AI VIET NAM
@aivietnam.edu.vn

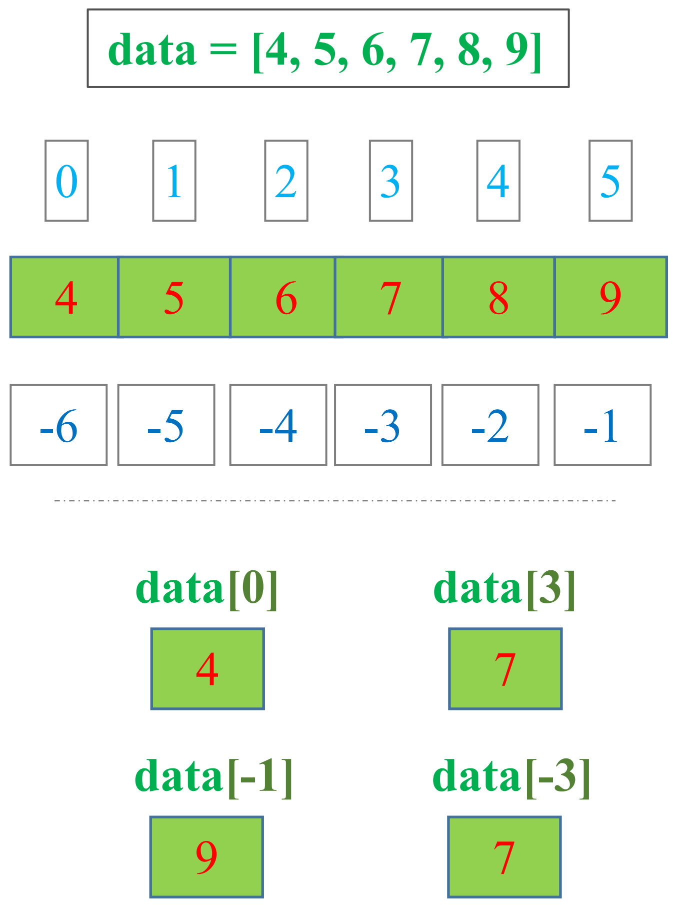
Data Structure

Delving into List

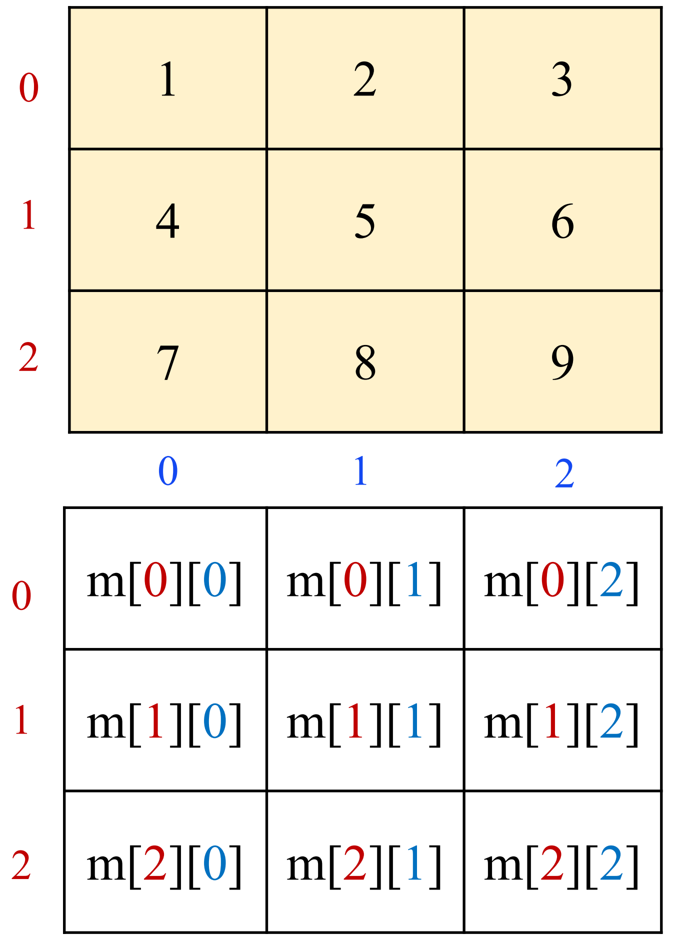
Quang-Vinh Dinh
PhD in Computer Science

Objectives

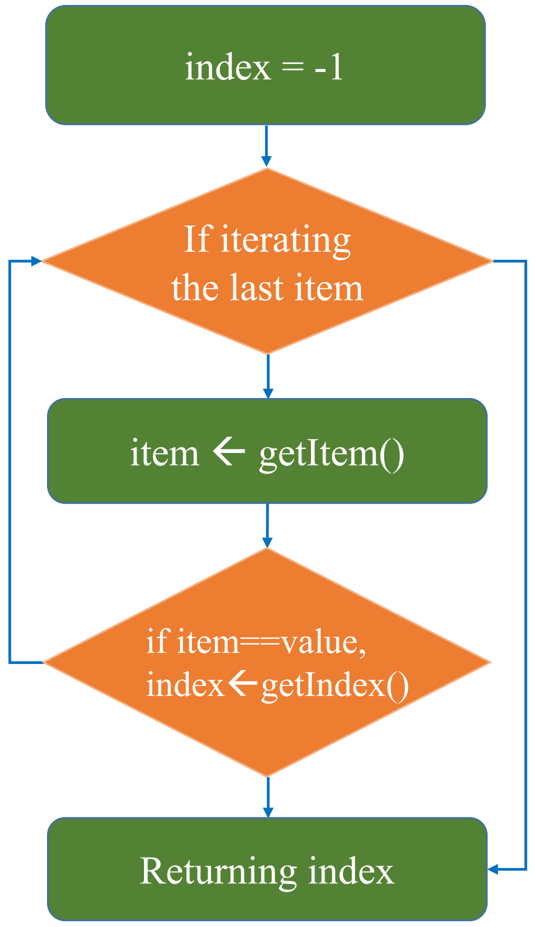
1D List



2D List

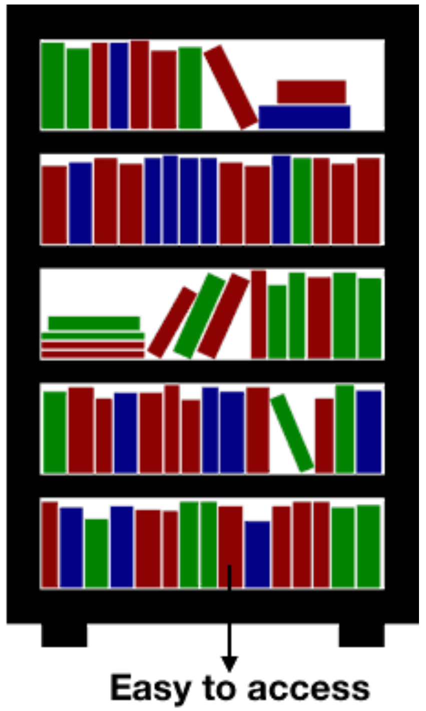


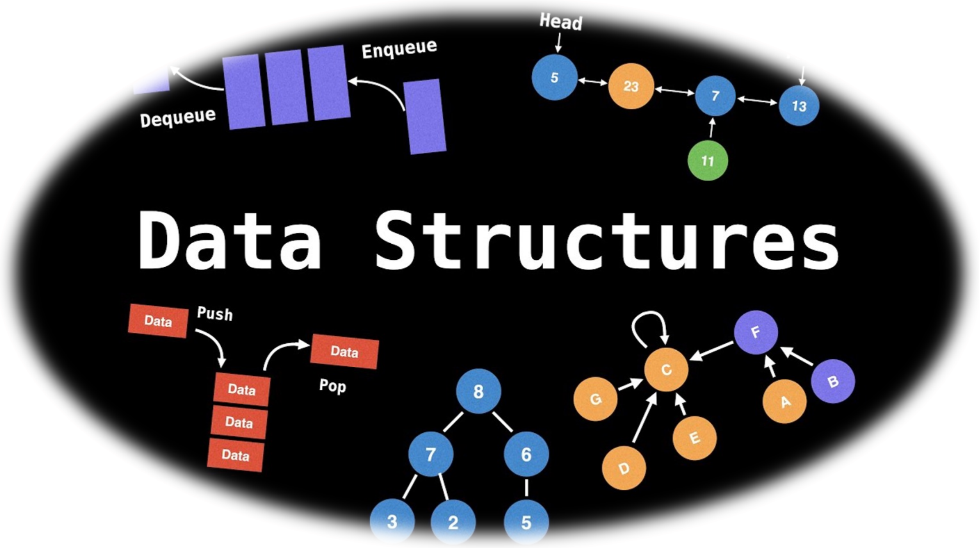
Algorithms



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.

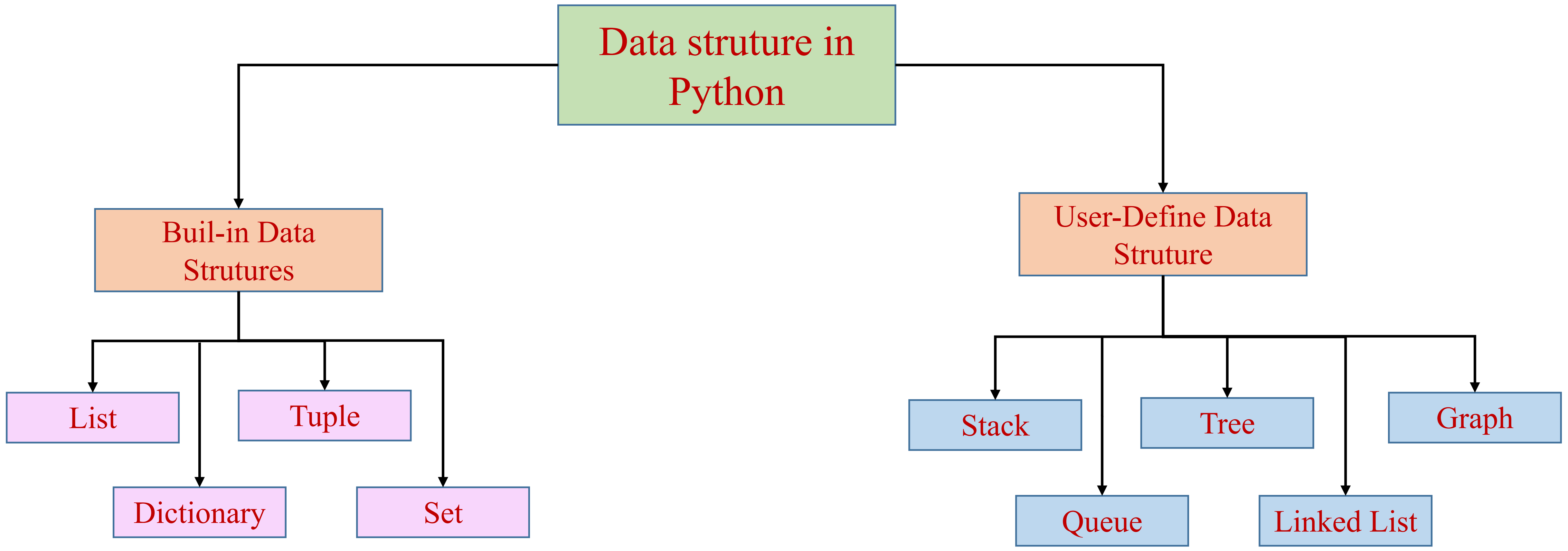




AI VIET NAM
All-in-One 2025

What is a Data Structure?

❖ Overview

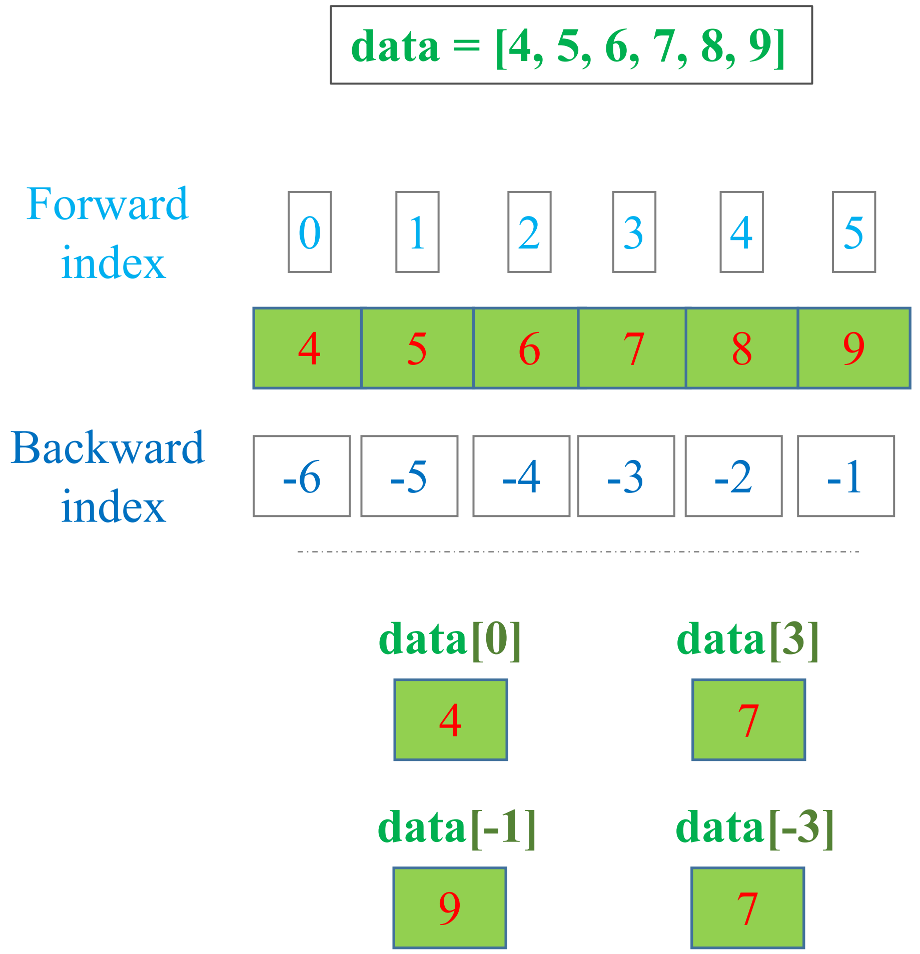


Outline

SECTION 1
1D List

SECTION 2
2D List

SECTION 3
Algorithms

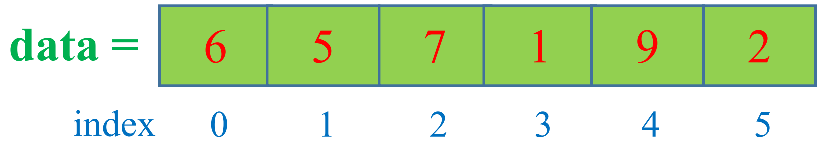


List

❖ A container that can contain elements

list_name = [element-1, …, element-n]

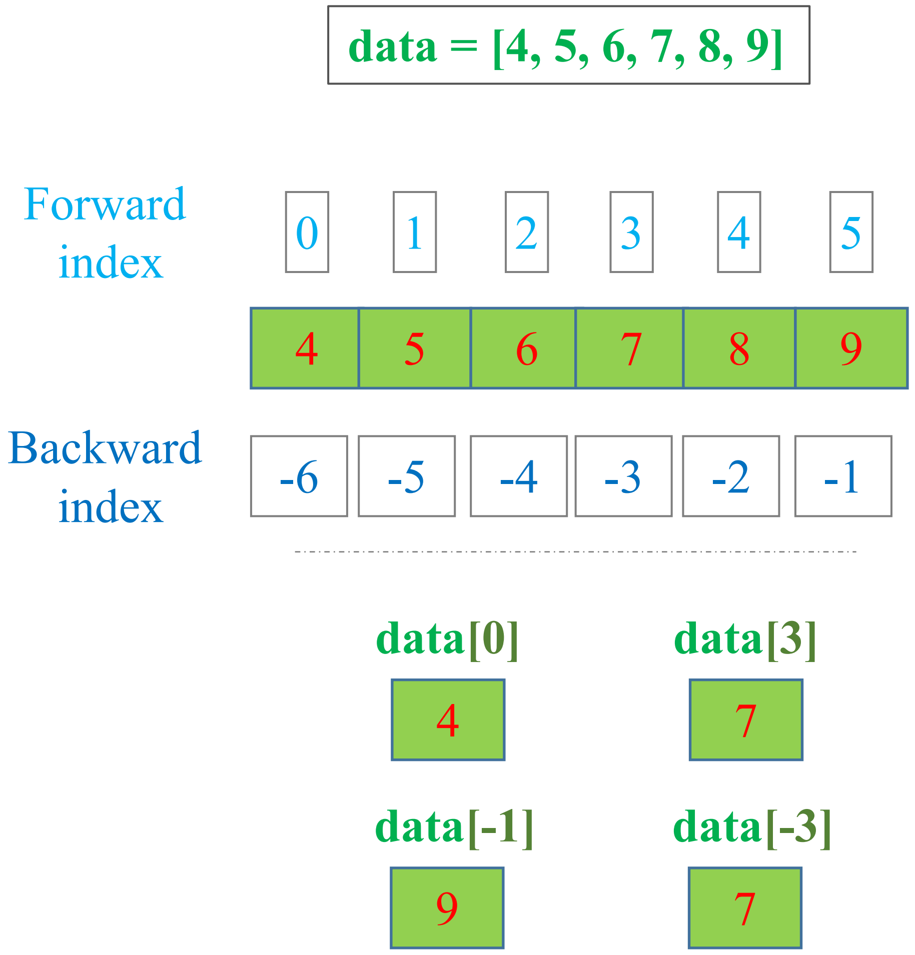
```
// create a list
data = [6, 5, 7, 1, 9, 2]
```



```
# 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']
```

