AI VIETNAM
All-in-One Course

Data Structure

String & List

Nguyen Dinh Vinh
Ph.D. in Computer Science

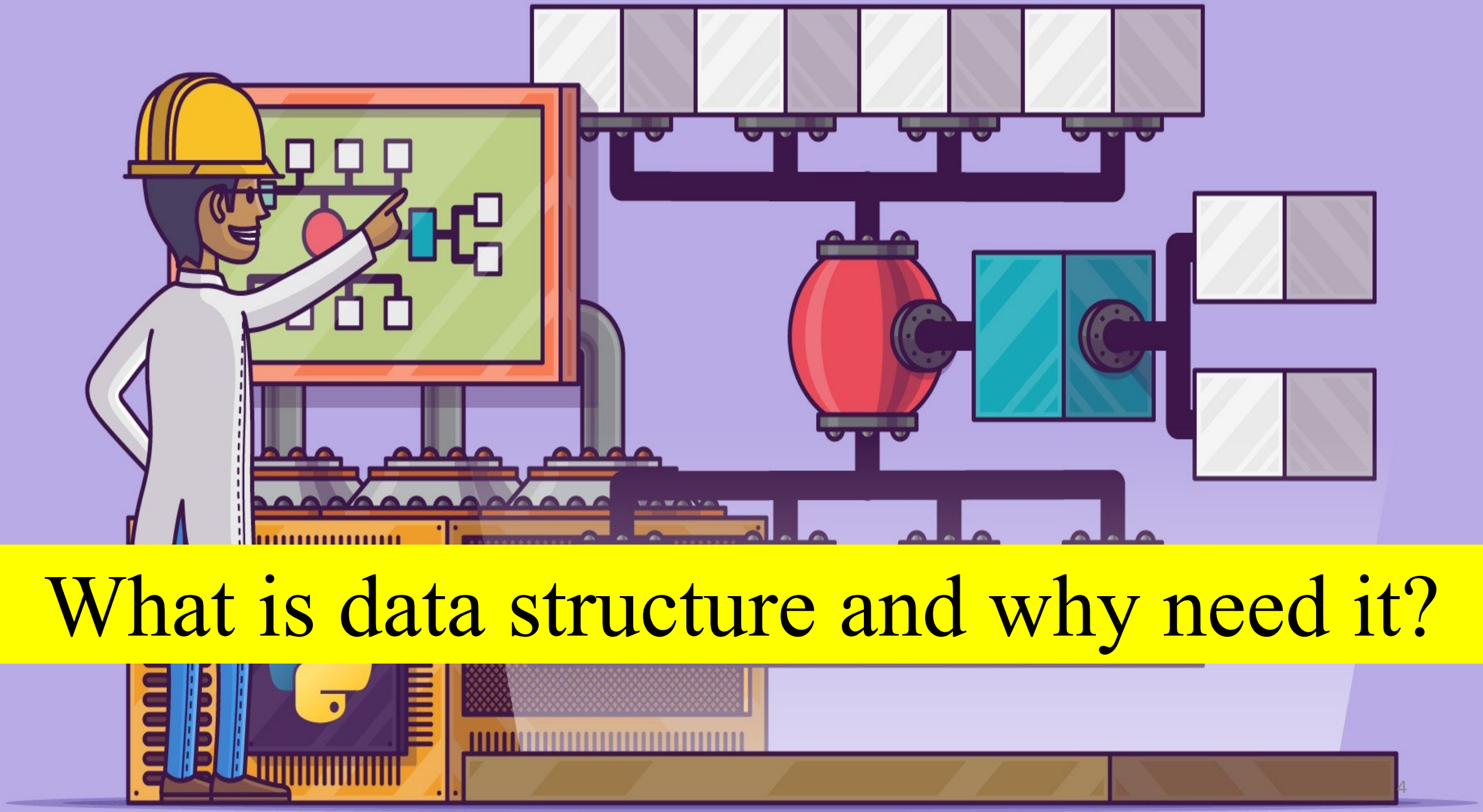
Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses and variable
- Summarize

Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

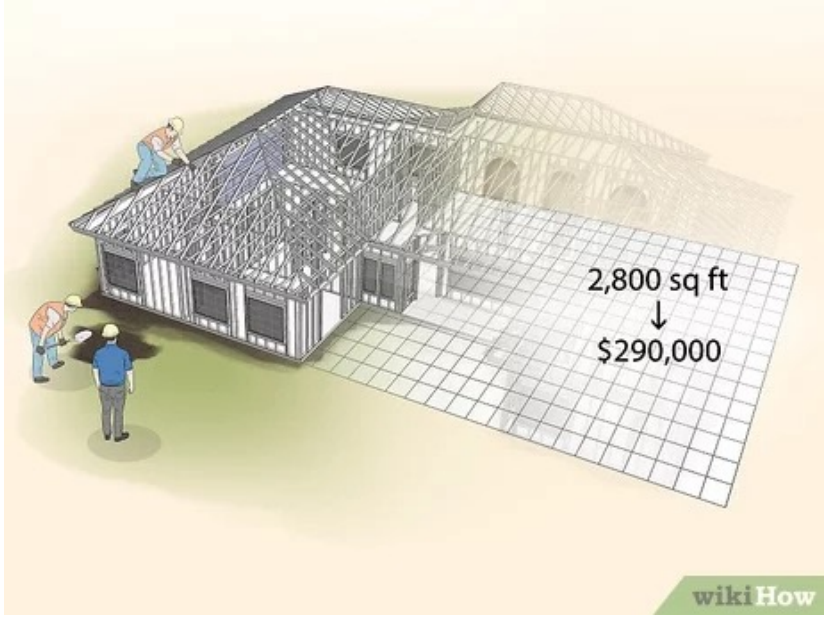
Credit by: https://realpython.com/python-data-structures/

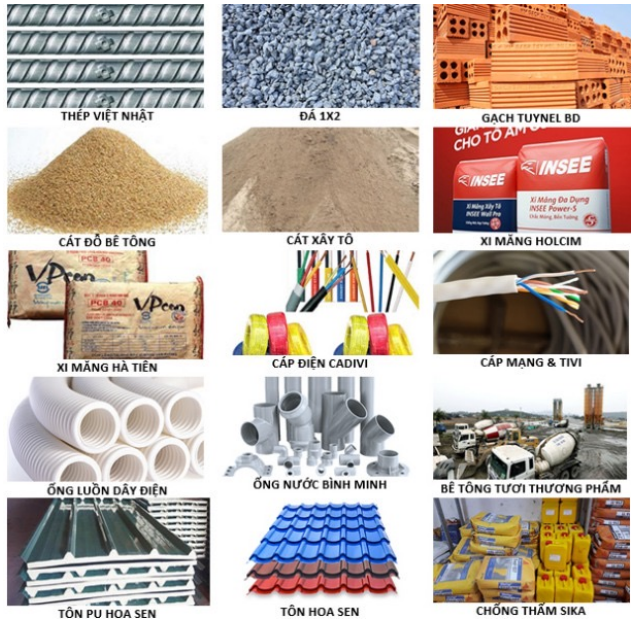


What is data structure and why need it?

Motivation

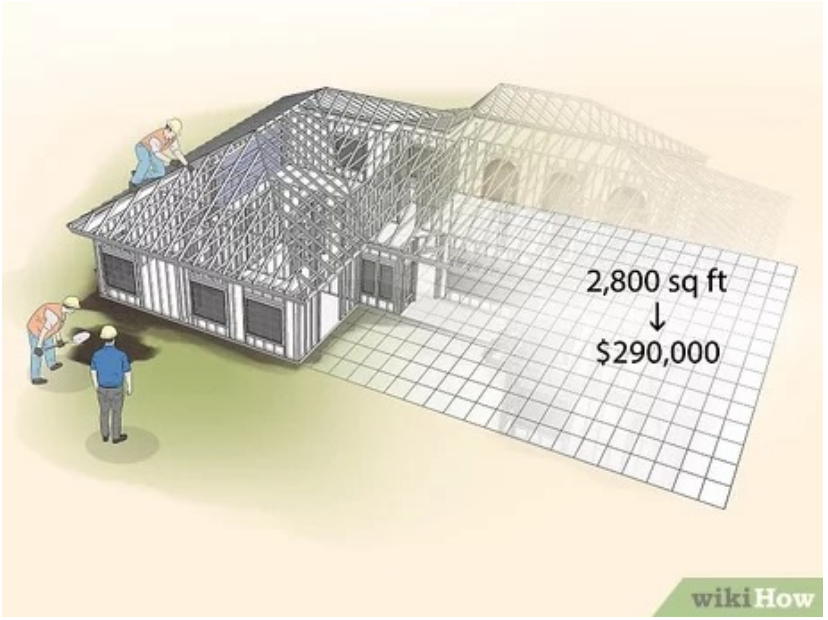
How to build a house





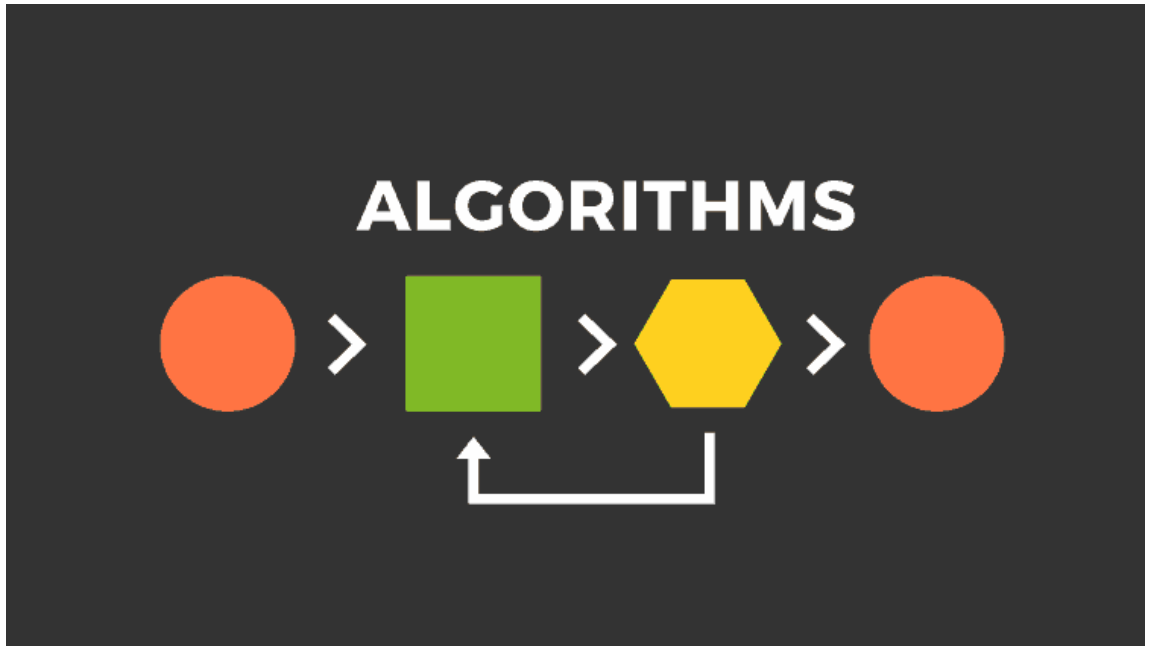
Building blocks & raw materials

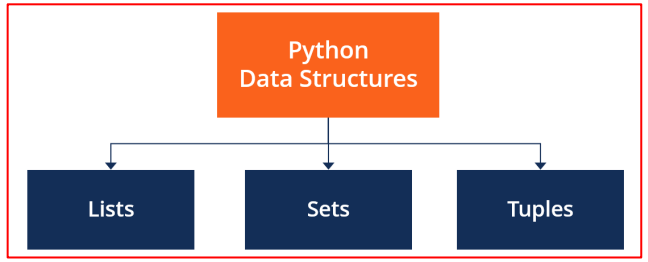
Home Building Instructions



Motivation

How to develop a Software/Algorithm





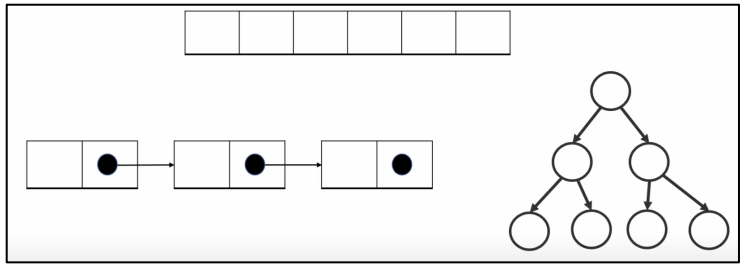
Building blocks & data structure

Code
Instructions

Data structure are building blocks or raw material for any software programs

Understand data structure → Good programmer

We need to use right data structure for a problem



Example

Problem: Store how Much You Spends for 3 days

```
1   day1 = 10 #$
2   day2 = 20 #$
3   day3 = 30 #$
```

Problem: Store how Much You Spends for 100 days

```
1   day1 = 10
2   day2 = 20
3   day3 = 30
4   day4 = 40
5   ...
6   day100 = 100
```

100 variables

Need new data structure to solve this isue

Limitations

Example

Problem: Store how Much You Spends for 3 days

main.py

```
1  spend_list = [10, 20, 30, 40, 50,..., 100]
```

Problem: Store How Much You Spends from April 4 to April 7

```
1  # Use data structure: Dictionary
2  spends_dict = {
3      'april 4' : 10,
4      'april 5' : 20,
5      'april 6' : 30
6  }
```

Problem: What was the spends on day 1

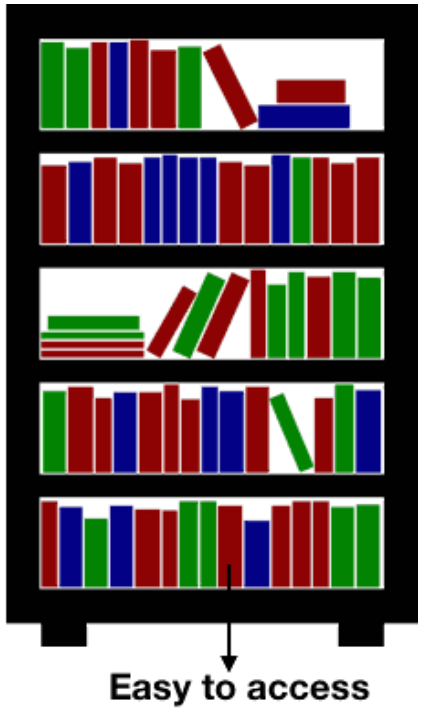
```
7  # Access the 1st element in the List
8  spend_list[0]
```

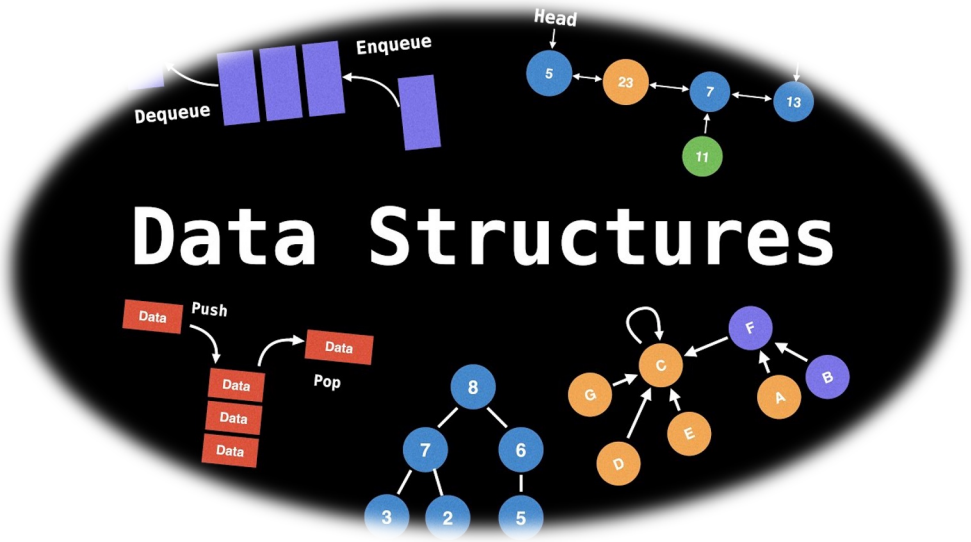
Problem: What was the spends on April 4

```
8  spends_dict['april 4']
```

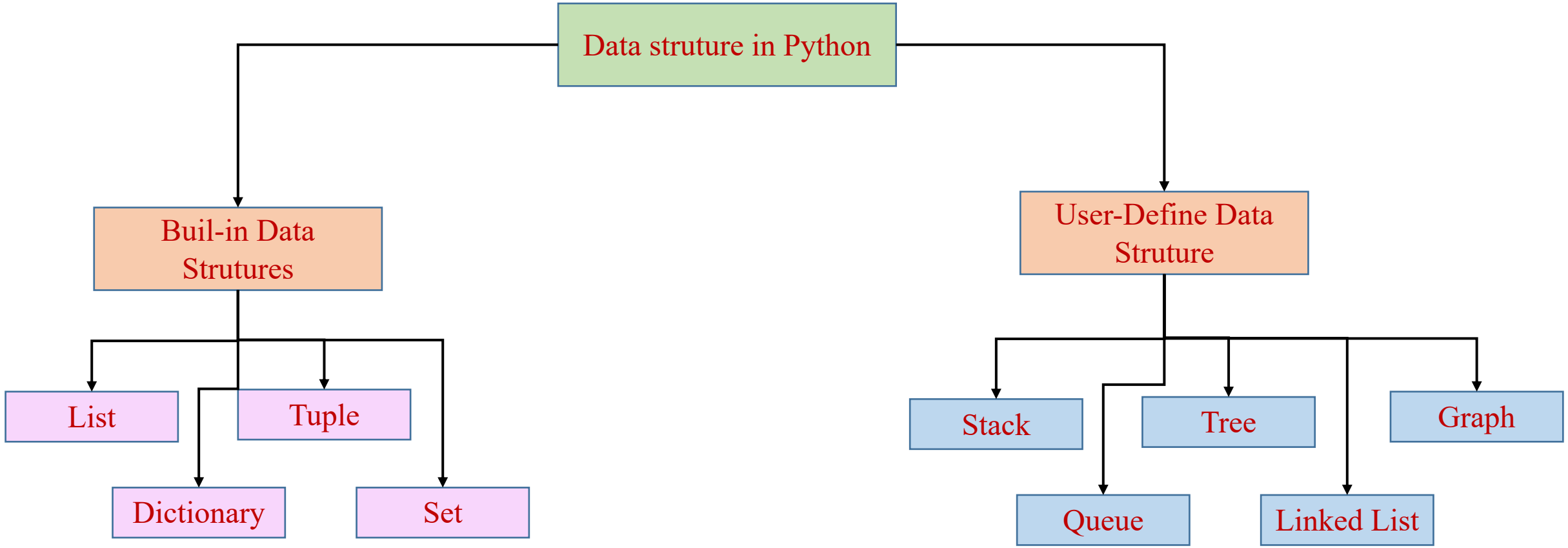
What is a Data Structure?

