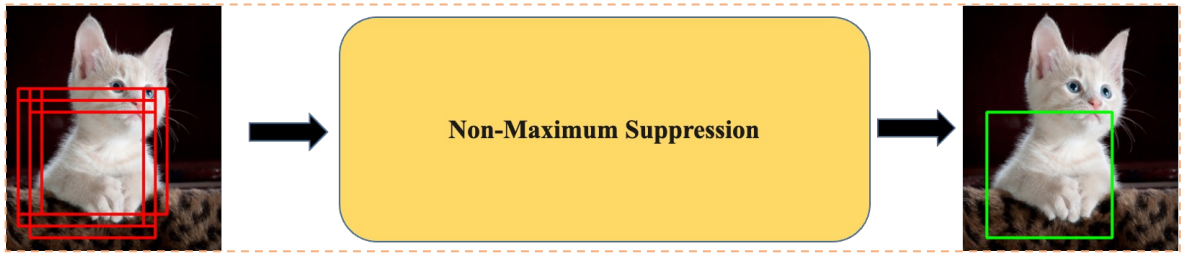
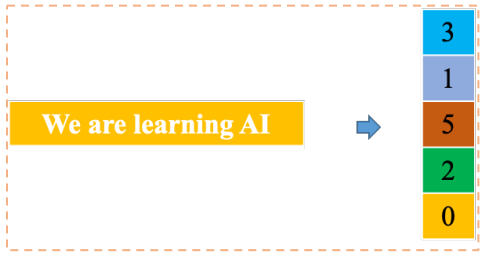
AI VIET NAM
@aivietnam.edu.vn

Data Structure

Tuple, Set and Dictionary



IOU, NMS using Tuple



Text embedding using Set

Histogram using Dictionary

Data & Code

Vinh Dinh Nguyen
PhD in Computer Science

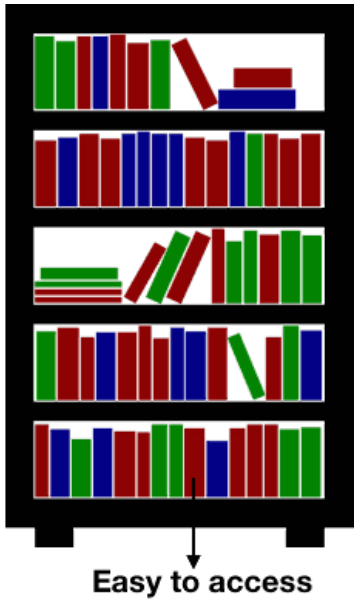
Outline

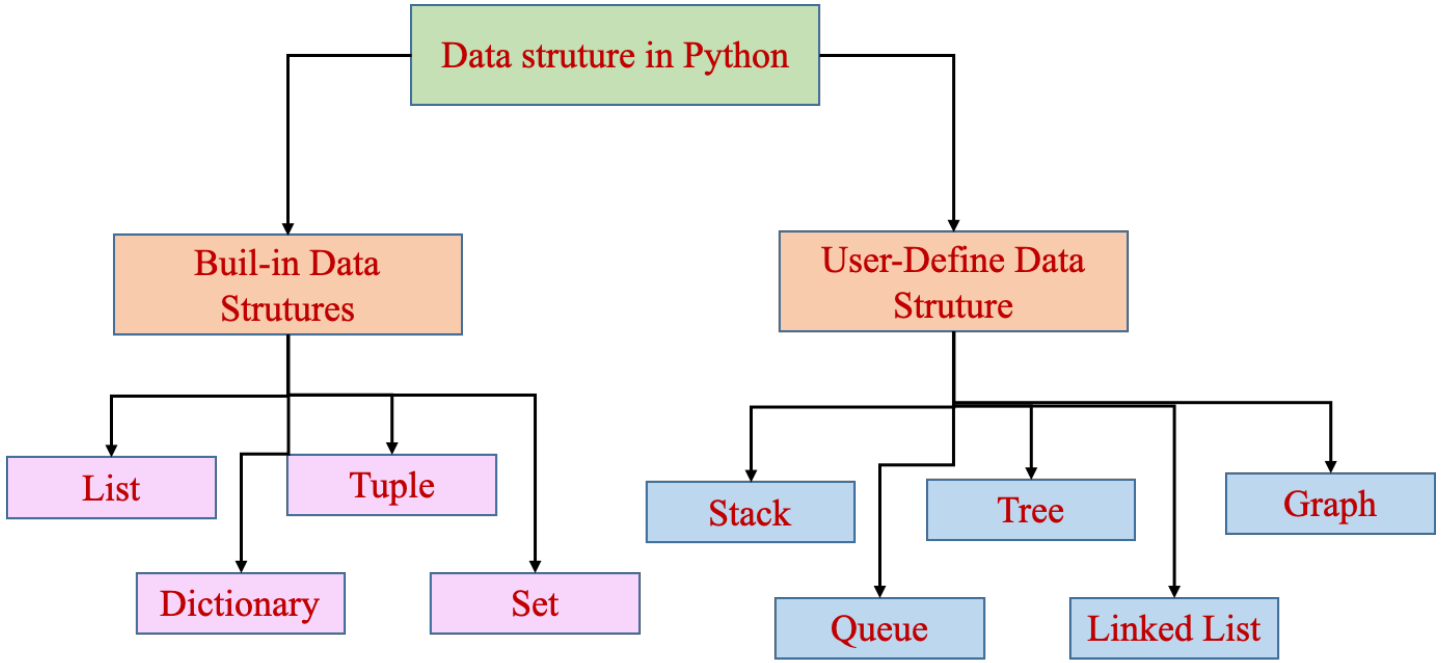
- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

