AI VIETNAM
All-in-One Course

Python Review

For, List, and Dictionary

Year 2023

Outline

- For Loop
- List
- Dictionary

For Loop

keyword

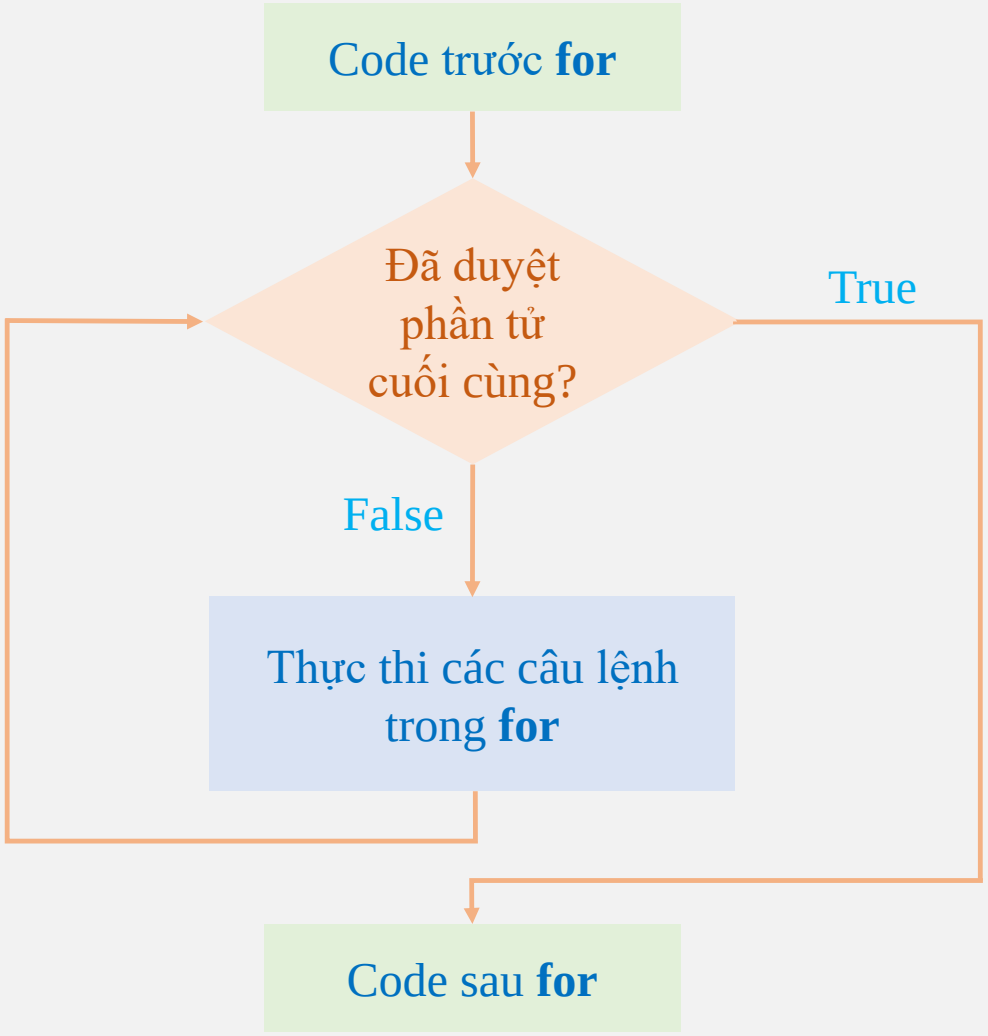
colon

```
# code trước for

for element in iterable:
    # code trong for

# code sau for
```

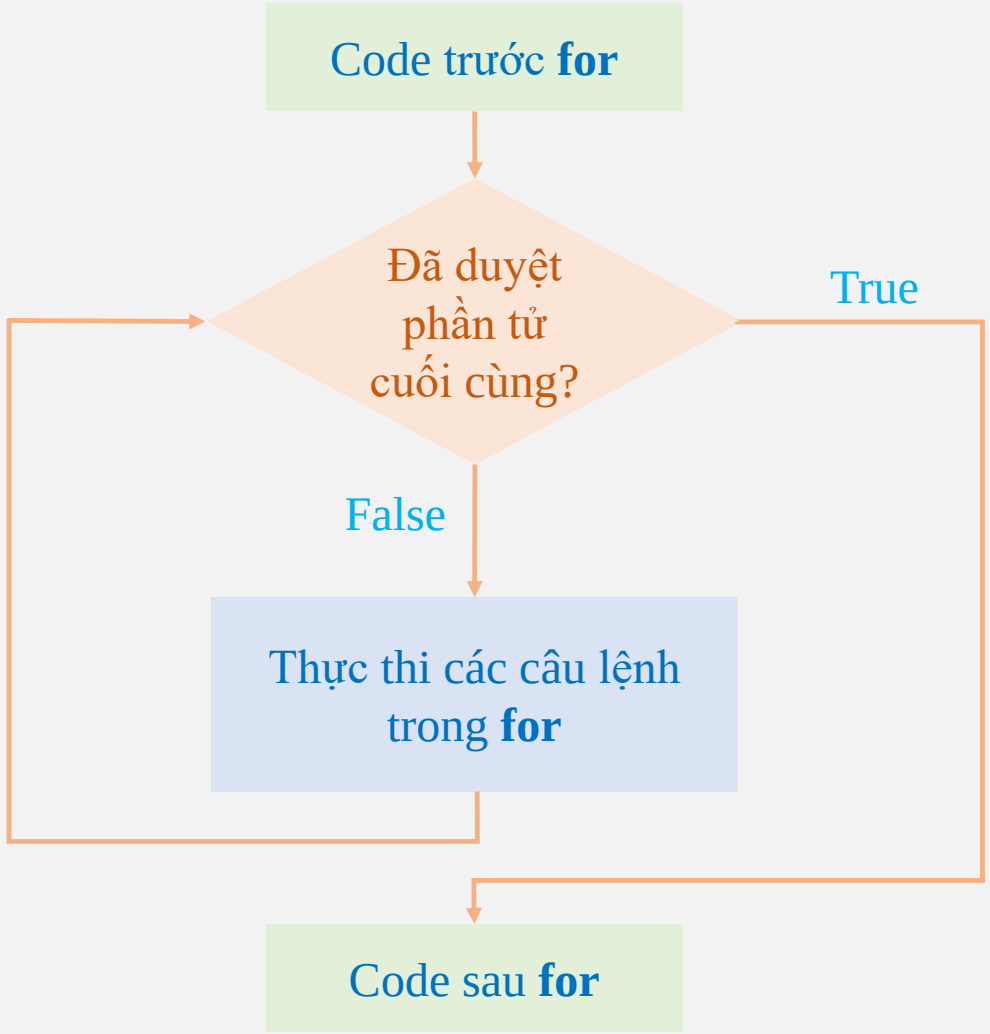
indentation

Iterables

- String
- Tuple
- List
- Dictionary
- range()