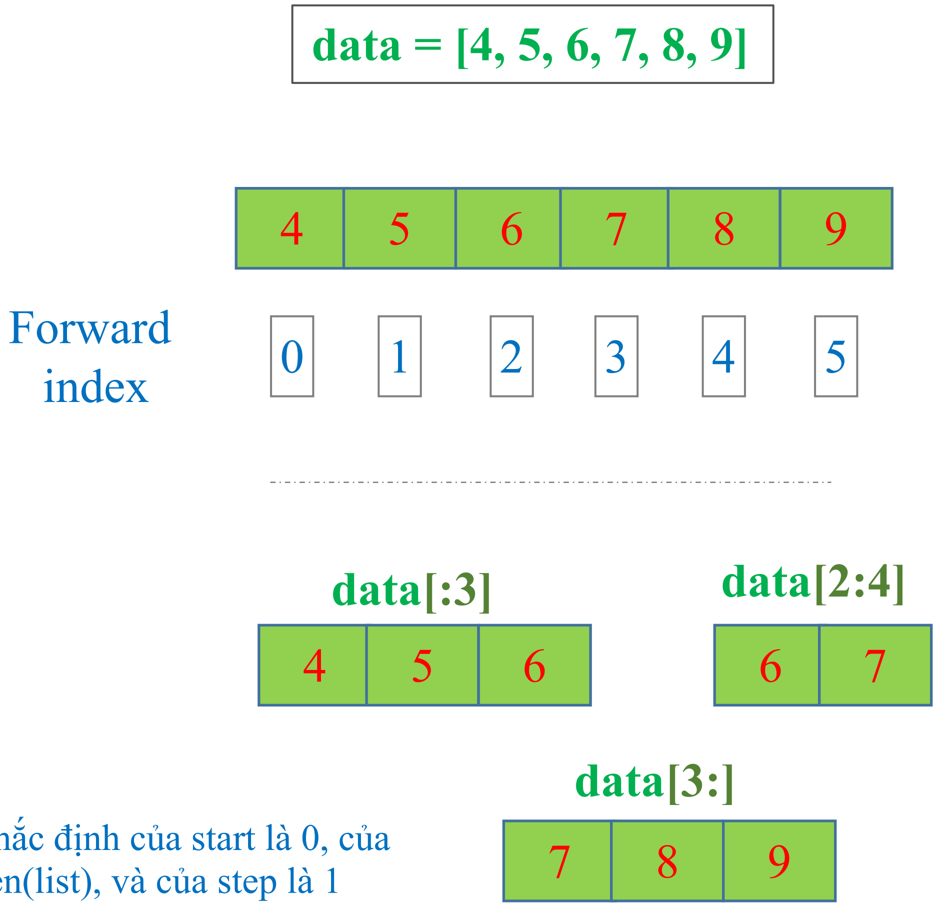
List

❖ Index



❖ Slicing

list[start:end:step]



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

List

❖ + and * operators

data1 =

6	5	7

data2 =

1	9	2

# nối 2 list
data = data1 + data2

data =

6	5	7	1	9	2

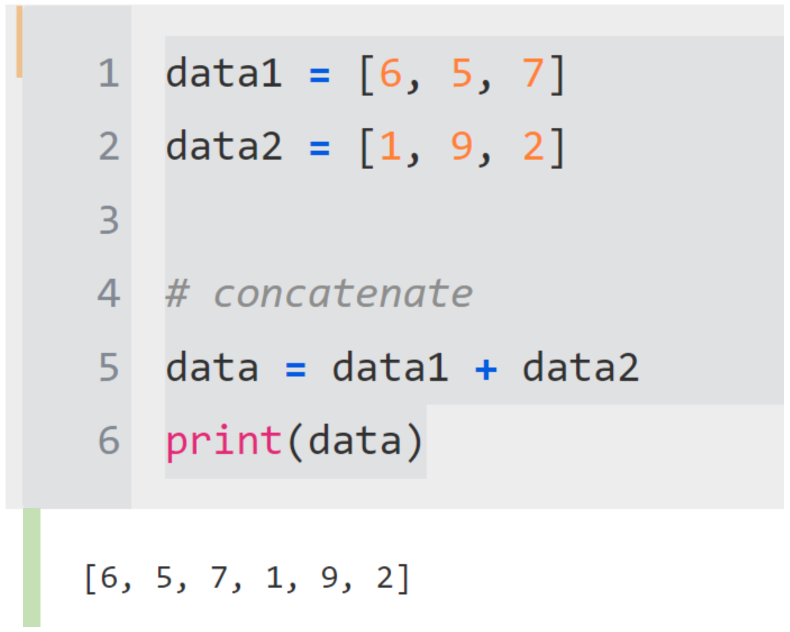
data =

6	5

# nhân list với một số nguyên
data_m = data * 3

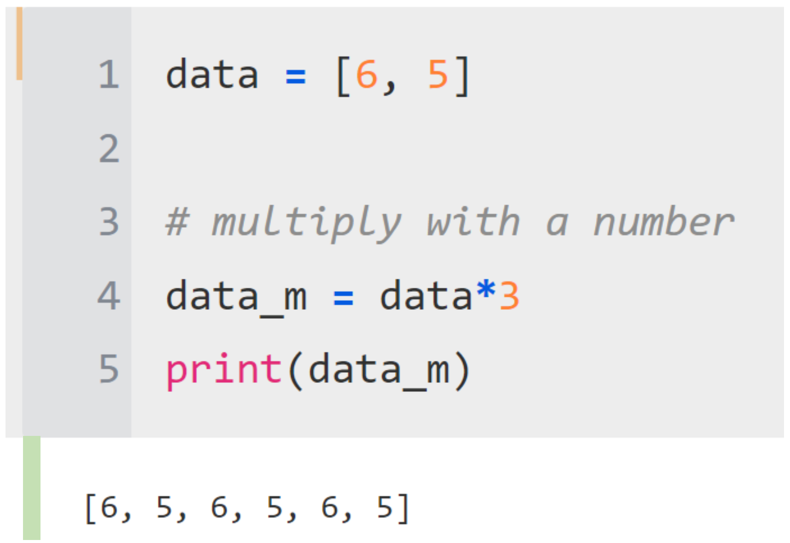
data_m =

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

❖ Add an element

data =

6	5	7	1	9	2

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

data =

6	5	7	1	9	2	4

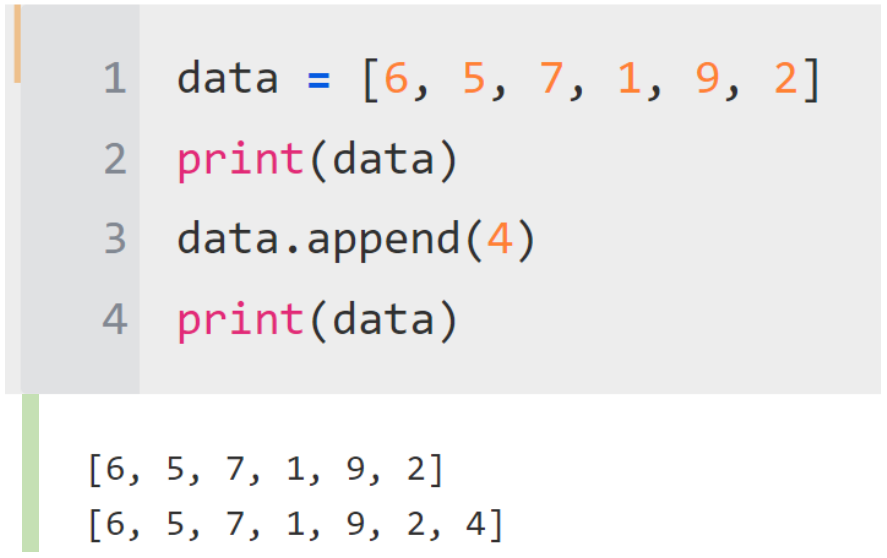
data =

6	5	7	1	9	2

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

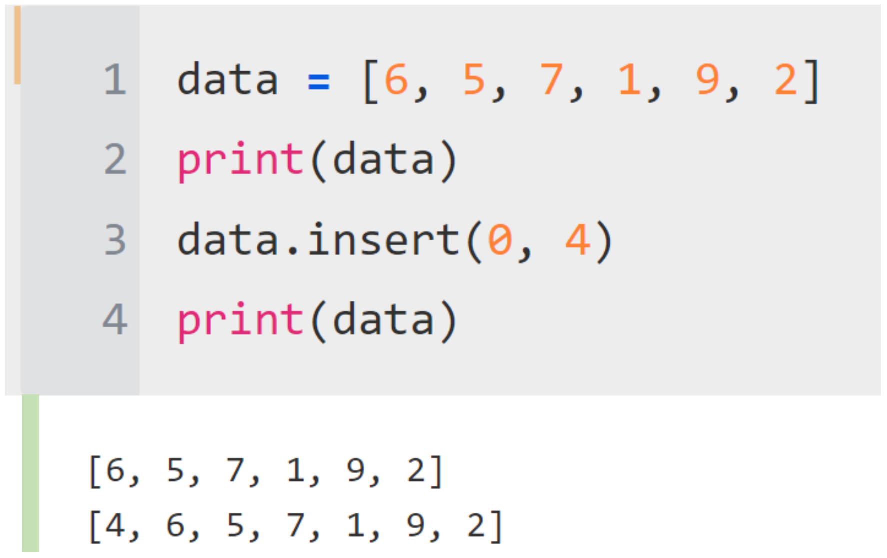
data =

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 trị cuối list

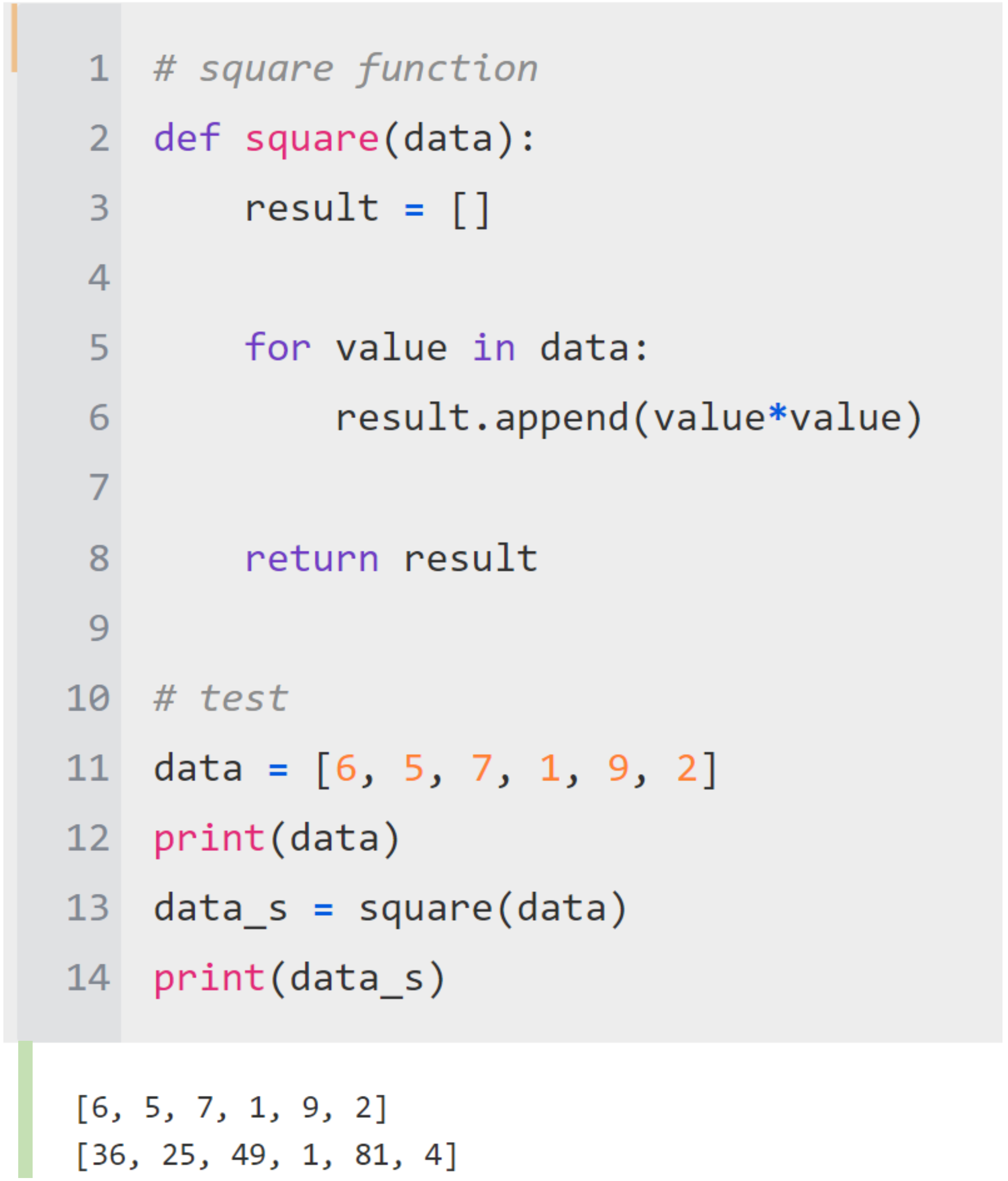
data =

6	5	7	1	9	2

Examples

square(aList)

data =	6	5	7	1	9	2
square(data) =	36	25	49	1	81	4



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

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

    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]
```

Mutable and Immutable

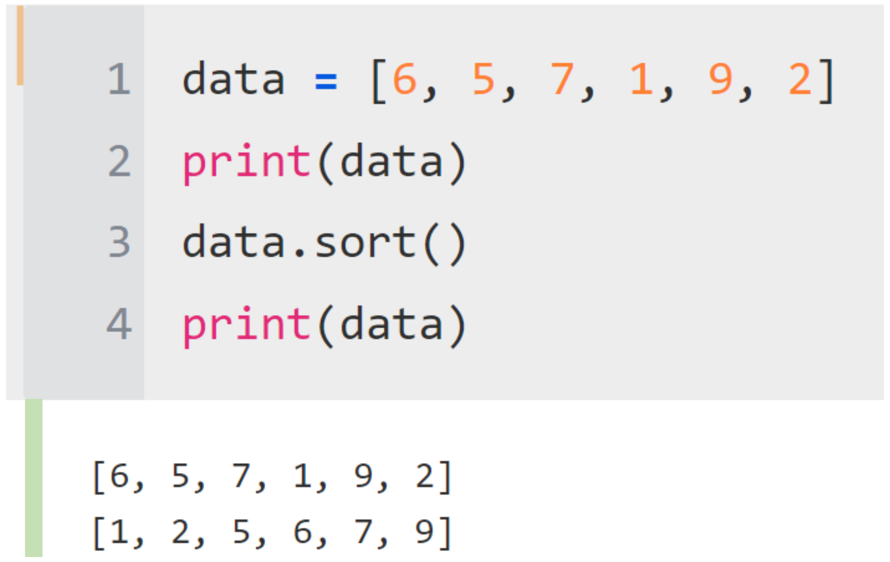
```python
# immutable
def square(data):
    result = []
    for value in data:
        result.append(value*value)

    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]
```

```python
# mutable
def square(data):
    length = len(data)

    for i in range(length):
        value = data[i]
        data[i] = value*value

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

square(data)
print(data)
```

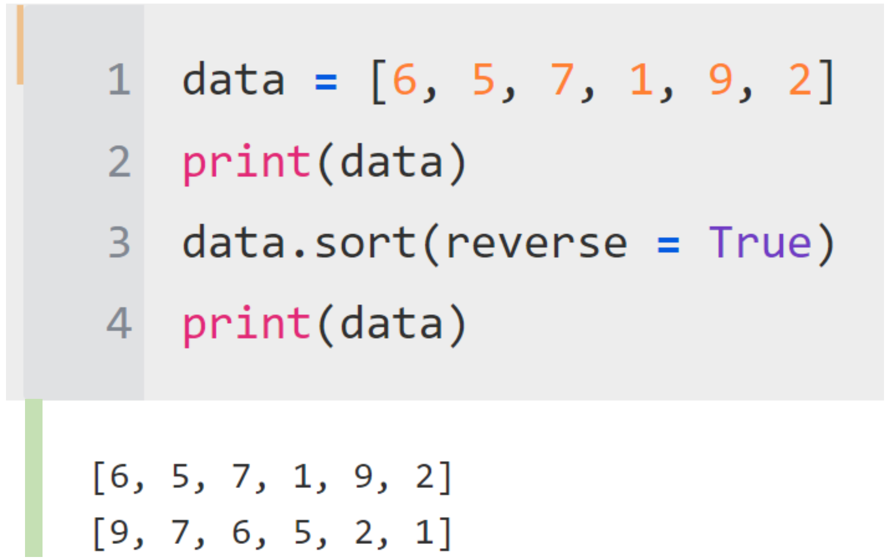
```
[6, 5, 7, 1, 9, 2]
[36, 25, 49, 1, 81, 4]
```

List



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

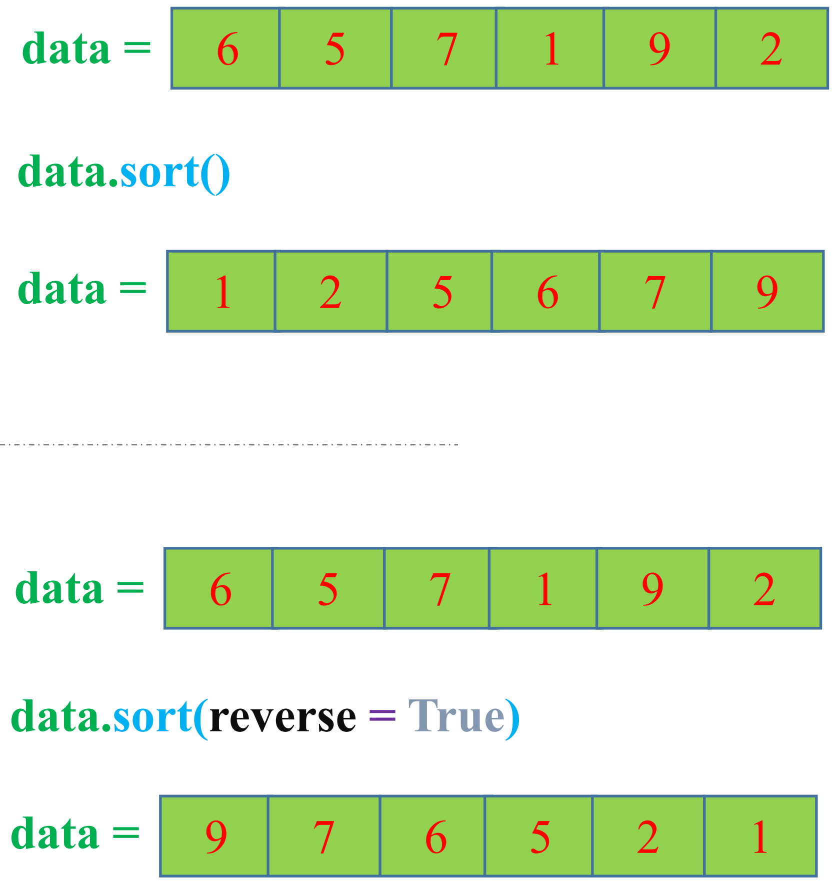
```
[6, 5, 7, 1, 9, 2]
[1, 2, 5, 6, 7, 9]
```