AI VIET NAM
@aivietnam.edu.vn

Data Structure?

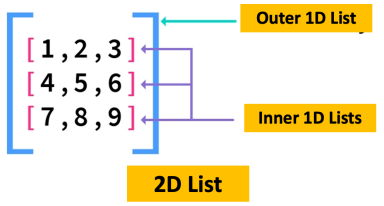
A data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.

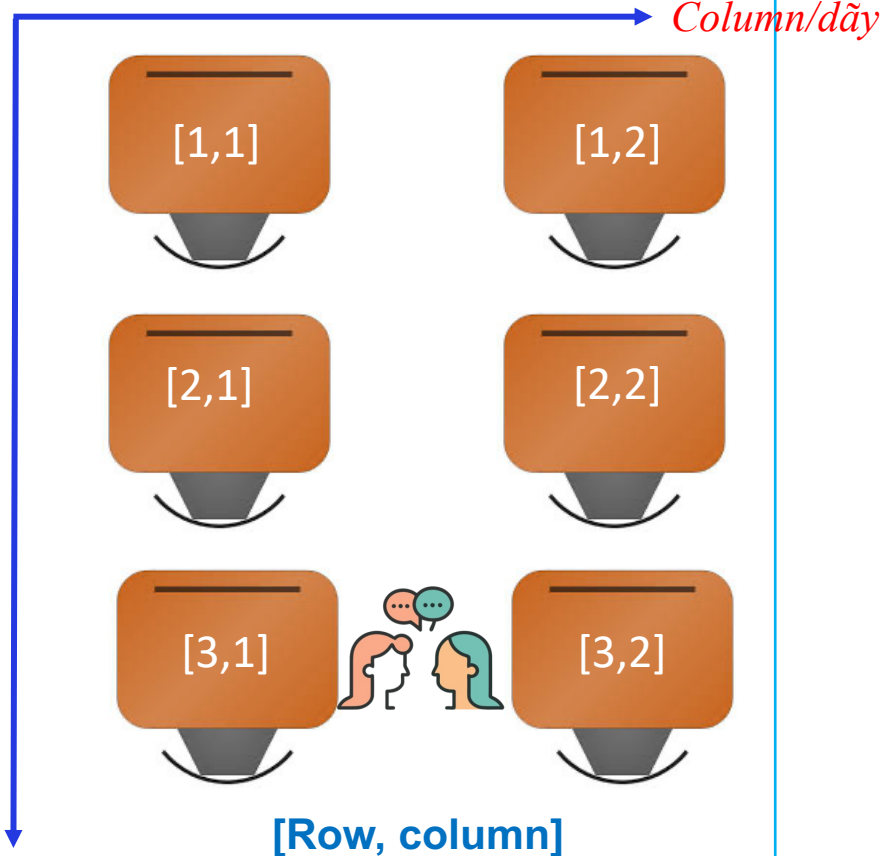




AI VIET NAM
@aivietnam.edu.vn

Review: From 1D to 2D List





Given that the teacher do not know students' names, please develop a program to help her find a student's name based on his/her seat location in the class.

