AI VIET NAM
@aivietnam.edu.vn

Module 01 – Extra Class

LIST

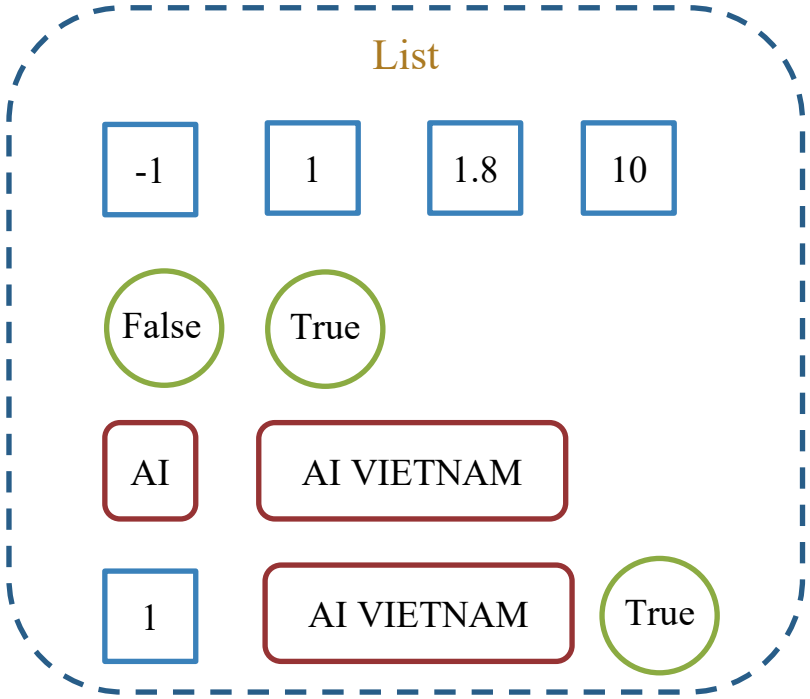
MSc. Nguyễn Quốc Thái

TA Nguyễn Đăng Nhã

Year 2025 code&data

Objectives

List



Build-in Function

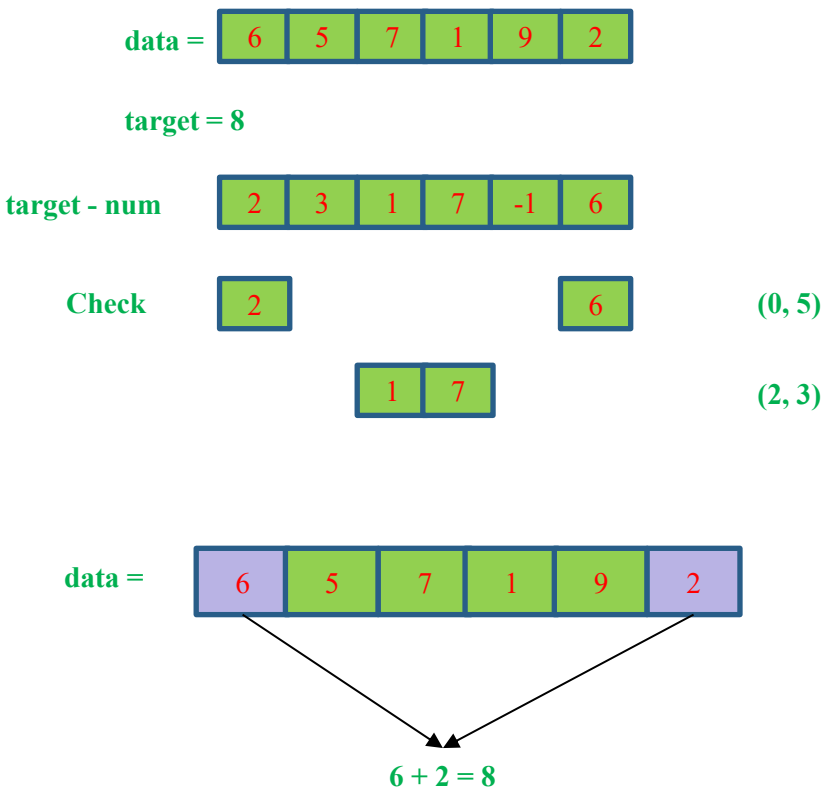
data = | 6 | 5 | 7 | 1 | 9 | 2 |

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

Example



Outline

SECTION 1

Review

SECTION 2

List

SECTION 3

Built-in Functions

SECTION 4

Practice

Review

❖ Function

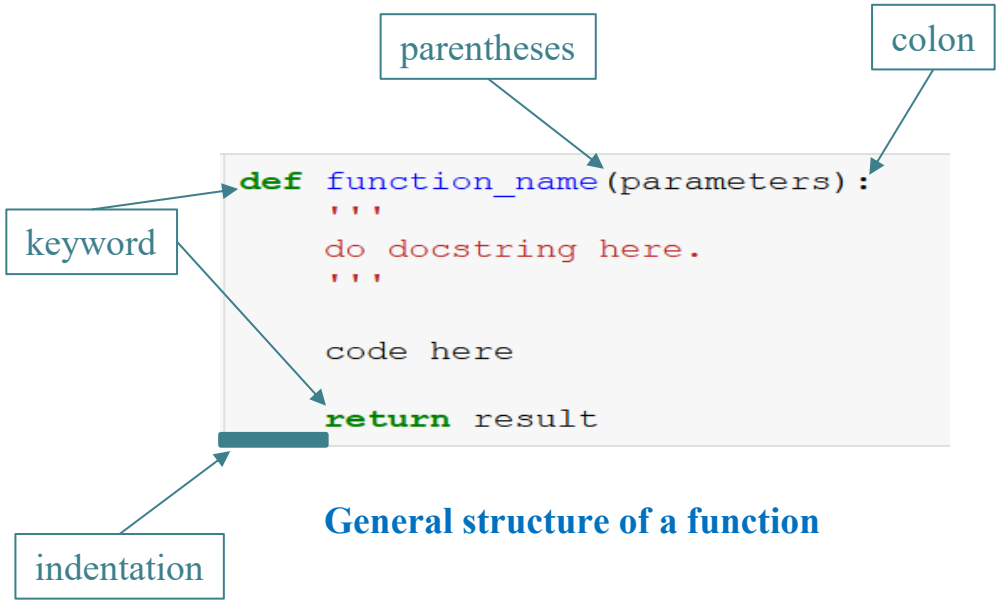
Input → **Function** (Do something with input) → Output