```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]
```

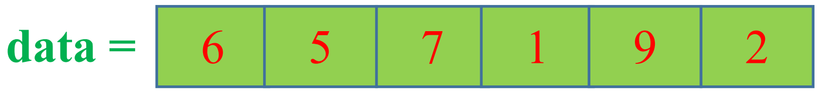
❖ sort() – Sắp xếp các phần tử



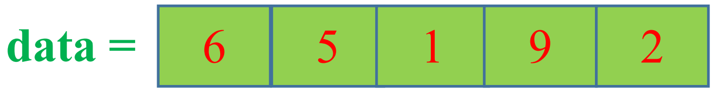
AI VIET NAM
All-in-One 2025

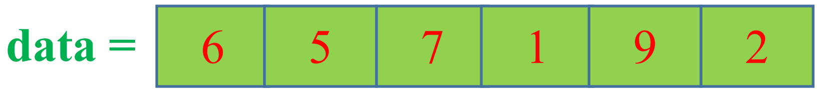
List

❖ Deleting an element

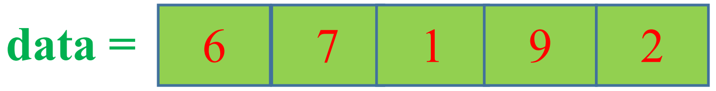


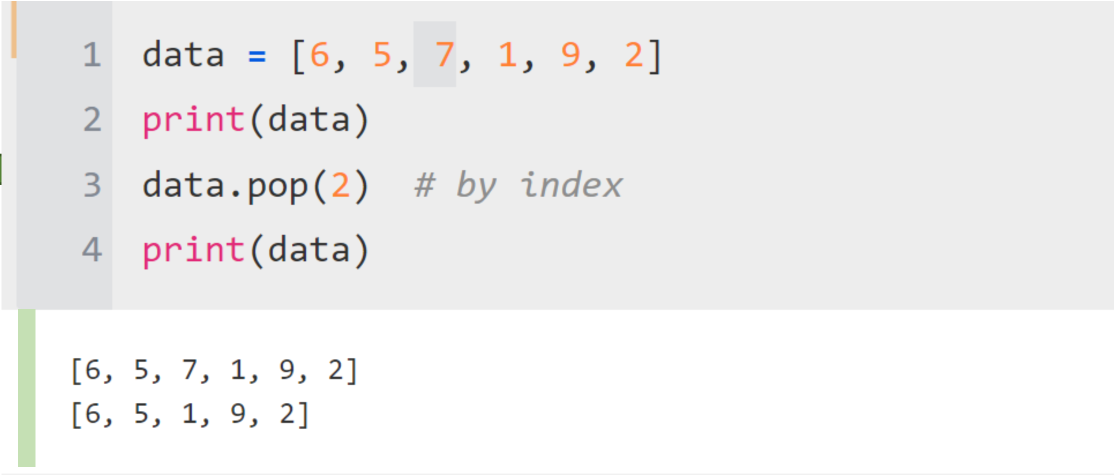
data.pop(2) # tại vị trí index = 2





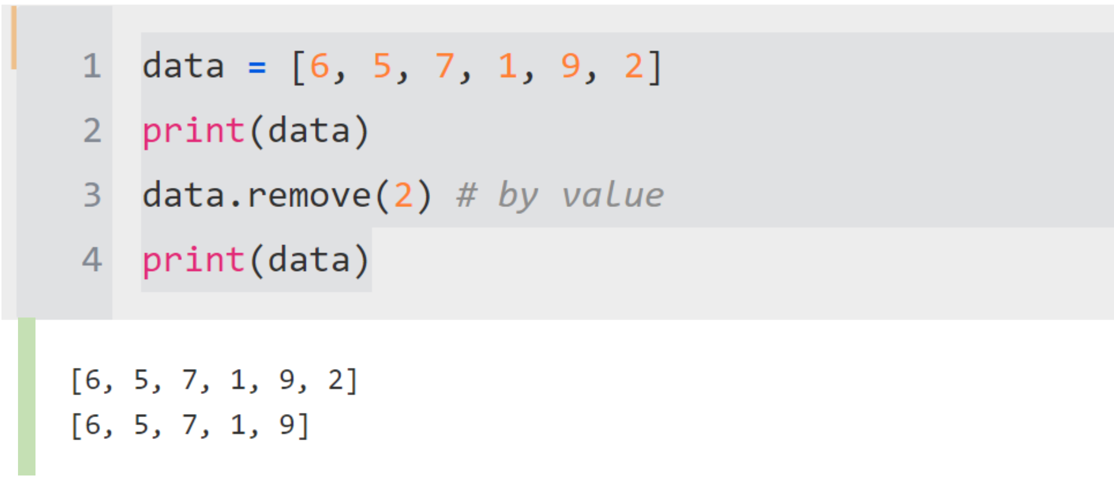
data.remove(5) # xóa phần tử đầu tiên
# có giá trị là 5





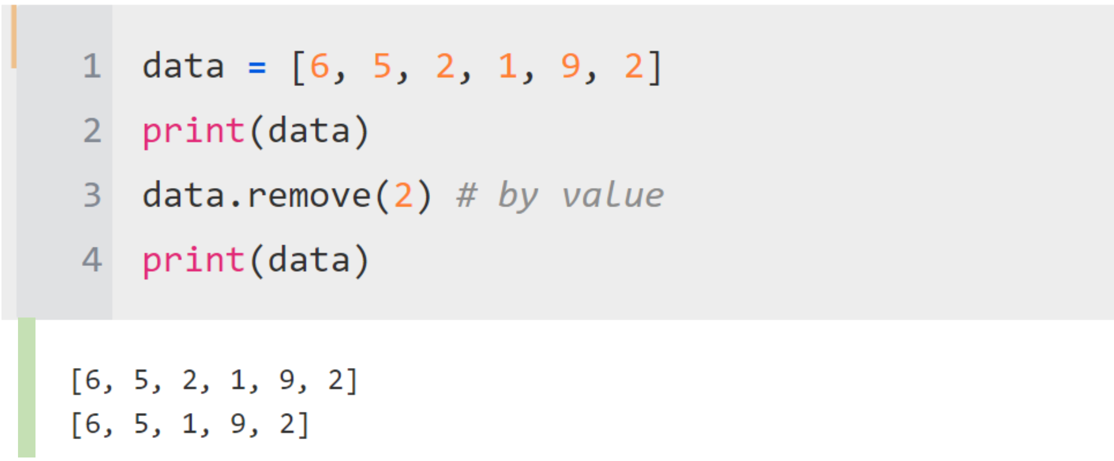
```python
data = [6, 5, 7, 1, 9, 2]
print(data)
data.pop(2)  # by index
print(data)
```

```
[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)
```

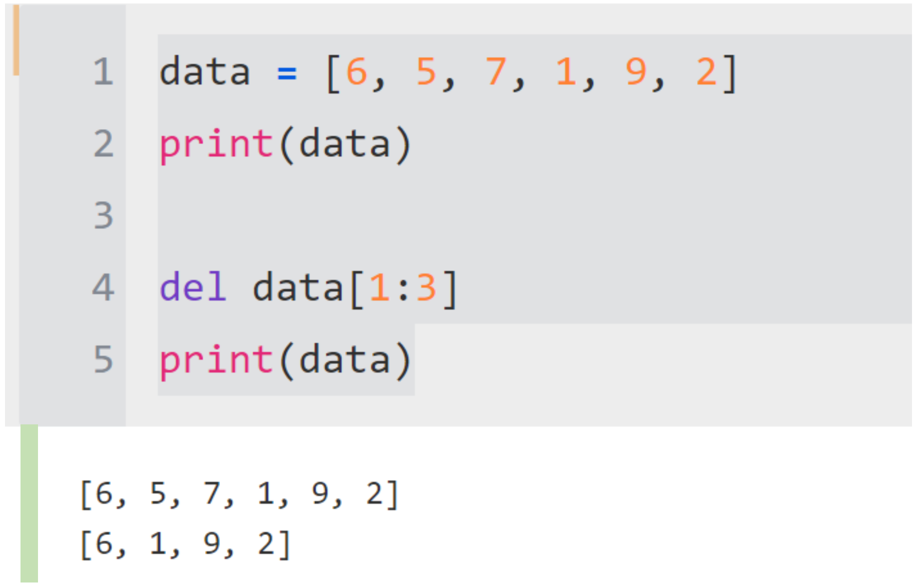
```
[6, 5, 7, 1, 9, 2]
[6, 5, 7, 1, 9]
```



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

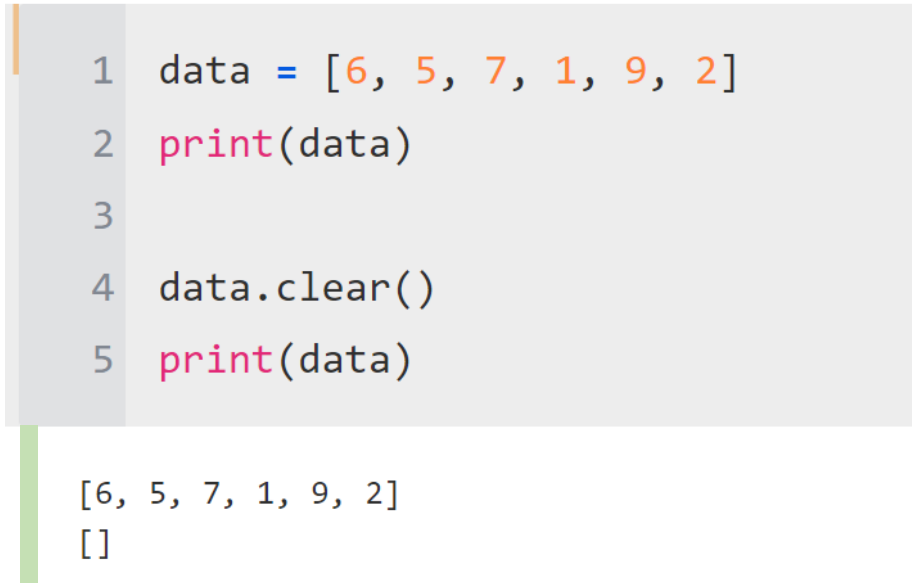
```
[6, 5, 2, 1, 9, 2]
[6, 5, 1, 9, 2]
```

List



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

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

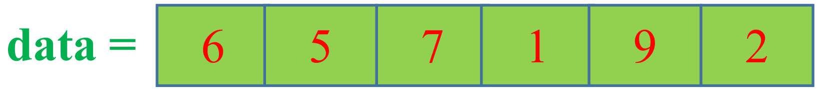
```
[6, 5, 7, 1, 9, 2]
[6, 1, 9, 2]
```



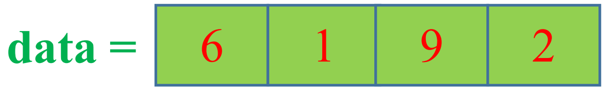
```python
data = [6, 5, 7, 1, 9, 2]
print(data)

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

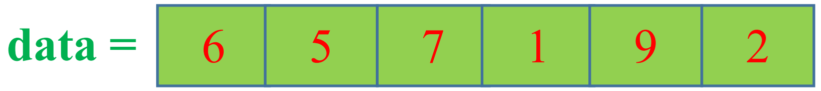
```
[6, 5, 7, 1, 9, 2]
[]
```

❖ Delete elements



```
# xóa phần tử thứ 1 và 2
del data[1:3]
```





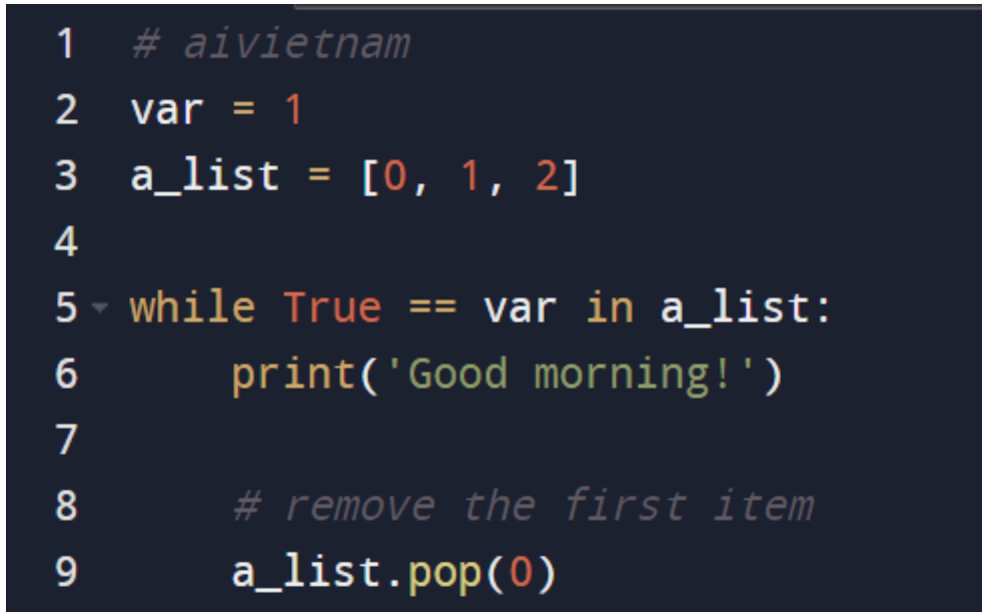
```
data.clear()
```

data = []

Quizzes

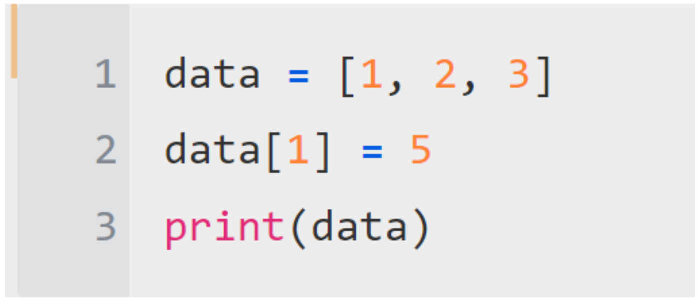
❖ **Using with for, while, and in**

Quiz 1

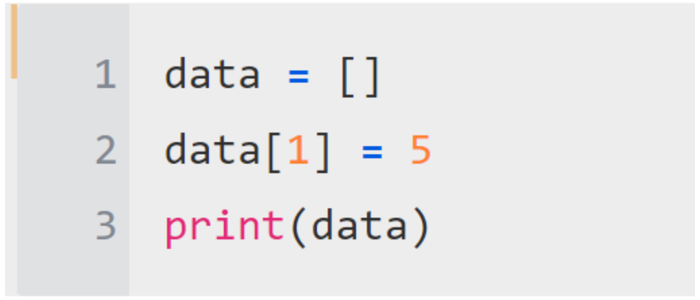


```python
# aivietnam
var = 1
a_list = [0, 1, 2]

while True == var in a_list:
    print('Good morning!')

    # remove the first item
    a_list.pop(0)
```

Quiz 2

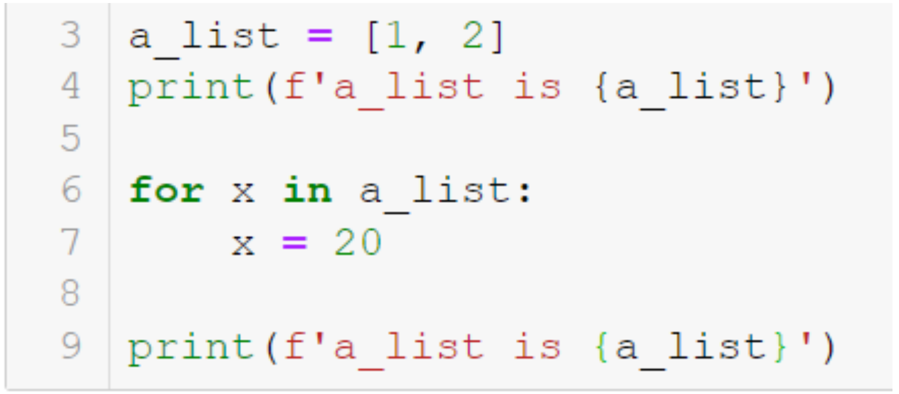


```python
data = [1, 2, 3]
data[1] = 5
print(data)
```



```python
data = []
data[1] = 5
print(data)
```

Quiz 3



```python
a_list = [1, 2]
print(f'a_list is {a_list}')

for x in a_list:
    x = 20

print(f'a_list is {a_list}')
```

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

reverse() – Đảo ngược vị trí các phần tử

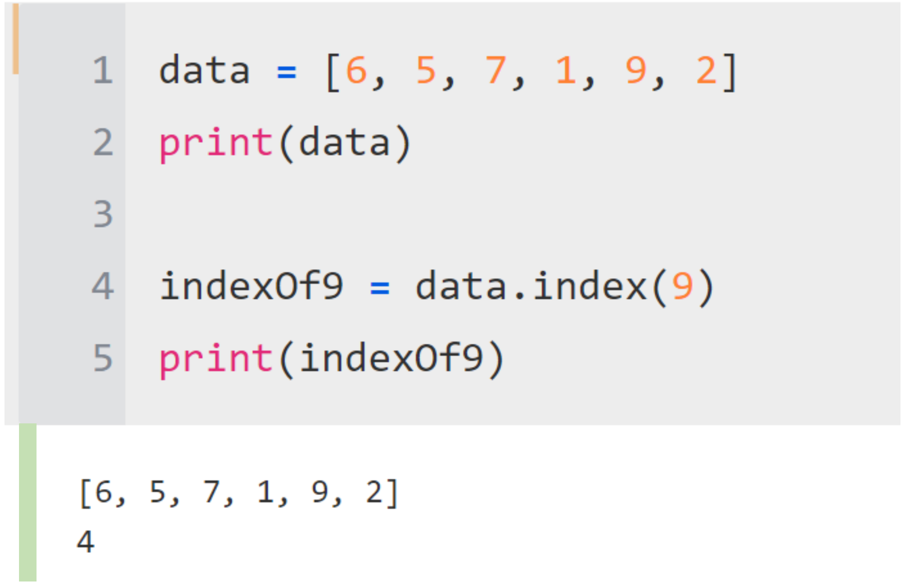
data =

6	5	7	1	9	2

data.reserse()