For Loop




```python
# iterate a list

fruits = ['apple', 'banana', 'melon', 'peach']

for fruit in fruits:
    print(fruit)
```

```
apple
banana
melon
peach
```

```python
# iterate a string

greeting = 'Hello'

for char in greeting:
    print(char)
```

```
H
e
l
l
o
```

```python
# iterate a dictionary

parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

for key in parameters:
    print(key, parameters.get(key))
```

```
learning_rate 0.1
optimizer Adam
metric Accuracy
```

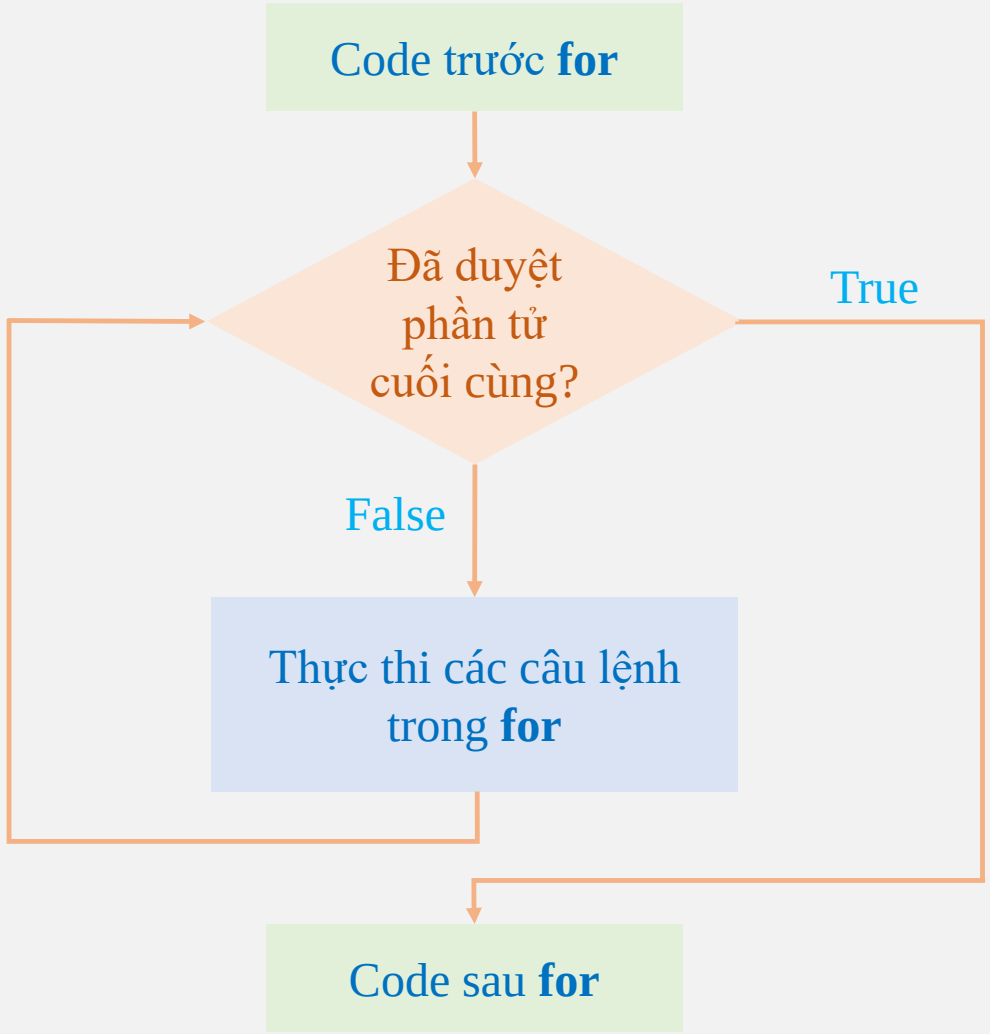
```python
# use range()

for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

```python
# iterate a tuple

fruits = ('apple', 'banana', 'melon')