Built-in Functions

print(parameters)

type(parameter)

User-defined Functions



General structure of a function

Review

❖ Built-in vs User-defined Functions

Built-in Functions

```python
sentence = "I Love AI"
print(sentence)
print(type(sentence))
```

```
I love AI
<class 'str'>
```

User-defined Functions

```python
def add_numbers(num_1, num_2):
    total = num_1 + num_2
    return total

num_1 = 10
num_2 = 8
print(add_numbers(num_1, num_2))
```

```
18
```

Review

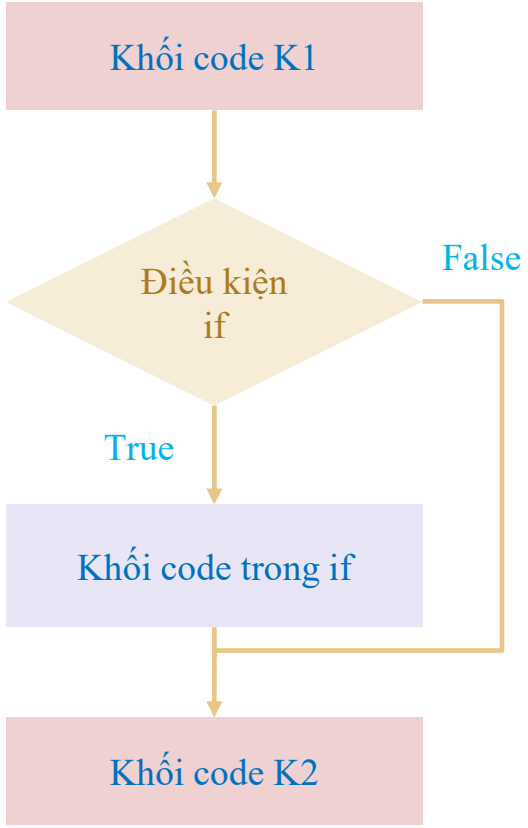
❖ If-Else

colon

keyword

```
# khối code trước if (K1)

if condition:
    # khối code trong if

# khối code sau if (K2)
```

indentation



Outline

Review

SECTION 2

List

Built-in Functions

Practice

List

❖ A container that can contain elements

Elements inside square brackets

list_name = [element-1, …, element-n]

Separated by commas

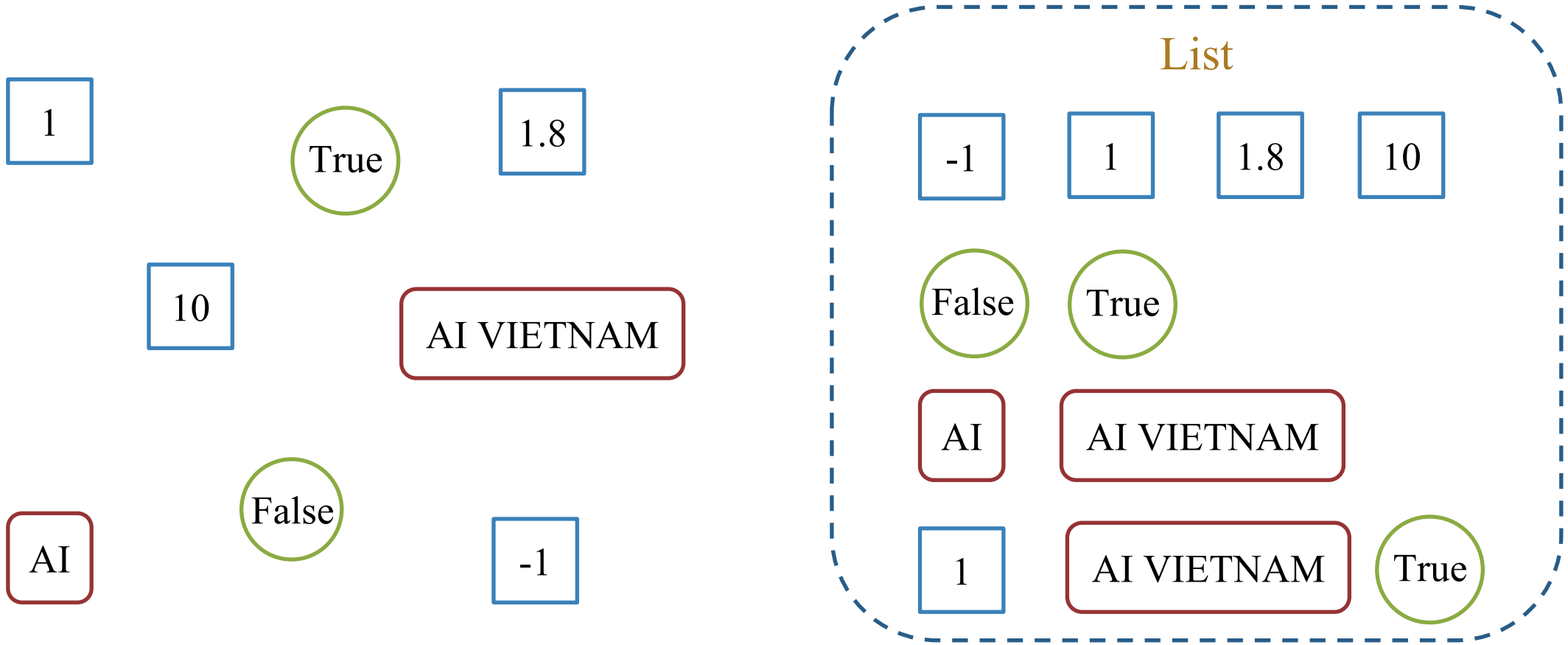
```
# Create a list
data = [4, 5, 6, 7, 8, 9]
print(data)
print(type(data))
print(len(data))
```

