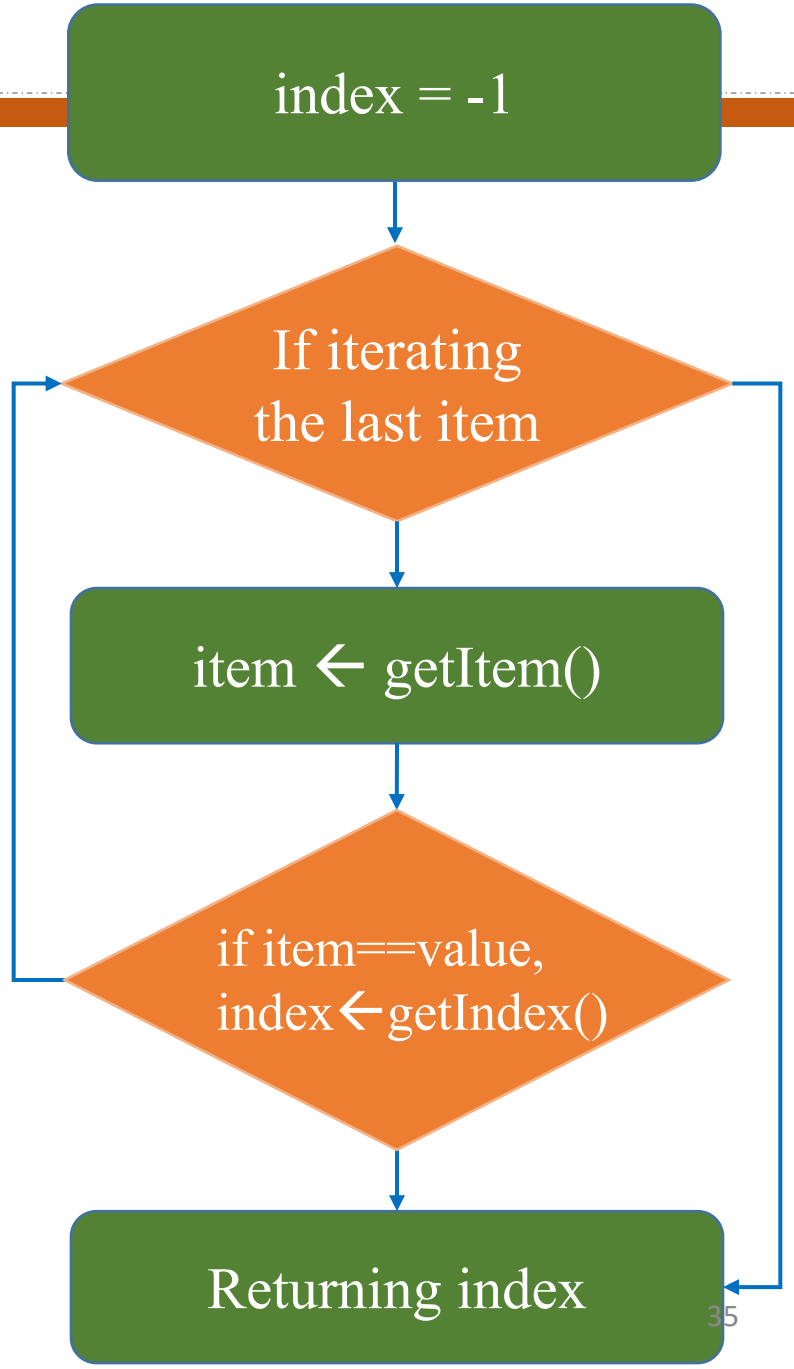


Returning 4

Searching for 8



Returning ?



Algorithms on List

❖ **Sorting using min(), remove(), and append()**

data =

6	5	7	1	9	2

result = [ ]