for fruit in fruits:
    print(fruit)
```

```
apple
banana
melon
```

For Loop



range(start=0, stop, step=1)

range(start=0, stop=5, step=1)

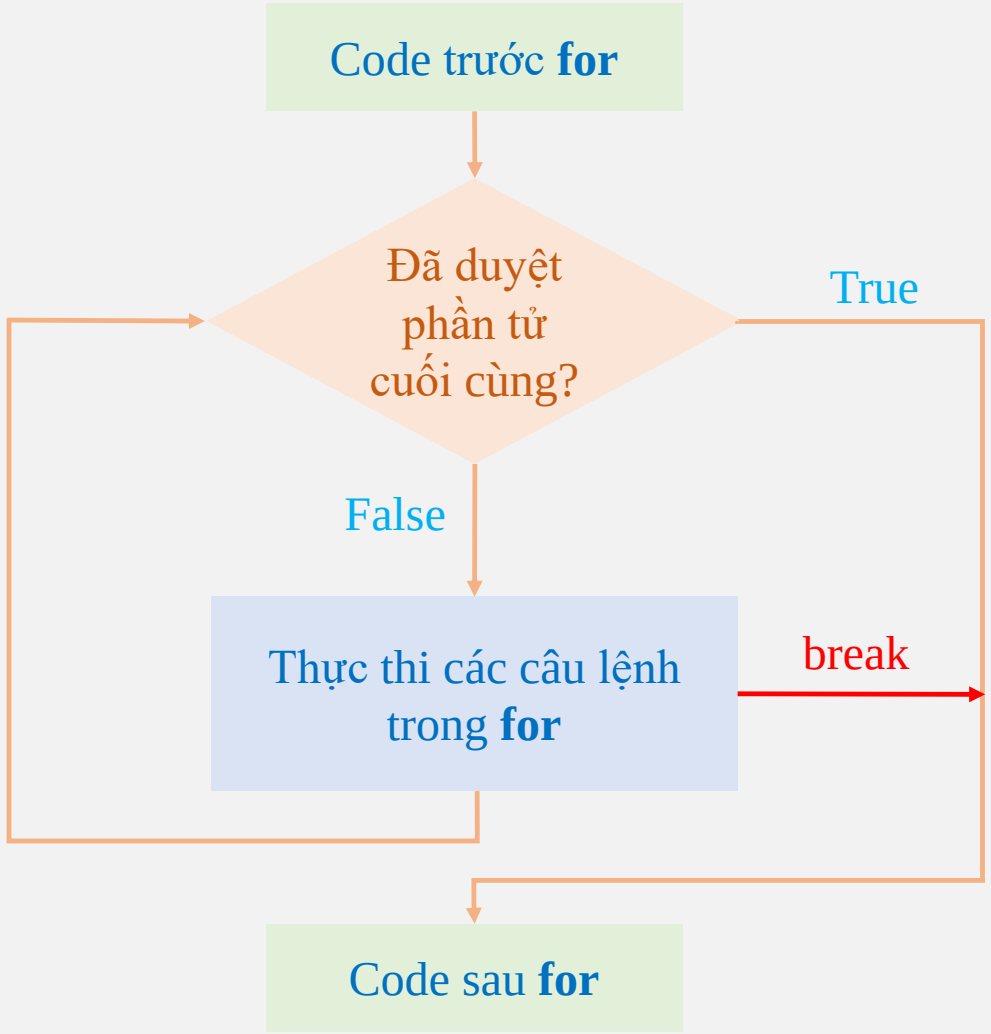
0, 1, 2, 3, 4

range(5)

0, 1, 2, 3, 4

Demo

For Loop



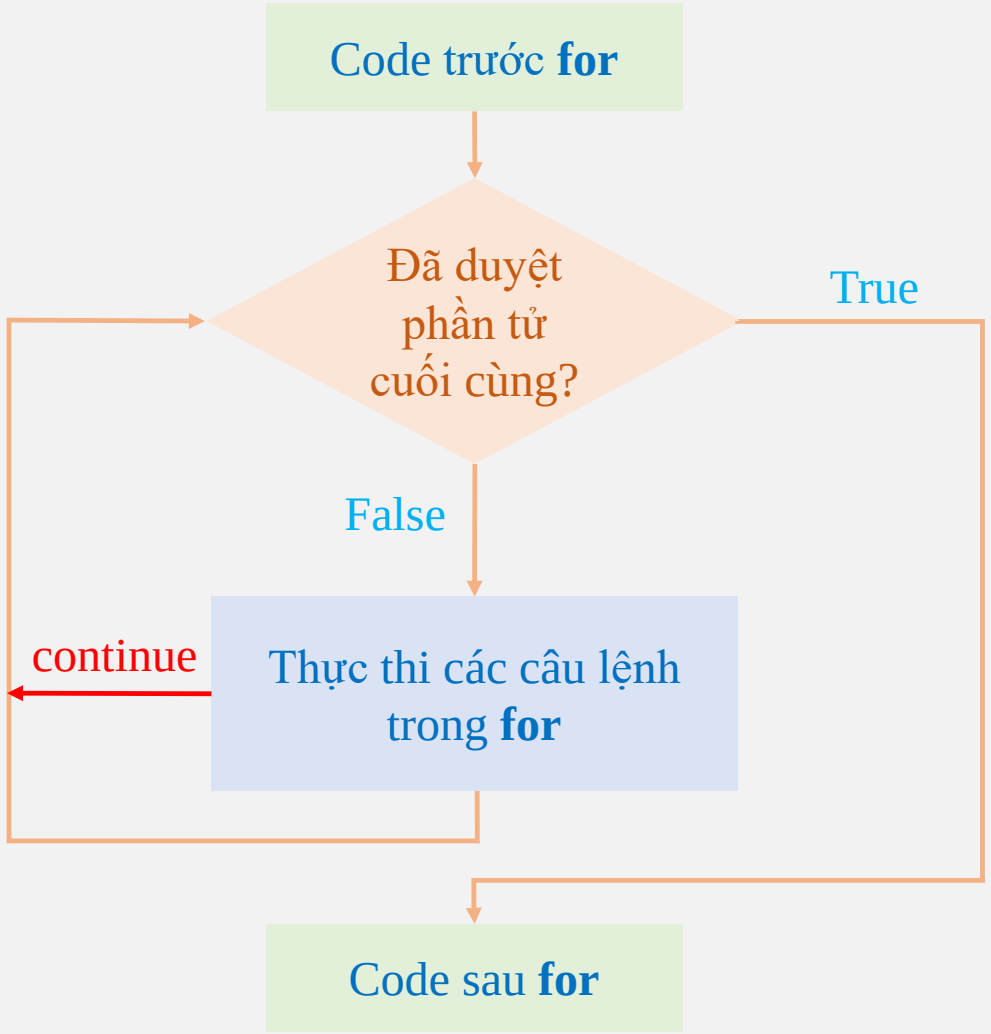
break keyword

```
1  # duyệt phần tử trong range(10)
2  for i in range(10):
3      # hỏi phần tử i có bằng 5 không?
4      if i == 5:
5          # nếu bằng thì thoát vòng lặp for này
6          break
7
8      # làm gì đó với i
9      print('Giá trị i là', i)
```

```
Giá trị i là 0
Giá trị i là 1
Giá trị i là 2
Giá trị i là 3
Giá trị i là 4
```

Demo

For Loop



continue keyword

```
1.  # duyệt phần tử trong range(10)
2.  for i in range(10):
3.      # hỏi phần tử i có bằng 5 không?
4.      if i == 5:
5.          # nếu bằng thì gọi continue
6.          # phần code sau continue sẽ không
7.          # được thực thi trong lần lặp này
8.          continue
9.
10.     # làm gì đó với i
11.     print('Giá trị i là', i)
```

```
Giá trị i là 0
Giá trị i là 1
Giá trị i là 2
Giá trị i là 3
Giá trị i là 4
Giá trị i là 6
Giá trị i là 7
Giá trị i là 8
Giá trị i là 9
```

Demo

Example

❖ PI estimation

Gregory-Leibniz Series

$$PI \approx 4 \sum_{i=1}^{n} \frac{(-1)^{i+1}}{2i-1}$$

```
# Gregory-Leibniz Series

n = 1000
PI = 0
for i in range(1, n):
    PI = PI + (-1)**(i+1) / (2*i - 1)
PI = PI*4

print('Estimated PI is ', PI)
```

```
Estimated PI is  3.142593654340044
```

Nilakantha Series

$$PI \approx 3 + 4 \sum_{i=0}^{n} \frac{-1^{i}}{(2i+2)(2i+3)(2i+4)}$$

```
# Nilakantha Series

n = 1000
PI = 0
for i in range(n):
    PI = PI + (-1)**(i) / ((2*i+2)*(2*i+3)*(2*i+4))
PI = 3 + 4*PI

print('Estimated PI is ', PI)
```

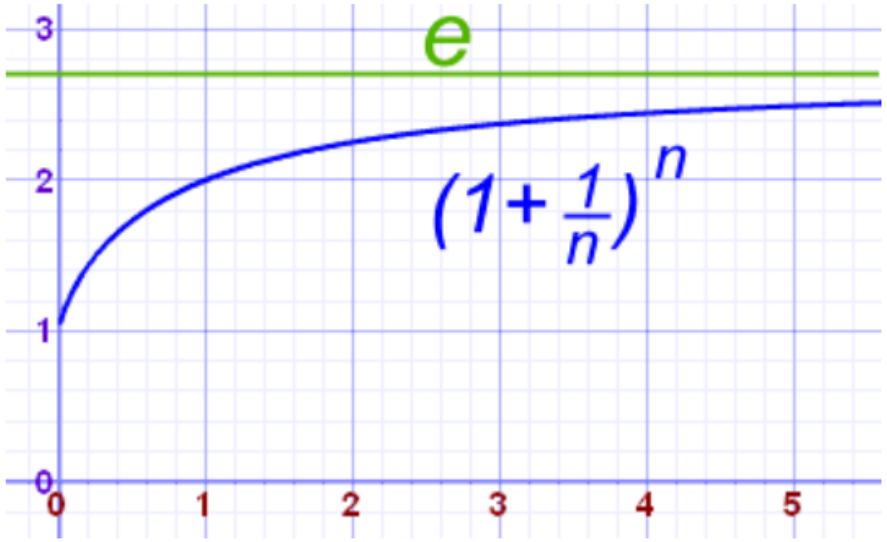
```
Estimated PI is  3.1415926533405423
```

Example

❖ Euler's number

$$e \approx \left(1 + \frac{1}{n}\right)^n$$

$$\lim_{n\to\infty}\left(1 + \frac{1}{n}\right)^n = e$$



n	$(1 + 1/n)^n$
1	2.00000
2	2.25000
5	2.48832
10	2.59374
100	2.70481
1,000	2.71692
10,000	2.71815
100,000	2.71827

https://www.mathsisfun.com/numbers/e-eulers-number.html

Example

❖ **Euler's number**

$$e = 2.71828$$

Formula

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}$$

1) Compute factorial

2) Compute sum

Example

❖ Euler's number

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \ldots + \frac{1}{n!}$$