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.

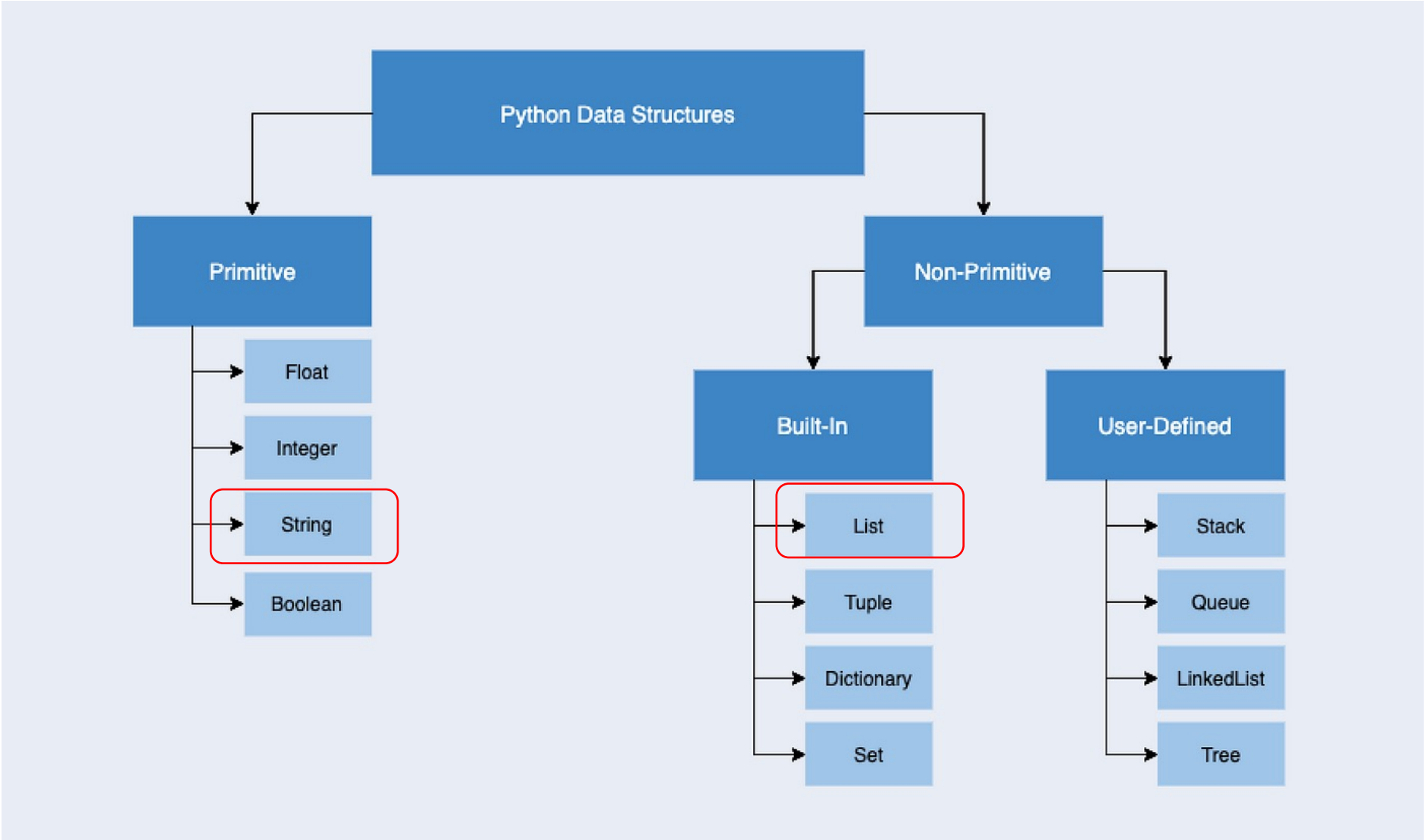




What is a Data Structure?



What is a Data Structure?



https://towardsdatascience.com/

Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

String

- A string is a sequence of characters
- A string literal uses quotes ‘AIVN' or ”AIVN"

Looking Inside Strings

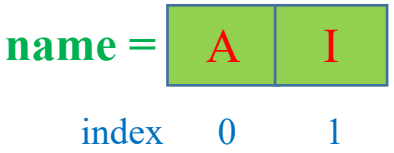
- We can get at any single character in a string using an index specified in square brackets
- The index value must be an integer and starts at zero
- The index value can be an expression that is computed


```
>>> name = ‘AIVINA'
>>> letter = name[1]
>>> print(letter)
I
>>> x = 3
>>> w = name[x - 1]
>>> print(w)
V
```

String

❖ Create and iterate a string

name = 'AI'




```
1  # create a string
2  name = 'AI'
3  print(name)
```

```
AI
```

```
1  # iterate a string
2  name = 'AI'
3  for character in name:
4      print(character)
```

```
A
I
```

```
1  # iterate a string
2  name = 'AI'
3  length = 2
4
5  for index in range(length):
6      print(name[index])
```

```
A
I
```

String

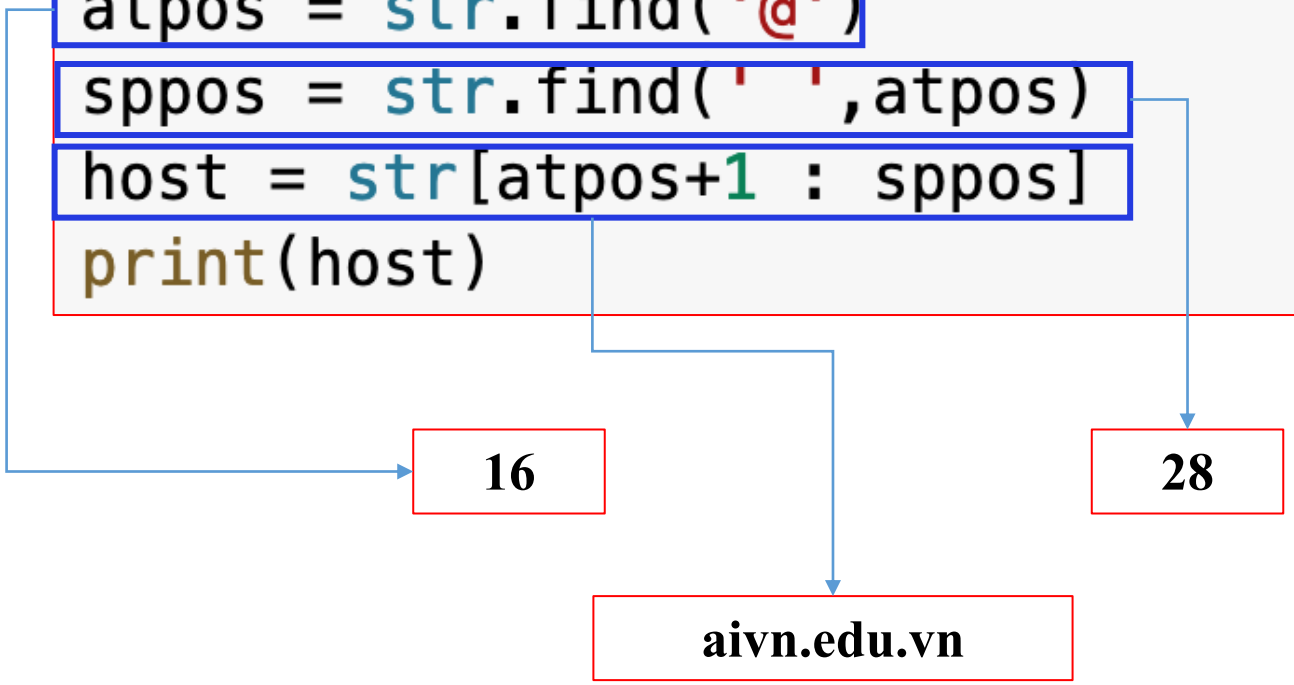
❖ **What is the result?**

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

String

❖ **What is the result?**

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



List Motivation

Suppose, If you want to store a sequence of numbers:
1,2,3,4,5,6,7

Can I do it?

```
num1, num2, num3, num4, num5, num6, num7 = 1,2,3,4,5,6,7
```

How about calculating sum of all numbers?

```
sum = num1 + num2 + num3 + num4 + num5 + num6 + num7
print("sum: ", sum)
```

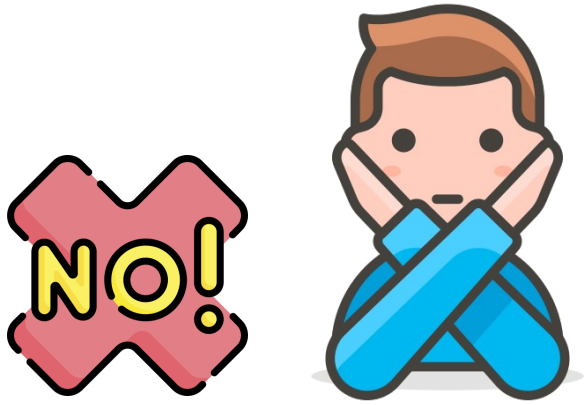
```
sum:  28
```

List Motivation

Suppose, If I want to store a sequence of numbers:
1,2,3,4,5,6,7

Can I do it?

```
numbers = "1234567"
```



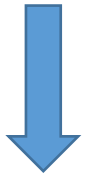
How about calculating sum of all numbers?

```
numbers = "1234567"
sum = 0
for i in numbers:
  sum = sum + i
print("sum: ", sum)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-30-cae26fbcb039> in <cell line: 3>()
      2 sum = 0
      3 for i in numbers:
----> 4   sum = sum + i
      5 print("sum: ", sum)

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

https://colab.research.google.com/drive/1Q3rrT4w24qIm51M5WFQjYNd00lSclwyX?usp=sharing

Are there any more efficent way to store data to solve this issue?



Python List

```
list_number = [1, 2, 3, 4, 5, 6, 7]
sum = 0
for i in list_number:
  sum = sum  + i
print("sum: ", sum)
```

```
sum:  28
```

Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

List

❖ A container that can contain elements

list_name = [element-1, …, element-n]

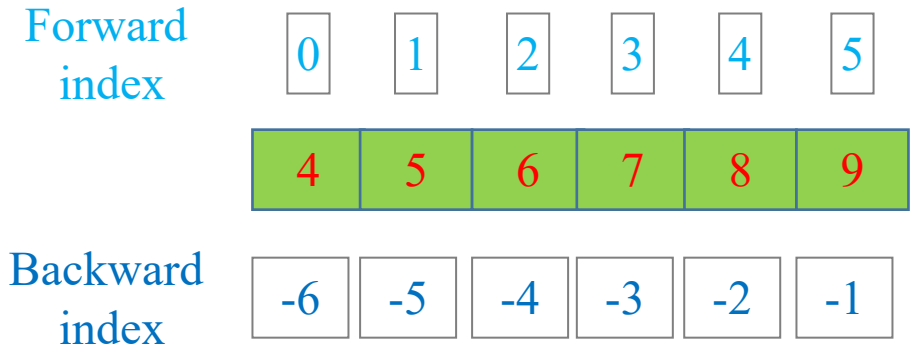
```
// create a list
data = [6, 5, 7, 1, 9, 2]
```


```
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

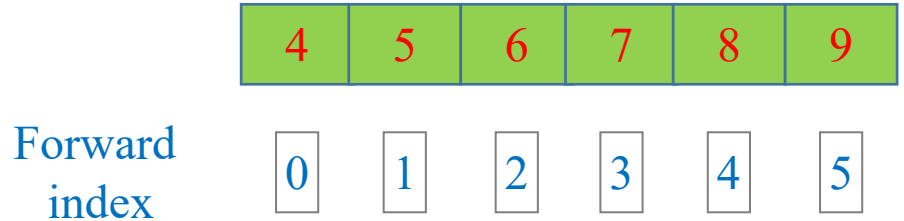
data[-1] → 9

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 |

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

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

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

```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

```
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

xóa phần tử thứ 1 và 2
del data[1:3]

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)

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()

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

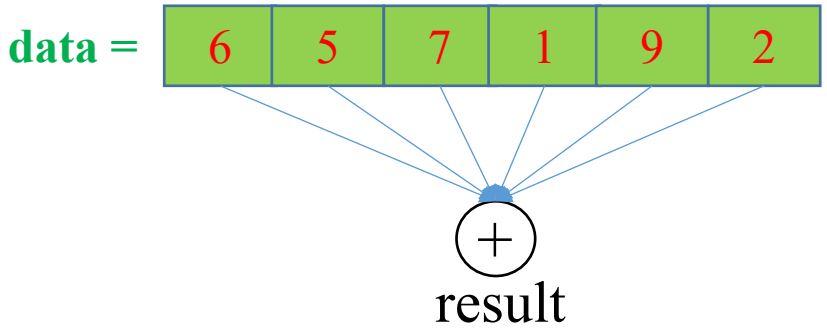
data =	6	5	7	1	9	2

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()

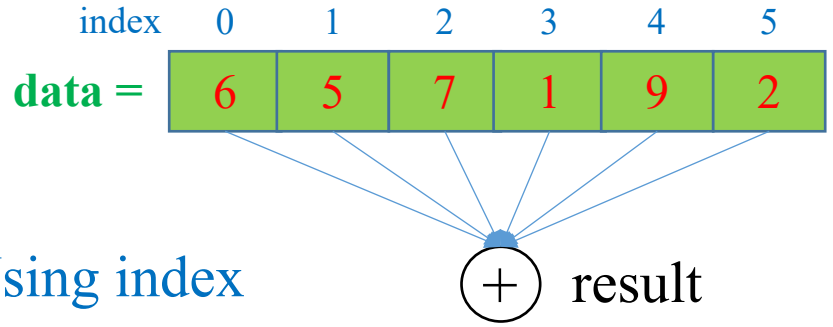
$$summation = \sum_{i=0}^{n} data_i$$

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
```



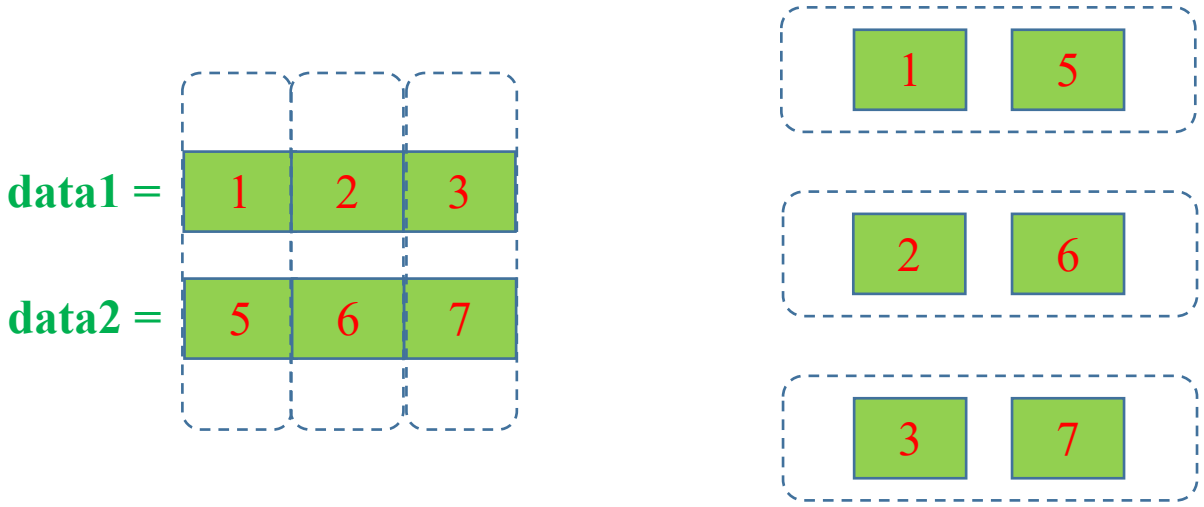
Using index