min(data) = 1

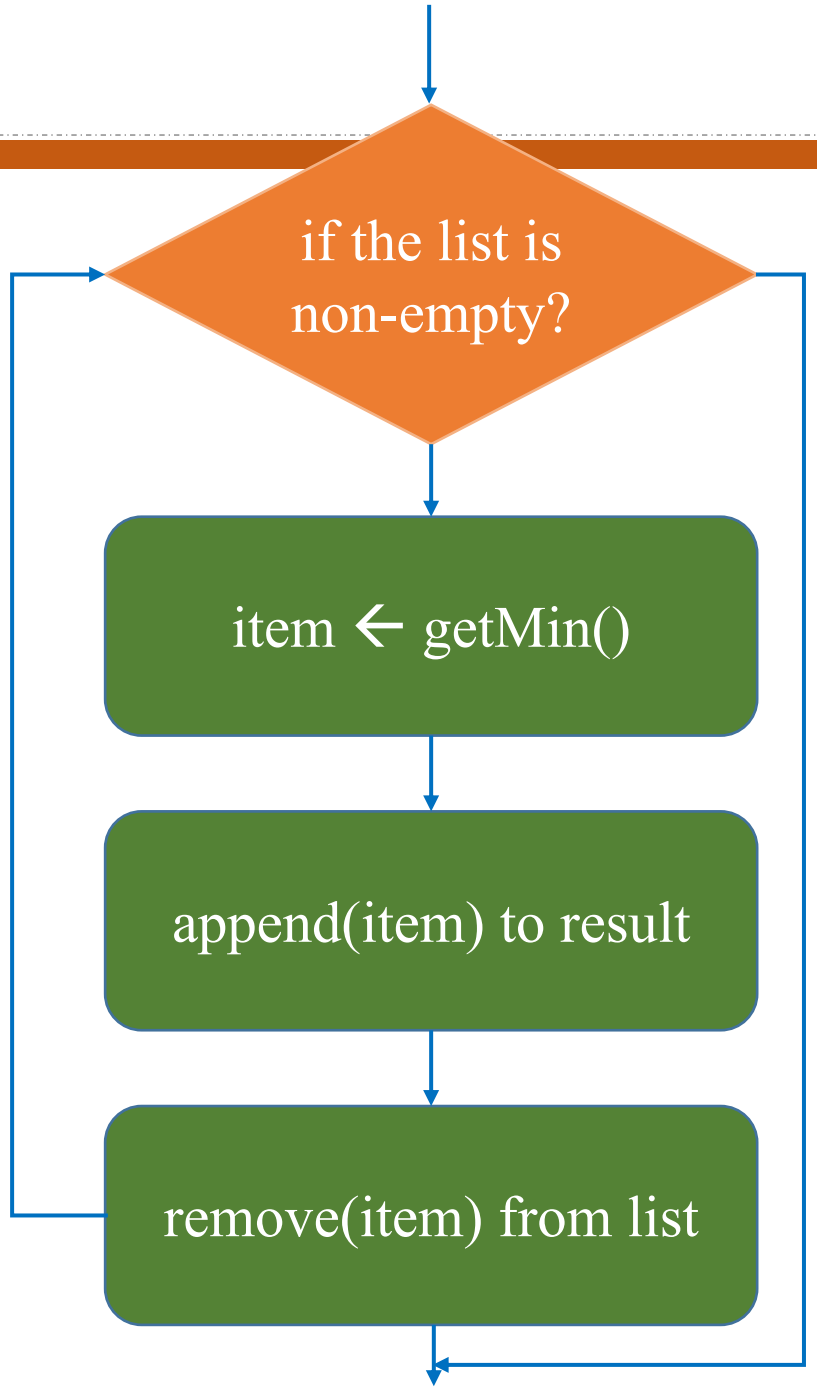
result.append(1) =

1

data.remove(1) =

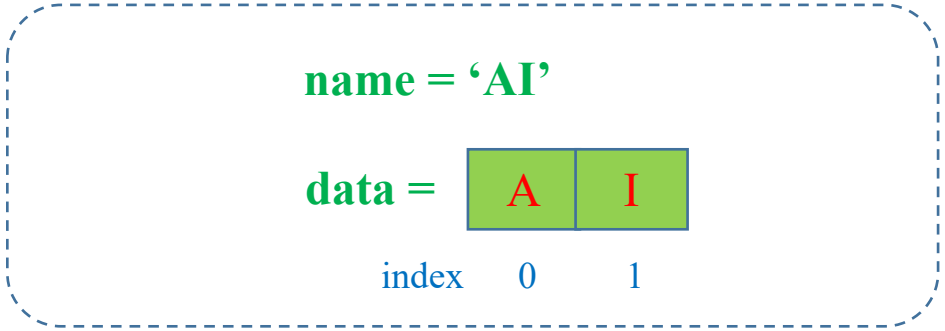
6	5	7	9	2

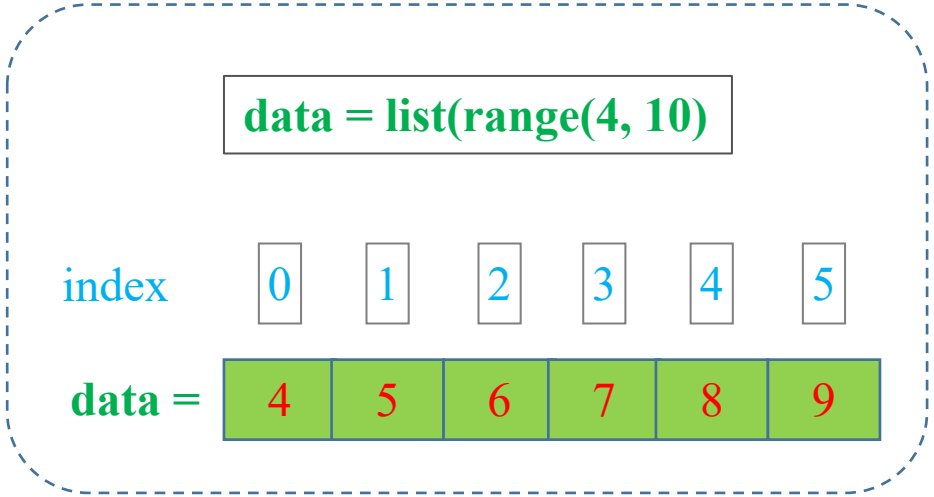
. . .



Converting to List

aList ← list(iterable)