```python
# aivietnam.ai

# hàm tính giai thừa
def factorial(n):
    result = 1

    for i in range(2, n+1):
        result = result*i

    return result

# hàm ước lượng số e
def estimate_e(n):
    result = 1

    for i in range(1, n+1):
        result = result + 1/factorial(i)

    return result

# ước lượng số e với n = 10
print(estimate_e(10))
```

Example: Quadratic Root

❖ Compute quadratic root for the number N

Newton Method

Set a value for x_0; n = 0
(x_0 = N/2)

$$x_{n+1} = \frac{x_n + \frac{N}{x_n}}{2}$$

n = n + 1

Compute $\sqrt{9}$

$$N = 9$$
$$set\ x_0 = \frac{9}{2} = 4.5$$
$$n = 0$$

$n = 0$

$$x_1 = \frac{x_0 + \frac{N}{x_0}}{2} = \frac{4.5 + \frac{9}{4.5}}{2} = \frac{6.5}{2} = 3.25$$

$n = 1$

$$x_2 = \frac{x_1 + \frac{N}{x_1}}{2} = \frac{3.25 + \frac{9}{3.25}}{2} = \frac{6.019}{2} = 3.009$$

$n = 2$

$$x_3 = \frac{x_2 + \frac{N}{x_2}}{2} = \frac{3.009 + \frac{9}{3.009}}{2} = 3.00001$$

Example: Quadratic Root

❖ Compute quadratic root for the number N

Newton Method

Set a value for x_0; n = 0
(x_0 = N/2)

$$x_{n+1} = \frac{x_n + \frac{N}{x_n}}{2}$$

n = n + 1

```python
def compute_square_root(N, num_loops):
    '''
    This function aims to compute square root for the number N

    N -- the number needs to take the square root
    num_loops -- number of loops used for this optimization
    '''

    x_n = N/2.0

    for i in range(num_loops):
        x_np1 = (x_n + N/x_n) / 2.0
        x_n = x_np1

    return x_np1

print(compute_square_root(N=9, num_loops=10))
print(compute_square_root(N=2, num_loops=10))
```

```
3.0
1.414213562373095
```

Outline

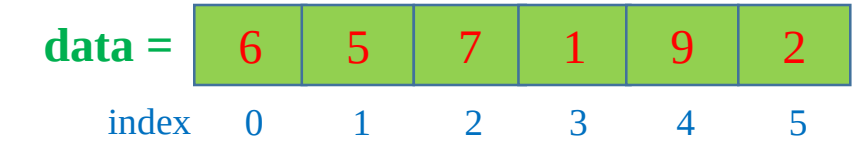
- For Loop
- List
- Dictionary

List

❖ A container that can contain elements

list_name = [element-1, …, element-n]

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

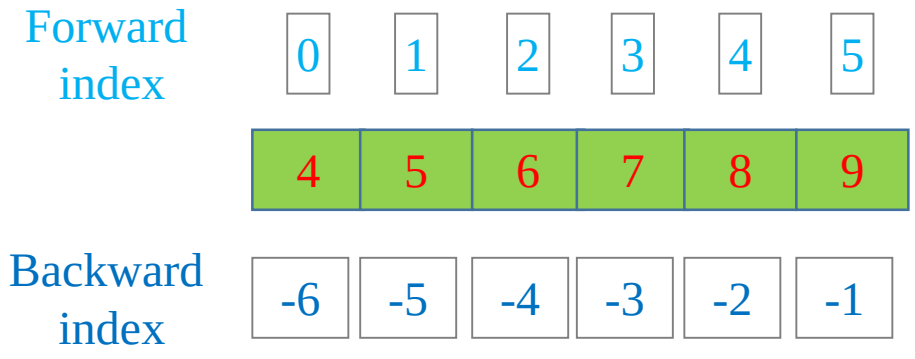



```
1.  # danh sách trống
2.  emty_list = []
3.
4.  # danh sách số tự nhiên nhỏ hơn 10
5.  my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
6.
7.  # danh sách kết hợp nhiều kiểu dữ liệu
8.  mixedList = [True, 5, 'some string', 123.45]
9.  n_list = ["Happy", [2,0,1,5]]
10.
11. #danh sách các loại hoa quả
12. shoppingList = ['táo', 'chuối', 'cherries', 'dâu', 'mận']
```

List

❖ Index

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



data[0] → 4

data[3] → 7

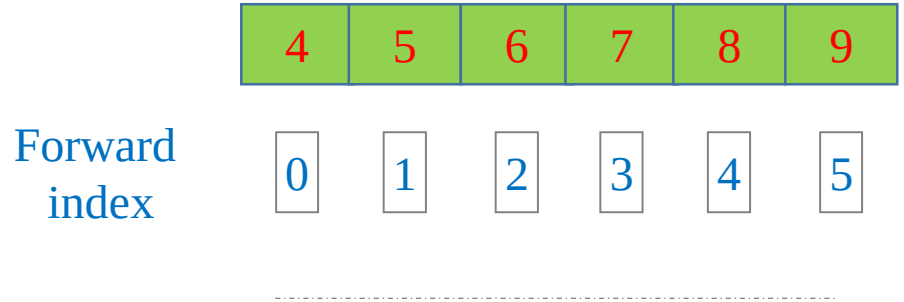
data[-1] → 9

data[-3] → 7

❖ Slicing

list[start:end:step]

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



data[:3]

data[2:4]

data[3:]

Giá trị mắc 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

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

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

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

```
1  data1 = [6, 5, 7]
2  data2 = [1, 9, 2]
3
4  # concatenate
5  data = data1 + data2
6  print(data)
```

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

```
1  data = [6, 5]
2
3  # multiply with a number
4  data_m = data*3
5  print(data_m)
```

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

List

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

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

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3  data.sort(reverse = True)
4  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 |

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

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

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

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

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

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

List

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

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

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  data.clear()
5  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

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  indexOf9 = data.index(9)
5  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

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  data.reverse()
5  print(data)
```

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

List

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  numOf7 = data.count(7)
5  print(numOf7)
```

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

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  aCopy = data.copy()
5  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()

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

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

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

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