Output

```
[4, 5, 6, 7, 8, 9]
<class 'list'>
6
```

List

❖ **Lists are used to store multiple items in a single variable**



List

❖ **Lists are used to store multiple items in a single variable**

```python
import sys

lst = [1, 2, 3, 4, 5]
for i, item in enumerate(lst):
    print(f"Index {i}: Value = {item}, ID = {id(item)}")
```

```
Index 0: Value = 1, ID = 10757736
Index 1: Value = 2, ID = 10757768
Index 2: Value = 3, ID = 10757800
Index 3: Value = 4, ID = 10757832
Index 4: Value = 5, ID = 10757864
```

```python
id(lst)
```

```
136873387510336
```

List

❖ A container that can contain elements

```python
# 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
mixed_list = [True, 5, 'some string', 123.45]
n_list = ["Happy", [2,0,1,5]]

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

Ordered, Duplicated

2	3	3	5	7	7

Indexable

2	3	3	5	7	7
0	1	2	3	4	5

Heterogeneous

2	3.1	“aio”	True	7	7

List

❖ Indexable

➢ Each element in a list is associated with a number, known as a index

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

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

Access elements using index

data[0] **data[3]**

4	7

```
data = [4, 5, 6, 7, 8, 9]
print(data[0])
print(data[3])
```

```
4
7
```

List

❖ Indexable

➢ Each element in a list is associated with a number, known as a 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

Access elements using index

data[0]	data[3]	data[-1]	data[-3]
4	7	9	7

```
data = [4, 5, 6, 7, 8, 9]
print(data[-1])
print(data[-3])
```

```
9
7
```

List

❖ Slicing

➢ Access a section of items from list using the slicing operator.

list[start:end:step]

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

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

data[:3]

4	5	6

data[2:4]

6	7

data[3:]

7	8	9

```
data = [4, 5, 6, 7, 8, 9]
print(data[:3])
print(data[2:4])
print(data[3:])
```

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

List

❖ Slicing

➢ Access a section of items from list using the slicing operator.

list[start:end:step]

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

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

data[::2]

4	6	8

```
data = [4, 5, 6, 7, 8, 9]
print(data[::2])
```

```
[4, 6, 8]
```

List

❖ Slicing

➢ Access a section of items from list using the slicing operator.

list[start:end:step]

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[:-3]

4	5	6

data[-2:-4]

∅

data[1:-3]

5	6

```
data = [4, 5, 6, 7, 8, 9]
print(data[:-3])
print(data[-2:-4])
print(data[1:-3])
```

```
[4, 5, 6]
[]
[5, 6]
```

List

❖ List Method

.append()	+ Add more element in the end of list
.insert()	+ Add element in particular index
.extend()	+ Add list, tuple in the end of list
.remove()	+ Removes the first occurrence value
.pop()	+ Removes item at index and return it
.clear()	+ Delete all elements in list

.sort()	+ Add more element in the end of list
.reverse()	+ Add element in particular index
.copy()	+ Add list, tuple in the end of list
.count()	+ Removes the first occurrence value
.index()	+ Removes item at index and return it

List

❖ Add elements to a Python List

➢ Use the **append()** method to add an element to the end of a Python list.

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

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

Output

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

List

❖ Add elements to a Python List

➢ Use the **insert()** method to add an element at the specified index of a Python list.

data =

6	5	7	1	9	2

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

data =

4	6	5	7	1	9	2

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

Output

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

List

❖ Add elements to a Python List

➢ Use the **extend()** method to add elements to a list from other iterables.

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

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

Output

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

List

❖ Updating an element

➢ Change the items of a list by assigning new values using the = operator .

data =

6	5	7	1	9	2

thay đổi phần tử thứ 1
data[1] = 4

data =

6	4	7	1	9	2

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

Output

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

List

❖ Delete an element from a list

➢ Using the **remove()** and **pop()** method.

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)
print(data)
```

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

Output

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

List

❖ Delete an element from a list

➢ Using ‘del’ keyword to delete objects or **clear()** to removal 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]
print(data)
del data[1:3]
print(data)
```

Output

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

data =

6	5	7	1	9	2

data.clear()

data = []

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

Output

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

List

❖ Index() method: Returns the index of the first matched item

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.index(9))
```

```
4
```

❖ Reverse() method: Reverses the item of the list

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

❖ Count() method: Returns the count of the specified item in the list

data =

6	5	7	1	9	2

trả về số lần phần tử 7 xuất hiện trong list
data.count(7) = 1

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

```
1
```

❖ Copy() method: Returns the shallow copy of the list

data =

6	5	7	1	9	2

data_copy = data.copy()

data_copy =

6	5	7	1	9	2

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

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

List

❖ Sort() method: Sorts the list in ascending/descending order

data =

6	5	7	1	9	2

data.sort()

data =

1	2	5	6	7	9

```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]
```

data =

6	5	7	1	9	2

data.sort(reverse = True)

data =

9	7	6	5	2	1

```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]
```

List

+ and * operators

data1 = | 6 | 5 | 7 |

data2 = | 1 | 9 | 2 |

data = data1 + data2 # nối 2 list

data = | 6 | 5 | 7 | 1 | 9 | 2 |

```
data1 = [6, 5, 7]
data2 = [1, 9, 2]
data = data1 + data2
print(data)
```

Output

```
[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 |

```
data = [6, 5]
print(data)
data_m = data*3
print(data_m)
```

Output

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

Outline

SECTION 1

Review

SECTION 2

List

SECTION 3

Built-in Functions

SECTION 4

Practice

Built-in Functions for List

❖ len(), min(), 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
# get a number of elements

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