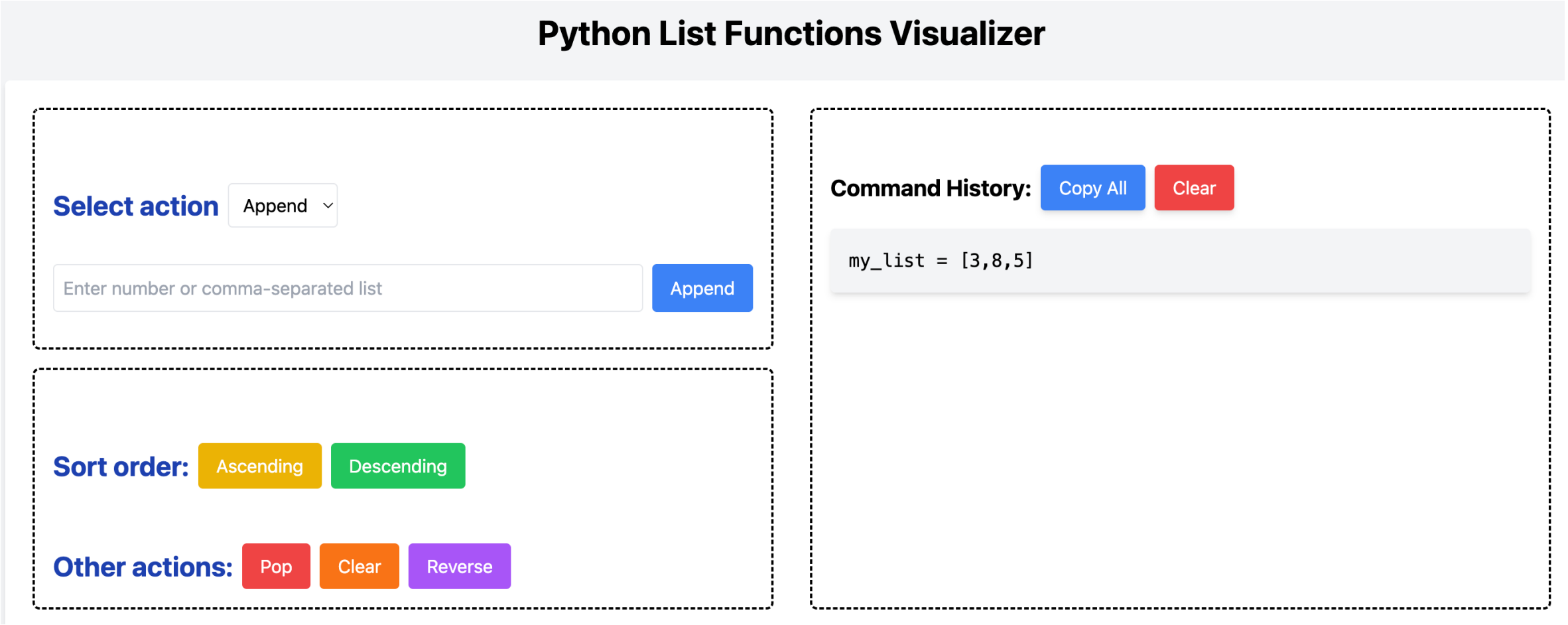

```python
name = 'AI'
data = list(name)

print(name)
print(data)
```

```
AI
['A', 'I']
```

```python
data = list(range(4, 10))
print(data)
```

```
[4, 5, 6, 7, 8, 9]
```

Python List Visualizer



https://ducnd58233.github.io/python-viz/

Outline

- Introduction to Data Structure
- String
- List and Algorithms
- 2D List and Example
- Data Pre-processing Using List
- Summary

2D List: Motivation

Develop a program to store information of 3 students

No.	Name	YOB	Address	Student ID
1	Vinh	2000	Ha Noi	001
2	An	2001	Ho Chi Minh	002
3	Tuan	2002	Da Nang	003

Using List

Student 1	Vinh	2000	Ha Noi	001
Student 2	An	2001	Ho Chi Minh	002
Student 2	Tuan	2002	Da Nang	003

```
student1 = ["Vinh", 2000, "Ha Noi", "001" ]
student2 = ["An", 2001, "Ho Chi Minh", "002" ]
student3 = ["Tuan", 2002, "Da Nang", "003"]

print( student1[0], student1[1], student1[2], student1[3])
print( student2[0], student2[1], student2[2], student2[3])
print( student3[0], student3[1], student3[2], student3[3])
```

How about storing 1000 students' information?

2D List: Motivation

Develop a program to store information of 3 students

No.	Name	YOB	Address	Student ID
1	Vinh	2000	Ha Noi	001
2	An	2001	Ho Chi Minh	002
3	Tuan	2002	Da Nang	003

Student's information

List of names	Vinh	An	Tuan
List of YOB	2000	2001	2002
List of Address	Ha Noi	HCM	Dang nang
List of SID	001	002	003

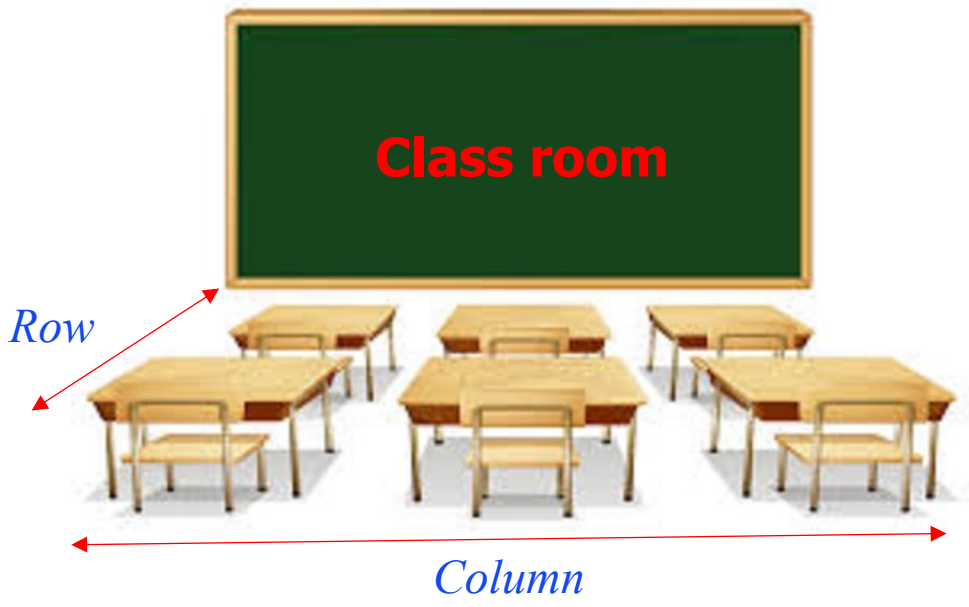
Index 0

```python
name_list = ["Vinh", "An", "Tuan" ]
age_list = [2000, 2001, 2002 ]
address_list = ["Ha Noi", "Ho Chi Minh", "Da Nang"]
studentID_list = ["001", "002", "003"]

print(name_list[0], age_list[0], address_list[0], studentID_list[0])
print(name_list[1], age_list[1], address_list[1], studentID_list[1])
print(name_list[2], age_list[2], address_list[2], studentID_list[2])
```

PROBLEM

How about storing 1000 students' information?