Danh sách sinh viên

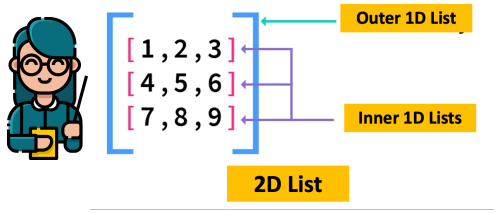
1. **Vinh**
2. **Ân**
3. **Toàn**
4. **Lâm**
5. **Tám**
6. **Chuyện**

"Cô mời 2 bạn: ngồi ở vị trí hàng thứ 3, dãy thứ 1, và bạn ngồi ở vị trí hàng thứ 3, dãy thứ 2" lên bảng giải thích về 2D List

Làm thế nào để biết tên 2 bạn sinh viên này nhỉ?

AI VIET NAM
@aivietnam.edu.vn

Review: From 1D to 2D List



"Cô mời 2 bạn: ngồi ở vị trí hàng thứ 3, dãy thứ 1, và ngồi ở vị trí hàng thứ 3, dãy thứ 2" lên bảng giải thích về 2D List

"Cô mời bạn Tám và bạn Chuyện" lên bảng giải thích về 2D List

Danh sách sinh viên

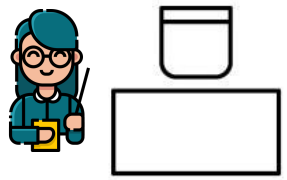
1. **Vinh**
2. **Ân**
3. **Toàn**
4. **Lâm**
5. **Tám**
6. **Chuyện**

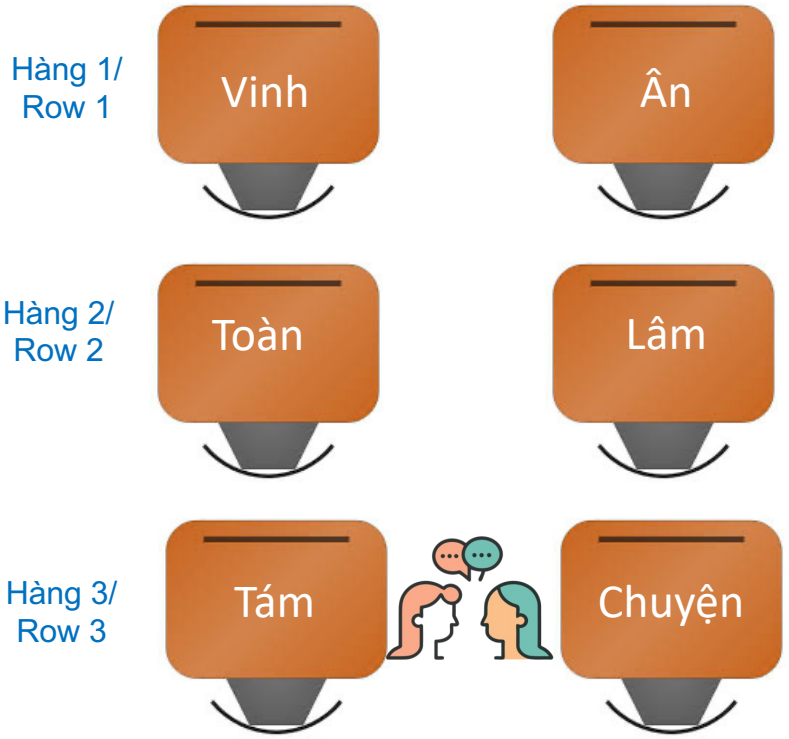
Program

find_student(row,col)