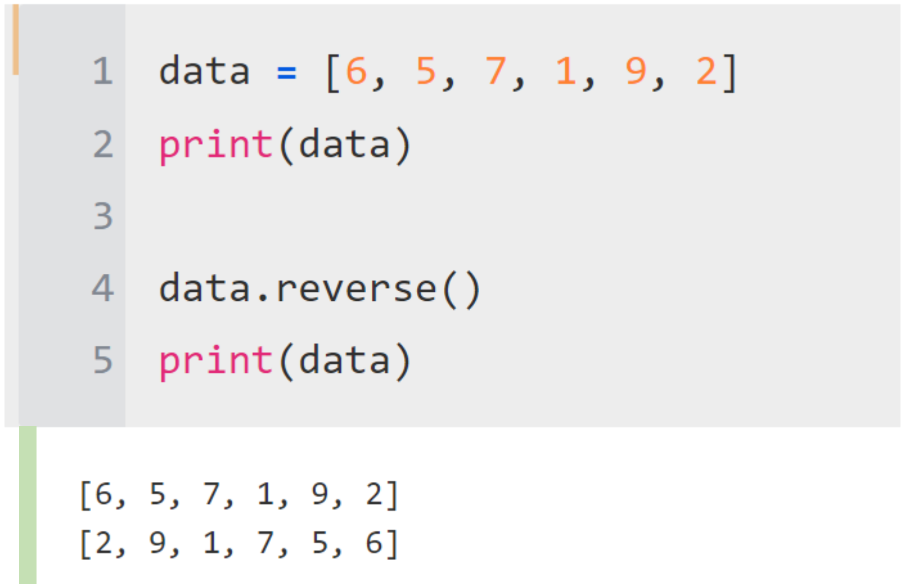
data =

2	9	1	7	5	6



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

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

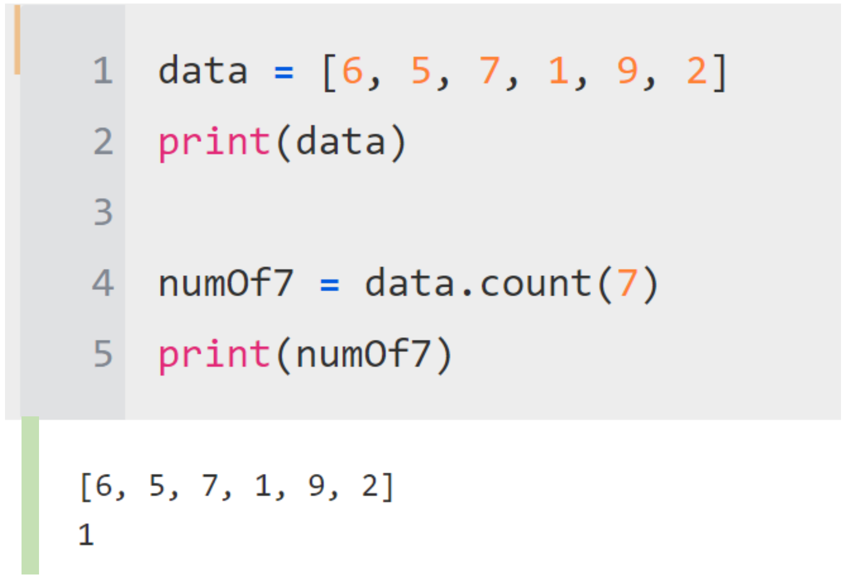
```
[6, 5, 7, 1, 9, 2]
4
```



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

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

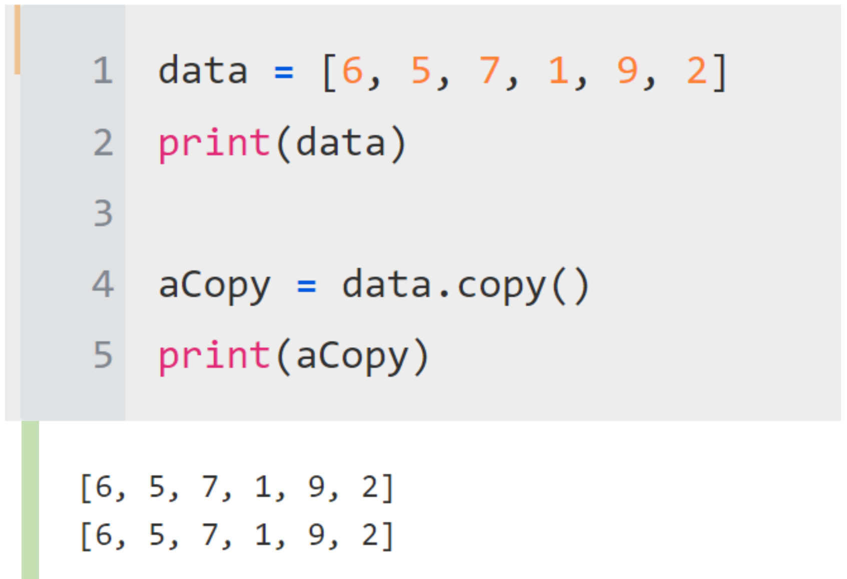
```
[6, 5, 7, 1, 9, 2]
[2, 9, 1, 7, 5, 6]
```

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

Built-in Functions for List

❖ **len(), min(), and max()**

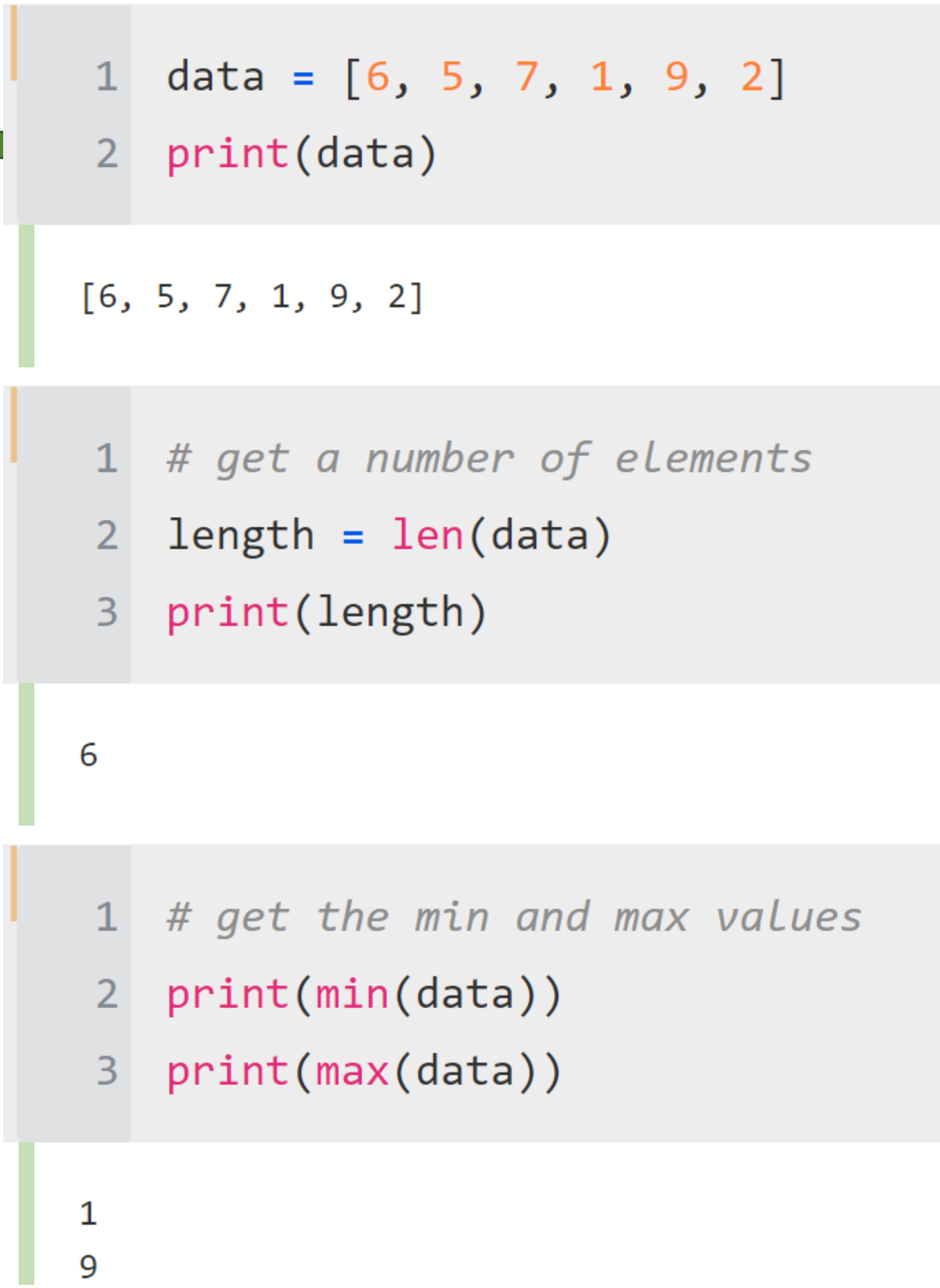
data =

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



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

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

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

```
6
```

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

```
1
9
```

Built-in Functions

❖ **sorted(aList) – Sắp xếp các phần tử**

sorted(iterable, reverse=reverse)

data =	6	5	7	1	9	2

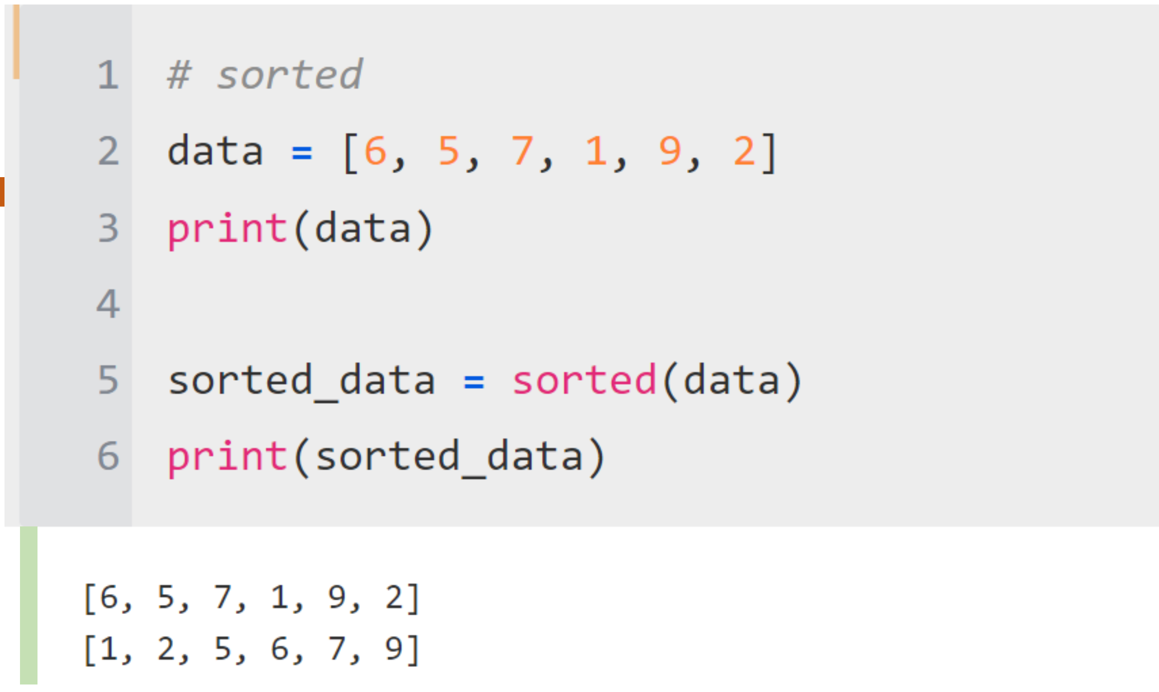
sorted_data = sorted(data)

sorted_data =	1	2	5	6	7	9

data =	6	5	7	1	9	2

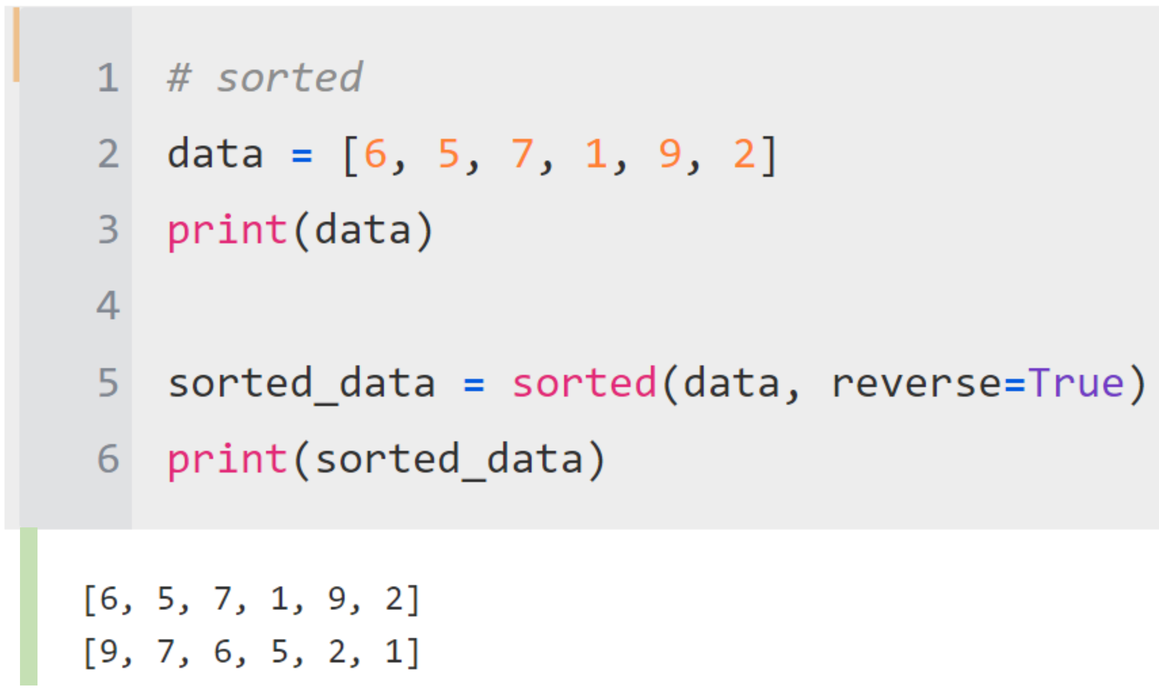
sorted_data = sorted(data, reverse=True**)**

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]
```



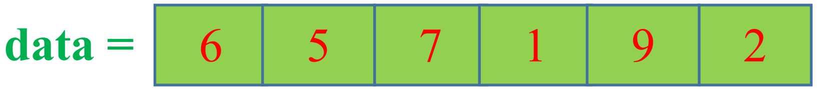
```python
# sorted
data = [6, 5, 7, 1, 9, 2]
print(data)

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

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

Built-in Functions

❖ sum()

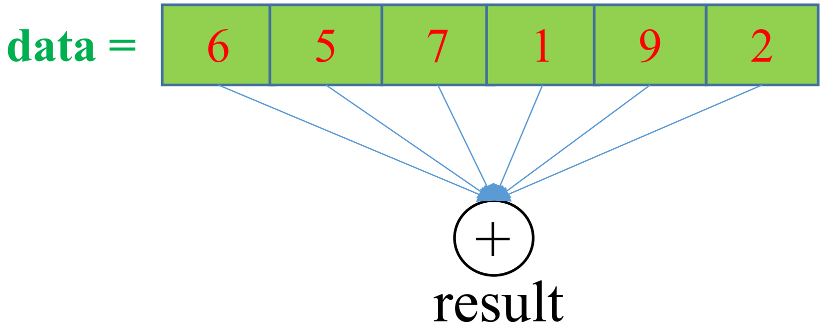
$$summation = \sum_{i=0}^{n-1} data_i$$



# 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
```

Built-in Functions

❖ sum()

$$summation = \sum_{i=0}^{n-1} data_i$$

data =

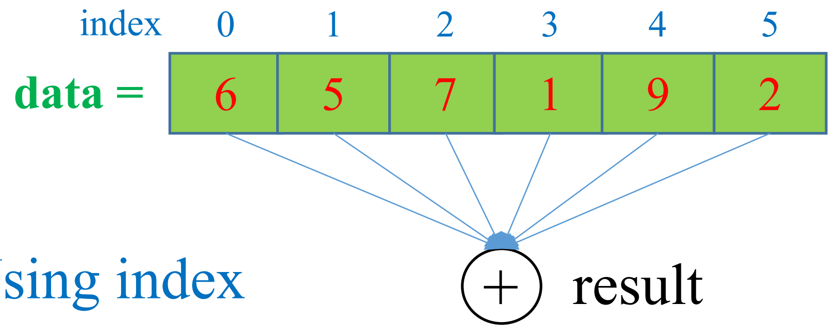
6	5	7	1	9	2

# 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
```