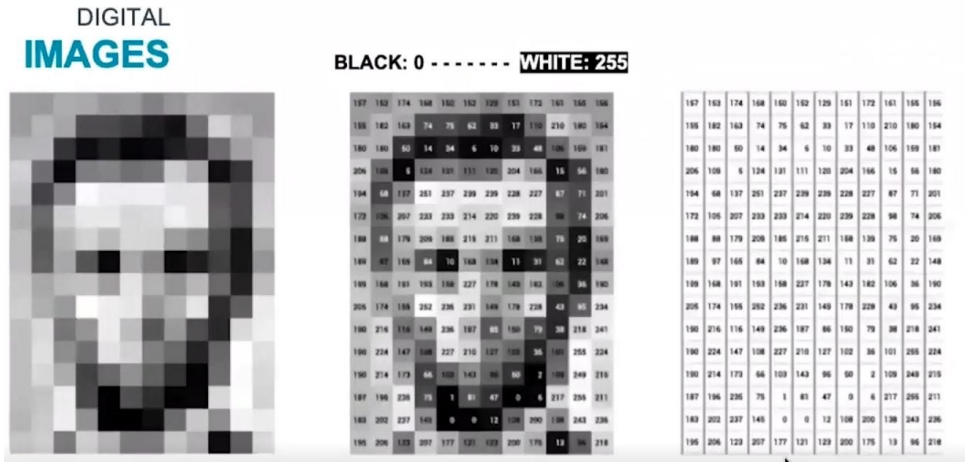
2D List: Motivation



Column

Row

Name	YOB	Address	Student ID
Vinh	2000	Ha Noi	001
An	2001	Ho Chi Minh	002
Tuan	2002	Da Nang	003



Column

Row

Chess board

2D List

❖ Create 2D Matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Column Index

Row Index	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

Column Index

Row Index	0	1	2
0	m[0][0]	m[0][1]	m[0][2]
1	m[1][0]	m[1][1]	m[1][2]
2	m[2][0]	m[2][1]	m[2][2]

Create 2D List

```
# Create a 2D list
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
numrows = len(m) # 3 rows in your example
numcols = len(m[0]) # 2 columns in your example
print(numrows)
print(numcols)
print(m)
```

Accessing Elements

m[r][c]: the value at row r and column c

m[0][0] = 1

m[2][2] = 9

2D List

❖ Iterate Over a 2D Matrix

Column Index

Row Index	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

```
# Create a 2D list
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
numrows = len(m) # 3 rows in your example
numcols = len(m[0]) # 2 columns in your example
```

Solution 1

```
for row in m:
    for elem in row:
        print(elem, end=' ')
    print()
```

Solution 2

```
for r in range(numrows):
    for c in range(numcols):
        print(m[r][c], end=' ')
    print()
```

```
1 2 3
4 5 6
7 8 9
```

2D List

❖ Update elements in 2D matrix

matrix[r][c] = new_value

Column Index

Row Index	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

```
# Create a 2D list
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
numrows = len(m) # 3 rows in your example
numcols = len(m[0]) # 2 columns in your example
```

```
# print 2d list
for r in range(numrows):
    for c in range(numcols):
        print(m[r][c], end=' ')
    print()
```

```
1 2 3
4 5 6
7 8 9
```

```
# Update element in the 2d list
m[1][1] = 0
```

```
# print 2d list
for r in range(numrows):
    for c in range(numcols):
        print(m[r][c], end=' ')
    print()
```

```
1 2 3
4 0 6
7 8 9
```

2D List: Hadamard Product

❖ 2D matrix: Hadamard Product (Element-wise Multiplication)

$$\overset{G}{\begin{bmatrix} 3 & 5 & 7 \\ 4 & 9 & 8 \end{bmatrix}} \circ \overset{H}{\begin{bmatrix} 1 & 6 & 3 \\ 0 & 2 & 9 \end{bmatrix}} = \overset{N}{\begin{bmatrix} 3\times1 & 5\times6 & 7\times3 \\ 4\times0 & 9\times2 & 8\times9 \end{bmatrix}}$$