```
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()



zip() function to perform parallel iterations on multiple iterables

```
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])
```

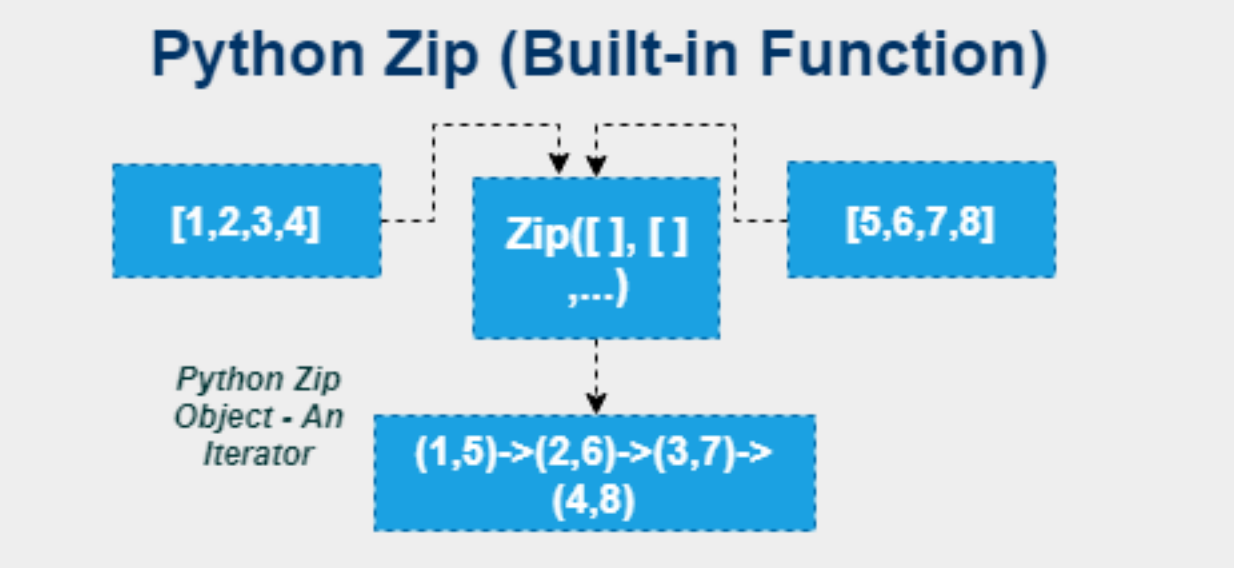
```
1 5
2 6
3 7
```

```
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

zip()



zip() function to perform parallel iterations on multiple iterables

```
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()

data = | 6 | 1 | 7 |

reversed(data) = | 7 | 1 | 6 |

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

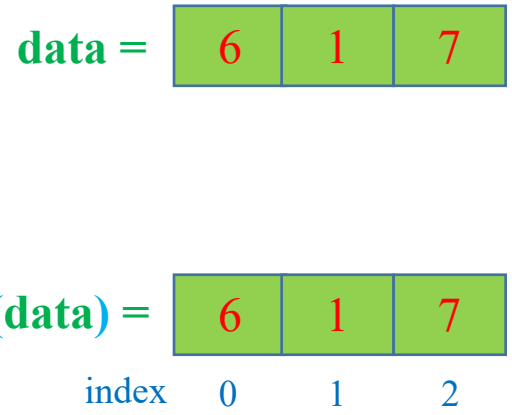
```
6
1
7
```

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

```
7
1
6
```

Built-in Functions

enumerate()




```
1 # get index and value
2 data = [6, 1, 7]
3
4 length = len(data)
5 for index in range(length):
6     print(index, data[index])
```

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

```
1 # enumerate
2 data = [6, 1, 7]
3 for index, value in enumerate(data):
4     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

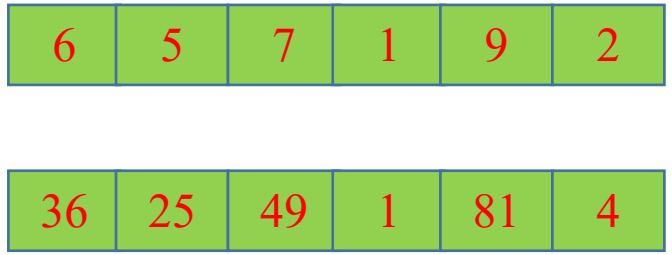
data = 


```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

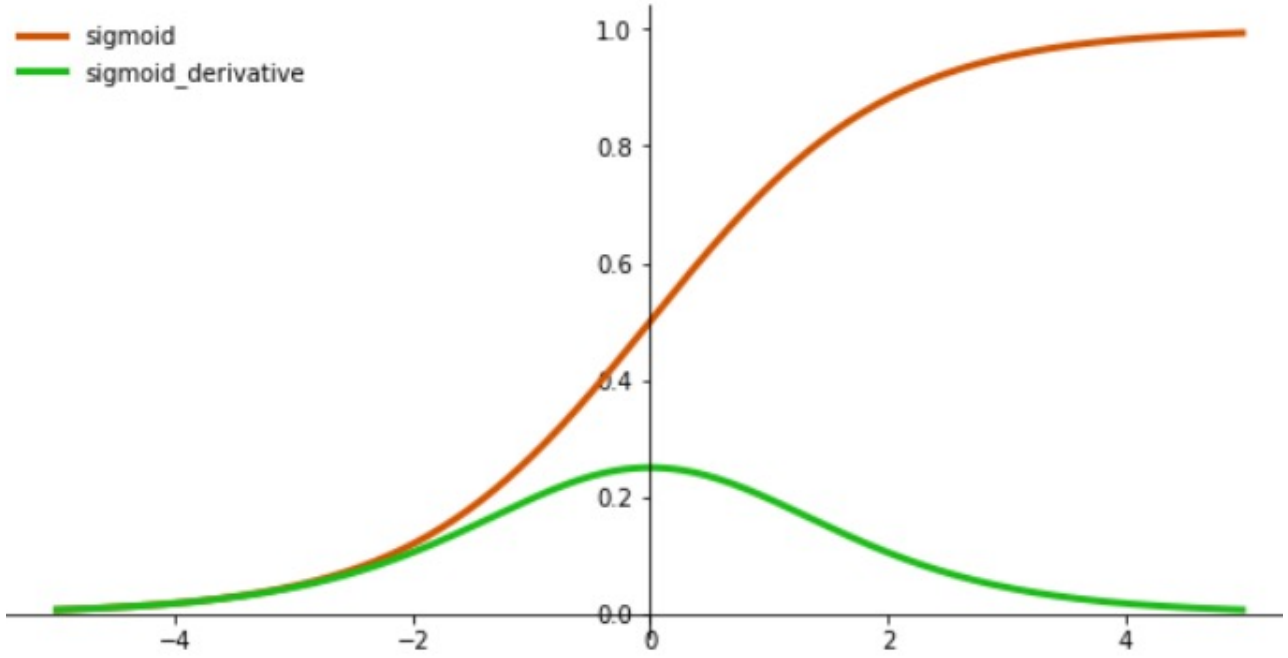
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

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

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

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

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

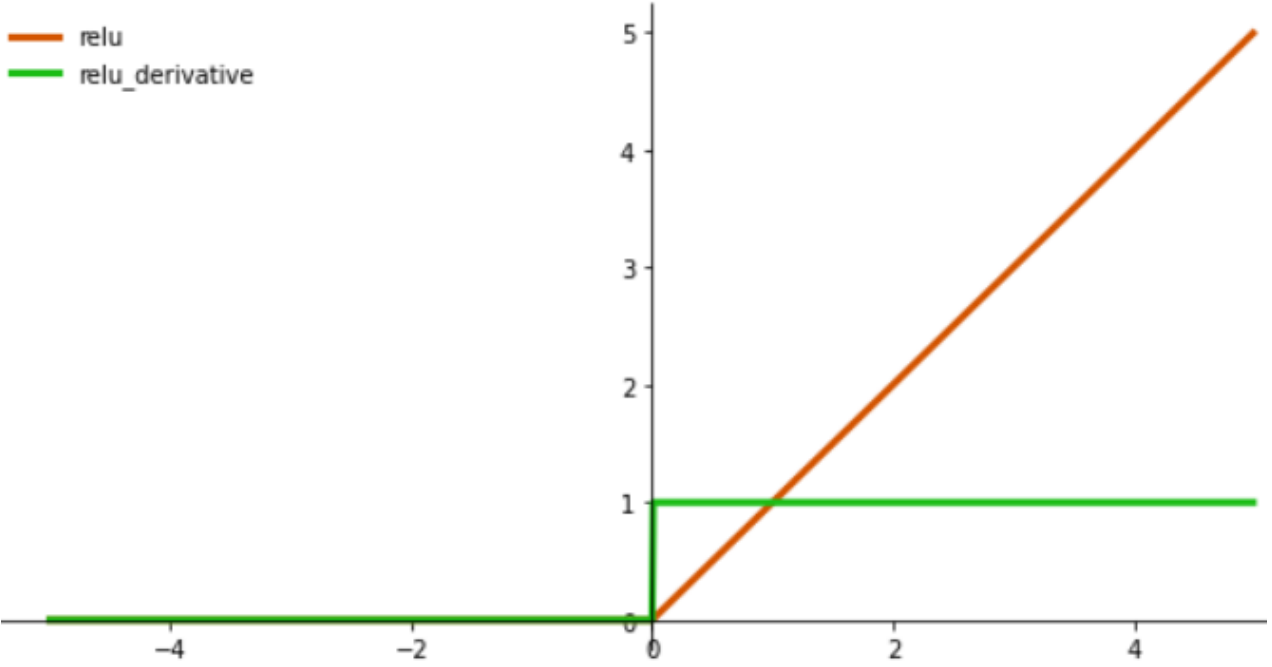
data =

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

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

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

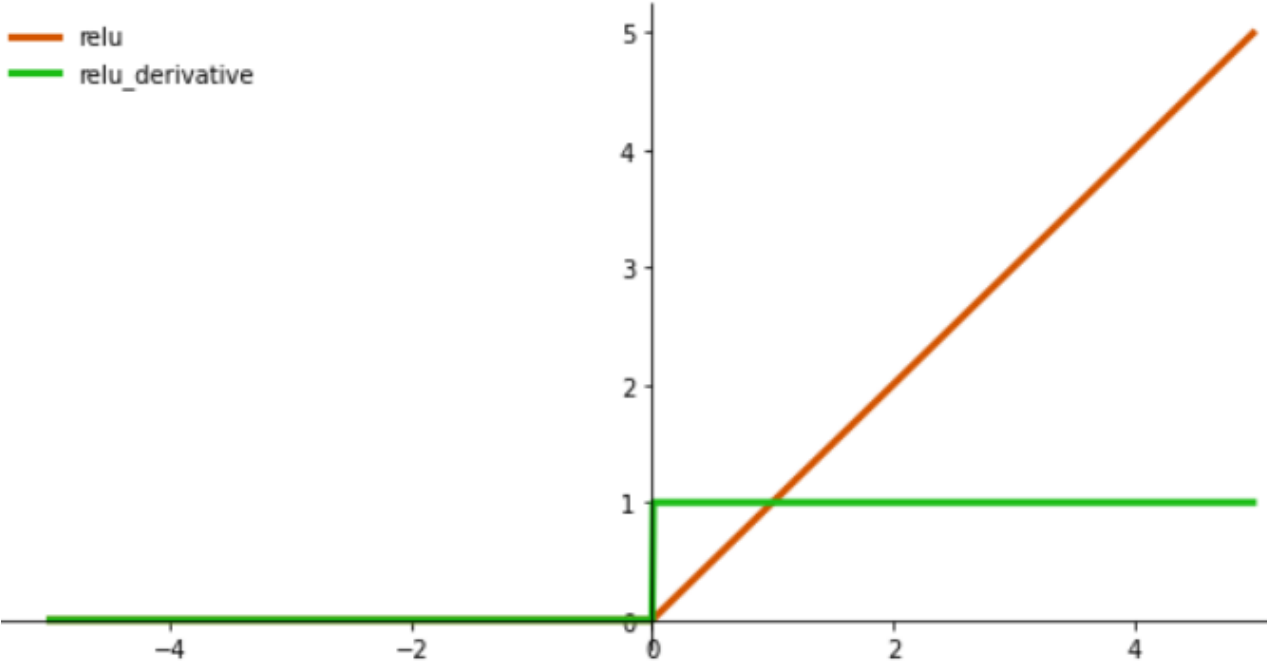
data =

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

https://colab.research.google.com/drive/1NtUb2-gR4HYaQLjdjEEJWQD3BWR51gje?usp=sharing

[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)
```

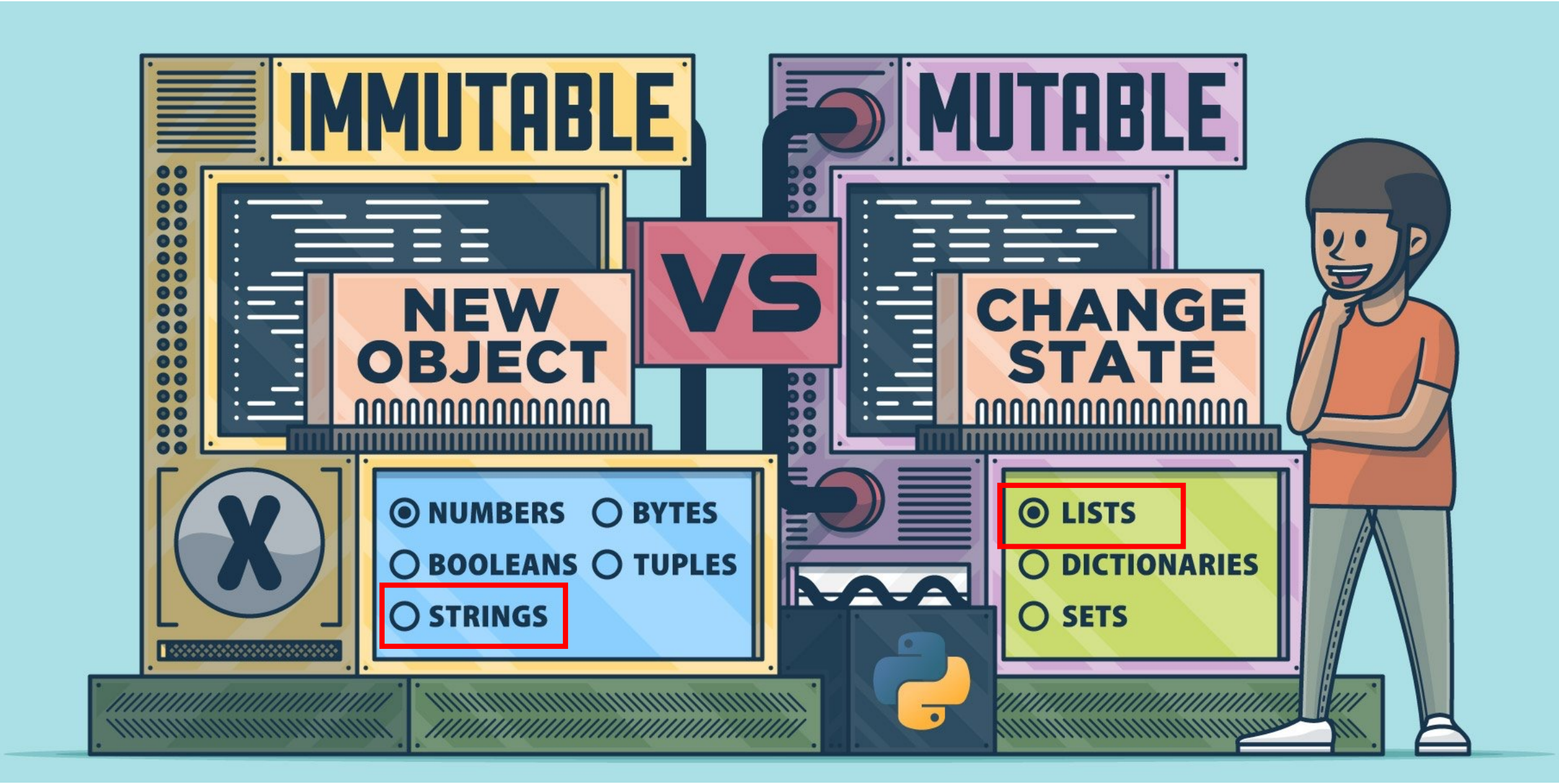
```
[6, 5, 7, 1, 9, 2]
[9, 7, 6, 5, 2, 1]
```