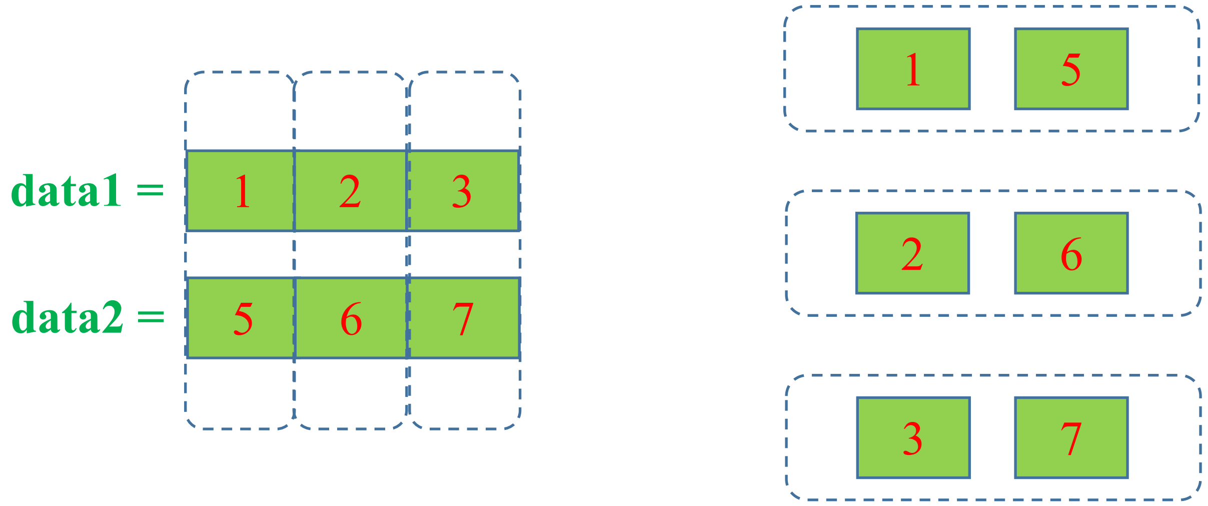
Using index

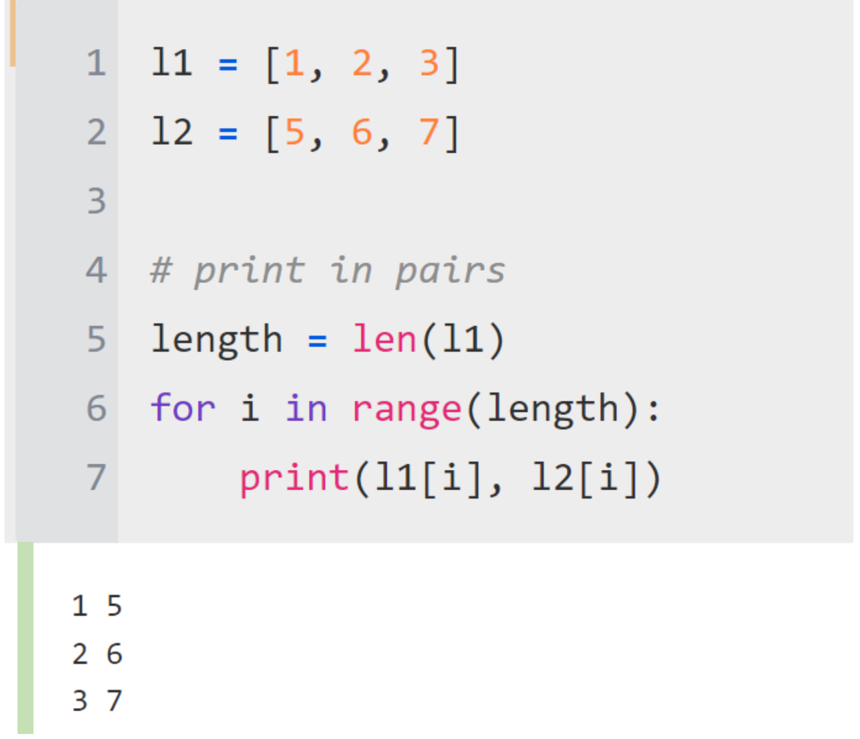
```
# custom summation - way 2
def computeSummation(data):
    result = 0

    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
```

Built-in Functions

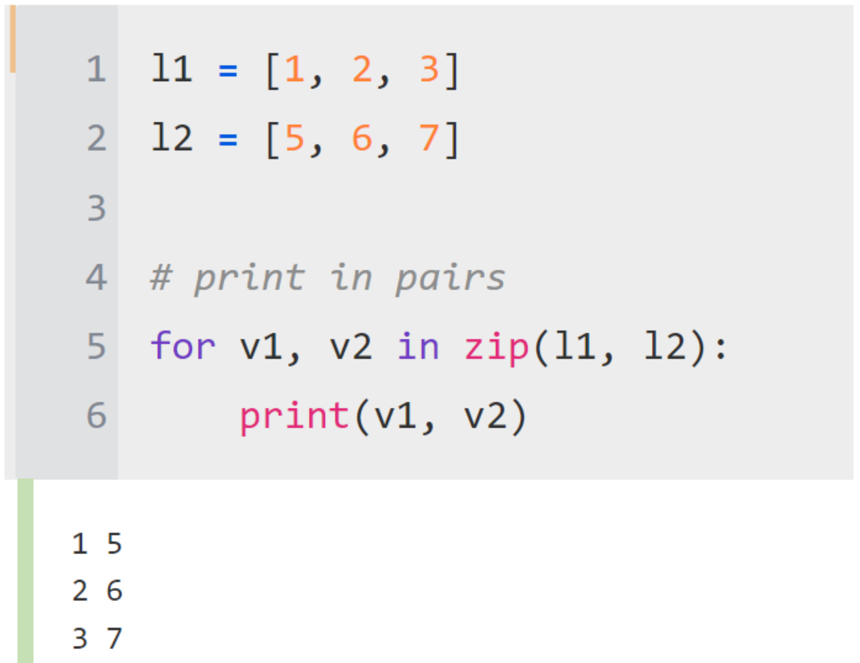
❖ zip()





```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
```

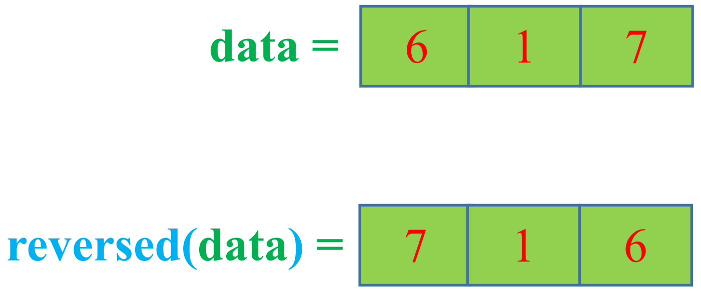


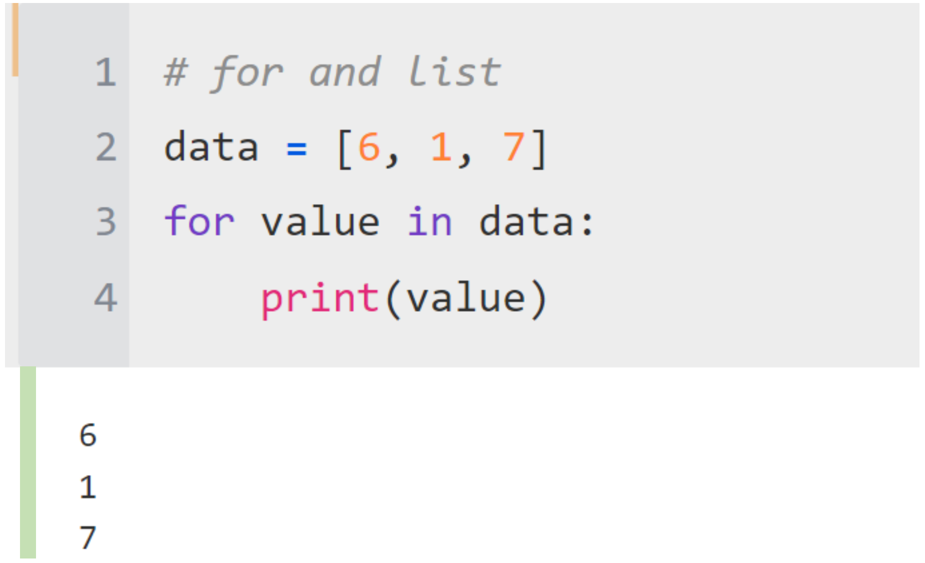
```python
l1 = [1, 2, 3]
l2 = [5, 6, 7]

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

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

Built-in Functions

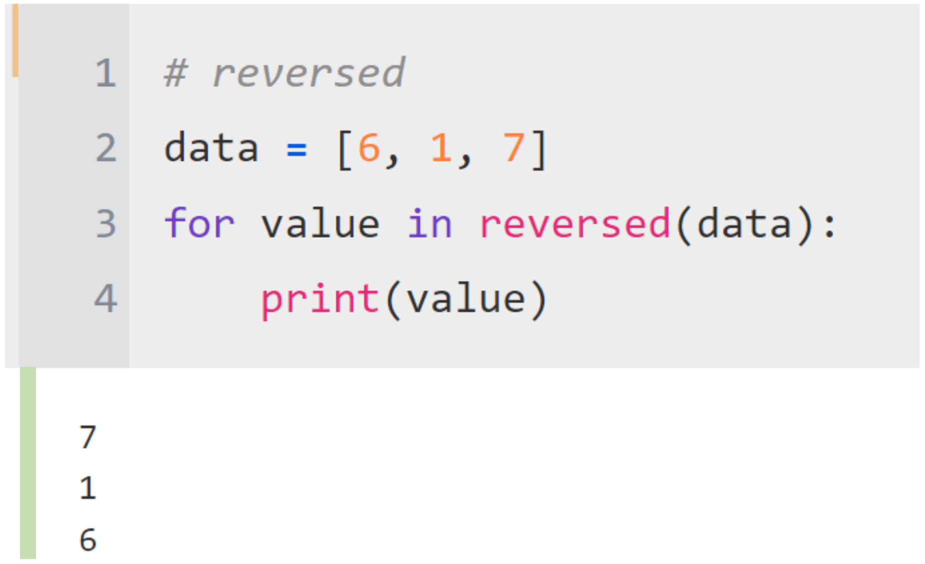
❖ reversed()





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

```
6
1
7
```

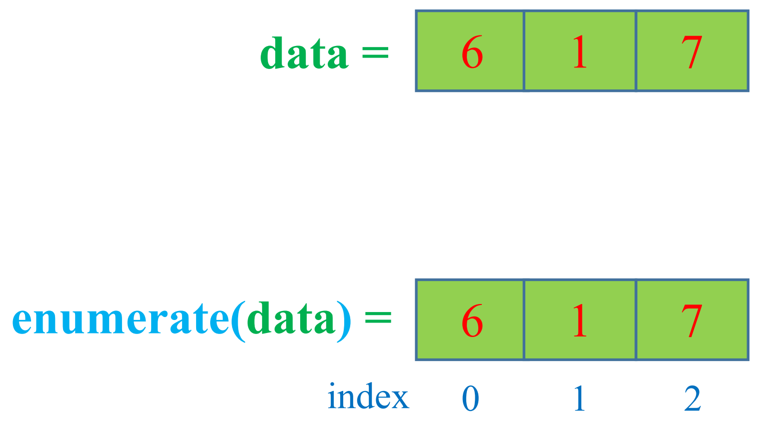


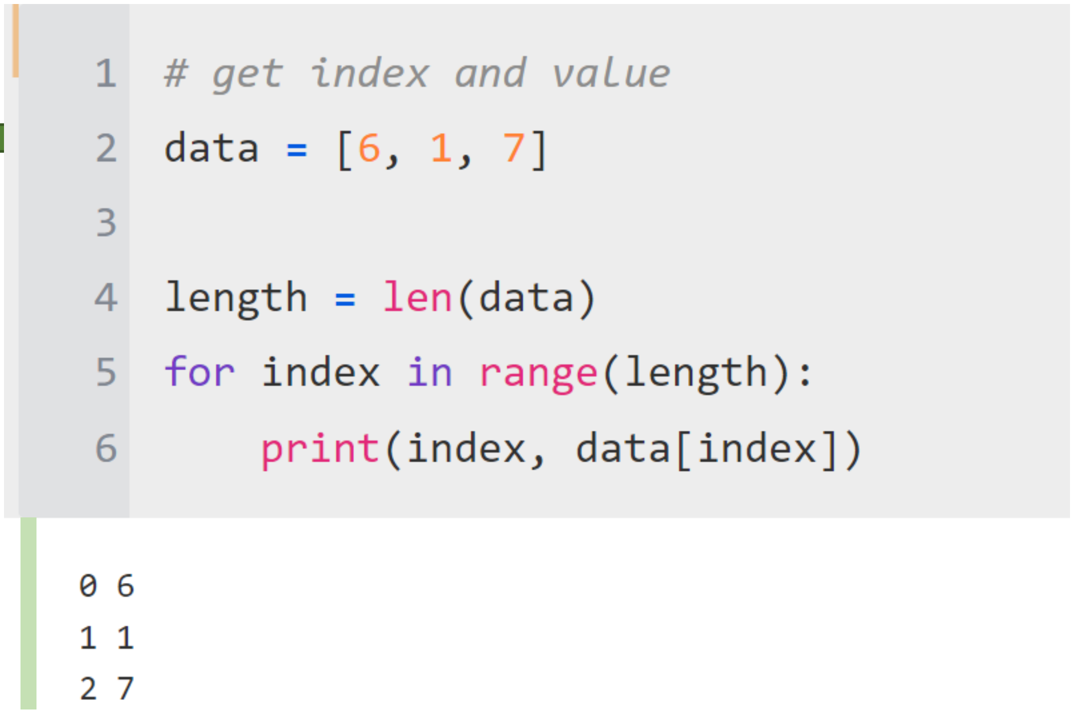
```python
# reversed
data = [6, 1, 7]
for value in reversed(data):
    print(value)
```

```
7
1
6
```

Built-in Functions

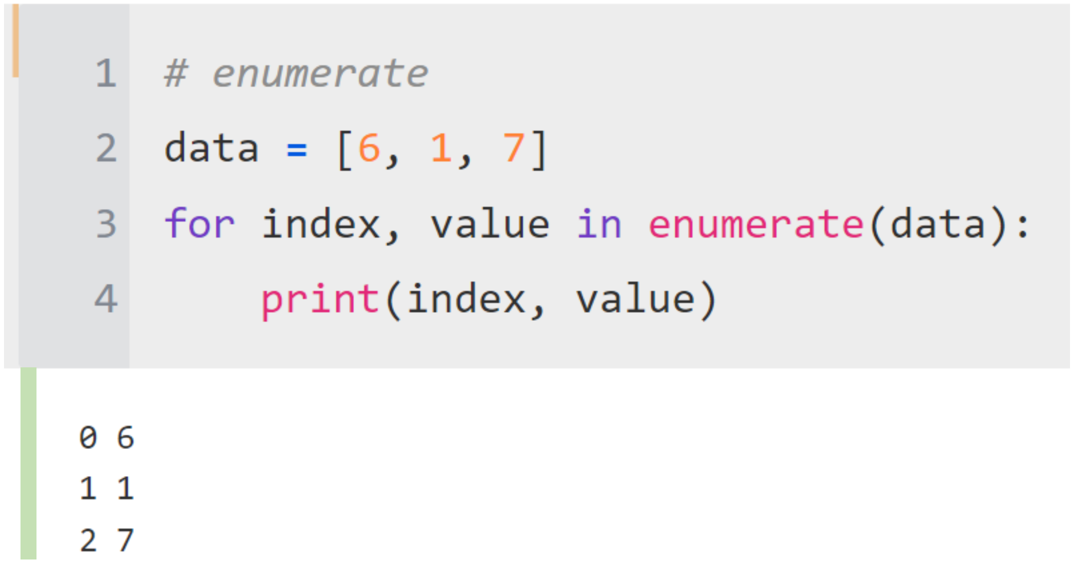
❖ **enumerate()**





```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)
```

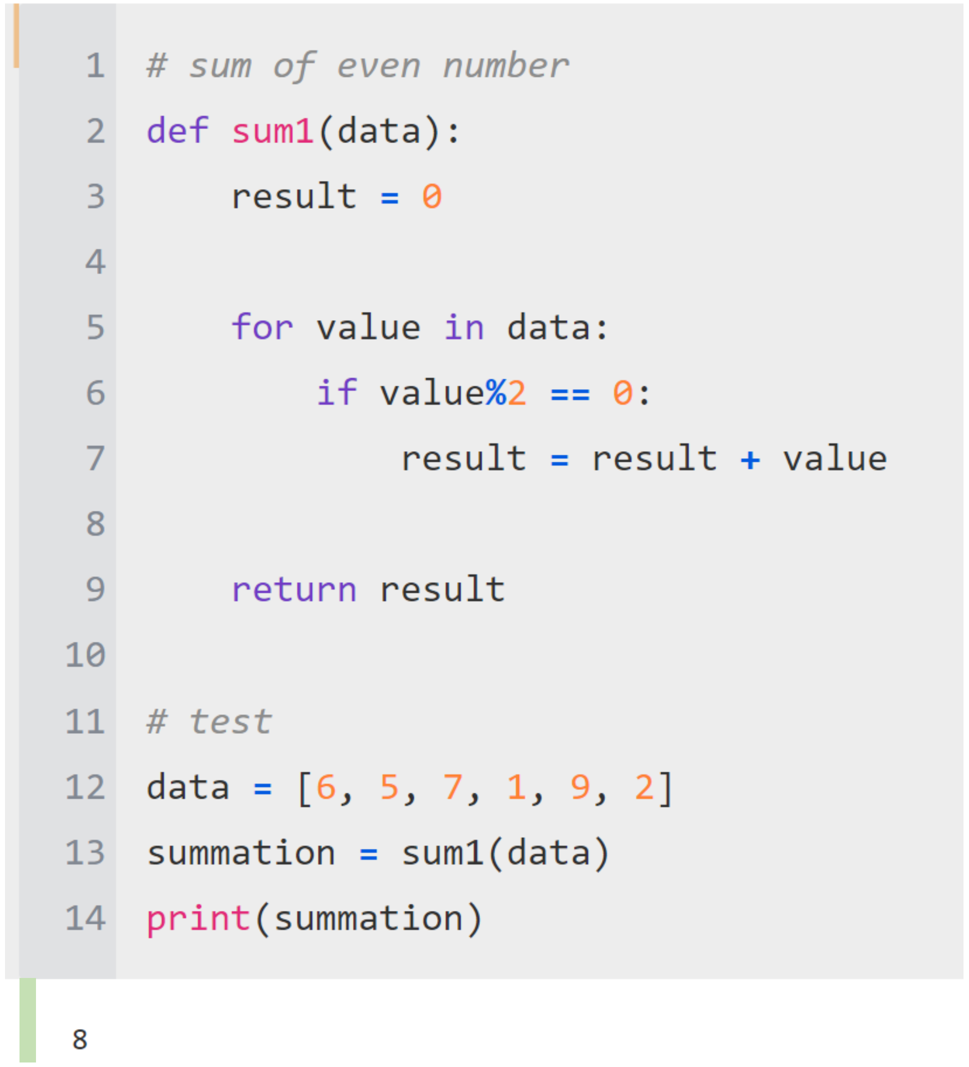
```
0 6
1 1
2 7
```

Examples

Sum of even numbers

data =

6	5	7	1	9	2



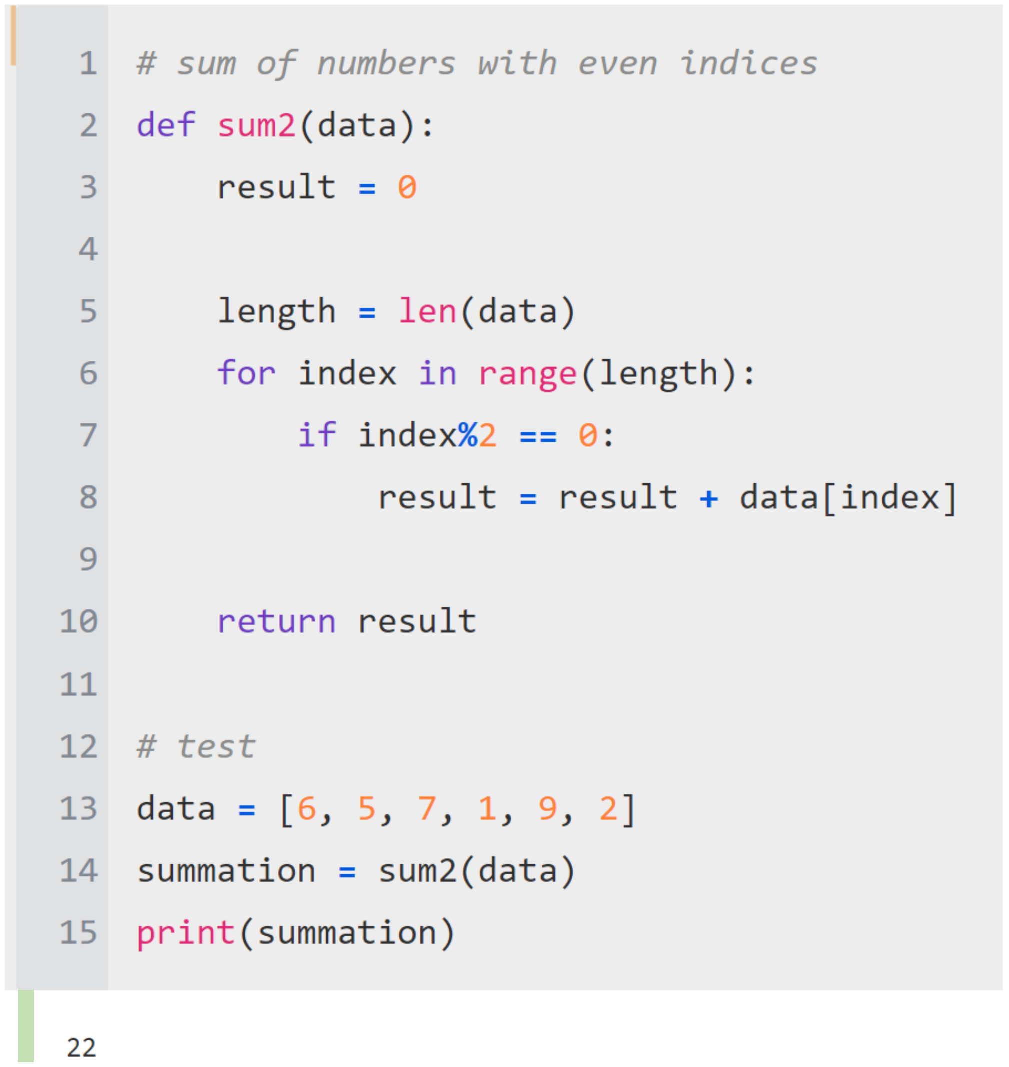
```python
# sum of even number
def sum1(data):
    result = 0