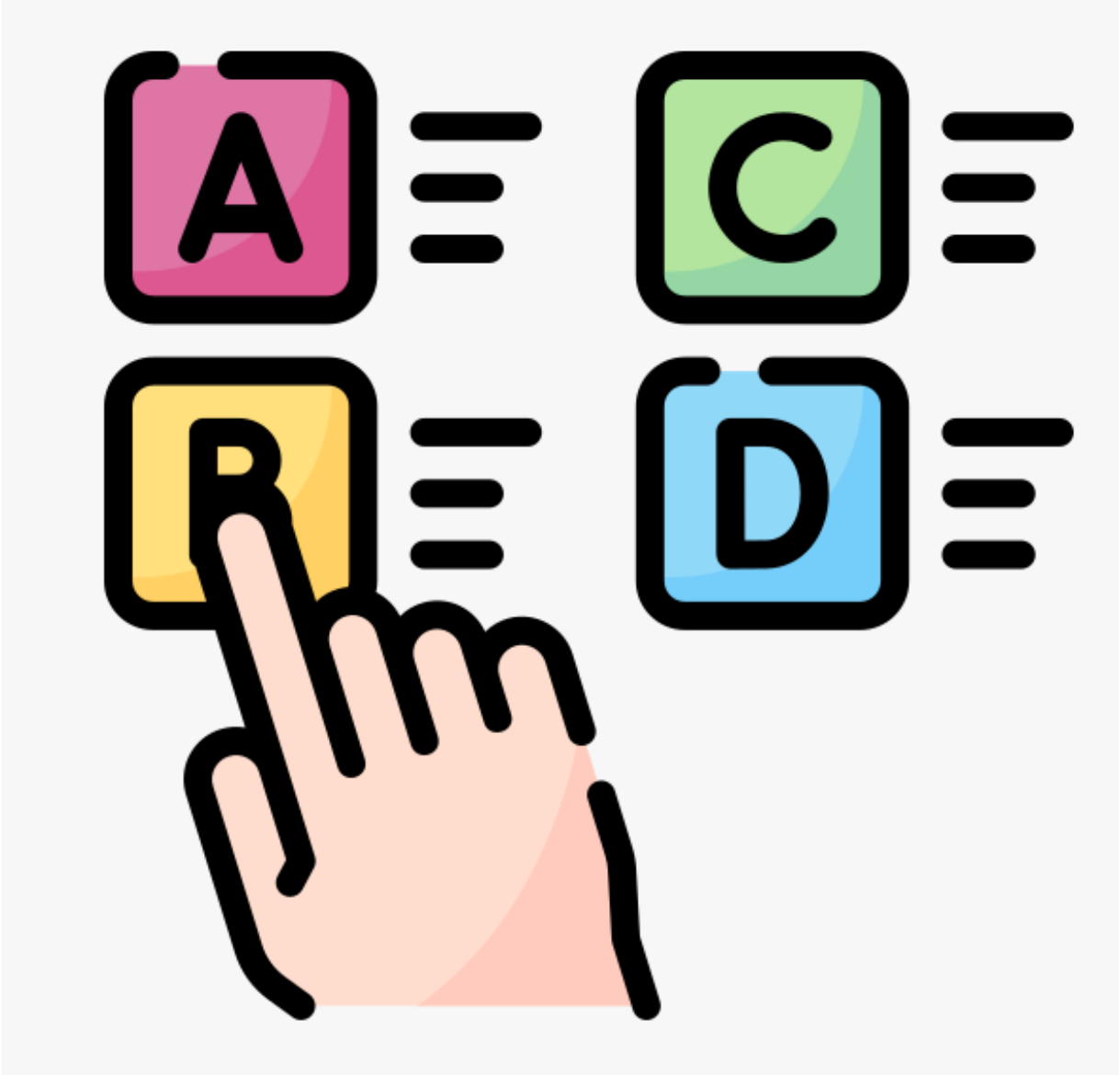
```python
# Create 2D matrix
G = [[3, 5, 7], [4, 9 , 8]]
H = [[1, 6, 3], [0, 2 , 9]]
rows = len(G)
cols = len(G[0])
N = x = [[None]*cols for _ in range(rows)]
```

```python
# Do Hadamard Product

for r in range(rows):
  for c in range(cols):
     N[r][c] = G[r][c] * H[r][c]
```

```python
# Print the results
for r in range(rows):
   for c in range(cols):
      print(N[r][c], end=' ')
      print()
```

```
3 30 21
0 18 72
```

Quizz



2D List: Example

Given the values of the first row and the first column in a 2D matrix [m,n], fill the remaining values by considering the indices (r, c-1), (r-1, c-1), and (r-1, c).

1	2	3	4
5			
9			
13			

Input matrix [4,4]

1	2		
5	8		

Output matrix [4,4]

(r-1,c-1)	(r-1,c)	(r-1,c+1)	
(r,c-1)	(r,c)	(r,c+1)	

Output matrix [4,4]

$$m[r][c] = m[r-1][c-1] + m[r-1][c] + m[r][c-1]$$

2D List: Example

Step 1: Create a matrix with dimensions based on the lengths of first_row and first_col, initially filled with zeros.

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

```
# create the matrix
first_row = [1, 2, 3, 4]
first_col = [1, 5, 9, 13]
rows = len(first_col)
cols = len(first_row)
matrix = [[0] * cols for _ in range(rows)]
```

2D List: Example

Step 2: Fill the First Row and First Column

1	2	3	4
5	0	0	0
9	0	0	0
13	0	0	0

```python
# create the matrix
first_row = [1, 2, 3, 4]
first_col = [1, 5, 9, 13]
rows = len(first_col)
cols = len(first_row)
matrix = [[0] * cols for _ in range(rows)]
```

```python
# Fill the first row and first column
for c in range(cols):
    matrix[0][c] = first_row[c]
for r in range(rows):
    matrix[r][0] = first_col[r]
```

2D List: Example

Step 3: Fill the Rest of the Matrix:

1	2	3	4
5	8	13	20
9	22	43	76
13	44	109	128

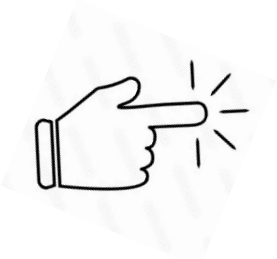
```
# create the matrix
first_row = [1, 2, 3, 4]
first_col = [1, 5, 9, 13]
rows = len(first_col)
cols = len(first_row)
matrix = [[0] * cols for _ in range(rows)]
```

```
# Fill the first row and first column
for c in range(cols):
    matrix[0][c] = first_row[c]
for r in range(rows):
    matrix[r][0] = first_col[r]
```

```
# Fill the rest of the matrix
for r in range(1, rows):
    for c in range(1, cols):
        matrix[r][c] = matrix[r][c-1] + matrix[r-1][c-1] +
                       matrix[r-1][c]
```