```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]
```



Credit by: https://realpython.com/python-mutable-vs-immutable-types/

Mutable	Immutable
We can change the value of a variable (which we have stored)	We can not change the value of a variable (which we have stored)

List is Mutable

```
words = ['I', 'like', 'AIVN']
for w in words:
    print(w, end=', ') #Printing the elements from the list cities, separated by a comma & space

print(hex(id(words))) #Printing the location of the object created in the memory address in hexadecimal format

words[len(words)-1] = "Kungfu Panda"

for w in words:
    print(w, end=', ') #Printing the elements from the list cities, separated by a comma & space

print(hex(id(words))) #Printing the location of the object created in the memory address in hexadecimal format
```

```
I, like, AIVN, 0x7fee1125d540
I, like, Kungfu Panda, 0x7fee1125d540
```

https://colab.research.google.com/drive/1idNgCzKlMUYSPvDzp74kya7fX2ksgAko?usp=sharing

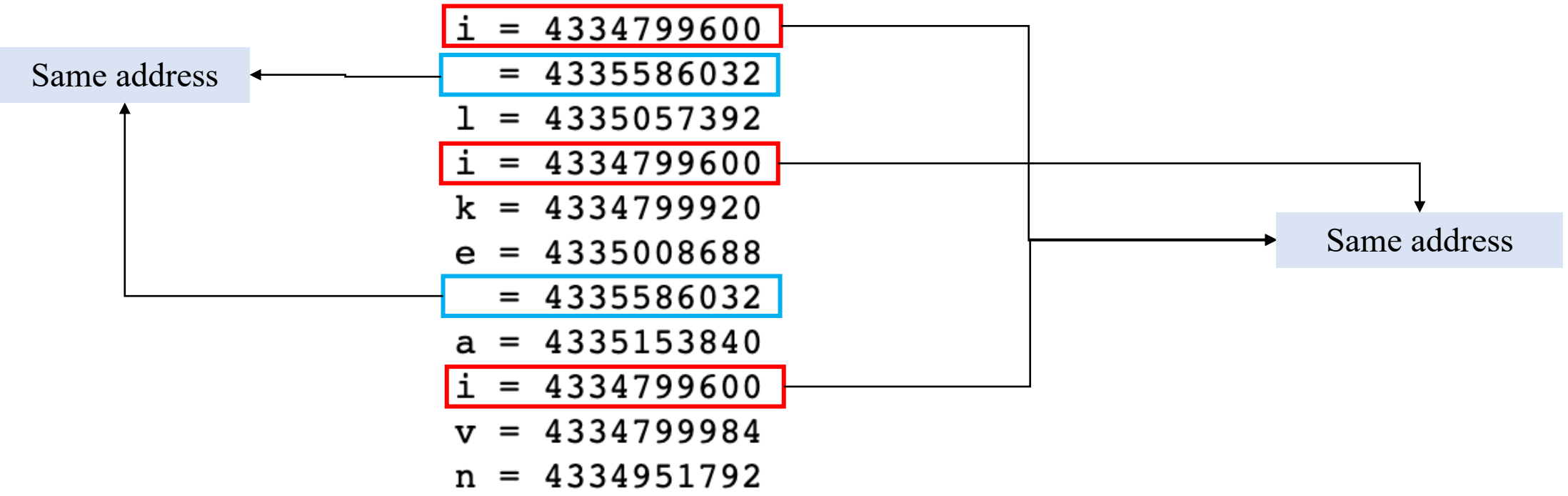
String is Immutable

```
words = "I like AIVN"
print(words)

words[len(words)-1] = '!'
```

```
I like AIVN
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-cb249155b27a> in <cell line: 4>()
      2 print(words)
      3
----> 4 words[len(words)-1] = '!'

TypeError: 'str' object does not support item assignment
```

String is Immutable

```
words = "i like aivn"

for idx in range (0,len(words)):
    print (words[idx], "=", id(words[idx]))
```



String is Immutable

```
1  word_1 = "aivn"
2  word_2 = "aivn"
3  print("id of word_1 = ", id(word_1))
4  print("id of word_2 = ", id(word_2))
```

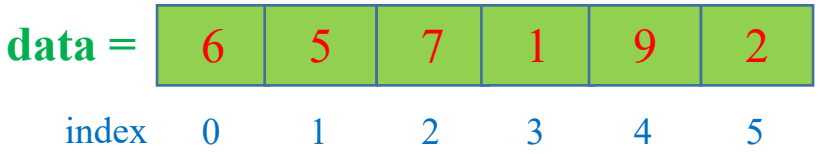
```
id of word_1 =  139893503508976
id of word_2 =  139893503508976
```

Outline

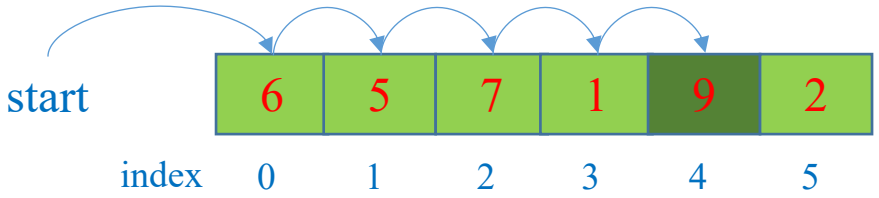
- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

Algorithms on List

❖ Linear searching

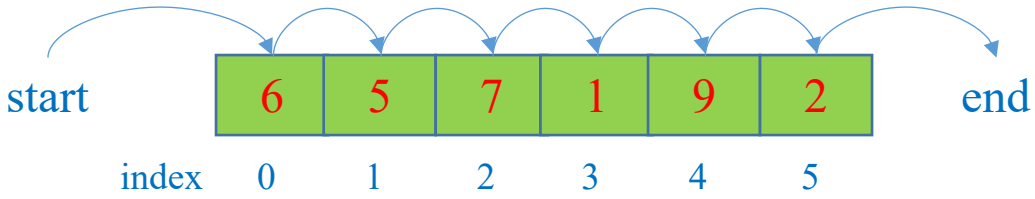


Searching for 9

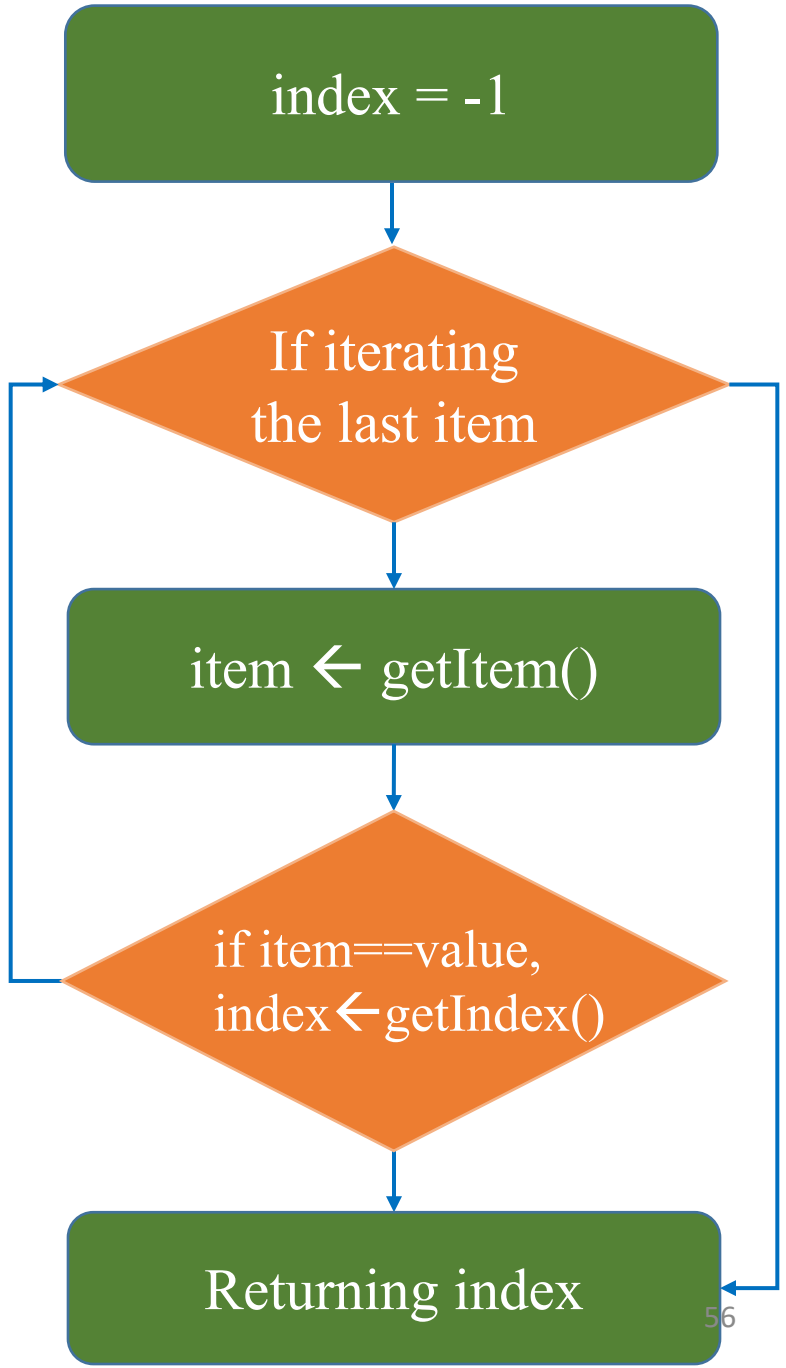


Returning 4

Searching for 8



Returning ?



Algorithms on List

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

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

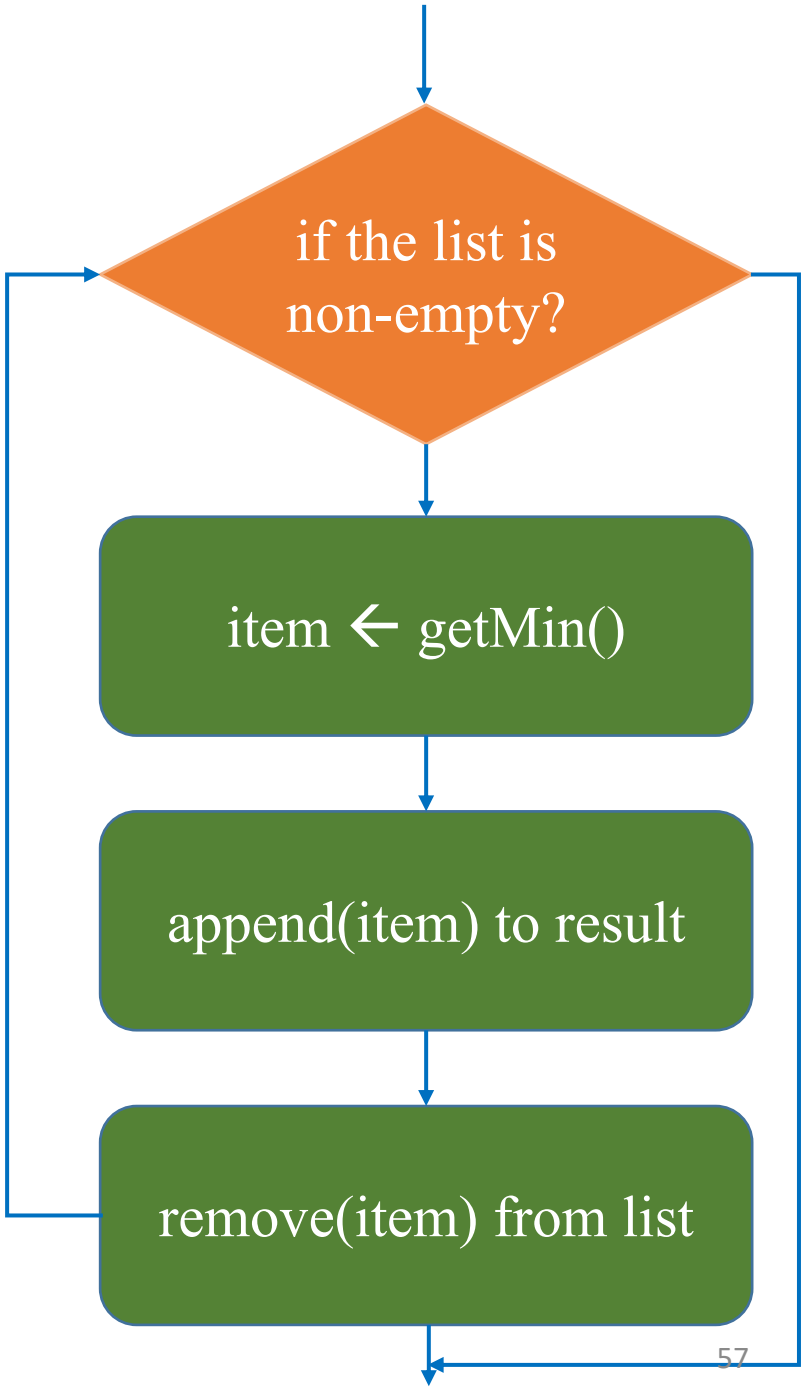
result = [ ]

min(data) = 1

result.append(1) = | 1 |

data.remove(1) = | 6 | 5 | 7 | 9 | 2 |

. . .



Algorithms on List

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

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

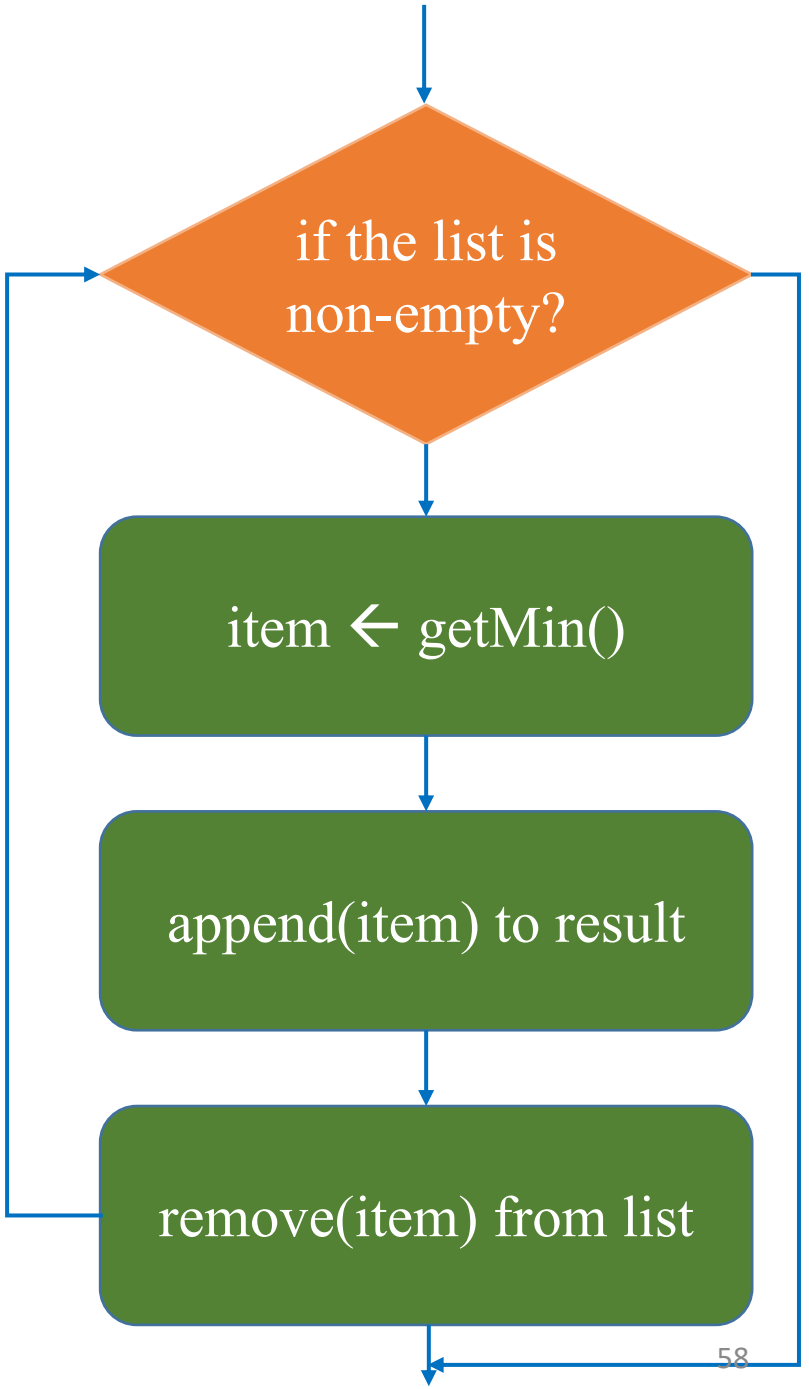
result = | 1 |

min(data) = 2

result.append(2) = | 1 | 2 |

data.remove(2) = | 6 | 5 | 7 | 9 |

. . .



Quizzes

Quiz 1

```
var = 1
a_list = [0, 1, 2]
while var in a_list:
  print("Good morning!")