AI VIET NAM
@aivietnam.edu.vn

1D List Solution



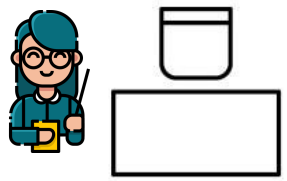


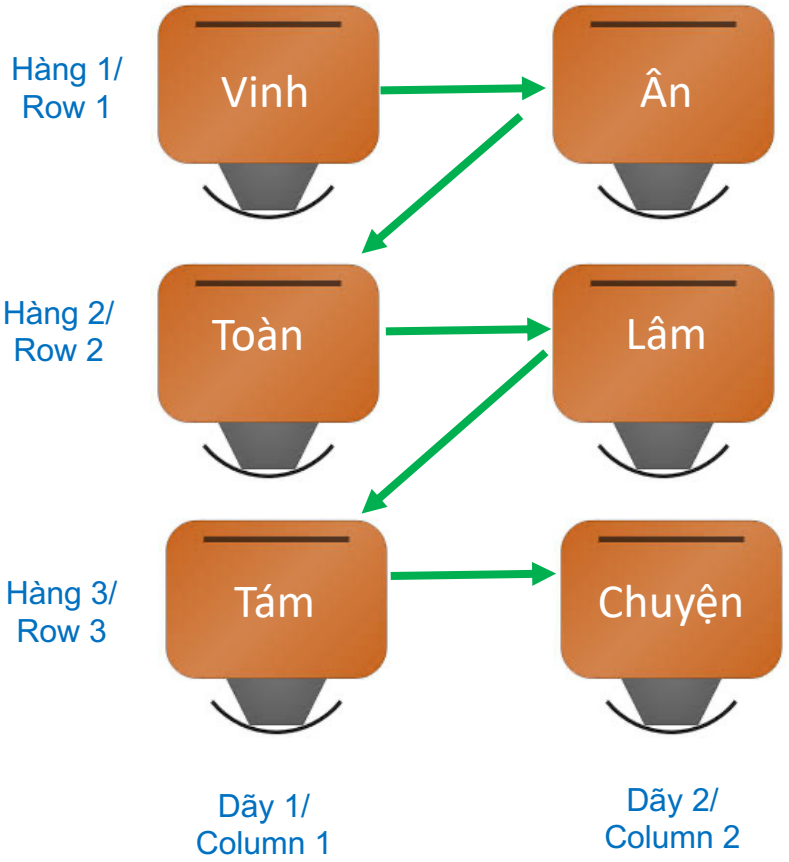

```python
student_list = ["Vinh", "An", "Toan", "Lam", "Tam", "Chuyen"]
row_class = 3
col_class = 2
def find_student(student_list, r, c):
    index = (r-1)*col_class + (c-1)
    return student_list[index]

#Find student a location (3, 1)
print(find_student(student_list, 3, 1))

#Find student a location (3, 2)
print(find_student(student_list, 3, 2))
```

```
Tam
Chuyen
```

Are there any easier solutions? This one is really difficult for newcomers

1D List Solution





student_list = ["Vinh", "An", "Toan", "Lam", "Tam", "Chuyen"]

Vinh	An	Toan	Lam	Tam	Chuyen

Example	Result	Analysis
find_student(student_list, 3, 1)	Tam	Hàng 1: bắt đầu từ index 0 Hàng 2: bắt đầu từ index 2 Hàng 3: bắt đầu từ index. 4
find_student(student_list, 3, 2)	Chuyen	
find_student(student_list, 2, 1)	Toan	
find_student(student_list, row, col)		Hàng 1 có index : (1 – 1)*2 = 0 Hàng 2 có index : (2 – 1)*2 = 2 Hàng 3 có index : (3 – 1)*2 = 4
find_student(student_list, 3, 1)	Tam	**Index = (3 – 1)*2** + (1 – 1) = 4
find_student(student_list, 3, 2)	Chuyen	**Index = (3 – 1)*2** + (2 – 1) = 5
index = (r - 1) * 2+ (c - 1) **index = (r - 1) * columns+ (c - 1)**		

2D List Solution

Prograprogramer View

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Teacher View

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

Use 1D List	Vinh	An
Use 1D List	Toan	Lam
Use 1D List	Tam	Chuyen

student_list = [element1, element2, element3)

AI VIET NAM
@aivietnam.edu.vn

2D List Solution

Teacher View

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Progrogramer View

```
row1_student = ["Vinh", "An"] #1D List
row2_student = ["Toan", "Lam"] #1D List
row3_student = ["Tam", "Chuyen"] #1D List

student_list = [row1_student, row2_student, row3_student] # 2D List
```

How to access element:
row 0 and column 1
row 0 and column 2

row1_retrieve = student_list[0] → Vinh | An

row1_retrieve[0] → Vinh

row1_retrieve[1] → An

student_list[0][0] → Vinh

student_list[0]1] → An

AI VIET NAM
@aivietnam.edu.vn

2D List Solution

	Column 1	Column 2
Row 1	Vinh	An
Row 2	Toan	Lam
Row 3	Tam	Chuyen

Teacher View

	Column index 0	Column index 1
Row index 0	Vinh	An
Row index 1	Toan	Lam
Row index 2	Tam	Chuyen

Prograprogramer View

```python
row1_student = ["Vinh", "An"] #1D List
row2_student = ["Toan", "Lam"] #1D List
row3_student = ["Tam", "Chuyen"] #1D List

student_list = [row1_student, row2_student, row3_student] # 2D List

num_row = len(student_list)
num_col = len(student_list[0])

def find_student(student_list, r, c):
    return student_list[r-1][c-1]

#Find student a location (3, 1)
print(find_student(student_list, 3, 1))

#Find student a location (3, 2)
print(find_student(student_list, 3, 2))
```

```
Tam
Chuyen
```

Thanks! This one is easy to understand

Mutable Vs. Immutable

In Python, everything is treated as an object. Every object has these three attributes:

- Identity – This refers to the address that the object refers to in the computer's memory.
- Type – This refers to the kind of object that is created.
- Value – This refers to the value stored by the object.