```
6
```

```python
# get the min and max values

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

```
1
9
```

Built-in Functions for List

❖ **sum(): returns a number, the sum of all elements in a list**

sum(iterable, start)

data =

6	5	7	1	9	2

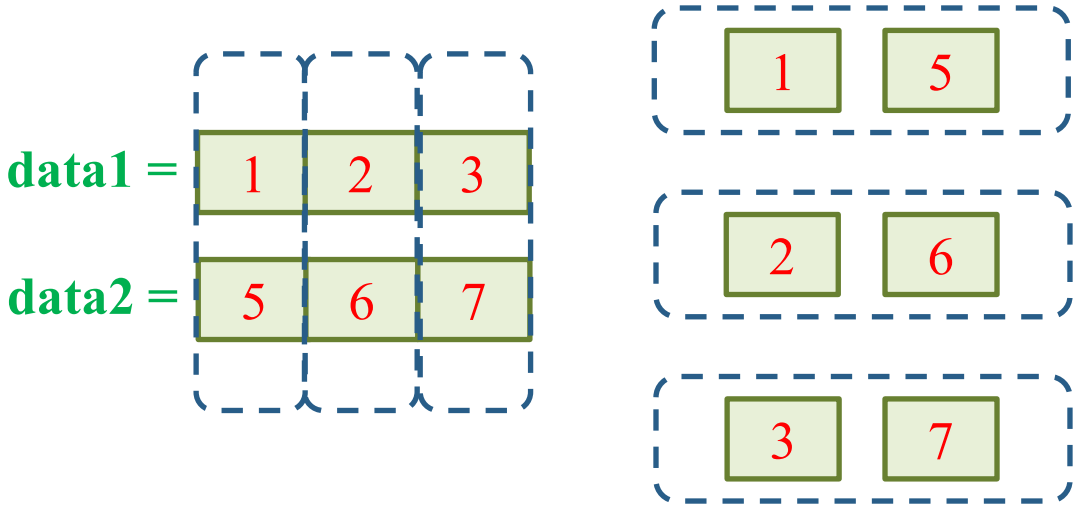
sum(data)
=> 30

```
data = [6, 5, 7, 1, 9, 2]
print(sum(data))
# Output 30
data = [6, 5, 7, 1, 9, 2]
# start: a value that is added to the return value
print(sum(data, 7))
# Output 37
```

Built-in Functions for List

❖ **zip(): takes iterable containers and returns a single iterator object, having mapped values from all the containers.**

zip(*iterators)




```python
data1 = [1, 2, 3]
data2 = [5, 6, 7]

for v1, v2 in zip(data1, data2):
    print(v1, v2)
```

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

Built-in Functions for List

❖ **reversed(): returns a reversed iterator object**

reversed(iterable)

data =

6	1	7

reversed(data) =

7	1	6

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

```
7
1
6
```

Built-in Functions for List

❖ **Sorted(): returns a arranged list**

sorted(iterable, is_reverse)

data =

6	5	7	1	9	2

sorted(data)

data =

1	2	5	6	7	9

sorted(data, reverse=True)

data =

9	7	6	5	2	1

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

Output

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

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

Output

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

Built-in Functions for List

❖ **enumerate(): adds a counter to an iterable and returns it as an enumerate object (iterator with index and the value)**

reversed(iterable, start)