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

Quiz 2

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

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

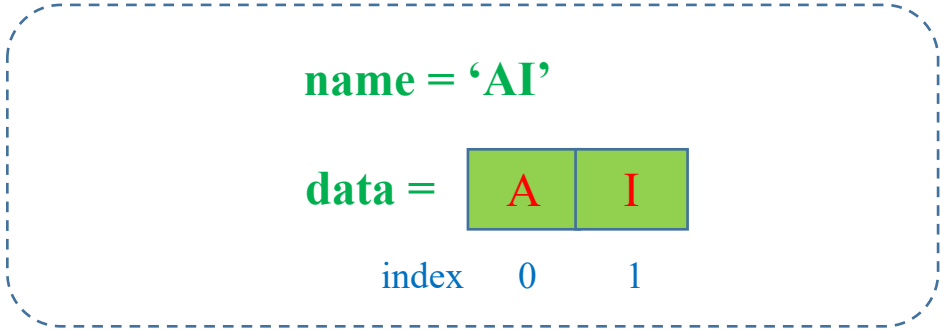
Quiz 3

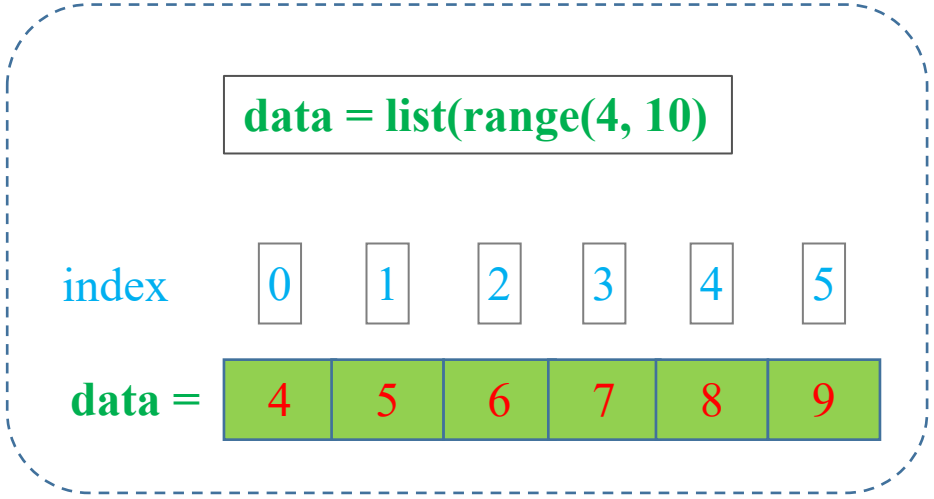
```
a_list = [1, 2]
print(f'a list before is {a_list}')

for x in a_list:
  x = 20

print(f'a list after is {a_list}')
```

https://colab.research.google.com/drive/1NtUb2-gR4HYaQLjdjEEJWQD3BWR51gje?usp=sharing

Converting to List

aList ← list(iterable)






```
1  name = 'AI'
2  data = list(name)
3
4  print(name)
5  print(data)
```

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

```
1  data = list(range(4, 10))
2  print(data)
```

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

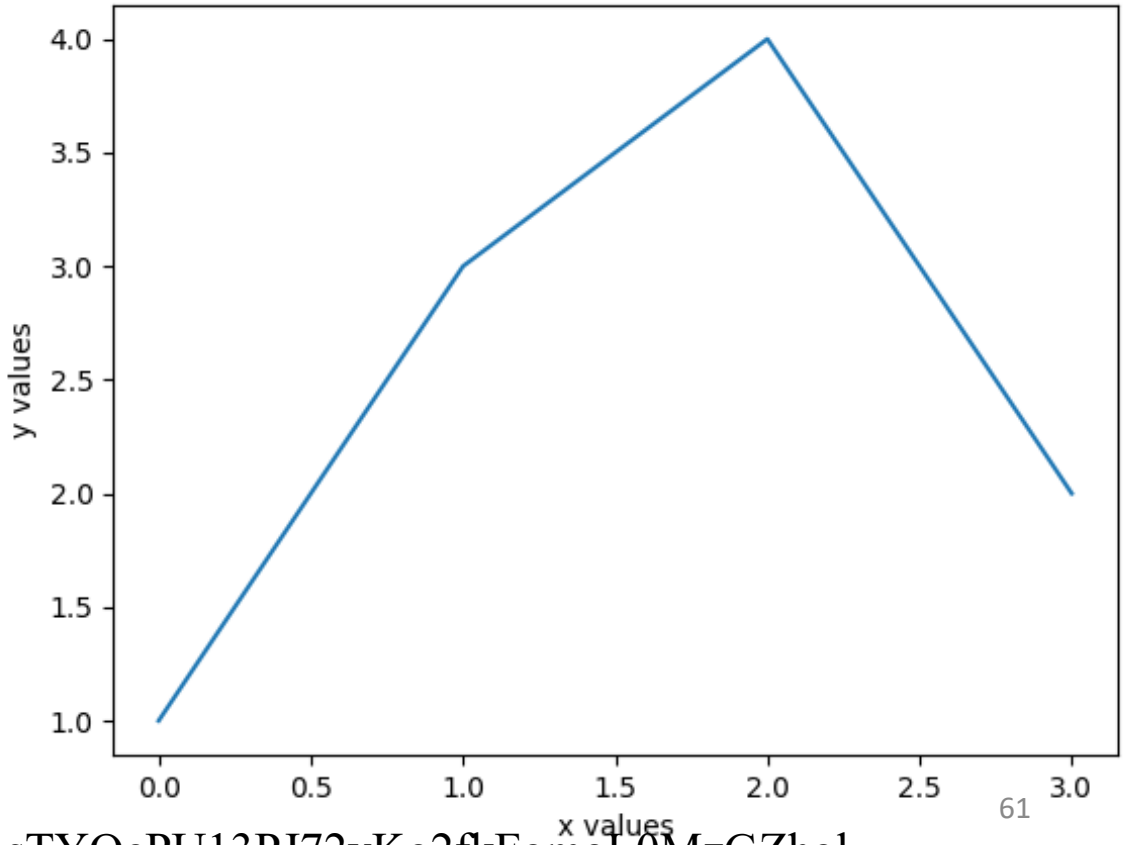
Plotting a function

❖ Using matplotlib

```
1  # ví dụ 1
2  import matplotlib.pyplot as plt
3
4  # data
5  x_data = [0, 1, 2, 3]
6  y_data = [1, 3, 4, 2]
7
8  # plot
9  plt.plot(x_data, y_data)
10 plt.ylabel('y values')
11 plt.xlabel('x values')
12
13 # show
14 plt.show()
```

x-data

y-data



https://colab.research.google.com/drive/1sTYOcPU13PJ72vKo2fkFomoL0MzGZhgl

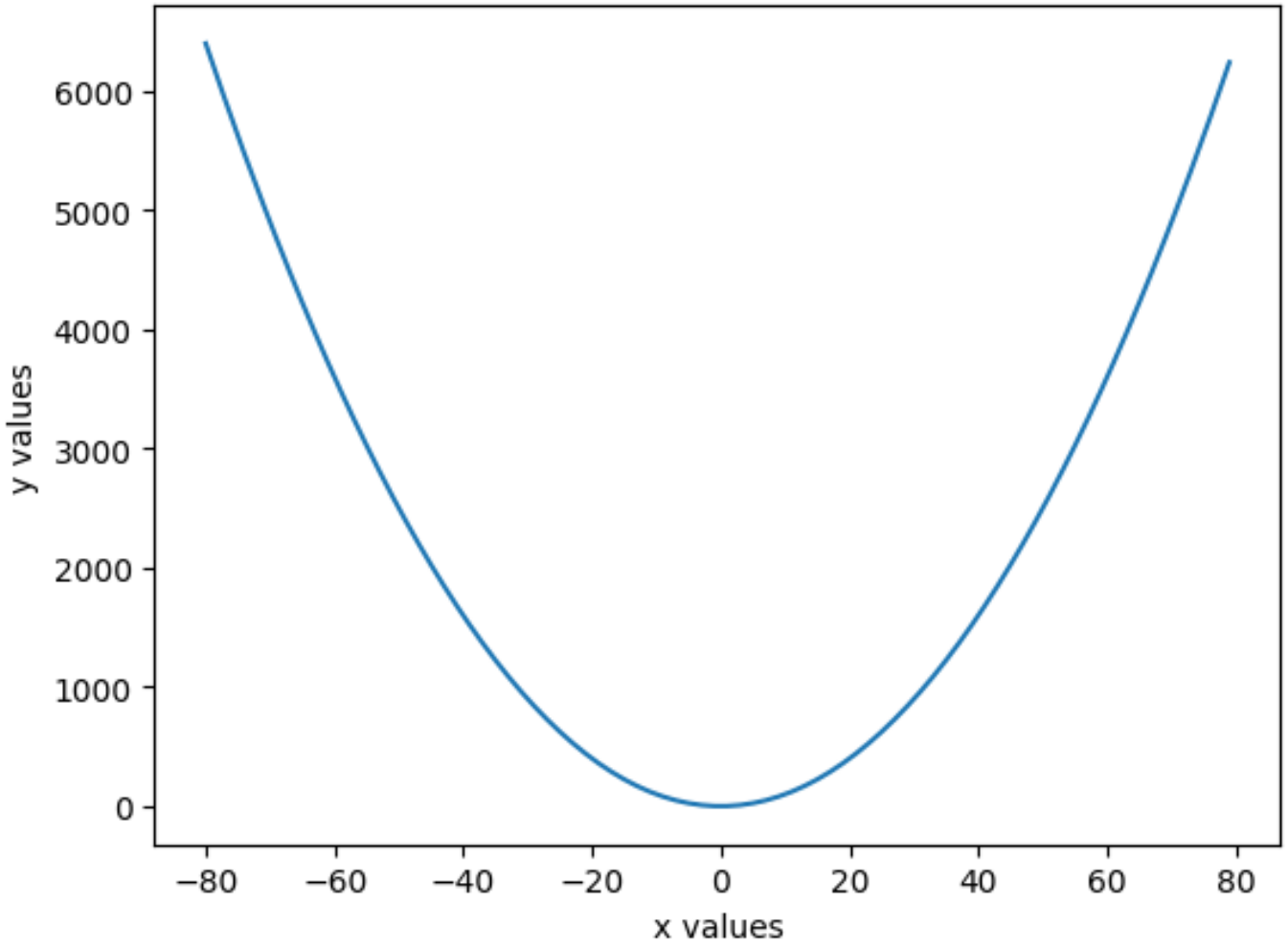
Plotting a function

❖ Using matplotlib

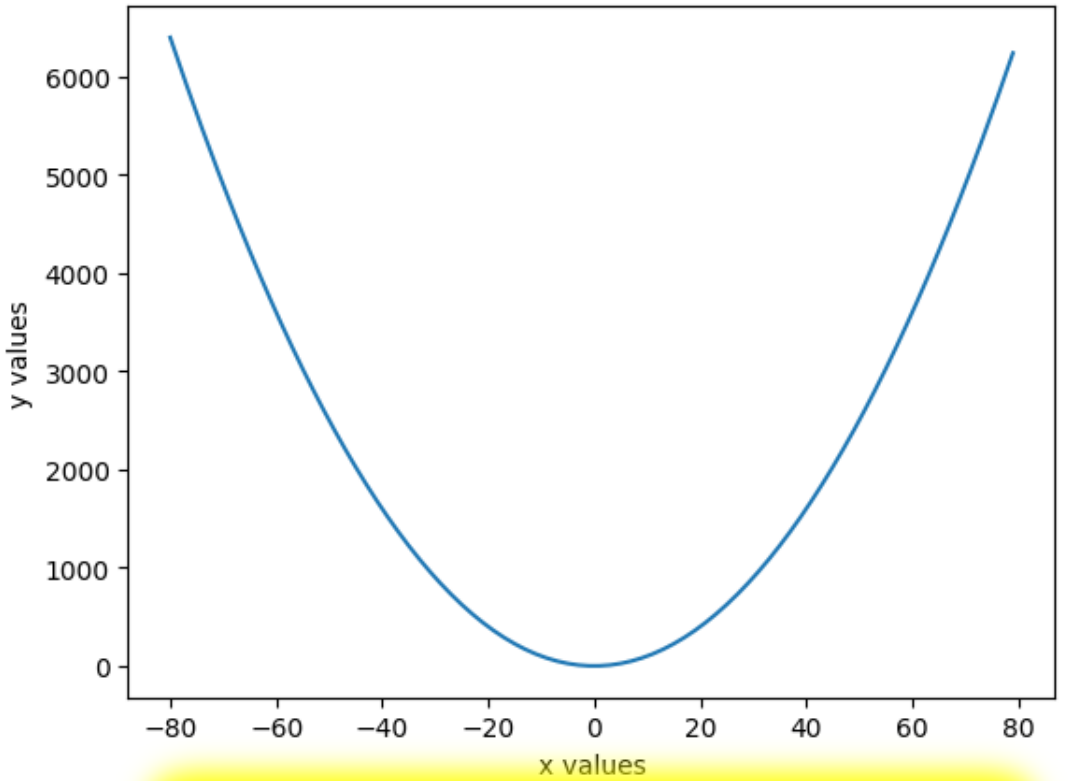
$x \in [-8, 8]$

$y = x^2$

How to do it?



```
x_data = list(range(-80, 80, 1))
y_data = []
for i in x_data:
  y_data.append(i**2)
```




```
plt.plot(x_data, y_data)
plt.show()
```