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, Dictionaries, Set, …	String, Tuples, …

While ID and Type cannot be changed once it's created, values can be changed for Mutable objects.

AI VIET NAM
@aivietnam.edu.vn

Mutable Vs. Immutable

❖List is Mutable

```
list=['a','b','c','d','e','f','g','h']
print("original list ")
print(list)

 # changing element at index 4 ,i.e., e to hello
list[4]='hello'
print("changed list")
print(list)
```

Output

```
original list
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
changed list
['a', 'b', 'c', 'd', 'hello', 'f', 'g', 'h']
```

❖String is Immutable

```
words = "AI is trending"
words[len(words)-1] = "o"
print(words)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-d66ddedf5312> in <cell line: 2>()
      1 words = "AI is trending"
----> 2 words[len(words)-1] = "o"
      3 print(words)

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

Mutable Vs. Immutable

❖List is Mutable

```
# Create a string and a list
list1 = [1, 2, 3, 4, 5]
list2 = [1, 2, 3, 4, 5]

# Get the memory addresses
list1_address = id(list1)
list2_address = id(list2)

# Print the memory addresses
print(f"The memory address of the list 1 is: {list1_address}")
print(f"The memory address of the list 2 is: {list2_address}")
```

```
The memory address of the list 1 is: 137537838201792
The memory address of the list 2 is: 137538543554816
```

❖String is Immutable

```
str1 = "AIO2024"
str2 = "AIO2024"

# Get the memory addresses
str1_address = id(str1)
str2_address = id(str2)

# Print the memory addresses
print(f"The memory address of the str1 is: {str1_address}")
print(f"The memory address of the str2 is: {str2_address}")
```

```
The memory address of the str1 is: 137537836287088
The memory address of the str2 is: 137537836287088
```

Object Interning In Python

Optimize memory usage

Tuple Motivation

❖ Develop a function to return personal information (multiple values)

```
# Function returning multiple values
def get_person():
    name = "Camedinh"
    age = 2004
    occupation = "Student"
    return [name, age, occupation]

# Test
person = get_person()
person[0] = " Your name has been hacked "
print(person[0], person[1], person[2])
```

List Example


```
Your name has been hacked, 2024, Student
```

❖ Can we prevent this happen?

Tuple Motivation

❖ **Develop a function to return personal information (multiple values)**

```
# Function returning multiple values
def get_person():
    name = "Camedinh"
    age = 2004
    occupation = "Student"
    return (name, age, occupation)

# Unpacking the tuple
person = get_person()
person[0] = "Your name is Hack"
print(person[0], person[1], person[2])
```

Tuple Example

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-22-3693fe48cbb2> in <cell line: 10>()
      8 # Unpacking the tuple
      9 person = get_person()
---> 10 person[0] = "Your name is Hack"
     11 print(person[0], person[1], person[2])

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

❖ **How to use a Tuple**

Tuples Are Like Lists

Tuples are another kind of sequence that functions much like a list - they have elements which are indexed starting at 0

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print(x[2])
Joseph
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

```
>>> for iter in y:
...     print(iter)
...
1
9
2
>>>
```

What is a Tuple

but... Tuples are “immutable”

Unlike a list, once you create a tuple, you cannot alter its contents - similar to a string

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>>[9, 8, 6]
>>>
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback:'str' object does
not support item
Assignment
>>>
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback:'tuple' object does
not support item
Assignment
>>>
```

What is a Tuple

Built-in Functions in List and Tuple

```
>>> l = list()
>>> dir(l)
['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']

>>> t = tuple()
>>> dir(t)
['count', 'index']
```

What is a Tuple

❖ Structure

tuple_name = (element-1, …, element-n)

```
1.  t = (1, 2, 3)
2.
3.  print(t[0])
4.  print(t[1])
5.  print(t[2])
```

```
1
2
3
```

Tuple unpacking

```
1  x1,y1,z1    = ('a','b','c')
2  (x2,y2,z2) = ('a','b','c')
3  print(x1)
4  print(x2)
```

```
a
a
```

What is a Tuple

❖ Structure

tuple_name = (element-1, …, element-n)