```
6
```

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

```
1
9
```

Built-in Functions

❖ sorted(aList) – Sắp xếp các phần tử

sorted(iterable, reverse=reverse)

sorted_data = sorted(data)

sorted_data = sorted(data, reverse=True)


```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()

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

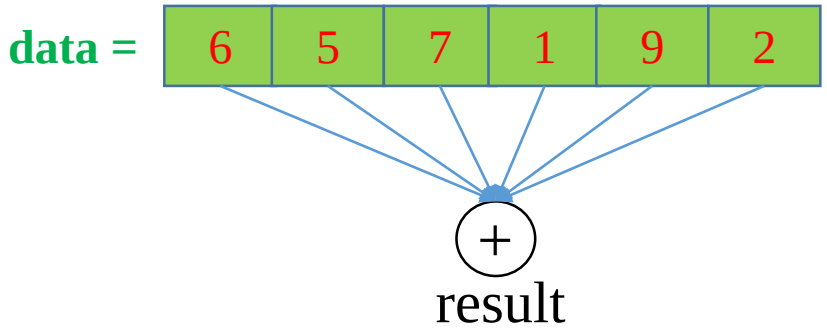
data =	6	5	7	1	9	2

tính tổng

sum(data) = 30

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  summation = sum(data)
5  print(summation)
```

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




```
1  # custom summation - way 1
2  def computeSummation(data):
3      result = 0
4
5      for value in data:
6          result = result + value
7
8      return result
9
10 # test
11 data = [6, 5, 7, 1, 9, 2]
12 summation = computeSummation(data)
13 print(summation)
```

```
30
```

Built-in Functions

sum()

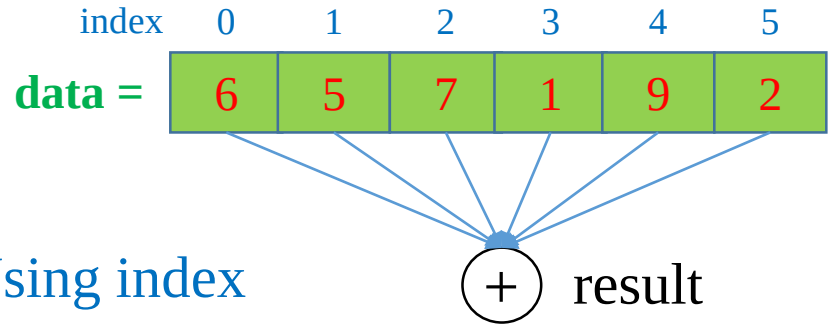
$$summation = \sum_{i=0}^{n} data_i$$

data =	6	5	7	1	9	2

tính tổng
sum(data) = 30

```
1  data = [6, 5, 7, 1, 9, 2]
2  print(data)
3
4  summation = sum(data)
5  print(summation)
```

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



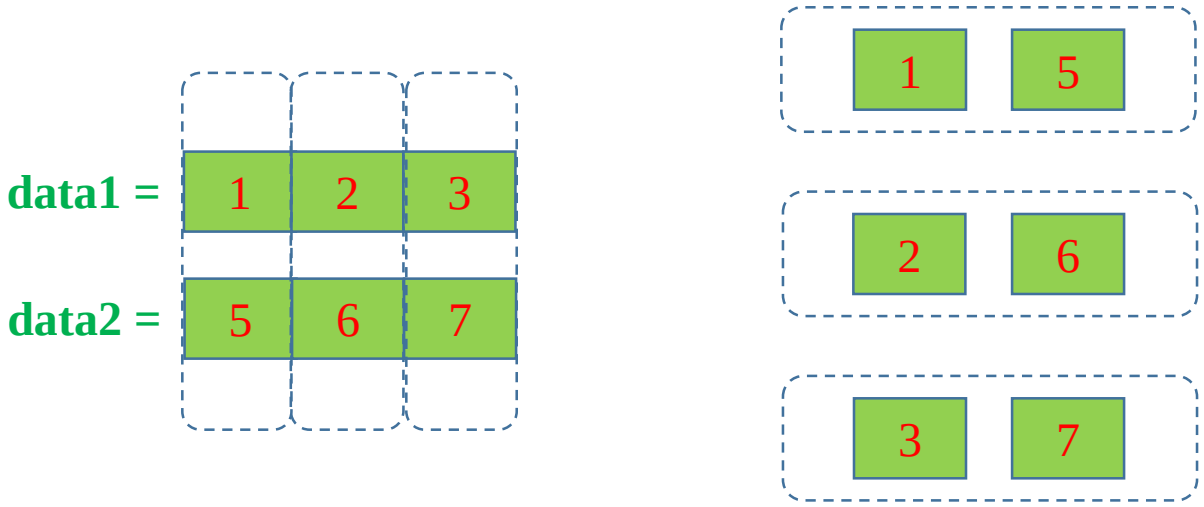
Using index

```
1  # custom summation - way 2
2  def computeSummation(data):
3      result = 0
4
5      length = len(data)
6      for index in range(length):
7          result = result + data[index]
8
9      return result
10
11 # test
12 data = [6, 5, 7, 1, 9, 2]
13 summation = computeSummation(data)
14 print(summation)
```

```
30
```

Built-in Functions

zip()




```
1  l1 = [1, 2, 3]
2  l2 = [5, 6, 7]
3
4  # print in pairs
5  length = len(l1)
6  for i in range(length):
7      print(l1[i], l2[i])
```

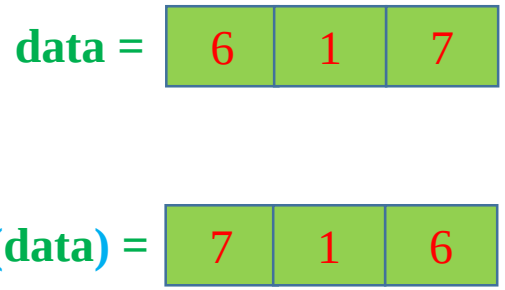
```
1 5
2 6
3 7
```

```
1  l1 = [1, 2, 3]
2  l2 = [5, 6, 7]
3
4  # print in pairs
5  for v1, v2 in zip(l1, l2):
6      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)
```

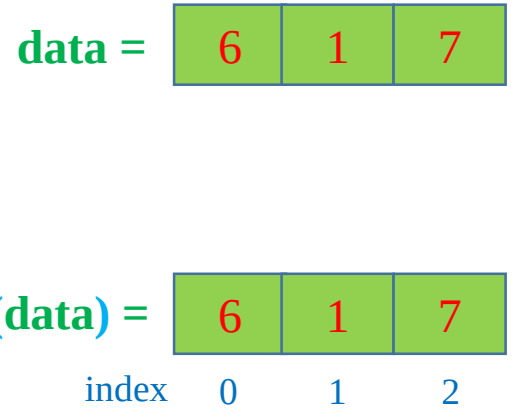
```
6
1
7
```