    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

```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]
```

Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$\sigma'(x) = \sigma(x)(1 - \sigma(x))$$

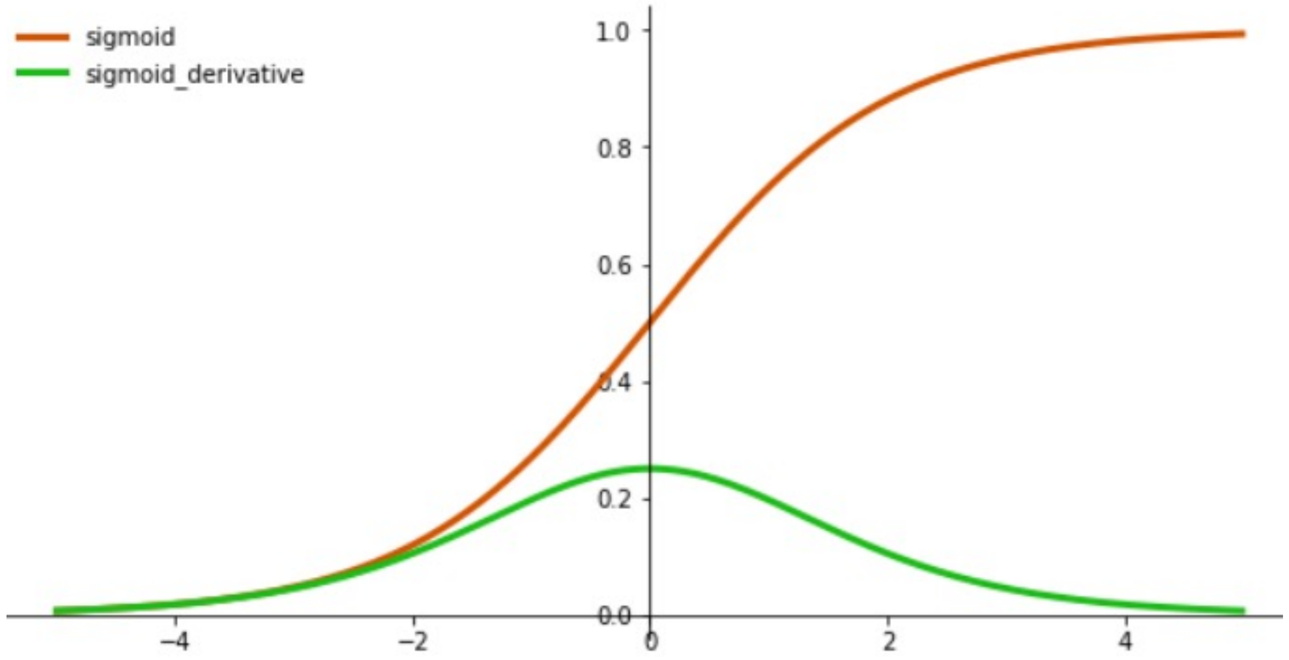
data =

1	5	-4	3	-2

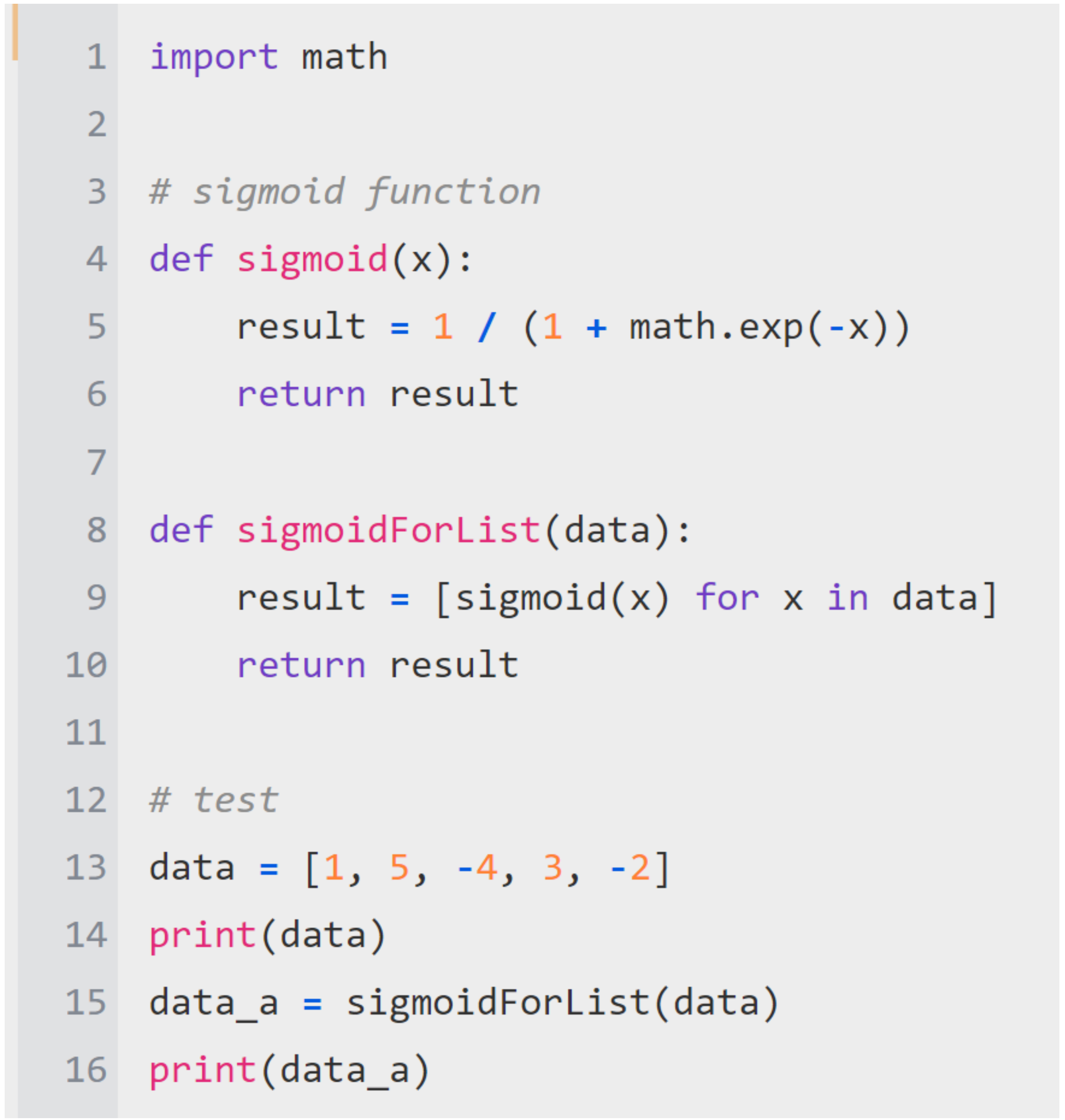
data_a = sigmoid(data)

data_a =

0.731	0.993	0.017	0.95	0.119



List Comprehension



```
import math

# sigmoid function
def sigmoid(x):
    result = 1 / (1 + math.exp(-x))
    return result

def sigmoidForList(data):
    result = [sigmoid(x) for x in data]
    return result

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

ReLU Function

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

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

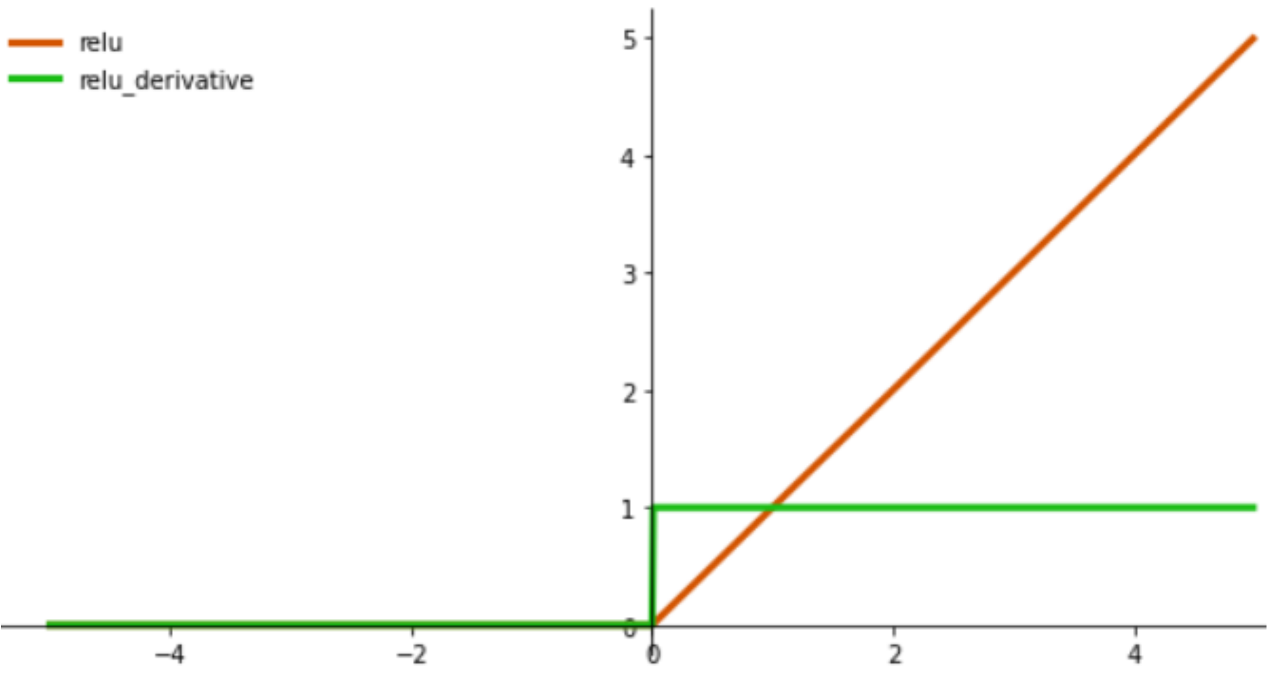
data =

1	5	-4	3	-2

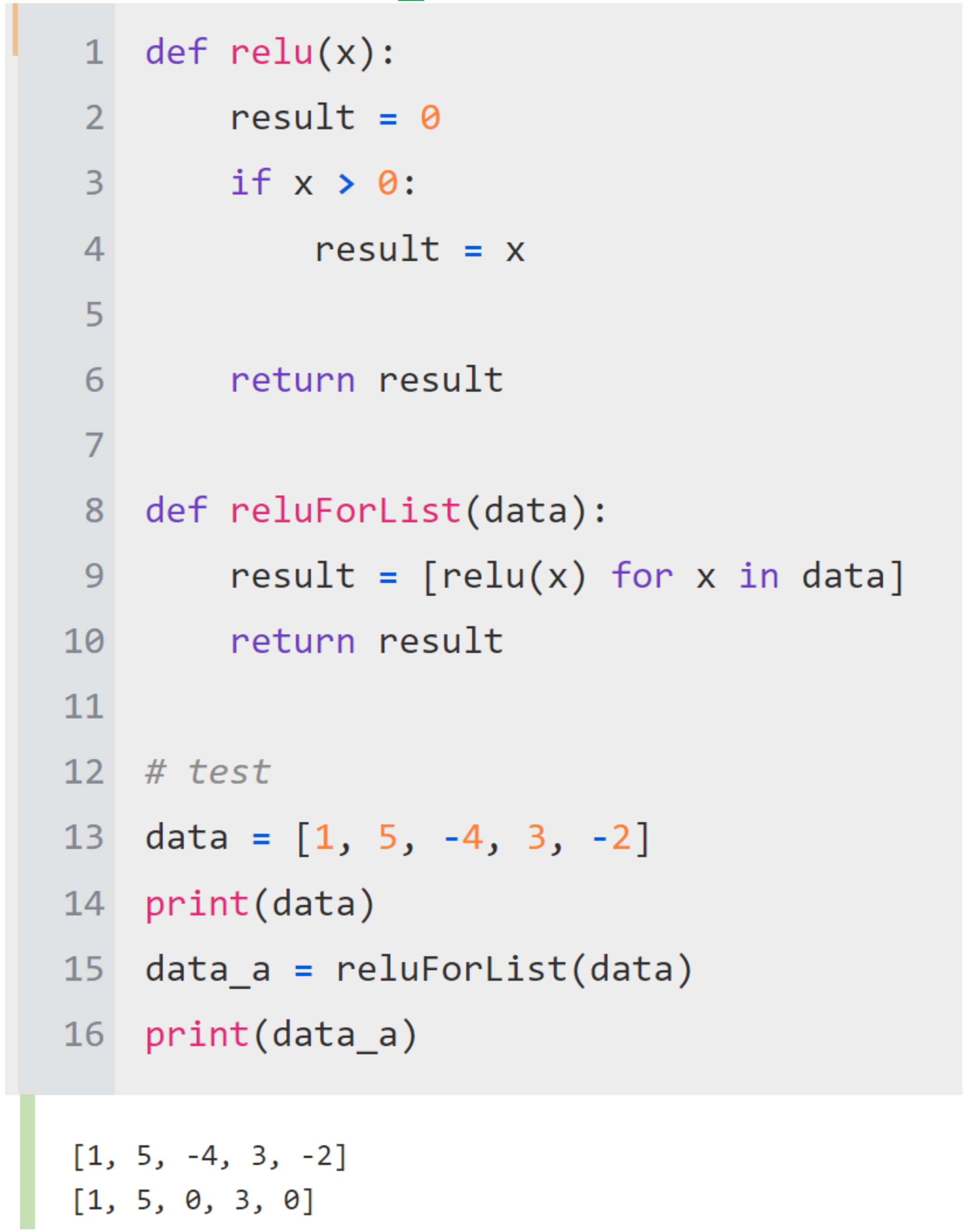
data_a = ReLU(data)

data_a =

1	5	0	3	0



List Comprehension