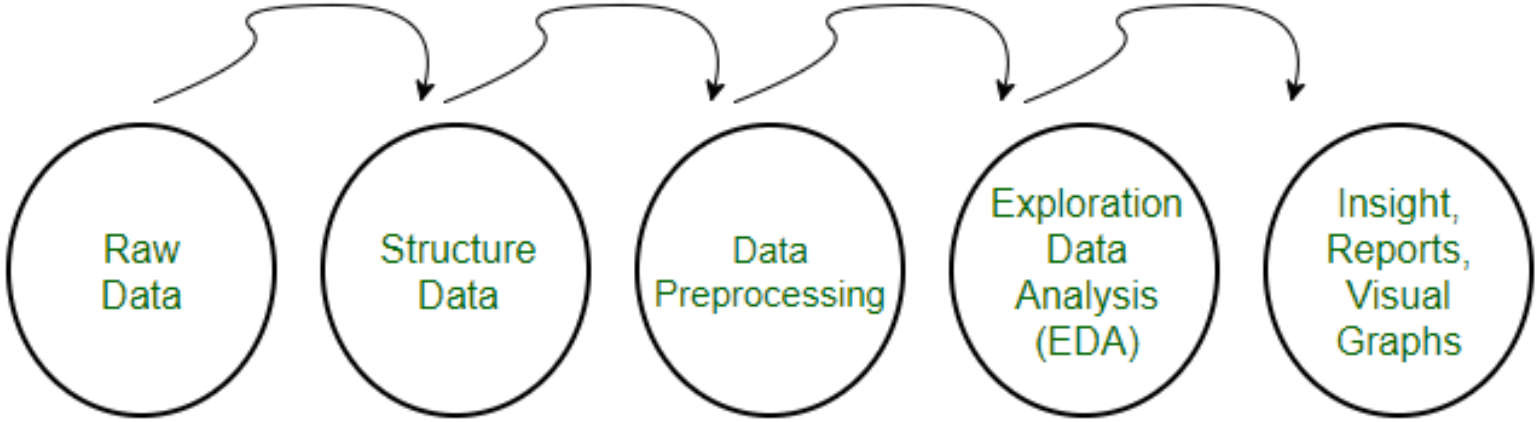
Outline

- Introduction to Data Structure
- String
- List and Algorithms
- 2D List and Example
- Data Pre-processing Using List
- Summary



Data Pre-processing using List

❖Data processing steps in ML



Read data from file

Python Pre-processing Program

Write data to file

Data with noise

Data without noise

data.txt

```
1 1, ,2,3,4,5,6,7,8,9
```

result.txt

```
1 1,5.0,2,3,4,5,6,7,8,9
```

Data Pre-processing using List

❖Step 1: Read data from file

Read data from a file (already exist)

(1)	open(file_path, 'r')
(2)	read()
(3)	close()

data.txt

```
1,,2,3,4,5,6,7,8,9
```

```python
file_name = "data.txt"
def read_data(file_name):
    with open(file_name, 'r') as file:
        data = file.read()
        file.close()
    return data.split(',')
```

Data Pre-processing using List

❖Step 2: Perform data preprocessing

```
input_data = [1,'', 2, 3, 4, 5, 6, 7, 8, 9]
```

```
def fill_missing_with_average(data):
    missing_values = [i for i, x in enumerate(data ) if len(x)==0]
    non_missing_values = [float(x) for x in data if len(x) > 0]
    average = sum(non_missing_values) / len(non_missing_values)
    for index in missing_values:
        data[index] = average
    return data
```

```
print(fill_missing_with_average(input_data))
Output:[1, 5.0, 2, 3, 4, 5, 6, 7, 8, 9]
```

Data Pre-processing using List

❖Step 3: Write data to file

Write data to a file (not exist)

(1)	open(file_path, 'w')
(2)	write()
(3)	close()