```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 = 


```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

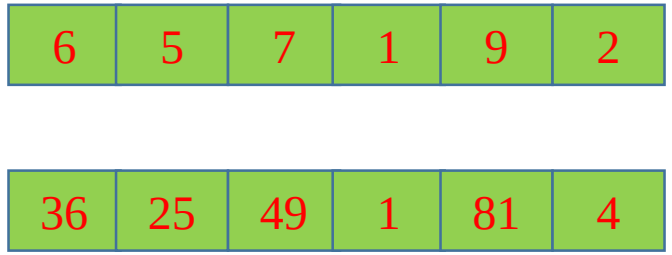
data = 


```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
```

Examples

square(aList)




```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]
```

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]
```

List Comprehension

Sigmoid Function

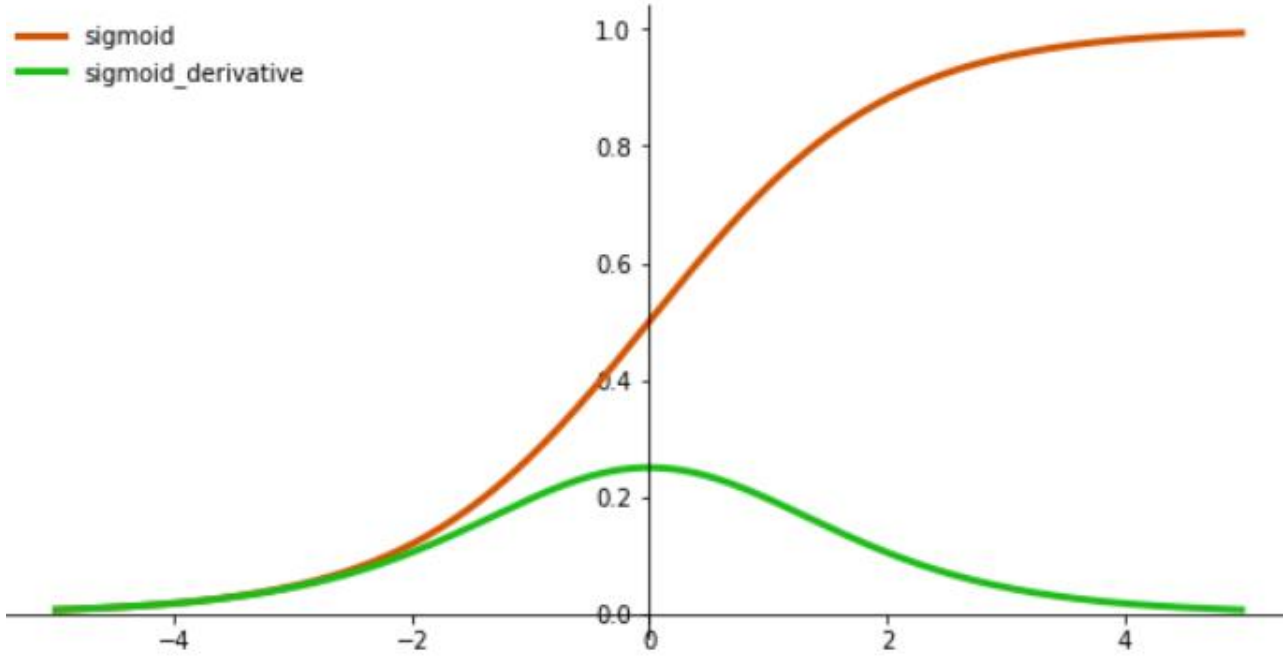
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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

data =	1	5	-4	3	-2

data_a = sigmoid(data)

data_a =	0.731	0.993	0.017	0.95	0.119




```python
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

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

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

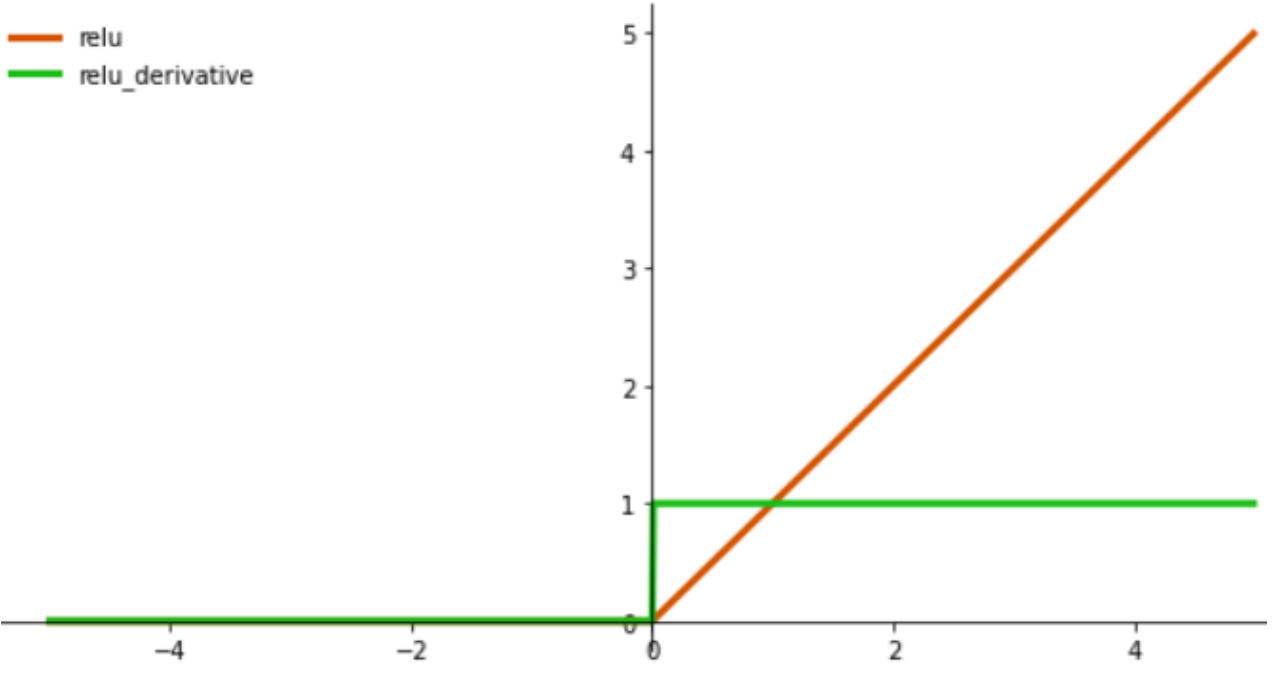
data =

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

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

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

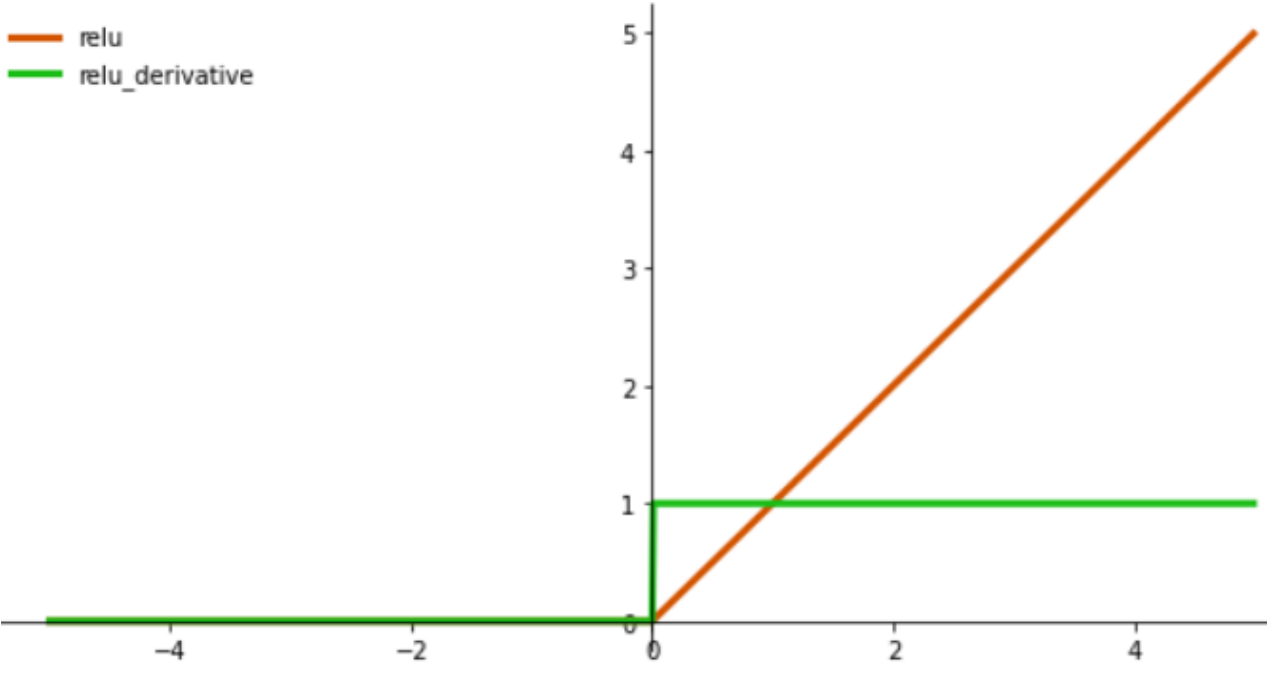
data =

1	5	-4	3	-2

data_a = ReLU(data)

data_a =

1	5	0	3	0



List Comprehension