```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)
```

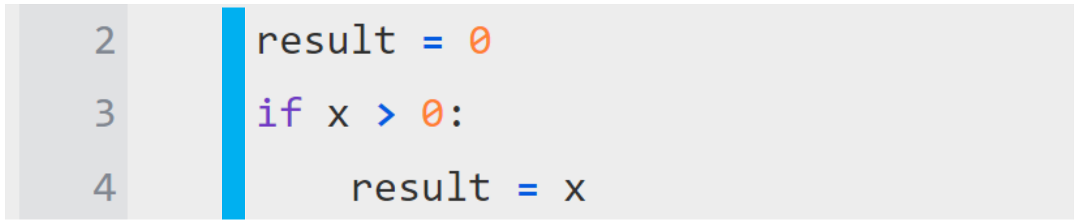
```
[1, 5, -4, 3, -2]
[1, 5, 0, 3, 0]
```

ReLU Function

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

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

List Comprehension



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

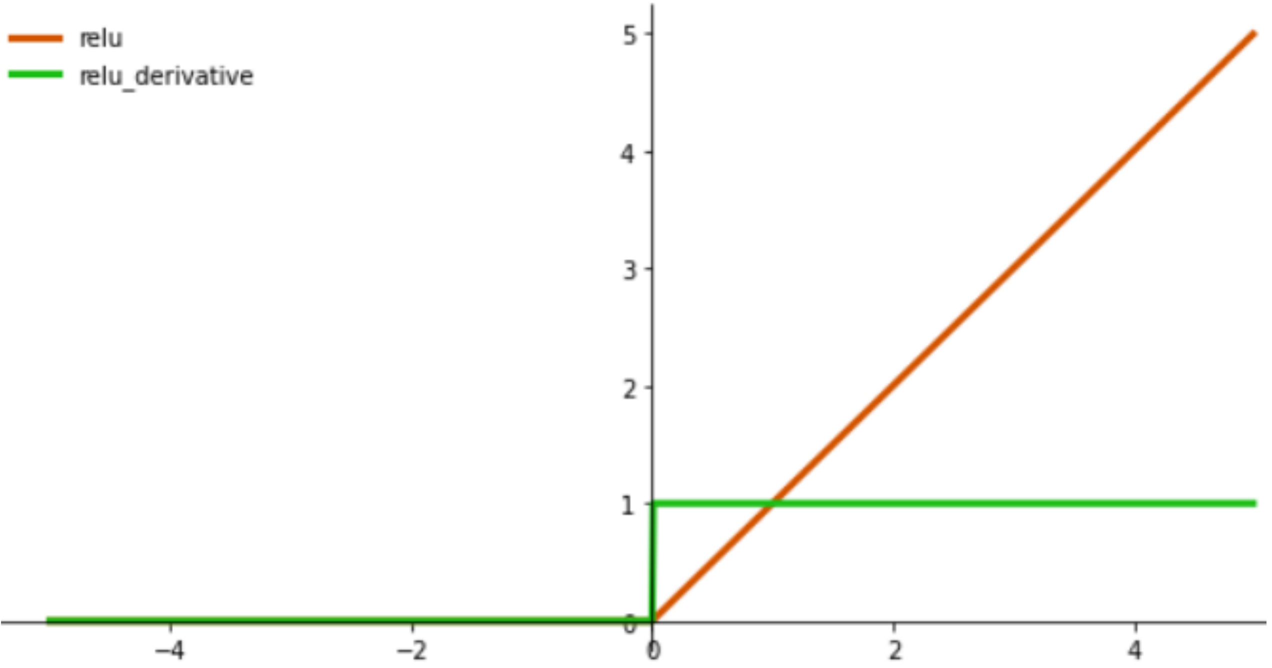
data =

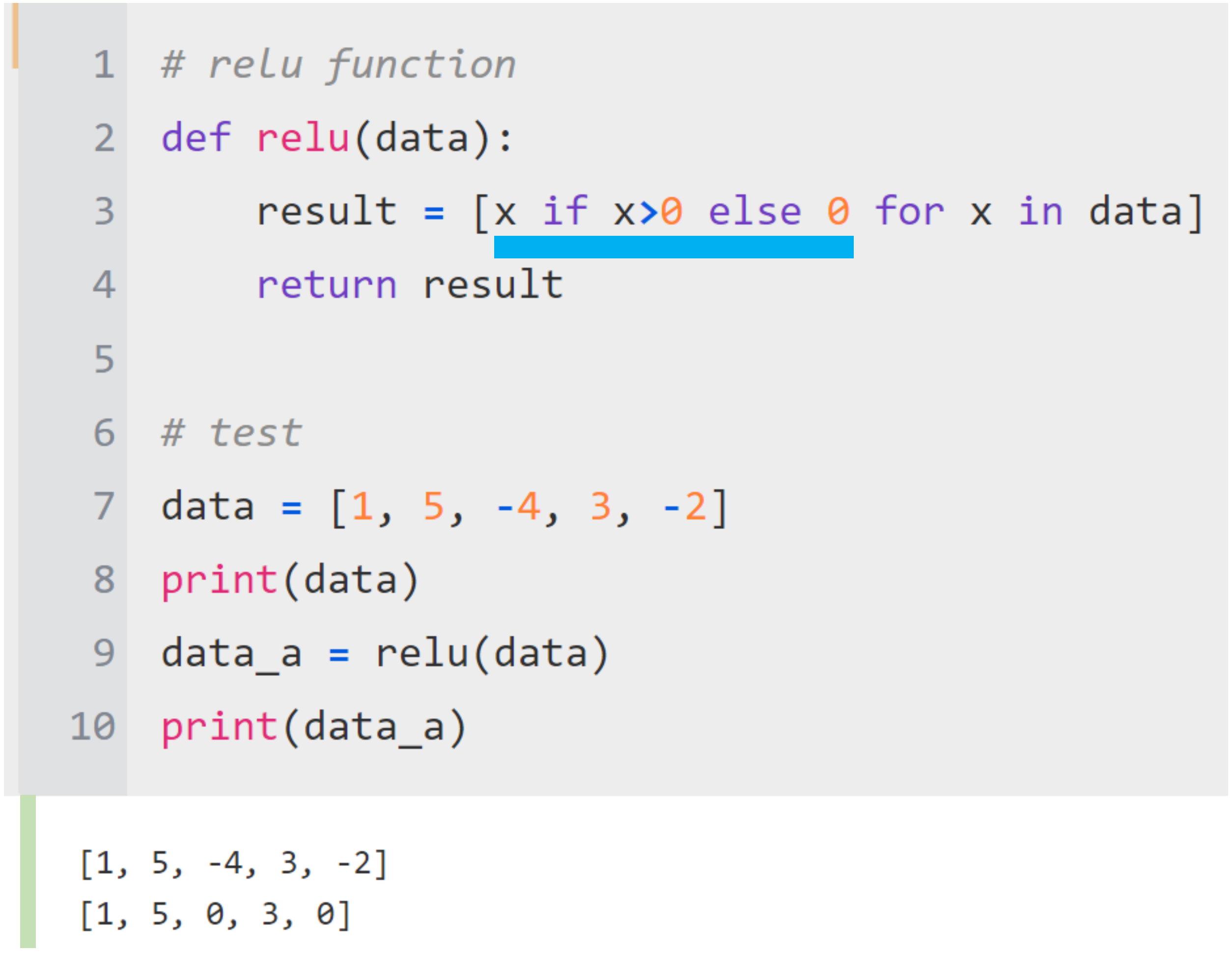
1	5	-4	3	-2

data_a = ReLU(data)

data_a =

1	5	0	3	0





```
# 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]
```

AI VIET NAM
All-in-One 2025

List Comprehension

định tuyến

bộ lọc (optional)

[condition_to_branch_x for x in data condition_to_filter_x]

```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)
```

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

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

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

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

```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)
```

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]
```

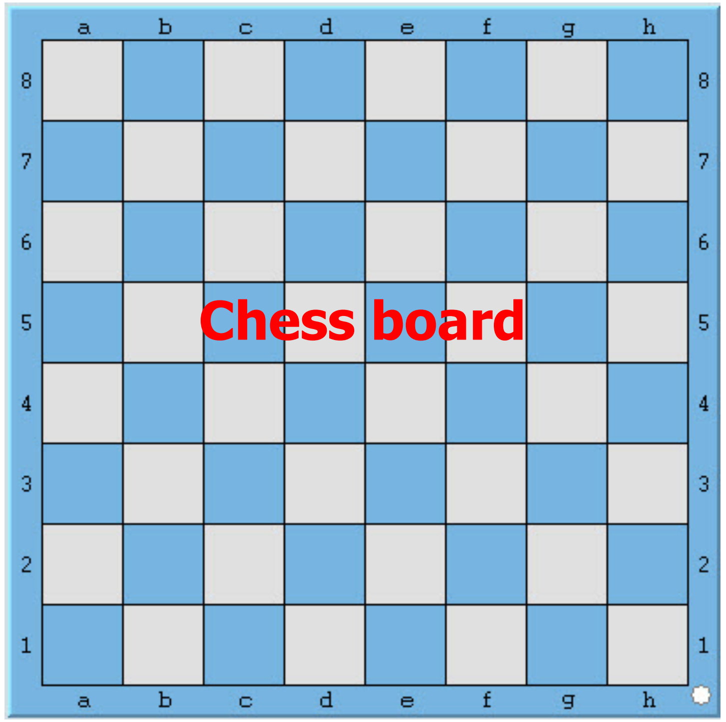
Outline

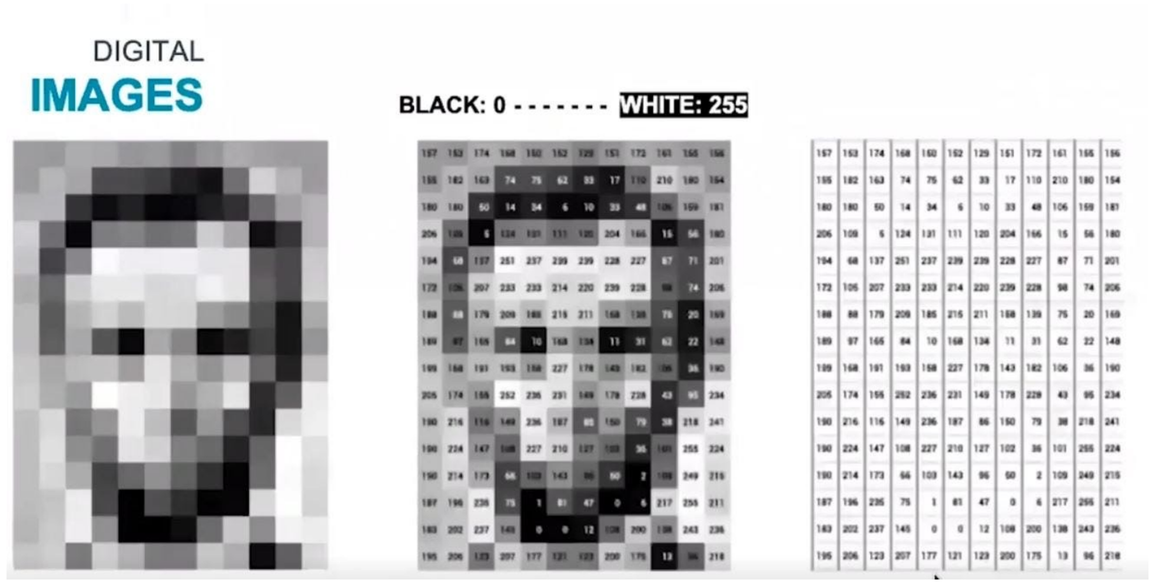
	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

	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]

AI VIET NAM
All-in-One 2025

2D List: Motivation





157	153	174	168	160	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	229	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

2D List

❖ Create 2D Matrix

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]

m[0][0] = 1

m[0][1] = 2

m[2][1] = 8

m[2][2] = 9

Create 2D List

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

Accessing Elements

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

2D List

❖ Iterating Over a 2D Matrix

Column Index

Row Index

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array}$$

```python
# Create a 2D list
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
num_rows = len(m)     # 3 rows
num_cols = len(m[0]) # 2 columns
```

Solution 1

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

Solution 2

```python
for r in range(num_rows):
    for c in range(num_cols):
        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

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 2 & 3 \\ 1 & 4 & 5 & 6 \\ 2 & 7 & 8 & 9 \end{array}$$

Row Index

Column Index

```python
# Create a 2D list
m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
num_rows = len(m)     # 3 rows
num_cols = len(m[0]) # 3 columns
```

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

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

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

```python
# print 2d list
for r in range(num_rows):
    for c in range(num_cols):
        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]]
num_rows = len(G)
num_cols = len(G[0])
N = [[None]*num_cols
     for _ in range(num_rows)]
```

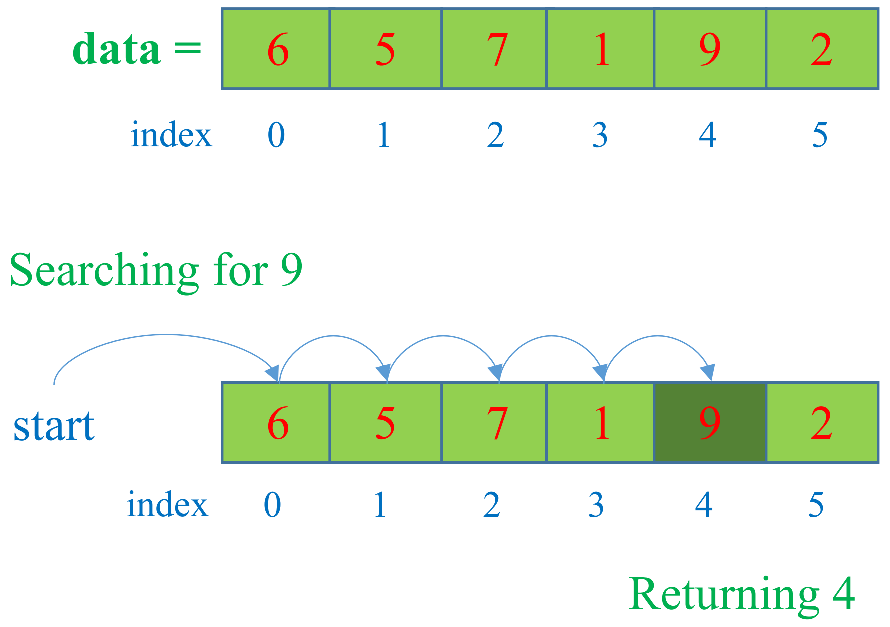
```python
# Do Hadamard Product

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

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

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

QUIZ TIME

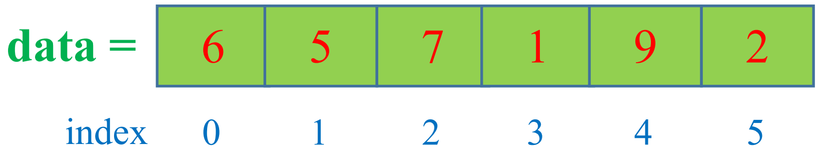
Outline



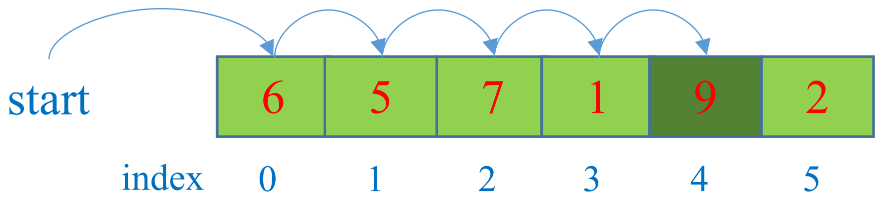
data =
6 5 7 1 9 2
index 0 1 2 3 4 5
Searching for 9
start
6 5 7 1 9 2
index 0 1 2 3 4 5
Returning 4

Algorithms on List

❖ Linear searching

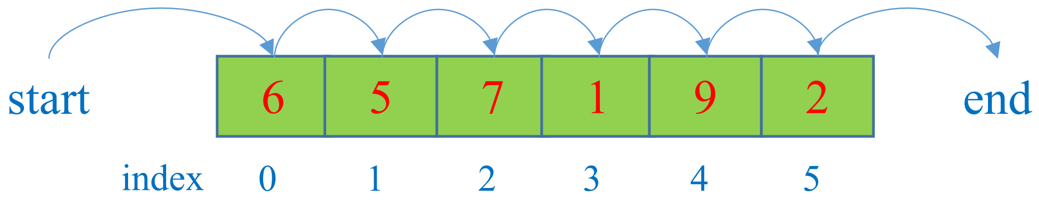


Searching for 9

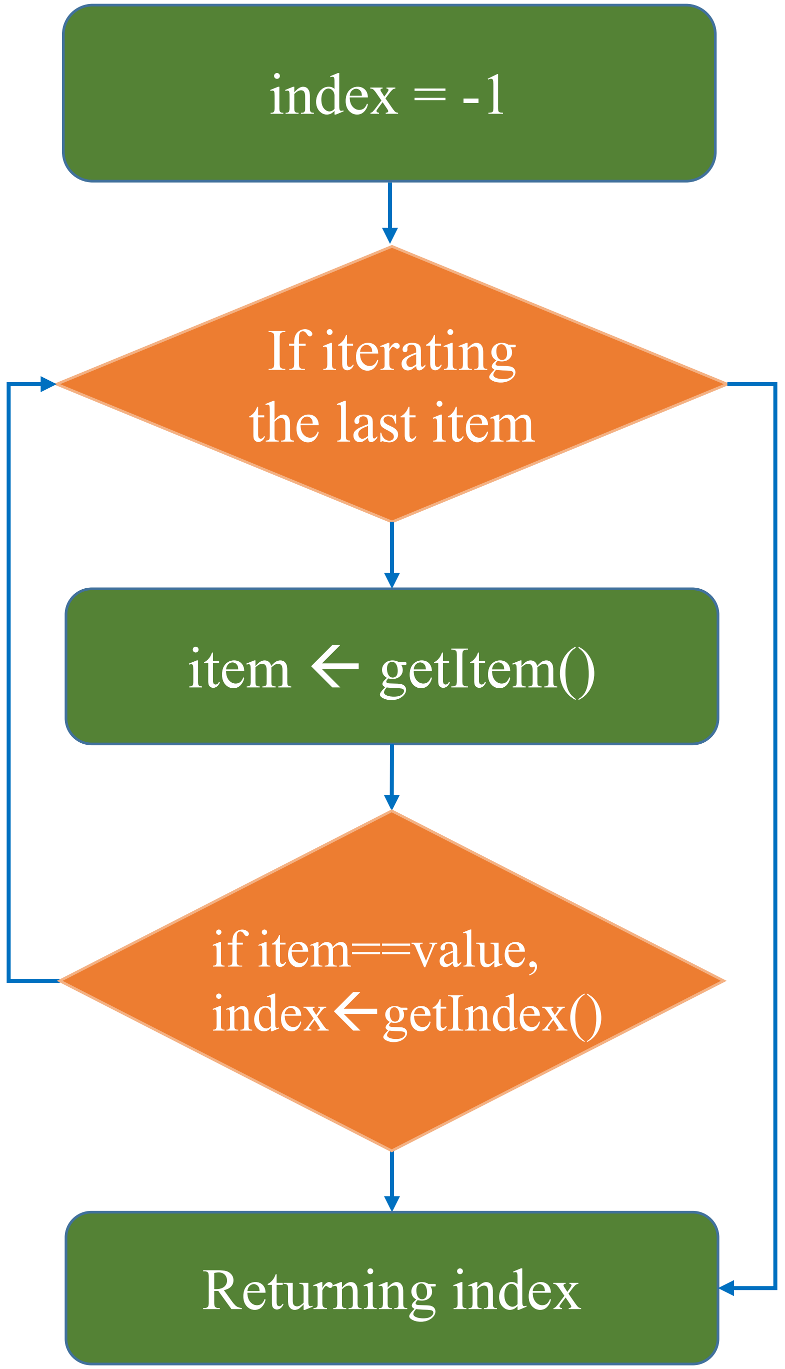


Returning 4

Searching for 8



Returning ?