( ) can be removed

```
1.  t = 1, 2
2.  print(t)
```

```
(1, 2)
```

Tuple with one element

```
1  var1 = (1 + 2) * 5
2  print(type(var1), ' ', var1)
3
4  var2 = (1)
5  print(type(var2), ' ', var2)
6
7  var3 = (1,)
8  print(type(var3), ' ', var3)
```

```
<class 'int'>   15
<class 'int'>   1
<class 'tuple'>   (1,)
```

AI VIET NAM
@aivietnam.edu.vn

What is a Tuple

❖ + and * operators

```
1.  t1 = (1, 0)
2.  print(t1)
3.
4.  t1 += (2,)
5.  print(t1)
```

```
(1, 0)
(1, 0, 2)
```

```
1.  t = (1,) * 5
```

```
(1, 1, 1, 1, 1)
```

count() - đếm số lần xuất hiện của một giá trị

index() - tìm vị trí xuất hiện của một giá trị

```
1.  t = (1,2,3,1)
2.  count = t.count(1)
3.  index = t.index(2)
4.
5.  print(count)
6.  print(index)
```

```
2
1
```

What is a Tuple

len() - Tìm chiều dài của một tuple

```
1.  t = (1, 2, 3, 4)
2.  len(t)
```

```
4
```

Lấy giá trị min và max của một tuple

```
1.  t = (1, 2, 3, 4, 5)
2.
3.  print(min(t))
4.  print(max(t))
5.  print(sum(t))
```

```
1
5
15
```

Dùng hàm zip() cho tuple

```
1.  t1 = (1, 2, 3, 4, 5)
2.  t2 = ('a', 'b', 'c', 'd', 'e')
3.
4.  print(t1)
5.  print(t2)
6.
7.  t3 = zip(t1, t2)
8.  for x,y in t3:
9.      print(x, y)
```

```
(1, 2, 3, 4, 5)
('a', 'b', 'c', 'd', 'e')
1 a
2 b
3 c
4 d
5 e
```

Sắp xếp các giá trị trong một tuple

```
1.  t = (4, 7, 3, 9, 6)
2.  t_s = sorted(t)
3.  print(t_s)
```

```
[3, 4, 6, 7, 9]
```

Tuple Examples

Swapping two variables

```python
def swap(v1, v2):
    (v2, v1) = (v1, v2)
    return (v1, v2)
```

```python
v1 = 2
v2 = 3
(v1, v2) = swap(v1, v2)

# print
print(v1)
print(v2)
```

```
3
2
```

Memory requirement

```python
# memory comparison
import sys

aList  = [3, 4, 5, 6, 7]
aTuple = (3, 4, 5, 6, 7)

print(sys.getsizeof(aList))
print(sys.getsizeof(aTuple))
```

```
120
80
```

Tuple slicing

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

```
(3, 4, 5)
(5, 4, 3, 2, 1)
```

list2tuple

```python
# convert from list to tuple
aList  = [3, 4, 5, 6, 7]
aTuple = tuple(aList)

print(aTuple)
print(type(aTuple))
```

```
(3, 4, 5, 6, 7)
<class 'tuple'>
```

tuple2list

```python
# convert from tuple to list
aTuple  = (3, 4, 5, 6, 7)
aList = list(aList)

print(aList)
print(type(aList))
```

```
[3, 4, 5, 6, 7]
<class 'list'>
```

AI VIET NAM
@aivietnam.edu.vn

Tuple Examples

❖ Example: Solve quadratic equation

```python
import math

def quadratic_equation(a, b, c):
    '''
    This function aims at solving the quadratic equation
    a, b, c --- three parameters and a =! 0
    '''

    # compute delta
    delta = b*b - 4*a*c

    if delta < 0:
        return ()
    elif delta == 0:
        x = (-b+math.sqrt(delta))/2*a
        return (x,)
    else:
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        return (x1,x2)

```

Case 1: delta<0

```python
result = quadratic_equation(a=5, b=0, c=1)
print(type(result))
print(len(result))
print(result)
```

```
<class 'tuple'>
0
()
```

Case 2: delta>0

```python
result = quadratic_equation(a=5, b=5, c=1)
print(type(result))
print(len(result))
print(result)
```

```
<class 'tuple'>
2
(-0.276393202250021, -0.7236067977499789)
```

Case 3: delta=0

```python
result = quadratic_equation(a=4, b=4, c=1)
print(type(result))
print(len(result))
print(result)
```

```
<class 'tuple'>
1
(-8.0,)
```

Data is protected

```python
result = quadratic_equation(a=4, b=4, c=1)
result[0] = 1
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most
<ipython-input-21-55f394e5ddc8> in <module>
      1 result = quadratic_equation(a=4, b=4, c=1)
----> 2 result[0] = 1
```

Outline

- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

Set Motivation

List

1	0	2	1	0	2

Set

1	3	2	4	5	6

Sets, unlike lists or tuples, cannot have multiple occurrences of the same element and store unordered values.