data =

6	1	7

enumerate(data) =

6	1	7

index 0 1 2

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

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

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

```
7 6
8 1
9 7
```

QUIZ TIME

Outline

SECTION 1

Review

SECTION 2

List

SECTION 3

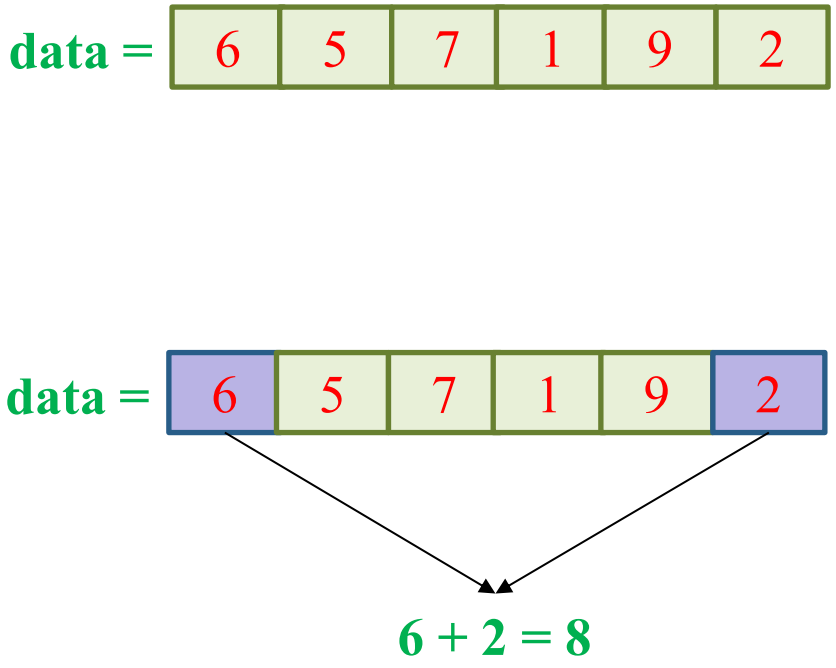
Built-in Functions

SECTION 4

Practice

Practice

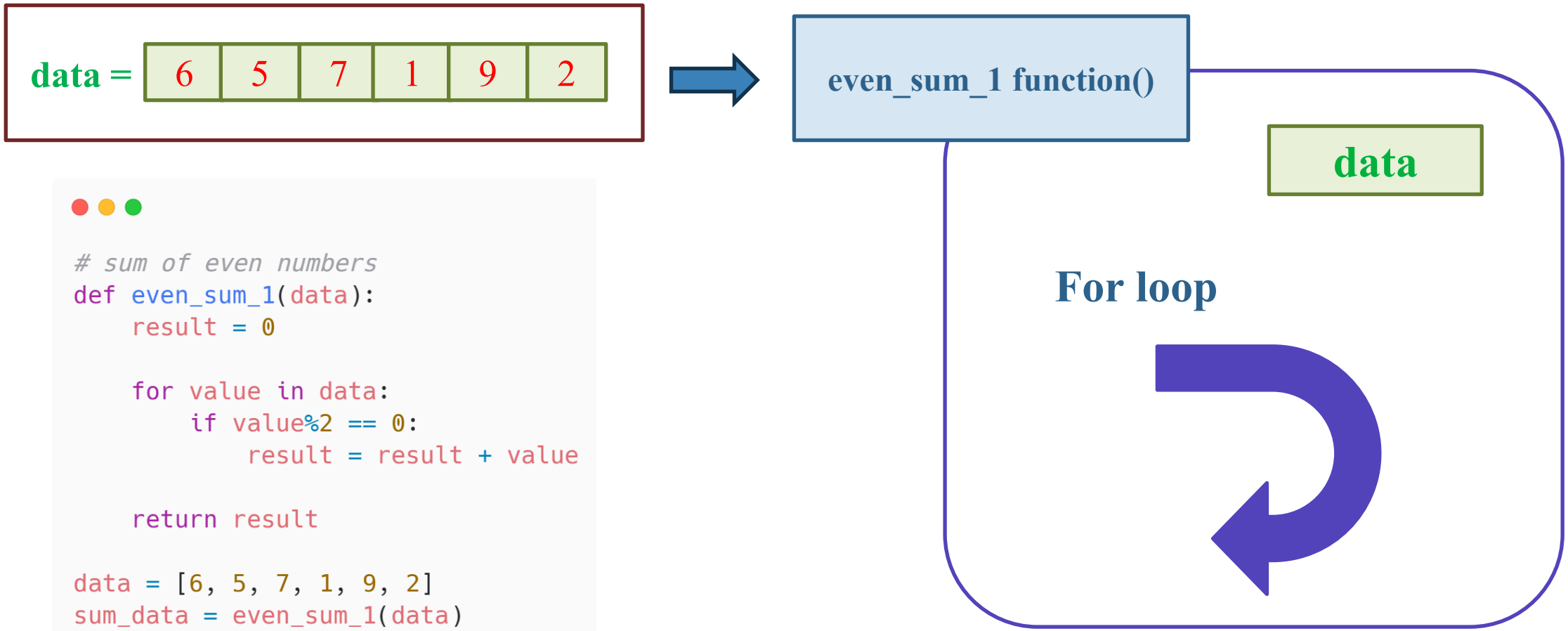
❖ Sum of even numbers




```python
# sum of even numbers
def even_sum_1(data):
    result = 0

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

    return result

data = [6, 5, 7, 1, 9, 2]
sum_data = even_sum_1(data)
print(sum_data)
```

Practice

❖ Sum of even numbers




```python
# sum of even numbers
def even_sum_1(data):
    result = 0

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

    return result

data = [6, 5, 7, 1, 9, 2]
sum_data = even_sum_1(data)
print(sum_data)
```