Algorithms on List

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

data =

6	5	7	1	9	2

result = [ ]

min(data) = 1

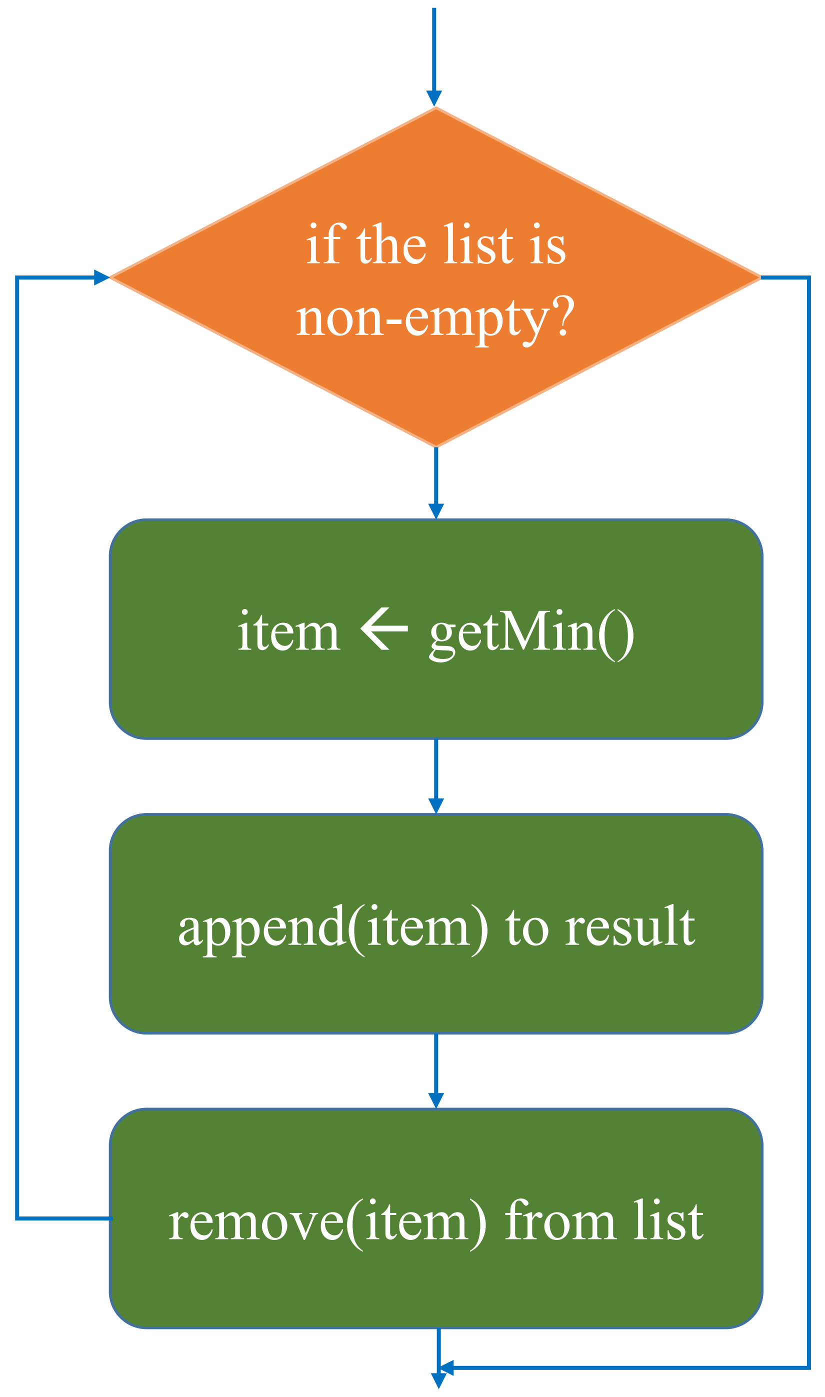
result.append(1) =

1

data.remove(1) =

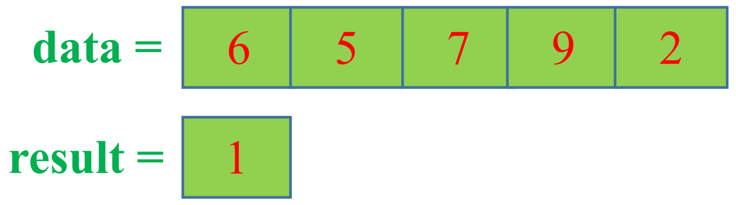
6	5	7	9	2

. . .

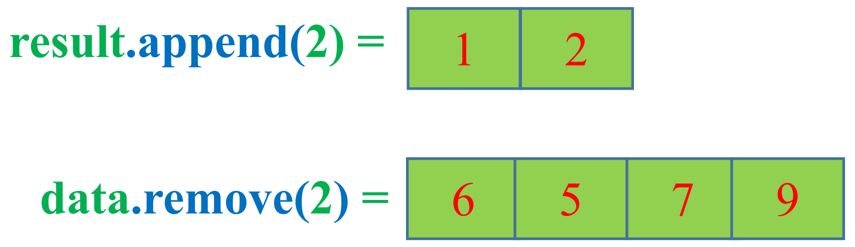


Algorithms on List

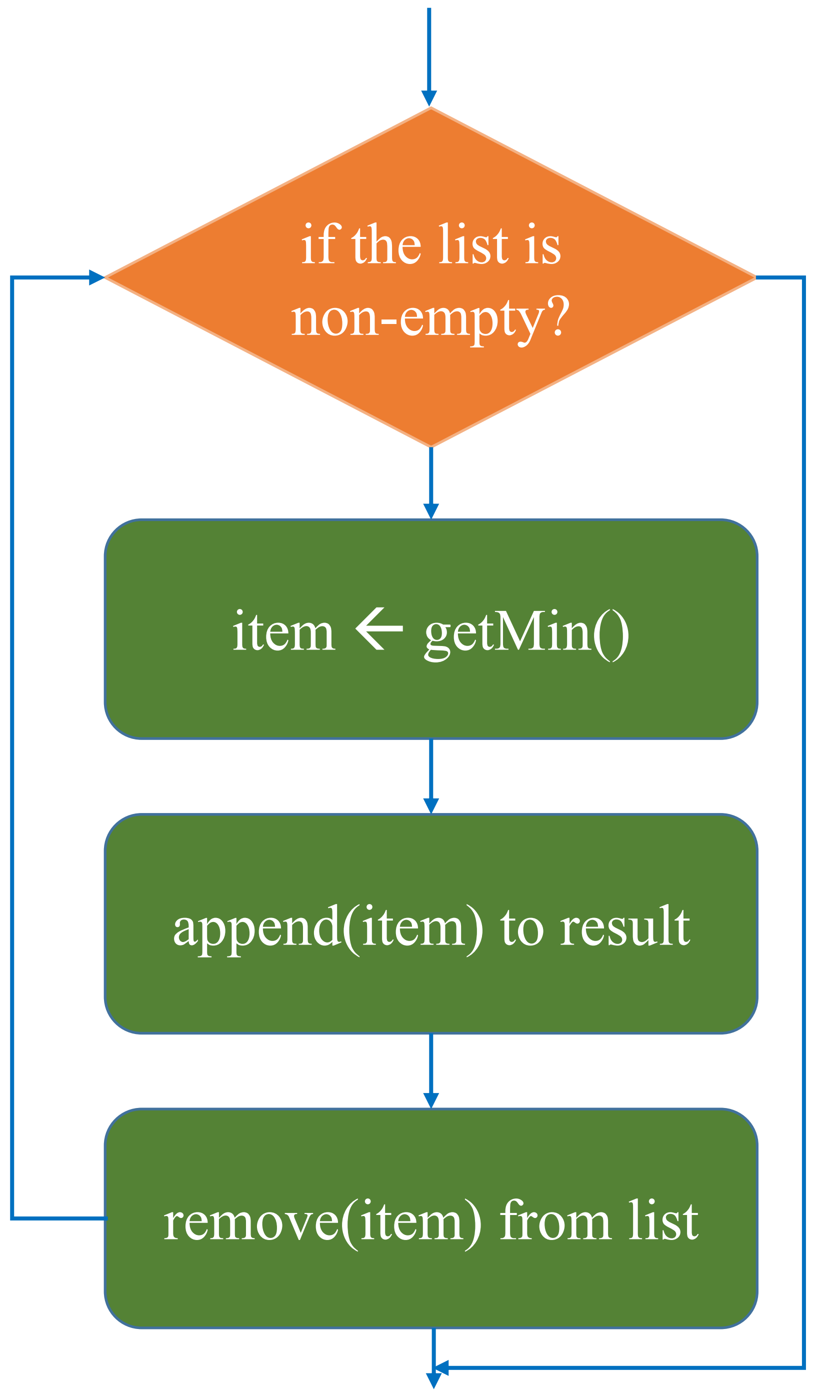
❖ **Sorting using min(), remove(), and append()**



min(data) = 2

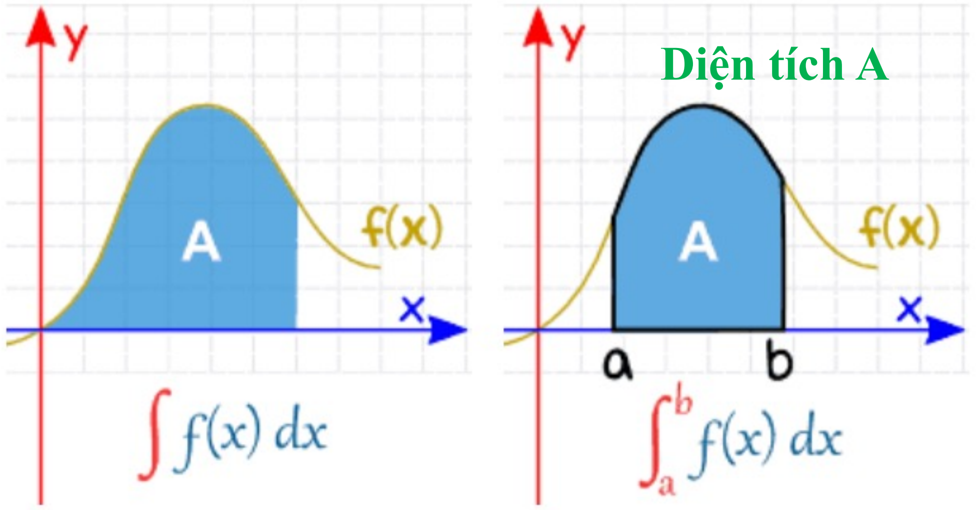


. . .



Case Study: (List and Integral)

Công thức



https://www.mathsisfun.com/calculus/integration-introduction.html

$$\mathrm{F(a)} = \int_{-\infty}^{a} f(x)d(x)$$

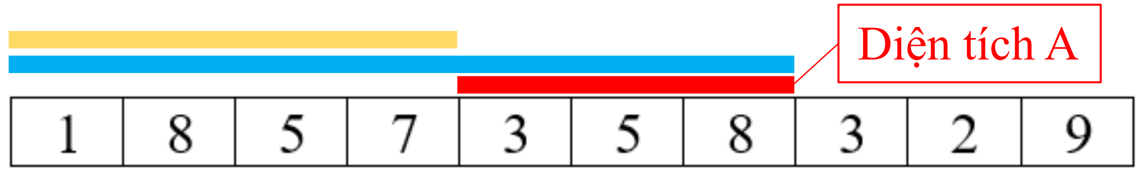
$$\mathrm{F(b)} = \int_{-\infty}^{b} f(x)d(x)$$

Diện tích A

$$\mathrm{A} = \mathrm{F(b)} - \mathrm{F(a)}$$

$$f(x) \geq 0$$

Áp dụng cho hàm rời rạc (1D)



$f(x)$	1	8	5	7	3	5	8	3	2	9
x	0	1	2	3	4	5	6	7	8	9

$$\mathrm{F(3)} = \sum_{x\leq 3} f(x) = f(0) + f(1) + f(2) + f(3)$$
$$= 1 + 8 + 5 + 7 = 21$$

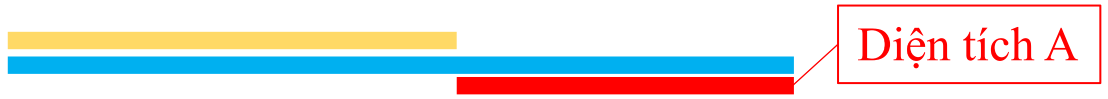
$$\mathrm{F(6)} = \sum_{x\leq 6} f(x) = 1 + 8 + 5 + 7 + 3 + 5 + 8 = 37$$

$$\mathrm{A} = \mathrm{F(6)} - \mathrm{F(3)} = \sum_{4\leq x\leq 6} f(x) = 3 + 5 + 8 = 16$$

AI VIET NAM
All-in-One 2025

Case Study: (List and Integral)

Áp dụng cho hàm rời rạc (1D)



	1	8	5	7	3	5	8	3	2	9
x	0	1	2	3	4	5	6	7	8	9

$$F(3) = \sum_{x\le 3} f(x) = f(0) + f(1) + f(2) + f(3)$$
$$= 1 + 8 + 5 + 7 = 21$$

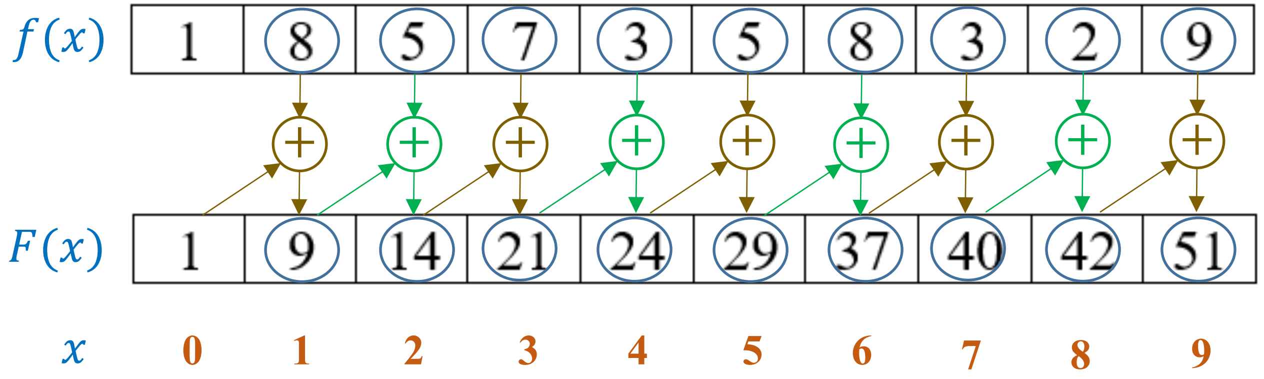
$$F(6) = \sum_{x\le 6} f(x) = 1 + 8 + 5 + 7 + 3 + 5 + 8 = 37$$

$$A = F(6) - F(3) = \sum_{4\le x\le 6} f(x) = 3 + 5 + 8 = 16$$

Tính chất

$$F(x) = f(x) + F(x-1)$$
$$F(7) = f(7) + F(6) = 3 + 37 = 40$$

Xây dựng integral array dùng tính chất $F(x) = f(x) + F(x-1)$



Tính tổng với độ phức tạp ~ O(1)

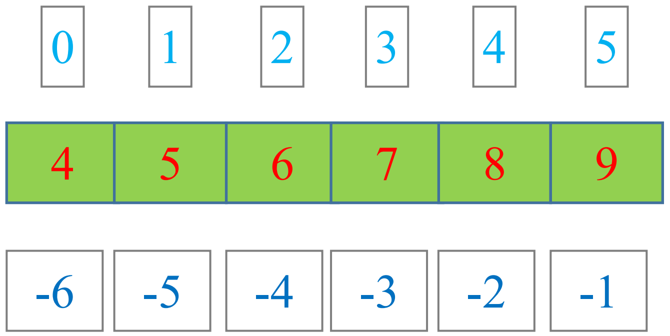
$$\sum_{a\le x\le b} f(x) = F(b) - F(a-1)$$

$$\sum_{4\le x\le 6} f(x) = F(6) - F(3) = 37 - 21 = 16$$

Summary

1D List

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



data[0]

4

data[3]

7

data[-1]

9

data[-3]

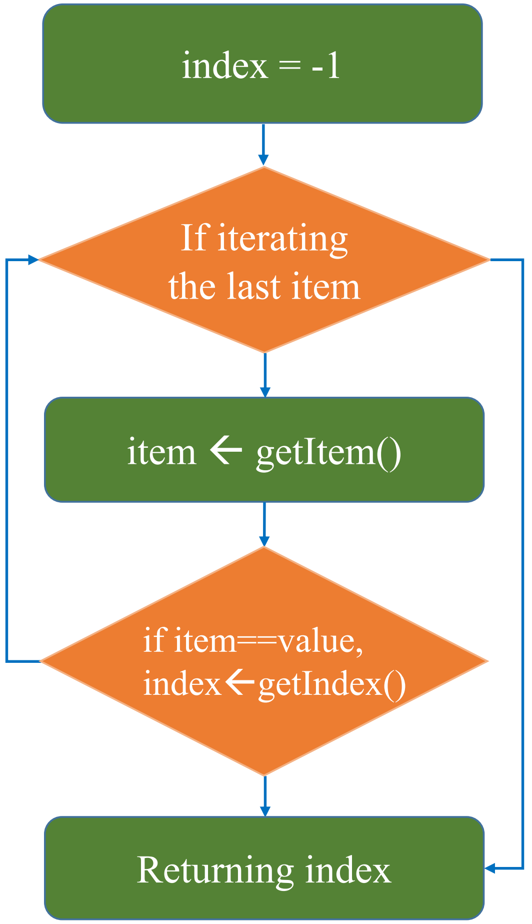
7

2D List

0	1	2	3
1	4	5	6
2	7	8	9
	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]

Algorithms


index = -1
If iterating the last item
item ← getItem()
if item==value, index←getIndex()
Returning index