Everytime you run the program, x_data and y_data have to initialize

Solution

Store and load data from a memory

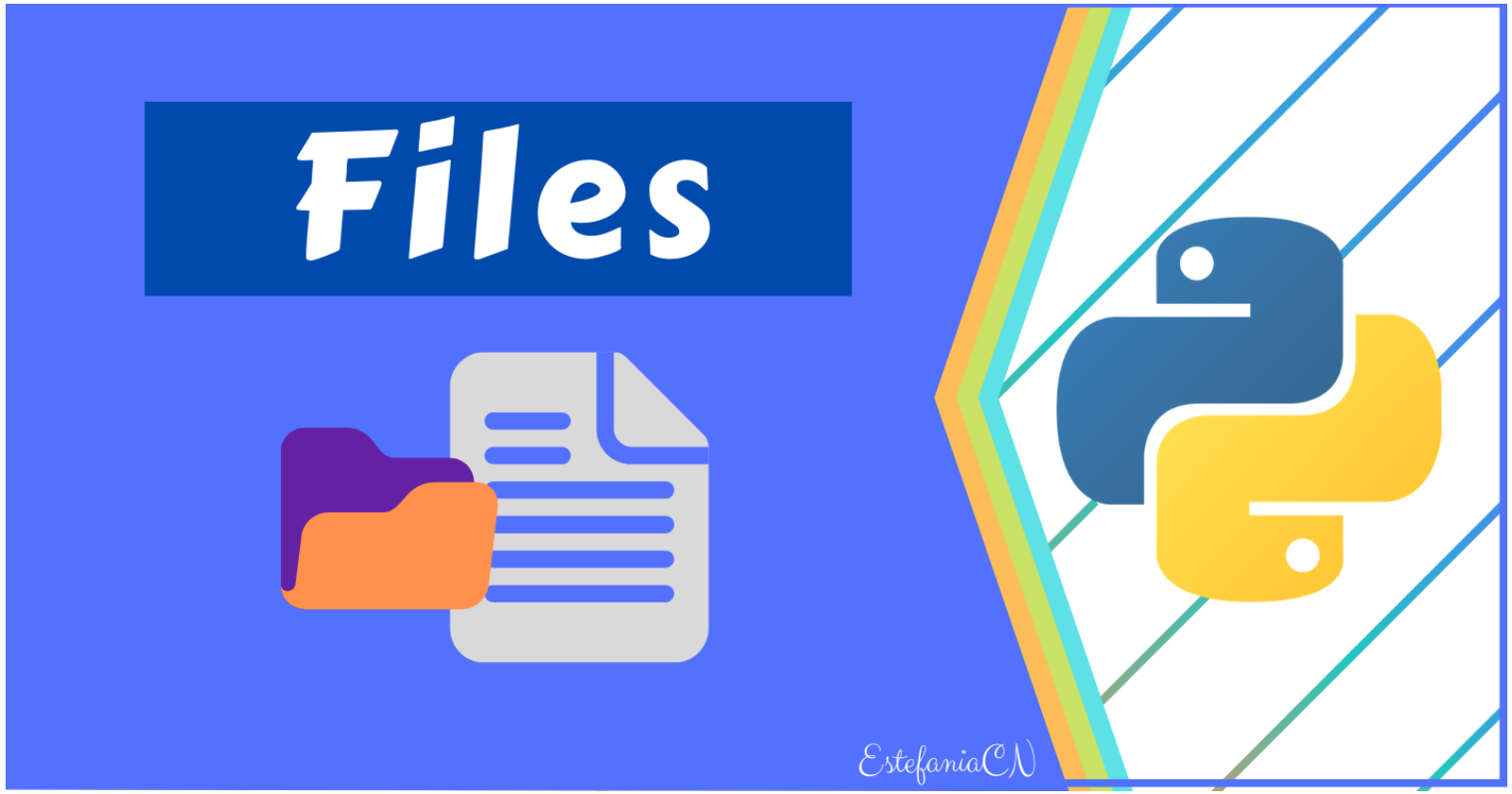
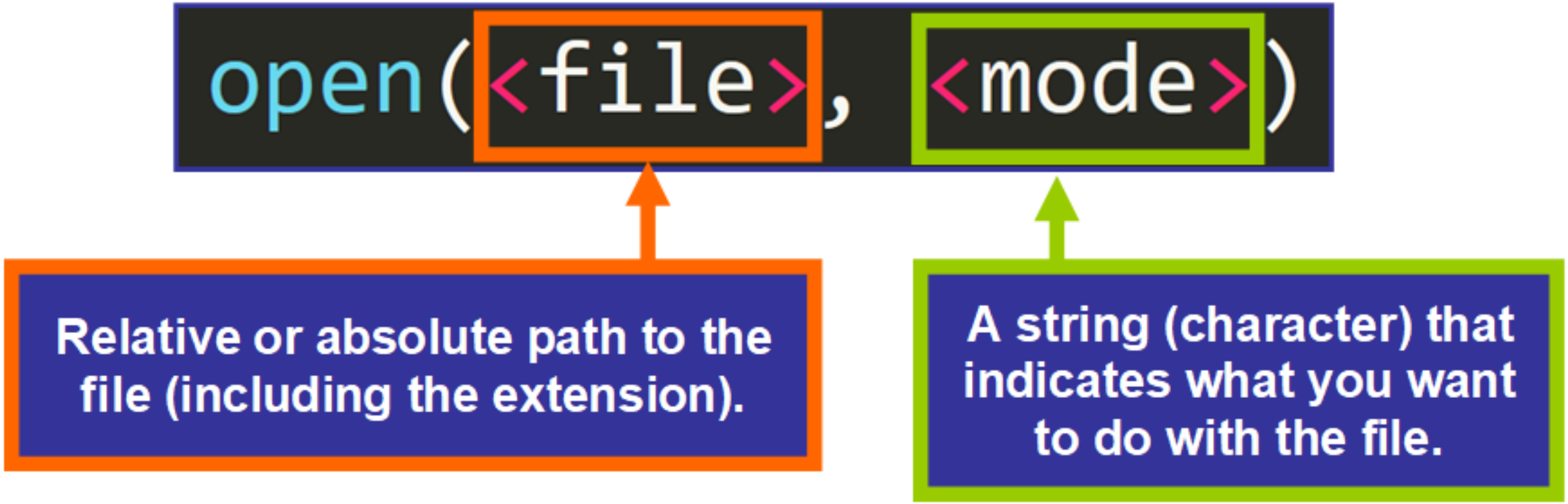


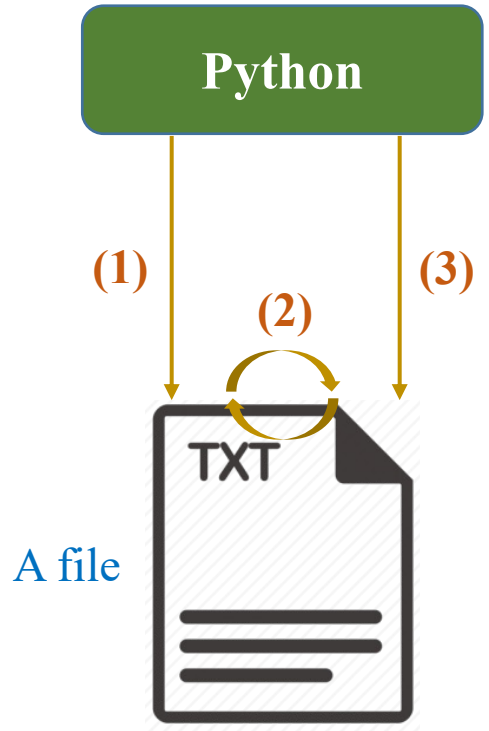
Image credits: https://www.freecodecamp.org/

How to handle File in Python



File

❖ Typical procedure

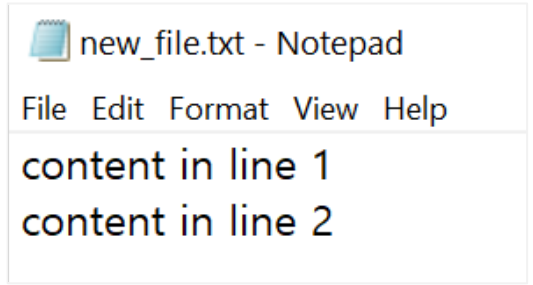


(1)	Connect to file
(2)	Read from/write to file
(3)	Disconnect to file

Write to a file (not exist)

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

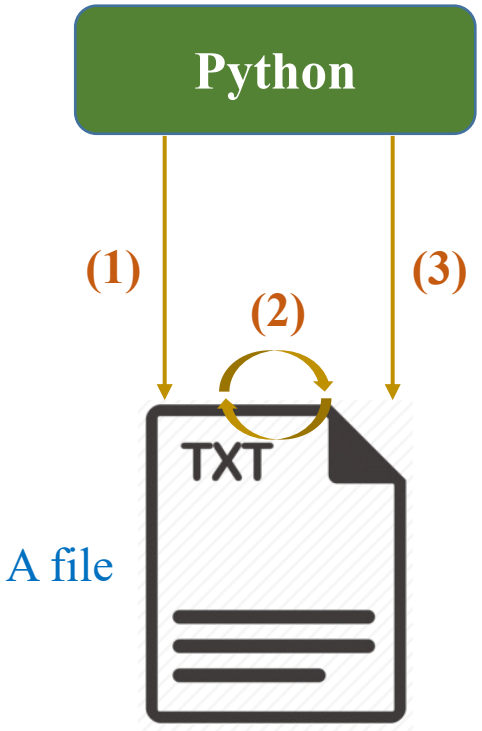
```
1  # kết nối với file
2  a_file = open('new_file.txt', 'w')
3
4  text1 = 'content in line 1 \n'
5  a_file.write(text1)
6
7  text2 = 'content in line 2 \n'
8  a_file.write(text2)
9
10 # Đóng kết nối với file
11 a_file.close()
```



https://colab.research.google.com/drive/11LAgyGCNW7Oj7NH2Njg34QH4s3j4vRut?usp=sharing

File

❖ Typical procedure



(1)	Connect to file
(2)	Read from/write to file
(3)	Disconnect to file

Write to a file (appending content if the file already exists)

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

new_file.txt - Notepad

File Edit Format View Help

content in line 1
content in line 2

```
1  # kết nối với file
2  a_file = open('new_file.txt', 'a')
3
4  text3 = 'content in line 3 \n'
5  a_file.write(text3)
6
7  # Đóng kết nối với file
8  a_file.close()
```

new_file.txt - Notepad

File Edit Format View Help

content in line 1
content in line 2
content in line 3

```
1  # kết nối với file
2  a_file = open('non_existing_file.txt', 'a')
3
4  text3 = 'content in line 3 \n'
5  a_file.write(text3)
6
7  # Đóng kết nối với file
8  a_file.close()
```

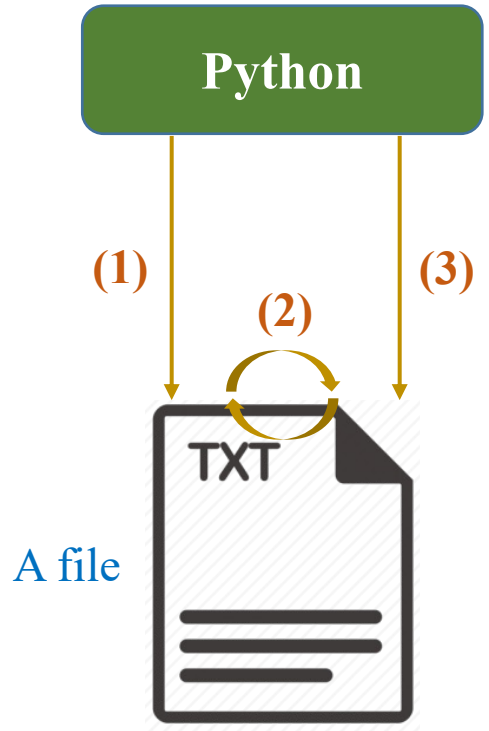
non_existing_file.txt - Notepad

File Edit Format View Help

content in line 3

File

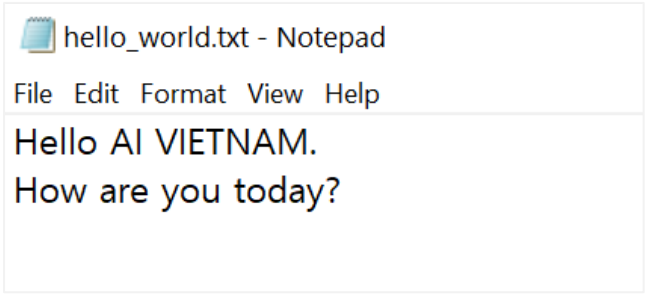
❖ Typical procedure



(1)	Connect to file
(2)	Read from/write to file
(3)	Disconnect to file

Read from a file (already exist)

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

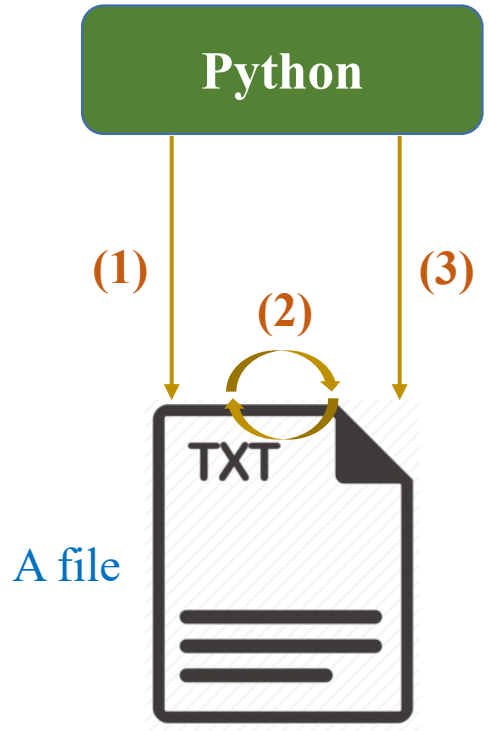



```python
# kết nối với file
a_file = open('hello_world.txt','r')

# read content as string
data = a_file.read()

print(type(data))
print(data)

# Đóng kết nối với file
a_file.close()
```

```
<class 'str'>
Hello AI VIETNAM.
How are you today?
```

File

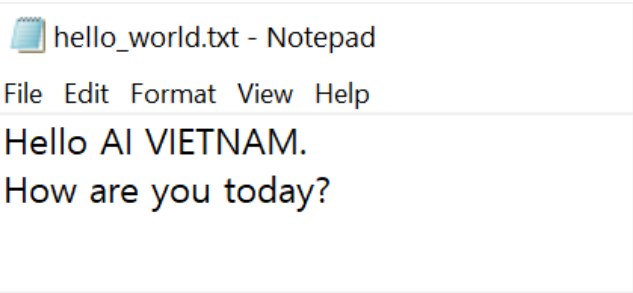
❖ Typical procedure



(1)	Connect to file
(2)	Read from/write to file
(3)	Disconnect to file

Read content from a file as lines

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




```
1  # kết nối với file
2  a_file = open('hello_world.txt','r')
3
4  # read content as string
5  lines = a_file.readlines()
6  for line in lines:
7      print(line)
8
9  # Đóng kết nối với file
10 a_file.close()
```

```
Hello AI VIETNAM.

How are you today?
```

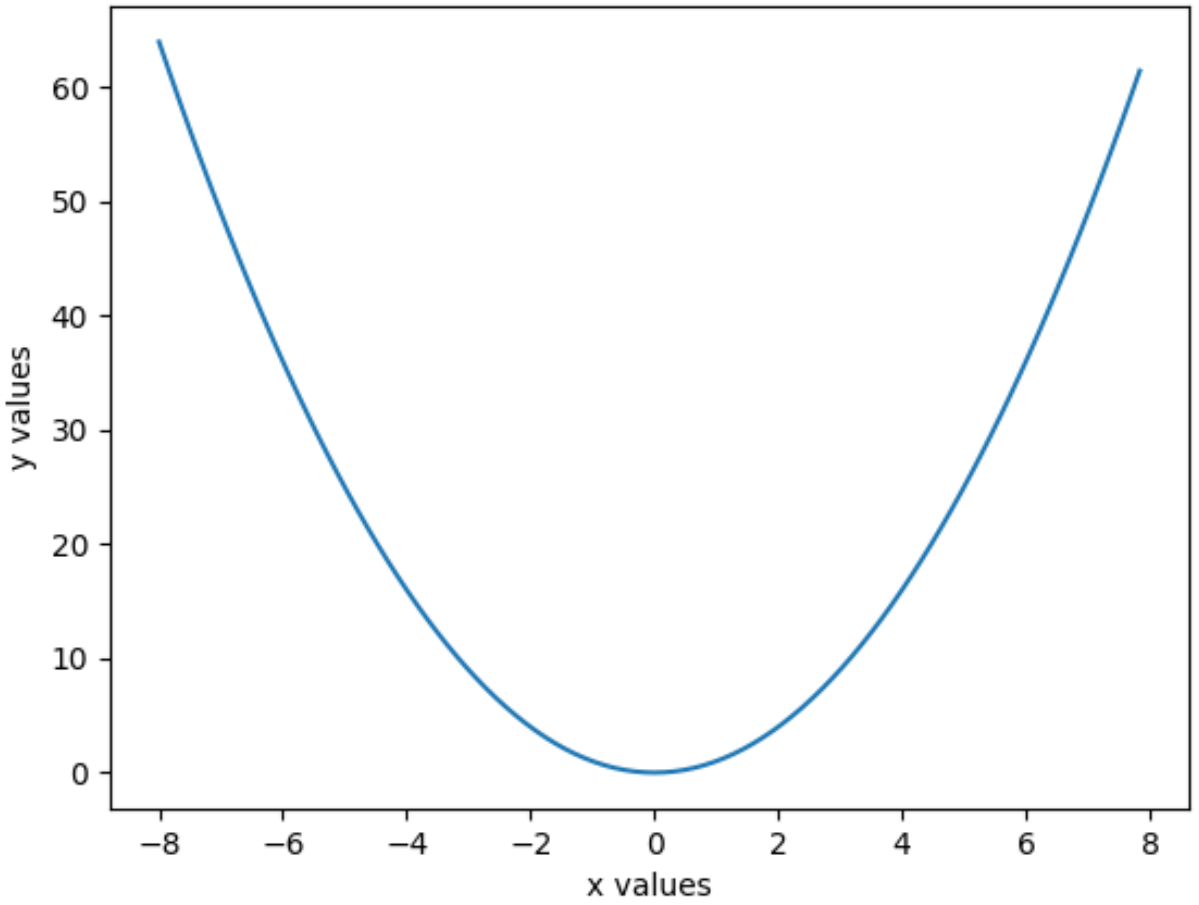
Example: List and File

❖ Read and Write from/to a log file

https://colab.research.google.com/drive/11LAgyGCNW7Oj7NH2Njg34QH4s3j4vRut?usp=sharing

```python
# write to a log file
with open(r'log.txt', 'w') as fp:
    for item in data:
        # write each item on a new line
        fp.write(f'{item}\n')
```

```python
# read from the log file
data = []
with open(r'log.txt', 'r') as fp:
    lines = fp.readlines()
    for line in lines:
        line = line.strip('\n')
        data.append(float(line))
```



Unlike open() where you have to close the file with the close() method, the with statement closes the file for you without you telling it to.