Practice

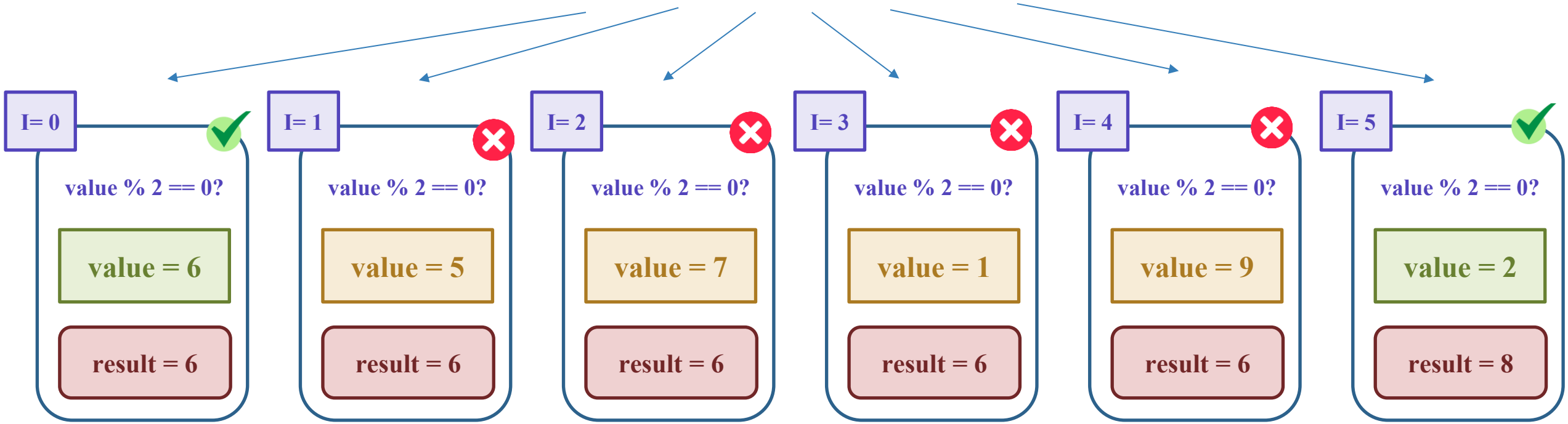
❖ Sum of even numbers

data = | 6 | 5 | 7 | 1 | 9 | 2 |

result = 0

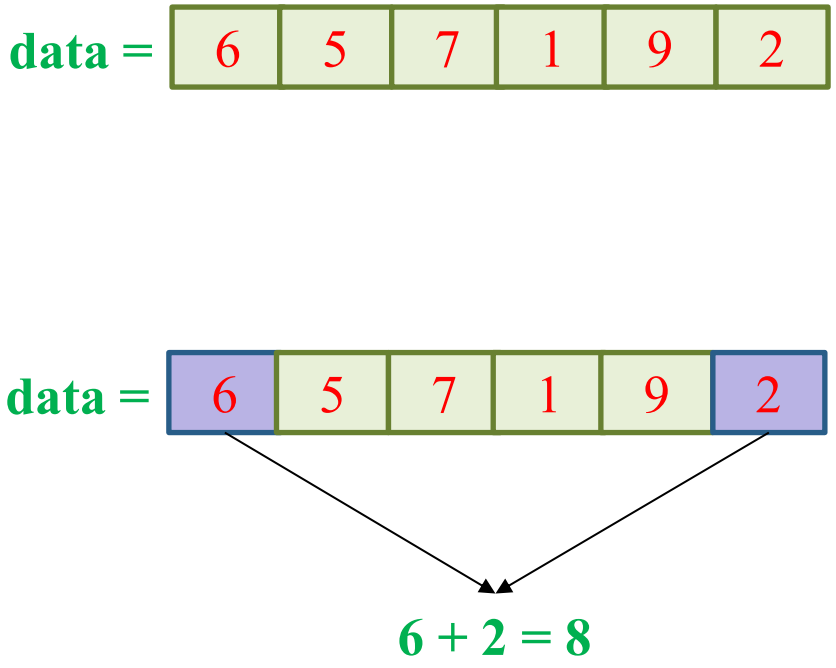
```
for value in data:
    if value%2 == 0:
        result = result + value
```

For Loop



Practice

❖ Sum of even numbers




```python
def even_sum_2(data):
    index = 0
    result = 0

    while index < len(data):
        if data[index]%2 == 0:
            result = result + data[index]
        index = index + 1

    return result

data = [6, 5, 7, 1, 9, 2]
sum_data = even_sum_2(data)
print(sum_data)
```

Practice

❖ Sum of even numbers

data = | 6 | 5 | 7 | 1 | 9 | 2 |

index = 0

result = 0

```
while index < len(data):
    if data[index]%2 == 0:
        result = result + data[index]
    index = index + 1
```

While Loop

Problem?

I= 0	I= 1	I= 2	I= 3	I= 4	I= 5
index<len(data)? ✓	index<len(data)? ✓	index<len(data)? ✓	index<len(data)? ✓	index<len(data)? ✓	index<len(data)? ✓
value % 2 == 0? ✓	value % 2 == 0? ✗	value % 2 == 0? ✗	value % 2 == 0? ✗	value % 2 == 0? ✗	value % 2 == 0? ✓
Value = 6	Value = 5	Value = 7	Value = 1	Value = 9	Value = 2
result = 6	result = 6	result = 6	result = 6	result = 6	result = 8

Practice

❖ Sum of even numbers

index = 0

result = 0

```
while index < len(data):
    if data[index]%2 == 0:
        result = result + data[index]
    index = index + 1
```

I= 6

index<len(data)? ✗

I= 0

index<len(data)? ✓

value % 2 == 0? ✓

Value = 6

result = 6

I= 1

index<len(data)? ✓

value % 2 == 0? ✗

Value = 5

result = 6

I= 2

index<len(data)? ✓

value % 2 == 0? ✗

Value = 7

result = 6

I= 3

index<len(data)? ✓

value % 2 == 0? ✗

Value = 1

result = 6

I= 4

index<len(data)? ✓

value % 2 == 0? ✗

Value = 9

result = 6

I= 5

index<len(data)? ✓

value % 2 == 0? ✓

Value = 2

result = 8

Practice

❖ Two sum

➢ Given an array of integers *data* and an integer *target*, return indices of the two numbers such that they add up to *target*

data =

6	5	7	1	9	2

target = 8

=> [2, 3] # [0, 5]

Practice

❖ Two sum

➢ Given an array of integers *data* and an integer *target*, return indices of the two numbers such that they add up to *target*

data =

6	5	7	1	9	2

target = 8

target - num

2	3	1	7	-1	6

Check

2					6

(0, 5)