```
2    result = 0
3    if x > 0:
4        result = x
```

```
1  # relu function
2  def relu(data):
3      result = [x if x>0 else 0 for x in data]
4      return result
5
6  # test
7  data = [1, 5, -4, 3, -2]
8  print(data)
9  data_a = relu(data)
10 print(data_a)
```

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

List Comprehension

[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)
```

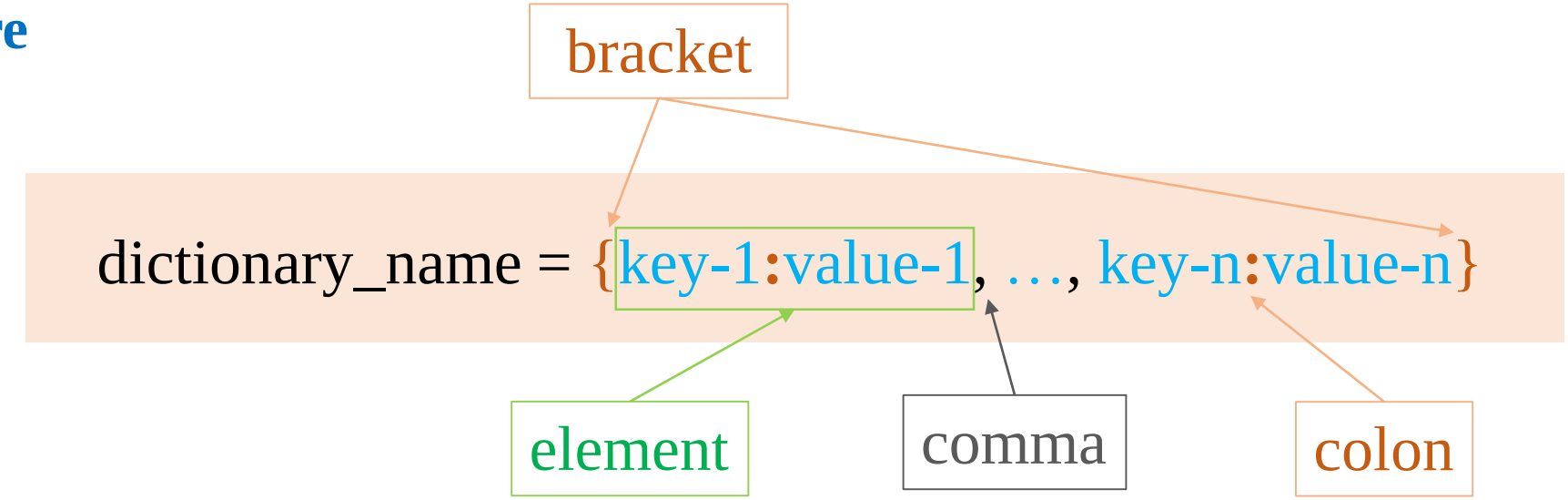
```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

Outline

- For Loop
- List
- Dictionary

Dictionary

❖ Structure



Create a dictionary

```
1  parameters = {'learning_rate': 0.1,
2                'optimizer': 'Adam',
3                'metric': 'Accuracy'}
4
5  print(parameters)
6  print(type(parameters))
```

```
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}
<class 'dict'>
```

Comparing Lists and Dictionaries

Dictionaries are like lists except that they use keys instead of numbers to look up values

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

```
>>> lst = list()
>>> lst.append(21)
>>> lst.append(183)
>>> print(lst)
[21, 183]
>>> lst[0] = 23
>>> print(lst)
[23, 183]
```

```
>>> ddd = dict()
>>> ddd['age'] = 21
>>> ddd['course'] = 182
>>> print(ddd)
{'course': 182, 'age': 21}
>>> ddd['age'] = 23
>>> print(ddd)
{'course': 182, 'age': 23}
```

List

Key	Value
[0]	21
[1]	183

lst

Dictionary

Key	Value
['course']	182
['age']	21

ddd

Dictionary

❖ Create a Dictionary

```python
# dic comprehension

a_dict = {str(i):i for i in range(5)}
print(a_dict)
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4}
```