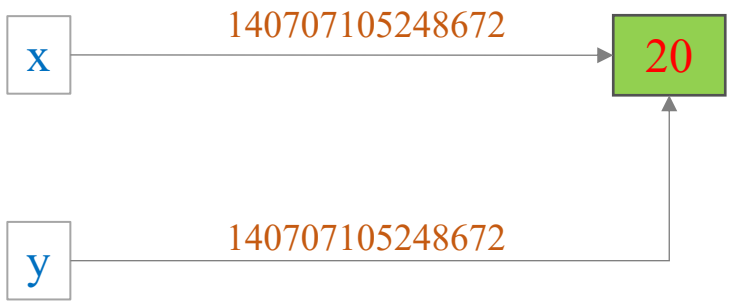
Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

Variables and Addresses

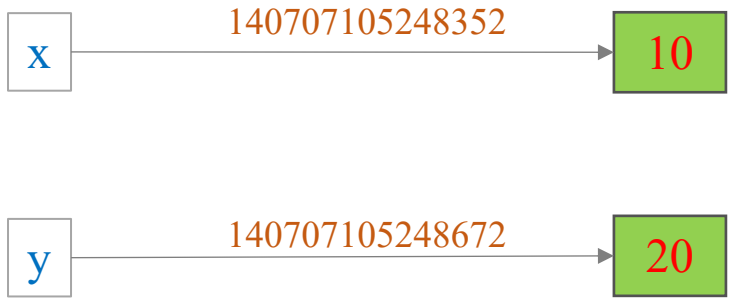
```
1  x = 20
2  y = x
3
4  print(x)
5  print(y)
6  print(id(x))
7  print(id(y))
```

```
20
20
140707105248672
140707105248672
```




```
1  x = 20
2  y = x
3  x = 10
4
5  print(x)
6  print(y)
7  print(id(x))
8  print(id(y))
```

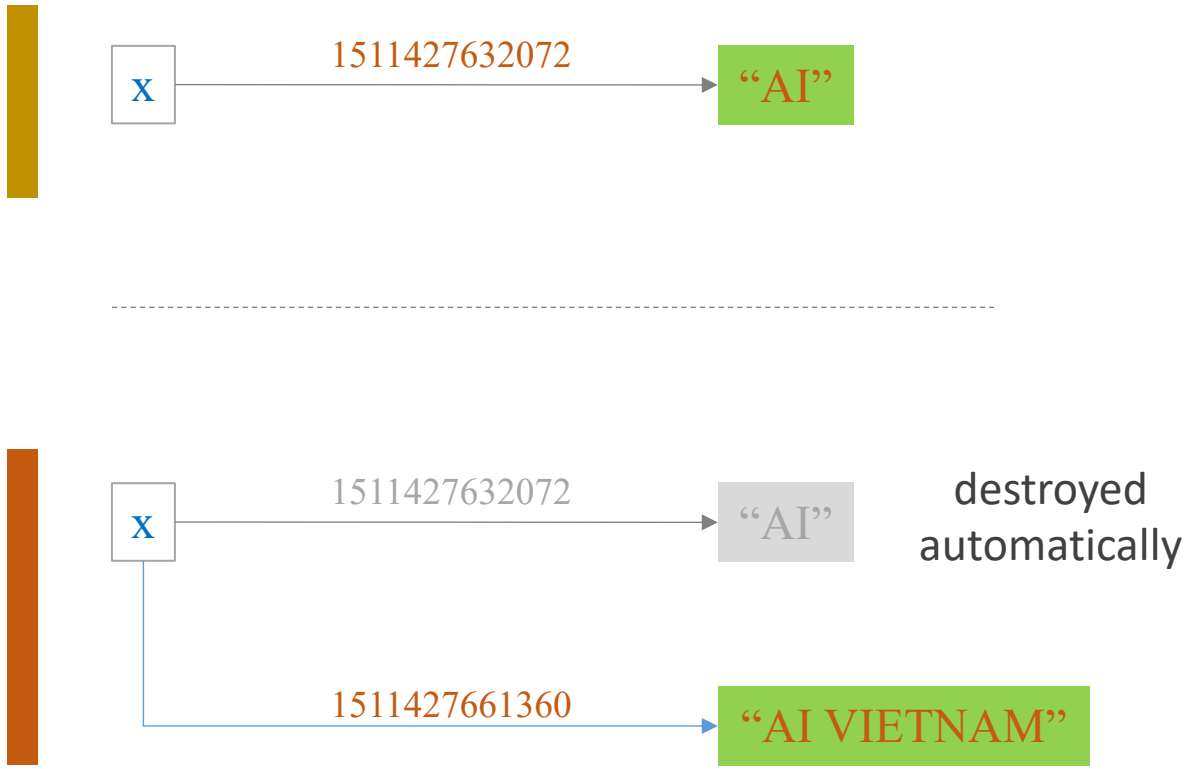
```
10
20
140707105248352
140707105248672
```



https://colab.research.google.com/drive/1Nx6IsouNci__nwvB-xa1-YhOTpdVMyIh?usp=sharing

Variables and Addresses

- Immutable types: Cannot be changed in place
- Including ints, floats, strings, and tuples



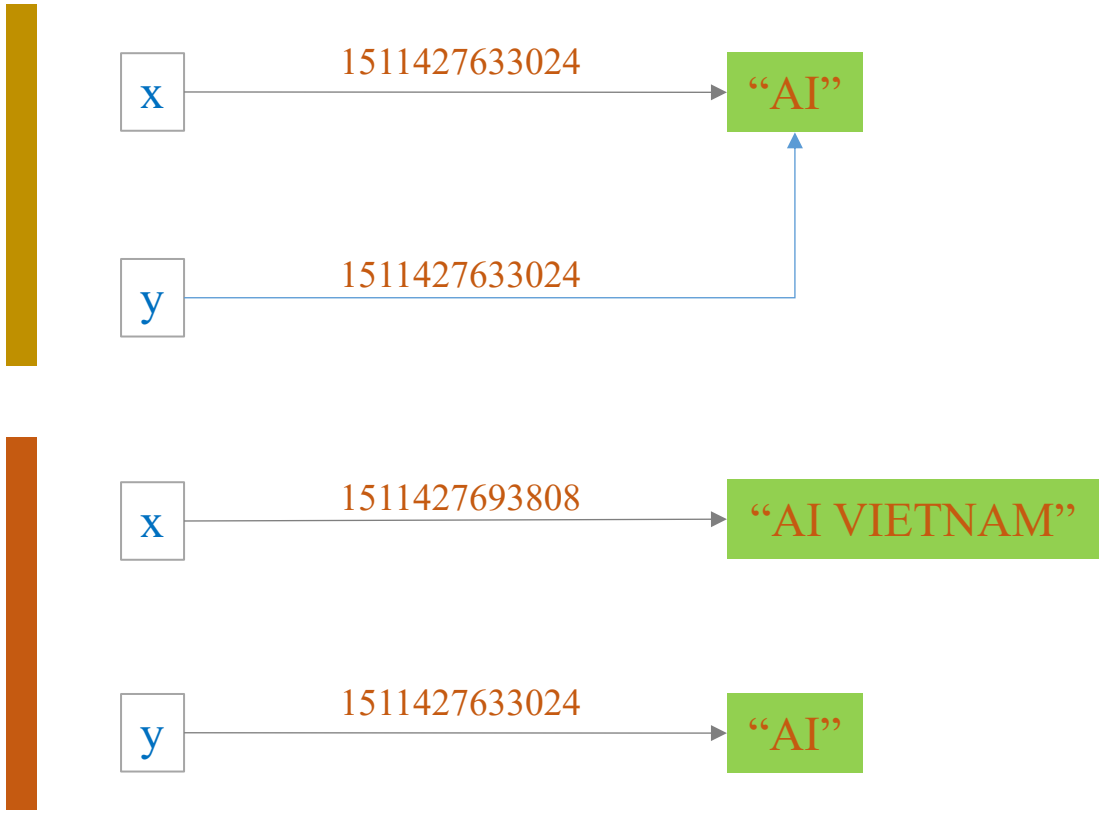

```
1  # immutable
2
3  x = "AI"
4  print(x)
5  print(id(x))
6  print()
7
8  x = x + " VIETNAM"
9  print(x)
10 print(id(x))
```

```
AI
1511427632072
```

```
AI VIETNAM
1511427661360
```

Variables and Addresses

- Immutable types: Cannot be changed in place
- Including ints, floats, strings, and tuples




```python
# immutable

x = "AI"
y = x
print('x: ', x)
print('y: ', y)
print('x address: ', id(x))
print('y address: ', id(y))
print()

x = x + " VIETNAM"
print('x: ', x)
print('x address: ', id(x))
print('y: ', y)
print('y address: ', id(y))
```

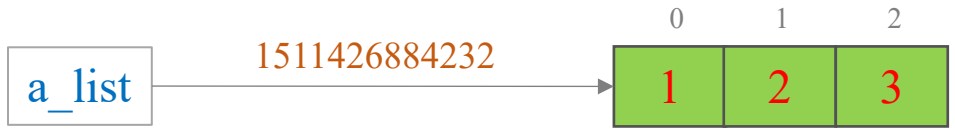
```
x:  AI
y:  AI
x address:  1511427633024
y address:  1511427633024
```

```
x:  AI VIETNAM
x address:  1511427693808
y:  AI
y address:  1511427633024
```

Variables and Addresses

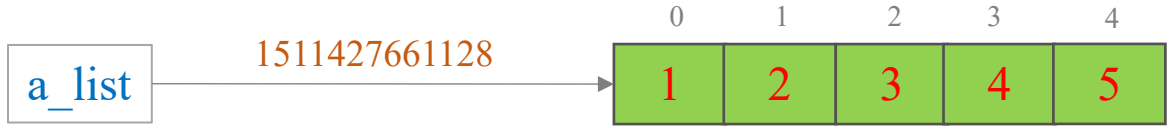
```
1  # aivietnam
2
3  a_list = [1, 2, 3]
4  print('a_list: ', a_list)
5  print('a_list address: ', id(a_list))
```

```
a_list:  [1, 2, 3]
a_list address:  1511426884232
```




```
1  a_list = a_list + [4, 5]
2  print('a_list: ', a_list)
3  print('a_list address: ', id(a_list))
```

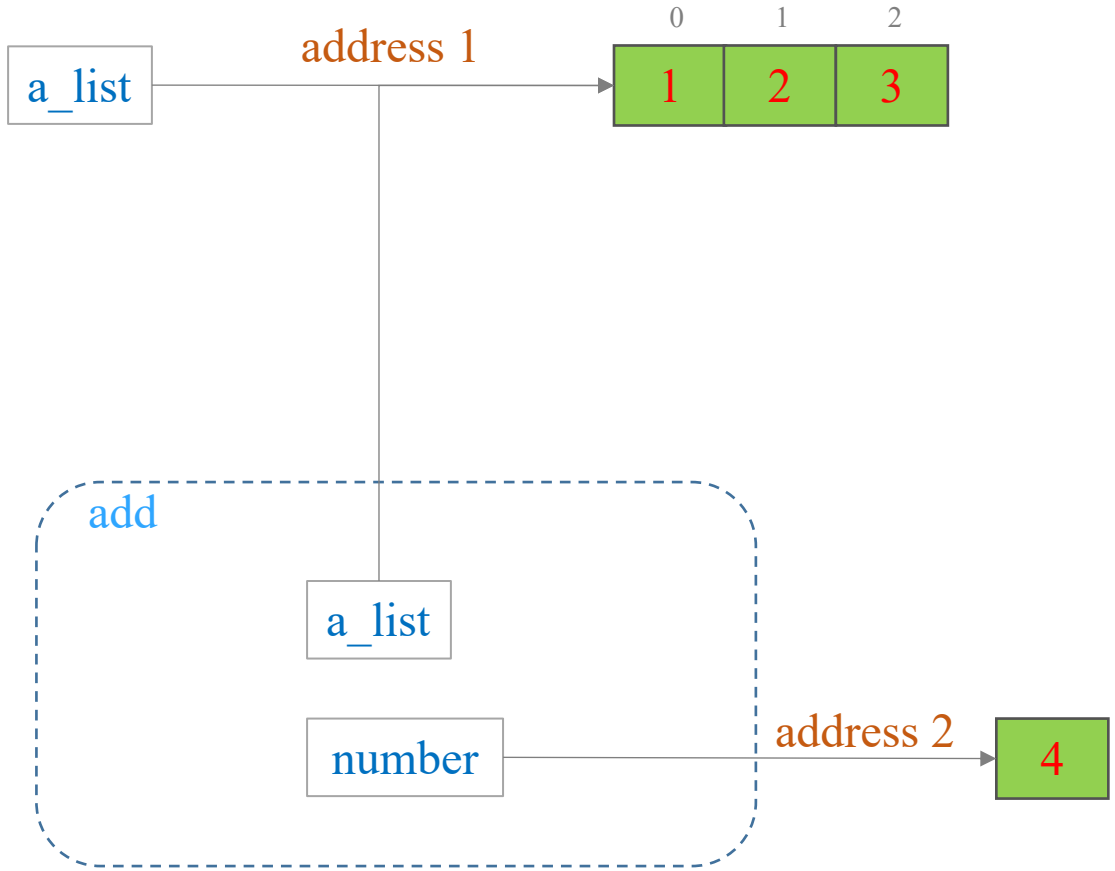
```
a_list:  [1, 2, 3, 4, 5]
a_list address:  1511427661128
```



Variables and Addresses

```python
# aivietnam

a_list = [1, 2, 3]
print(f'a_list is {a_list}')

def add(a_list, number):
    a_list = a_list + [number]

    return a_list

# test the add function
a_list = add(a_list, 4)
print(f'a_list is {a_list}')
```

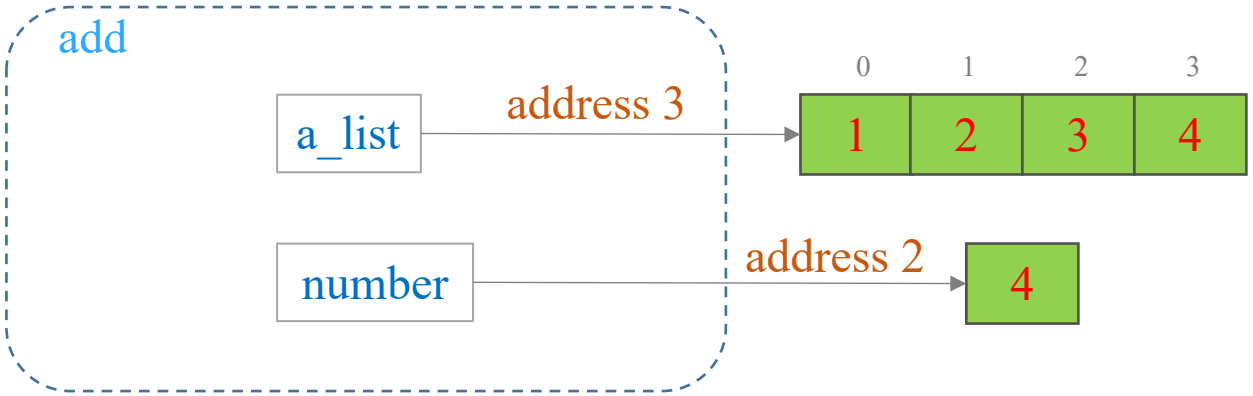
```
a_list is [1, 2, 3]
a_list is [1, 2, 3, 4]
```



Variables and Addresses

```python
# aivietnam

a_list = [1, 2, 3]
print(f'a_list is {a_list}')

def add(a_list, number):
    a_list = a_list + [number]

    return a_list

# test the add function
a_list = add(a_list, 4)
print(f'a_list is {a_list}')
```