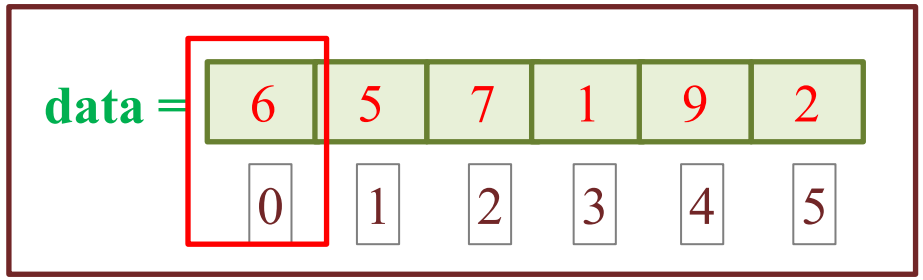
1	7

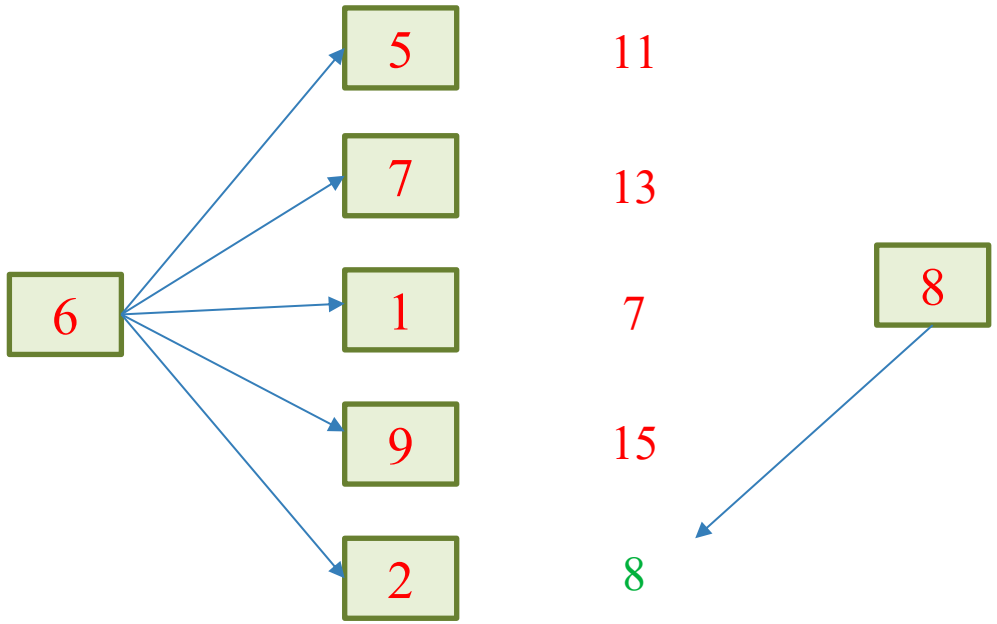
(2, 3)

Practice

❖ Two sum

```
for i in range(n):
    for j in range(i + 1, n):
        if nums[i] + nums[j] == target:
            ans.append([i, j])
```

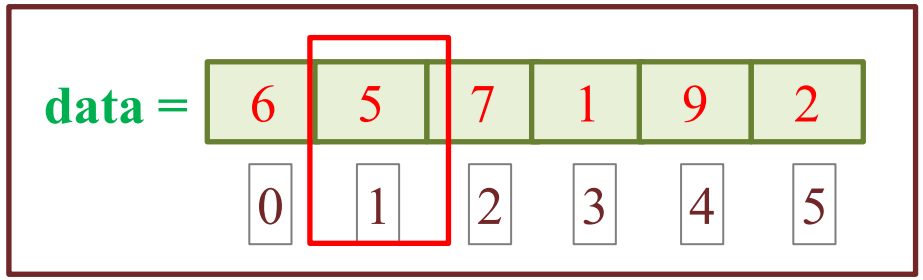


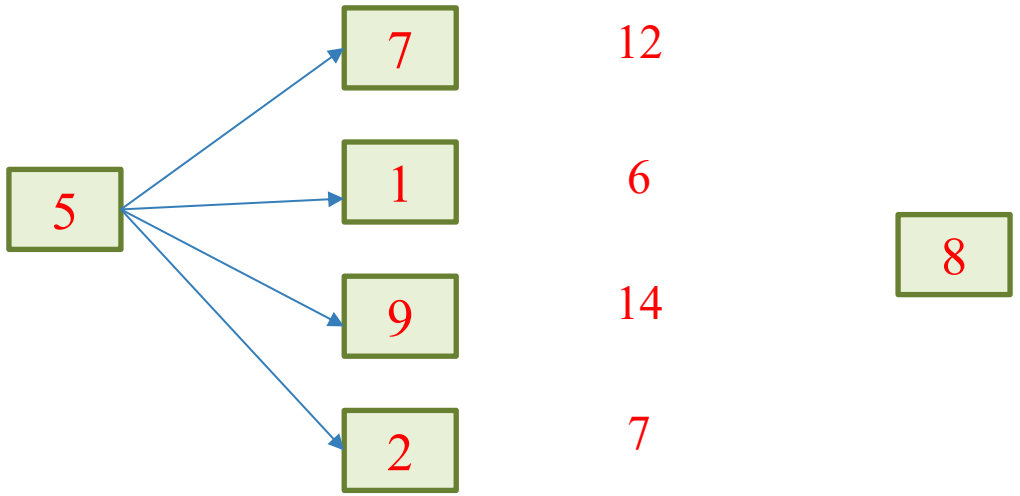


Practice

❖ Two sum

```
for i in range(n):
    for j in range(i + 1, n):
        if nums[i] + nums[j] == target:
            ans.append([i, j])
```





Practice

❖ Two sum

```python
def twoSum(nums, target):
    n = len(nums)
    ans = []
    for i in range(n):
        for j in range(i + 1, n):
            if nums[i] + nums[j] == target:
                ans.append([i, j])
    return ans
```

Practice

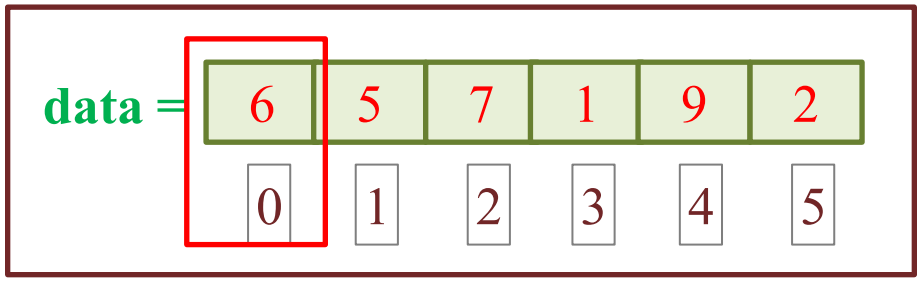
❖ Two sum

```python
def two_sum(data, target):
        num_indices = {}
        ans = []

        for i, num in enumerate(data):
            complement = target - num
            if complement in num_indices:

                ans.append([num_indices[complement], i])

            num_indices[num] = i

        return ans

data = [6, 5, 7, 1, 9, 2]
target = 8
result = two_sum(data, target)
print(result)
```

Practice

❖ Two sum

ans = []