```python
# from zip

tuple1 = (1, 2, 3)
tuple2 = (4, 5, 6)

a_dict = dict(zip(tuple1, tuple2))
print(type(a_dict))
print(a_dict)
```

```
<class 'dict'>
{1: 4, 2: 5, 3: 6}
```

```python
# from zip

set1 = {1, 2, 3}
set2 = {4, 5, 6}

a_dict = dict(zip(set1, set2))
print(type(a_dict))
print(a_dict)
```

```
<class 'dict'>
{1: 4, 2: 5, 3: 6}
```

```python
# from zip

list1 = [1, 2, 3]
list2 = [4, 5, 6]

a_dict = dict(zip(list1, list2))
print(type(a_dict))
print(a_dict)
```

```
<class 'dict'>
{1: 4, 2: 5, 3: 6}
```

Dictionary

❖ Update a value

```
parameters = {'learning_rate': 0.1,
              'metric': 'Accuracy'}

parameters['learning_rate'] = 0.2
print(parameters)
```

```
{'learning_rate': 0.2, 'metric': 'Accuracy'}
```

❖ Copy a dictionary

```
parameters = {'learning_rate': 0.1,
              'metric': 'Accuracy'}

a_copy = parameters.copy()
a_copy['learning_rate'] = 0.2

print(parameters)
print(a_copy)
```

```
{'learning_rate': 0.1, 'metric': 'Accuracy'}
{'learning_rate': 0.2, 'metric': 'Accuracy'}
```

Dictionary

❖ Hàm copy() chỉ sao chép kiểu shallow

```
d1 = {'a': [1,2], 'b': 5}
d2 = d1.copy()

# thay đổi giá trị d2 sẽ ảnh hưởng đến d1
d2['a'][0] = 3
d2['a'][1] = 4

print('d1:', d1)
print('d2:', d2)
```

```
d1: {'a': [3, 4], 'b': 5}
d2: {'a': [3, 4], 'b': 5}
```

❖ Sử dụng hàm deepcopy() trong module copy

```
import copy

d1 = {'a': [1,2], 'b': 5}
d2 = copy.deepcopy(d1)

# thay đổi giá trị d2
d2['a'][0] = 3
d2['a'][1] = 4

print('d1:', d1)
print('d2:', d2)
```

```
d1: {'a': [1, 2], 'b': 5}
d2: {'a': [3, 4], 'b': 5}
```

Dictionary

❖ Get keys and values

Get keys

```
keys = parameters.keys()
for key in keys:
    print(key)
```

```
learning_rate
optimizer
metric
```

Get values

```
values = parameters.values()
for value in values:
    print(value)
```

```
0.1
Adam
Accuracy
```

Get keys

```
parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

for key in parameters:
    print(key)
```

```
learning_rate
optimizer
metric
```

Get keys and values

```
parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

items = parameters.items()
for key, value in items:
    print(key, value)
```

```
learning_rate 0.1
optimizer Adam
metric Accuracy
```

Dictionary

❖ Get a value by a key

Get value using get() function

```
parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

value = parameters.get('learning_rate')
print(value)

print('\nAfter using get() function')
print(parameters)
```

```
0.1

After using get() function
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}
```

Get value and delete the corresponding item

```
parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

value = parameters.pop('learning_rate')
print(value)

print('\nAfter using pop() function')
print(parameters)
```

```
0.1

After using pop() function
{'optimizer': 'Adam', 'metric': 'Accuracy'}
```

Dictionary

popitem() - lấy ra một phần tử ở cuối dictionary

```
parameters = {'learning_rate': 0.1,
              'optimizer': 'Adam',
              'metric': 'Accuracy'}

item = parameters.popitem()

print(item)
print(parameters)
```

```
('metric', 'Accuracy')
{'learning_rate': 0.1, 'optimizer': 'Adam'}
```

clear() - xóa tất cả các phần tử của một dictionary

```
parameters = {'learning_rate': 0.1,
              'metric': 'Accuracy'}

print('Before using clear() function')
print(parameters)

parameters.clear()

print('\nAfter using clear() function')
print(parameters)
```

```
Before using clear() function
{'learning_rate': 0.1, 'metric': 'Accuracy'}

After using clear() function
{}
```

Use del keyword to delete an item

```
parameters = {'learning_rate': 0.1,
              'metric': 'Accuracy'}
print(parameters)

del parameters['metric']
print(parameters)
```

```
{'learning_rate': 0.1, 'metric': 'Accuracy'}
{'learning_rate': 0.1}
```

Dictionary

❖ Key that does not exist

Try to delete a non-existing item

```
1  parameters = {'learning_rate': 0.1,
2                'metric': 'Accuracy'}
3
4  del parameters['algorithm']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback
<ipython-input-29-e759e1753a28> in <module>
      2                'metric': 'Accuracy'}
      3
----> 4 del parameters['algorithm']

KeyError: 'algorithm'
```

Try to get an item by a non-existing key

```
1  parameters = {'learning_rate': 0.1,
2                'optimizer': 'Adam',
3                'metric': 'Accuracy'}
4
5  value = parameters.pop('algorithm')
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback
<ipython-input-30-8310550c04f4> in <module>
      3                'metric': 'Accuracy'}
      4
----> 5 value = parameters.pop('algorithm')

KeyError: 'algorithm'
```

Dictionary

setdefault() function

```
1  # setdefault()
2
3  fruits = {'banana': 2}
4  fruits.setdefault('apple', 0)
5
6  print(fruits)
```

```
{'banana': 2, 'apple': 0}
```

```
1  # setdefault()
2
3  fruits = {'banana': 2, 'apple': 4}
4  fruits.setdefault('apple', 0)
5
6  print(fruits)
```

```
{'banana': 2, 'apple': 4}
```

example

```
1  # setdefault()
2
3  fruits = {'banana': 2}
4  fruits.setdefault('apple', 0)
5
6  fruits['apple'] += 10
7  print(fruits)
```

```
{'banana': 2, 'apple': 10}
```

Result ???

```
1  # setdefault()
2
3  fruits = {'banana': 2}
4
5  fruits['apple'] += 10
6  print(fruits)
```

Dictionary

❖ Get a value via a key

Method 1

```
1  # access value via key
2
3  fruits = {'banana': 2, 'apple': 4}
4  print(fruits['apple'])
5  print(fruits['corn'])
```

```
4

---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-21-bda9d1f64a8f> in <module>
      3 fruits = {'banana': 2, 'apple': 4}
      4 print(fruits['apple'])
----> 5 print(fruits['corn'])

KeyError: 'corn'
```

Method 2

```
1  # access value via key
2
3  fruits = {'banana': 2, 'apple': 4}
4  print(fruits.get('apple'))
5  print(fruits.get('corn'))
```

```
4
None
```

Dictionary

Merge two dictionaries

```python
# merge two dicts

fruits = {'banana': 2, 'apple': 4}
cereal = {'rice': 3, 'corn': 7}

result = {**fruits, **cereal}
print(result)
```

```
{'banana': 2, 'apple': 4, 'rice': 3, 'corn': 7}
```

Check if a key exists

```python
# check if a key exists

fruits = {'banana': 2, 'apple': 4}

print('apple' in fruits)
print('corn' in fruits)
```

```
True
False
```

Remove empty items

```python
# remove empty items

fruits = {'banana': 2, 'apple': None}

dict1 = {key:value for (key, value)
                    in fruits.items()
                    if value is not None}
print(dict1)
```

```
{'banana': 2}
```

Dictionary comprehension

```python
# dic comprehension

aDict = {str(i):i for i in range(5)}
print(aDict)
```

```
{'0': 0, '1': 1, '2': 2, '3': 3, '4': 4}
```