Set

❖ Create a set

Using curly brackets

```
1  # create a set
2  animals = {"cat", "dog", "tiger"}
3
4  print(type(animals))
5  print(animals)
```

```
<class 'set'>
{'dog', 'cat', 'tiger'}
```

Items with different data types

```
1  # create a set
2  a_set = {"cat", 5, True, 40.0}
3
4  print(type(a_set))
5  print(a_set)
```

```
<class 'set'>
{40.0, 'cat', 5, True}
```

Set comprehension

```
1  # set comprehension
2
3  a_set = {i*i for i in range(10)}
4  print(a_set)
```

```
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

Set

Access the items of a set

```
1  # accessing items
2  animals = {"cat", "dog", "tiger"}
3  for animal in animals:
4      print(animal)
```

```
dog
cat
tiger
```

Copy a set

```
1  # copy
2  animals = {"cat", "dog", "tiger"}
3  print("Animals:", animals)
4
5  a_copy = animals.copy()
6  print("Copy:", a_copy)
```

```
Animals: {'dog', 'cat', 'tiger'}
Copy: {'dog', 'cat', 'tiger'}
```

Set

Add an item

```python
# add an item
animals = {"cat", "dog", "tiger"}
animals.add("bear")
print(animals)
```

```
{'dog', 'bear', 'cat', 'tiger'}
```

Insert a set to another set

```python
# insert a set to another set
animals = {"cat", "dog", "tiger"}
animals.update({"chicken","Duck"})
print(animals)
```

```
{'Duck', 'tiger', 'dog', 'cat', 'chicken'}
```

Join two sets

```python
# join two sets
set1 = {"cat", "dog"}
set2 = {"duck", "tiger"}

set3 = set1.union(set2)
print(set3)
```

```
{'duck', 'dog', 'cat', 'tiger'}
```

Not allow duplicate values

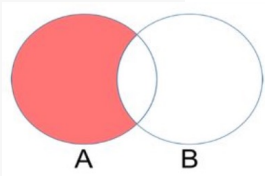
```python
# No duplication
animals = {"cat", "dog", "tiger"}
print(animals)

animals.add("cat")
print(animals)
```

```
{'tiger', 'cat', 'dog'}
{'tiger', 'cat', 'dog'}
```

Set

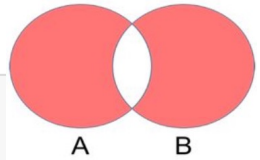
difference function

```
1  # difference
2
3  set1 = {"apple", "banana", "cherry"}
4  set2 = {"pineapple", "apple"}
5
6  set3 = set1.difference(set2)
7
8  print(set3)
```




```
{'cherry', 'banana'}
```

symmetric_difference




```
1  # symmetric_difference
2
3  set1 = {"apple", "banana", "cherry"}
4  set2 = {"pineapple", "apple"}
5
6  set3 = set1.symmetric_difference(set2)
7
8  print(set3)
```

```
{'pineapple', 'cherry', 'banana'}
```

difference_update function

```
1  # difference_update
2
3  set1 = {"apple", "banana", "cherry"}
4  set2 = {"pineapple", "apple"}
5
6  set1.difference_update(set2)
7
8  print(set1)
```

```
{'cherry', 'banana'}
```

symmetric_difference_update

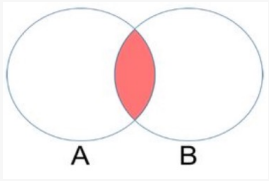
```
1  # symmetric_difference_update
2
3  set1 = {"apple", "banana", "cherry"}
4  set2 = {"pineapple", "apple"}
5
6  set1.symmetric_difference_update(set2)
7
8  print(set1)
```

```
{'pineapple', 'cherry', 'banana'}
```

Set

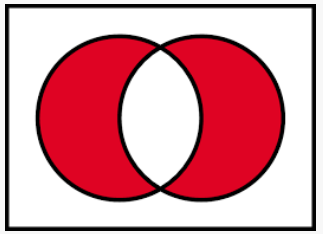
❖ Bitwise operator

```
1  # AND (&)
2
3  set1 = {1, 2, 3}
4  set2 = {3, 4, 5}
5
6  print(set1 & set2)
```



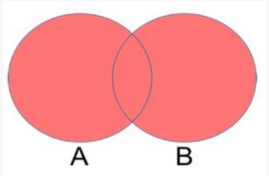