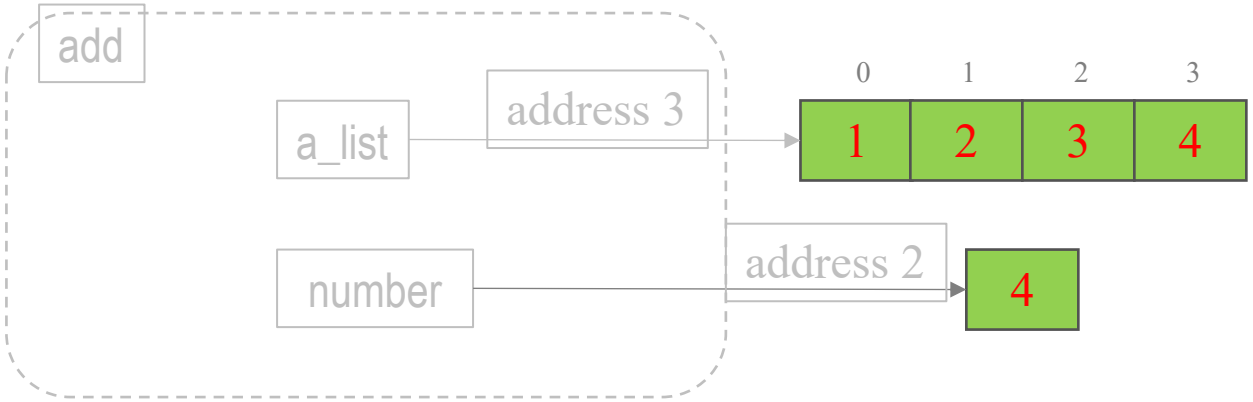
```
a_list is [1, 2, 3]
a_list is [1, 2, 3, 4]
```



Variables and Addresses

```python
# aivietnam

a_list = [1, 2, 3]
print(f'a_list is {a_list}')

def add(a_list, number):
    a_list = a_list + [number]

    return a_list

# test the add function
a_list = add(a_list, 4)
print(f'a_list is {a_list}')
```

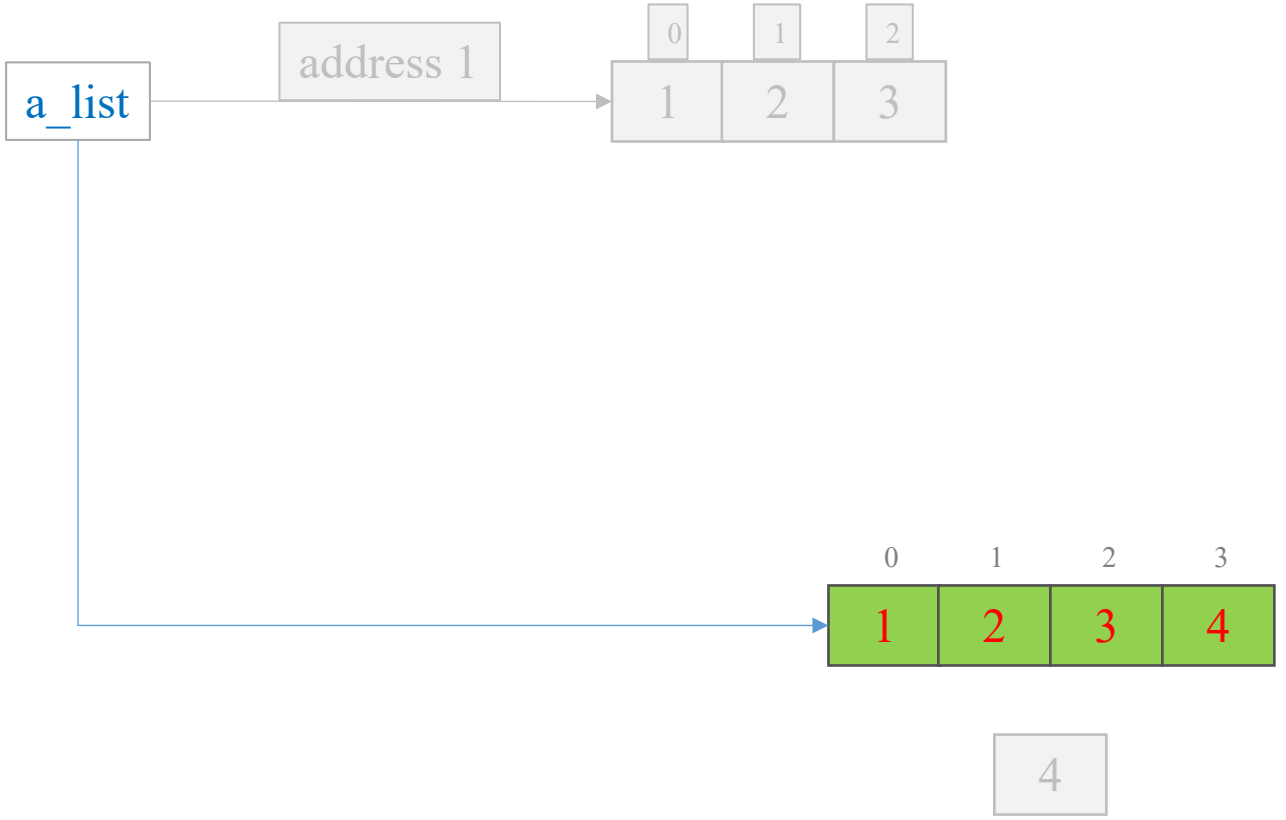
```
a_list is [1, 2, 3]
a_list is [1, 2, 3, 4]
```



Variables and Addresses

```python
# aivietnam

a_list = [1, 2, 3]
print(f'a_list is {a_list}')

def add(a_list, number):
    a_list = a_list + [number]

    return a_list

# test the add function
a_list = add(a_list, 4)
print(f'a_list is {a_list}')
```

```
a_list is [1, 2, 3]
a_list is [1, 2, 3, 4]
```



Quizzes

❖ Small number caching

```python
# aivietnam

a = 1
b = 2

print(f'a={a} and b={b}')
print(f'a address is {id(a)}')
print(f'b address is {id(b)}')
```

```
a=1 and b=2
a address is 1923776538928
b address is 1923776538960
```

```python
# aivietnam

a = a + 2
b = b + 1

print(f'a={a} and b={b}')
print(f'a address is {id(a)}')
print(f'b address is {id(b)}')
```

```
a=3 and b=3
a address is 1923776538992
b address is 1923776538992
```

Caching integers in [-5, 256]

Quizzes

```
# aivietnam

a_list = [1, 2, 3]
for i in range(3):
    print(f'a_list[{i}] address is {id(a_list[i])}')

print(type(a_list[0]))
for x in a_list:
    print(f'x address is {id(x)}')
```

```
a_list[0] address is 140381652861168
a_list[1] address is 140381652861200
a_list[2] address is 140381652861232
<class 'int'>
x address is 140381652861168
x address is 140381652861200
x address is 140381652861232
```

Quizzes

```python
# quiz 1
index = 5
def my_function():
    print(index)


my_function()
```

```python
# quiz 2
index = 5
def my_function():
    data = index + 1
    print(index)

my_function()
```

```python
# quiz 3
index = 5
def my_function():
    index = index + 1
    print(index)

my_function()
```

```python
# quiz 4
index = 5
def my_function(index):
    index = index + 1
    print(index)

my_function(7)
```

You could reference the global name variable from inside the function but if you assign a value to the variable in the function's body, the local variable shadows the global one.

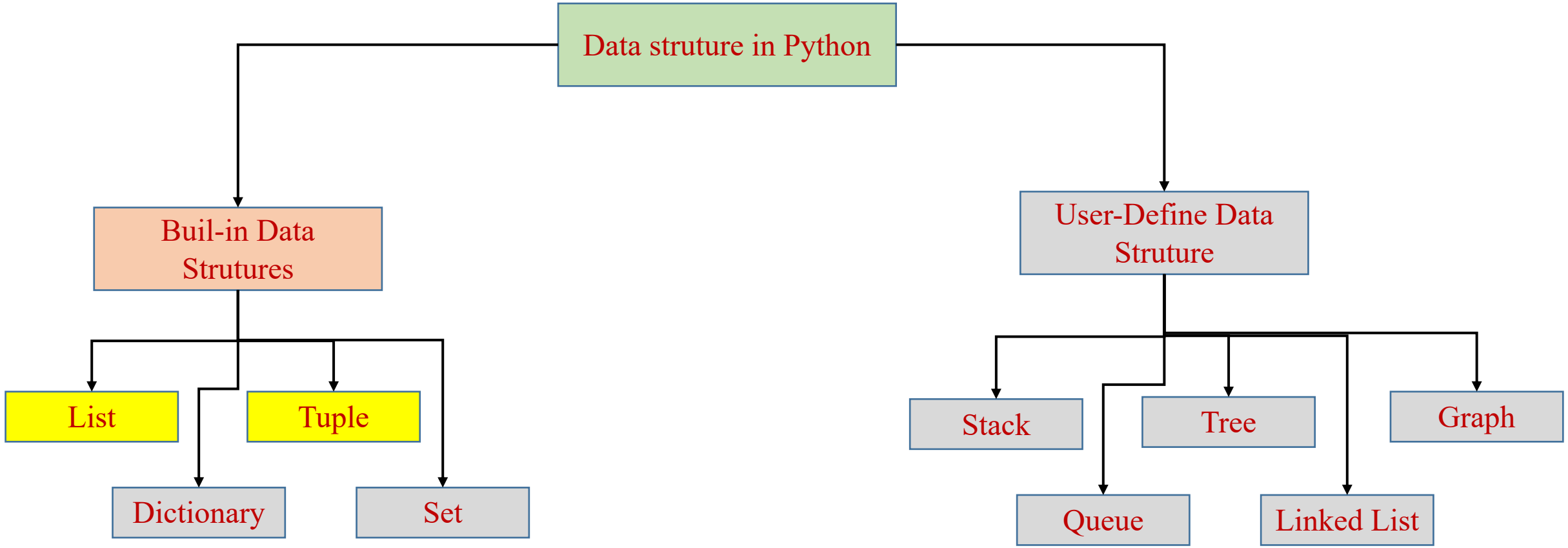
```
1  # quiz 2
2  index = 5
3  def my_function():
4      data = index + 1
5      print(index)
6
7  my_function()
```

```
1  # quiz 3
2  index = 5
3  def my_function():
4      index = index + 1
5      print(index)
6
7  my_function()
```

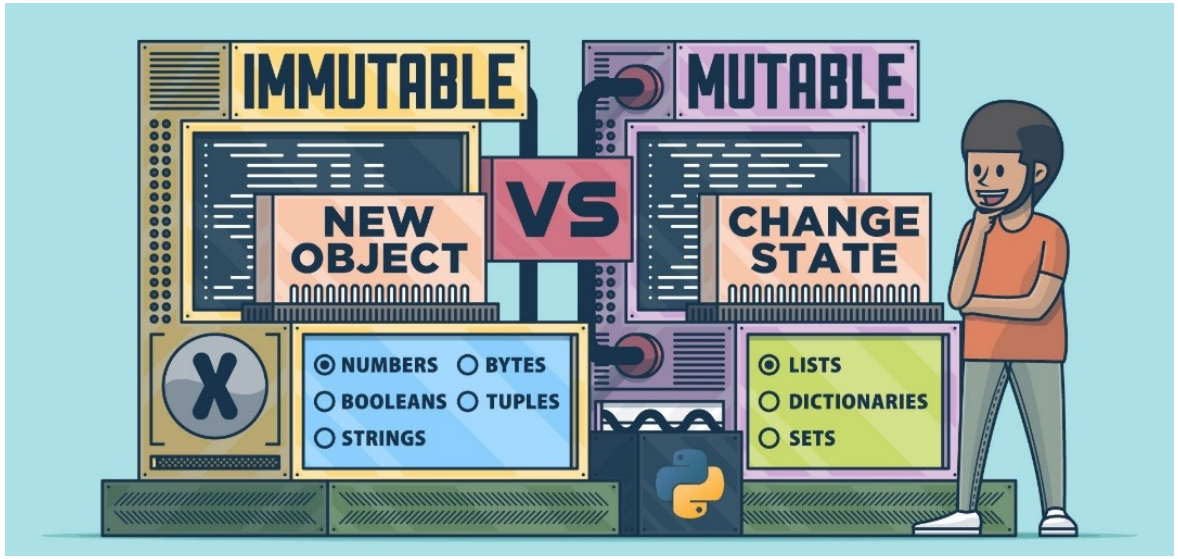
Outline

- Introduction
- String
- List
- Algorithms on List
- Addresses
- Summarize

Summarize


Data struture in Python
Buil-in Data Strutures
User-Define Data Struture
List
Tuple
Dictionary
Set
Stack
Tree
Graph
Queue
Linked List

Summarize



List

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

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

Strings

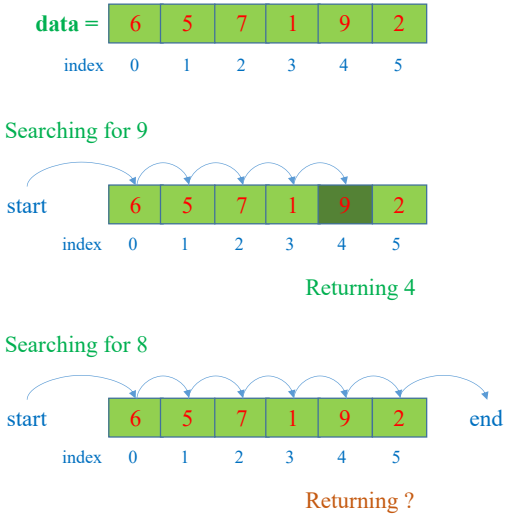
name = ‘A’ ‘I’

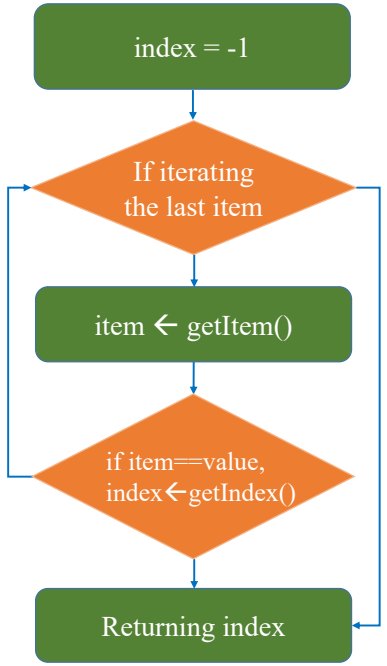
index 0 1

Summarize

Algorithms on List

❖ Linear searching



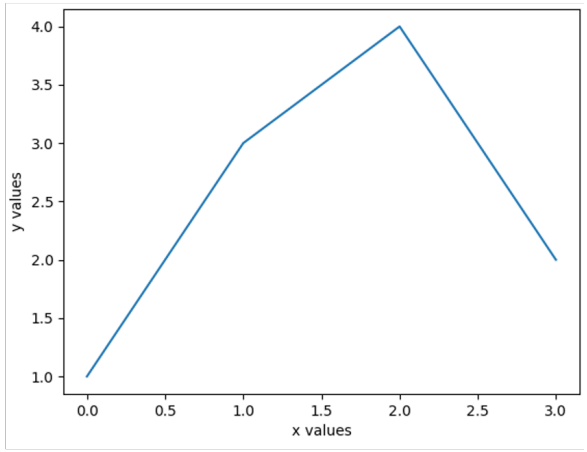


Plotting a function

❖ Using matplotlib

```
1  # ví dụ 1
2  import matplotlib.pyplot as plt
3
4  # data
5  x_data = [0, 1, 2, 3]
6  y_data = [1, 3, 4, 2]
7
8  # plot
9  plt.plot(x_data, y_data)
10 plt.ylabel('y values')
11 plt.xlabel('x values')
12
13 # show
14 plt.show()
```

x-data
y-data

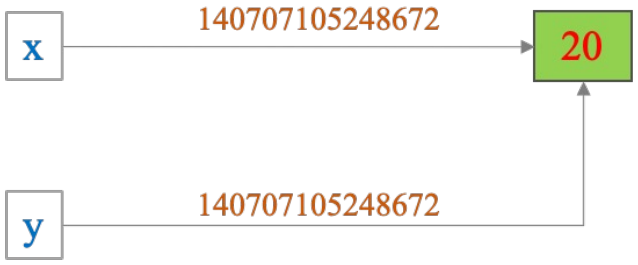


Summarize

Variables and Addresses

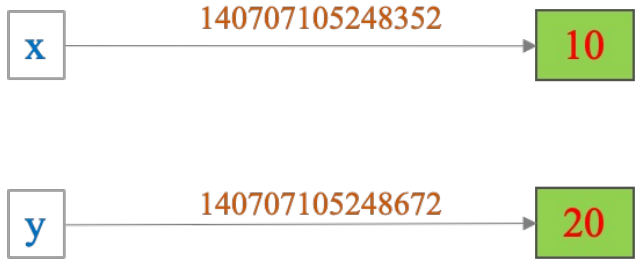
```
1  x = 20
2  y = x
3
4  print(x)
5  print(y)
6  print(id(x))
7  print(id(y))
```

```
20
20
140707105248672
140707105248672
```




```
1  x = 20
2  y = x
3  x = 10
4
5  print(x)
6  print(y)
7  print(id(x))
8  print(id(y))
```

```
10
20
140707105248352
140707105248672
```



Summarize

Read and Write Data to File