result.txt

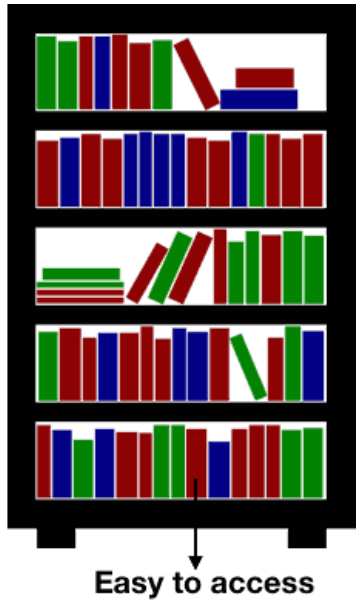
```
1,5.0,2,3,4,5,6,7,8,9
```

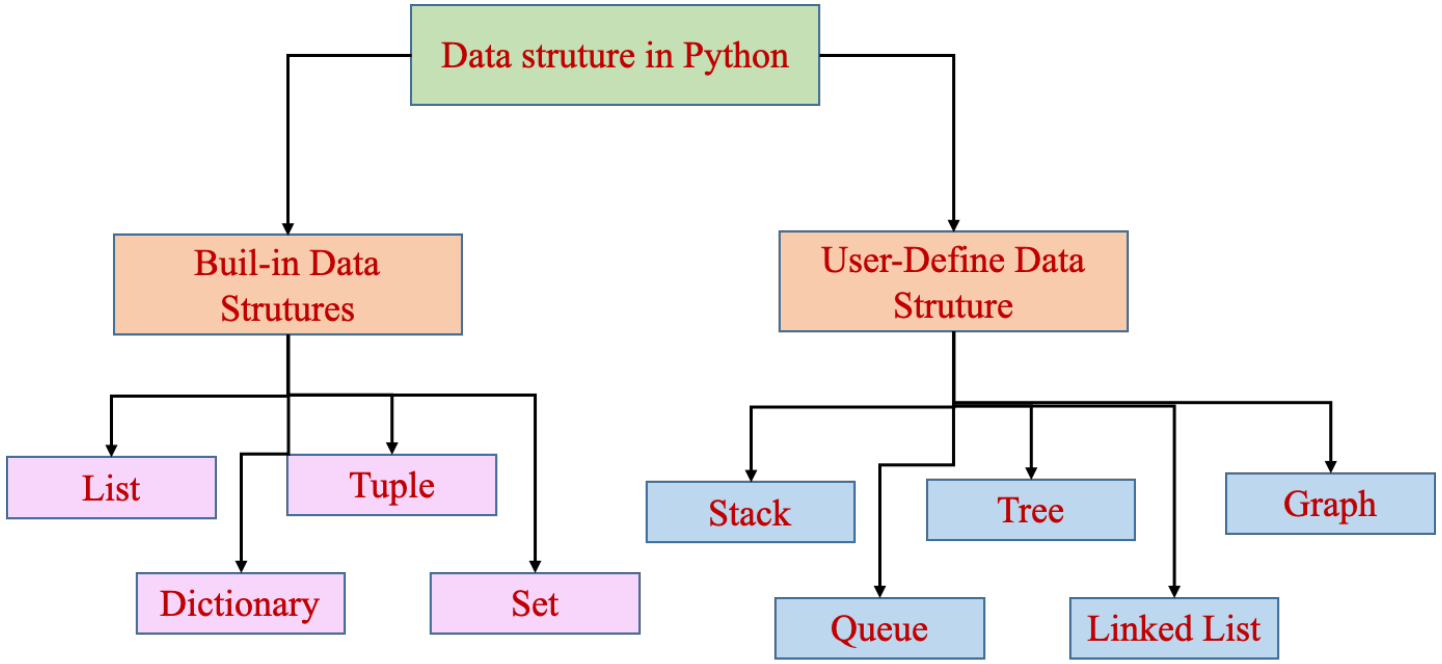
```python
# write data to file
def write_data(file_name, data):
  with open(file_name, 'w') as file:
    file.write(','.join(str(x) for x in data))
    file.close()
```

Outline

- Introduction
- String
- List
- Algorithm on List
- Data Pre-processing Using List
- Summary

Summary






```python
# create a string
name = 'AI'
print(name)
```

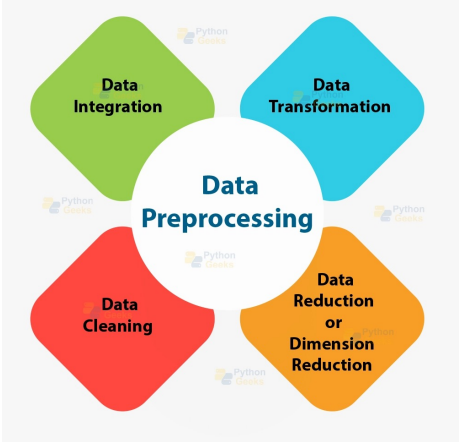
```
AI
```

```
// create a list
data = [6, 5, 7, 1, 9, 2]
```