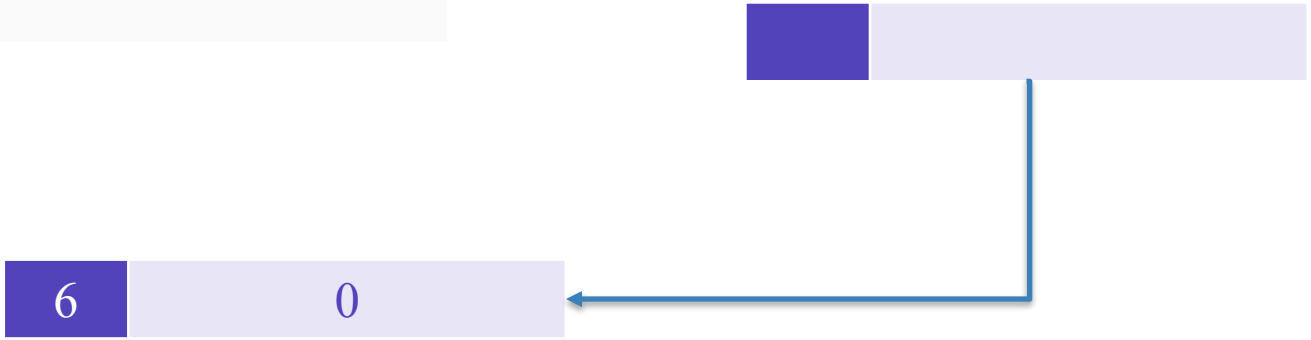



```
for i, num in enumerate(data):
    complement = target - num
    if complement in num_indices:

        ans.append([num_indices[complement], i])

    num_indices[num] = i
```

comp = 2

target = 8

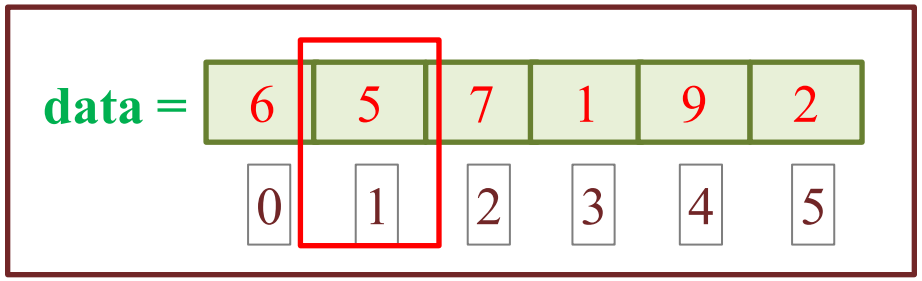
?



Practice

❖ Two sum

ans = []




```
for i, num in enumerate(data):
    complement = target - num
    if complement in num_indices:

        ans.append([num_indices[complement], i])

    num_indices[num] = i
```

comp = 3

target = 8

?

6	0

6	0
5	1

Practice

❖ Two sum

ans = []

data =

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

```
for i, num in enumerate(data):
    complement = target - num
    if complement in num_indices:

        ans.append([num_indices[complement], i])

    num_indices[num] = i
```

comp = 1

target = 8

?

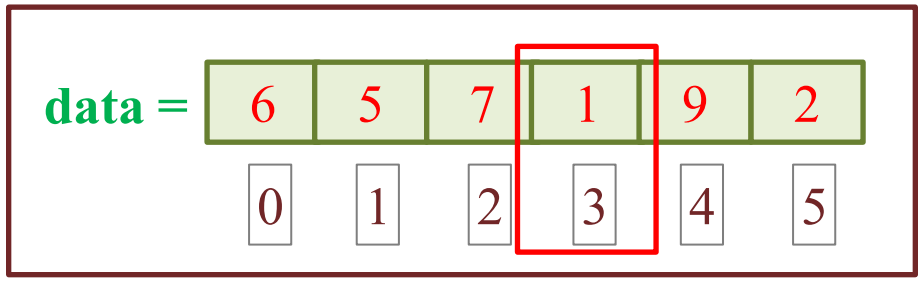
6	0
5	1

6	0
5	1
7	2

Practice

❖ Two sum

ans = []




```
for i, num in enumerate(data):
    complement = target - num
    if complement in num_indices:

        ans.append([num_indices[complement], i])

    num_indices[num] = i
```

comp = 7

target = 8

?

6	0
5	1
7	2

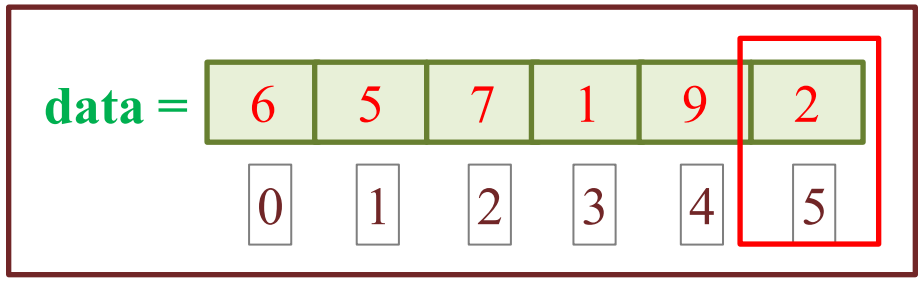
[2, 3]

ans = [[2,3]]

Practice

❖ Two sum

ans = [[2,3]]




```
for i, num in enumerate(data):
    complement = target - num
    if complement in num_indices:

        ans.append([num_indices[complement], i])

    num_indices[num] = i
```

comp = 6

target = 8

?

6	0
5	1
7	2
9	4

[0, 5]

ans = [[2,3], [0, 5]]

Summary

List

- Create: nums = [1, 2, 3]
- Index: nums[0] => 1
- Slicing: nums[:2] => [1, 2]
- Add an element: nums.append(3)
- Update: nums[0] = 2
- Delete: nums.remove(3), nums.pop(0)
- Reverse: nums.reverse()
- Count: nums.count(1)
- Copy: new_nums = nums.copy()
- Sort: nums.sort(reverse=True/False)

Built-in Functions

- len(nums)
- min(nums)
- max(nums)
- sum(nums)
- reversed(nums)
- enumerate
- zip

Thanks!

Any questions?