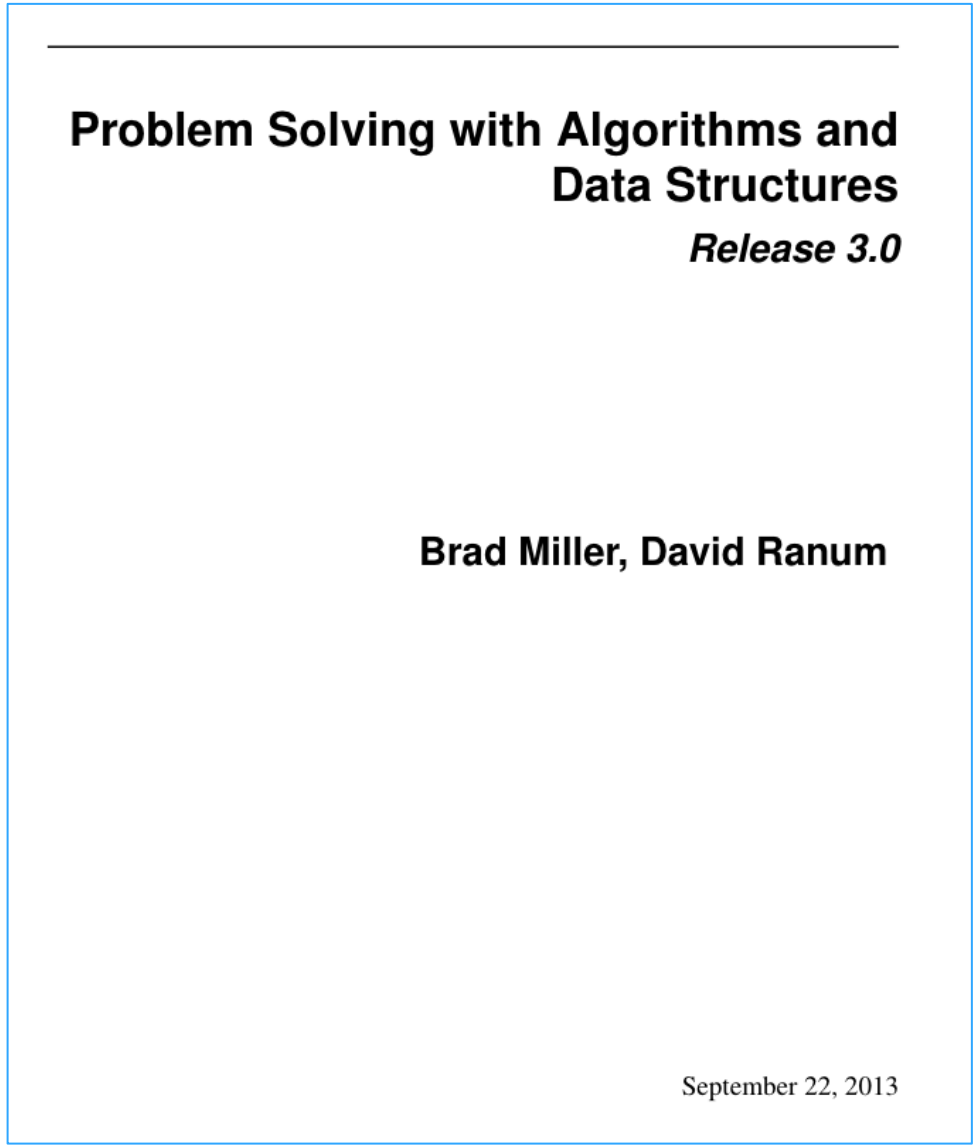
data =

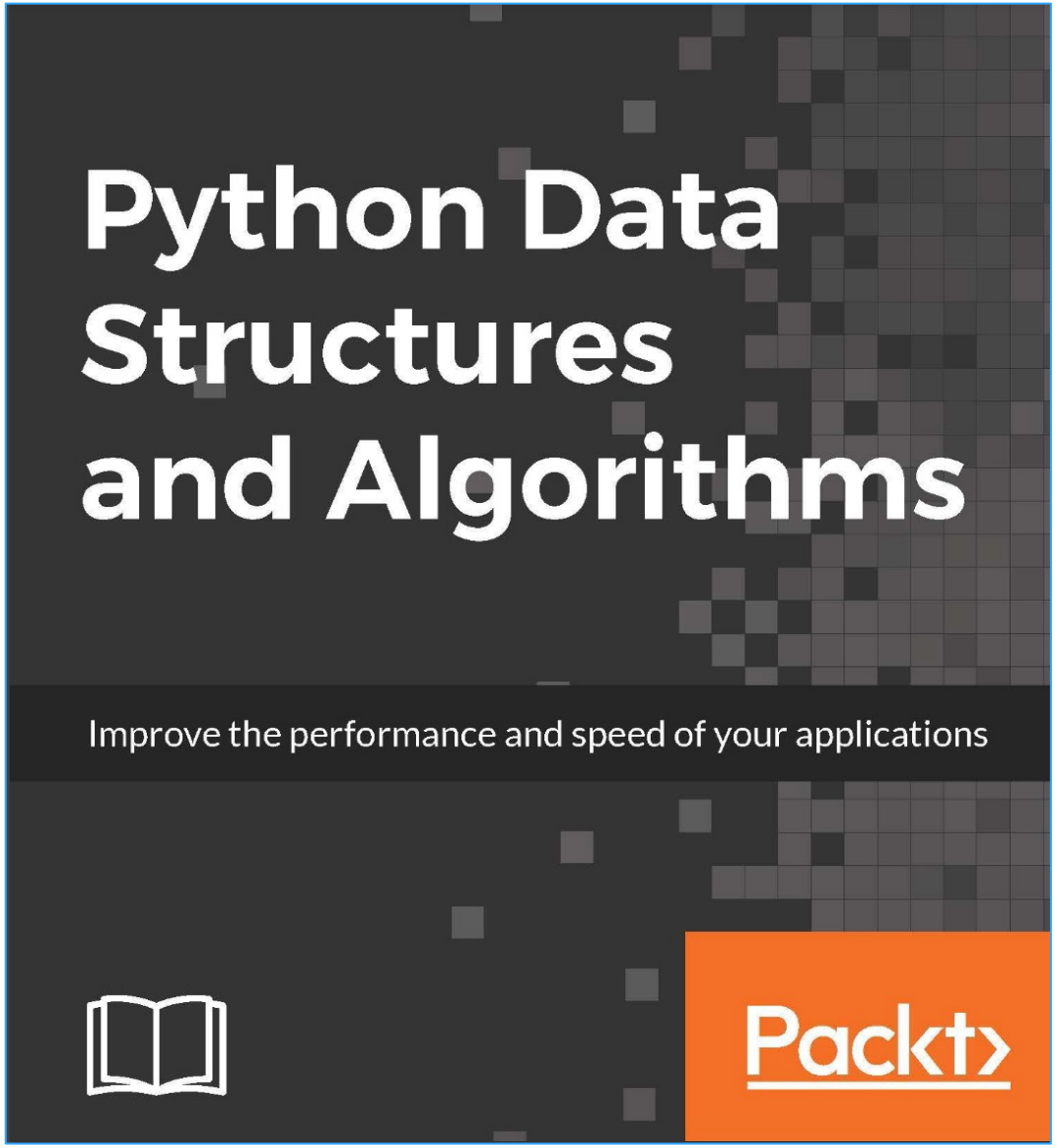
6	5	7	1	9	2
0	1	2	3	4	5

index



References


Problem Solving with Algorithms and Data Structures
Release 3.0
Brad Miller, David Ranum
September 22, 2013


Python Data Structures and Algorithms
Improve the performance and speed of your applications
Packt