```
{3}
```

```
1  # XOR (^)
2
3  set1 = {1, 2, 3}
4  set2 = {3, 4, 5}
5
6  print(set1 ^ set2)
```



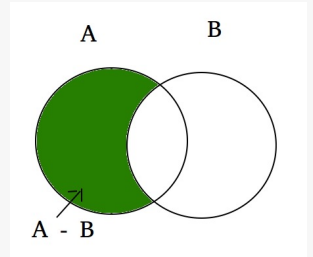
```
{1, 2, 4, 5}
```

```
1  # OR (|)
2
3  set1 = {1, 2, 3}
4  set2 = {3, 4, 5}
5
6  print(set1 | set2)
```




```
{1, 2, 3, 4, 5}
```

```
1  # subtraction (-)
2
3  set1 = {1, 2, 3}
4  set2 = {3, 4, 5}
5
6  print(set1 - set2)
```




```
{1, 2}
```

Set

❖ Remove an item

remove(item)

Remove an item from the set.

```
# remove an item
animals = {"cat", "dog", "tiger"}
animals.remove("dog")
print(animals)
```

```
{'cat', 'tiger'}
```

discard(item)

Remove an item from the set if it is present.

```
# remove an item
animals = {"cat", "dog", "tiger"}
animals.discard("tiger")
print(animals)
```

```
{'dog', 'cat'}
```

Set comprehension

```
# set comprehension

aSet = {i*i for i in range(10)}
print(aSet)
```

```
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```

https://docs.python.org/3/library/stdtypes.html?t#set

Set

❖ Remove an item

remove(item)

Remove an item from the set.

Raises KeyError if elem is not contained in the set.

```
1 # remove an item
2 animals = {"cat", "dog", "tiger"}
3 animals.remove("duck")
4 print(animals)
```

```
---------------------------------------
KeyError
<ipython-input-29-604e724f3515> in <module
      1 # remove an item
      2 animals = {"cat", "dog", "tiger"}
----> 3 animals.remove("duck")
      4 print(animals)

KeyError: 'duck'
```

discard(item)

Remove an item from the set if it is present.

```
1 # remove an item
2 animals = {"cat", "dog", "tiger"}
3 animals.discard("duck")
4 print(animals)
```

```
{'dog', 'cat', 'tiger'}
```

https://docs.python.org/3/library/stdtypes.html?t#set

Set

❖ Create a set

Unordered and unindexed

```
1  # not support indexing
2  animals = {"cat", "dog", "tiger"}
3  print(animals[1])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-24-a1b489f88deb> in <module>
      1 # not support indexing
      2 animals = {"cat", "dog", "tiger"}
----> 3 print(animals[1])

TypeError: 'set' object is not subscriptable
```

Cannot contain unhashable types

```
1  # create a set
2  a_list = [1, 2, 3]
3  a_set  = {"cat", a_list}
4  print(a_set)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-5-35efb5d3ad33> in <module>
      1 # shallow copy
      2 a_list = [1, 2, 3]
----> 3 a_set  = {"cat", a_list}
      4 print(a_set)

TypeError: unhashable type: 'list'
```

Set

Set ←→ List and Tuple

```python
# convert from set to list
aSet = {1, 2, 3, 4, 5}

aList = list(aSet)
print(aList)
print(type(aList))
```

```
[1, 2, 3, 4, 5]
<class 'list'>
```

```python
# convert from list to set
aList = [1, 2, 3, 2, 1]

aSet = set(aList)
print(aSet)
print(type(aSet))
```

???

```python
# convert from set to tuple
aSet = {1, 2, 3, 4, 5}

aTuple = tuple(aSet)
print(aTuple)
print(type(aTuple))
```

```
(1, 2, 3, 4, 5)
<class 'tuple'>
```

```python
# convert from tuple to set
aTuple = (1, 2, 3, 2, 1)

aSet = set(aTuple)
print(aSet)
print(type(aSet))
```

???

Q&A Time

Outline

- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

Dictionary Motivation

number_list = [11, 15, 35, 20, 18, 9]

11	15	35	20	18	9
0	1	2	3	4	5

Index

index

value

number_list[0] => 11

It's very hard to remember that the 1st element in a list has index 0, is there an easier way to remember this?

number_dict ={"One":11, "Two":15, "Three":35, "Four":20, "Five":18, "Six:9" }

11	15	35	20	18	9
"One	Two	Three	Four	Five	Six

key

key

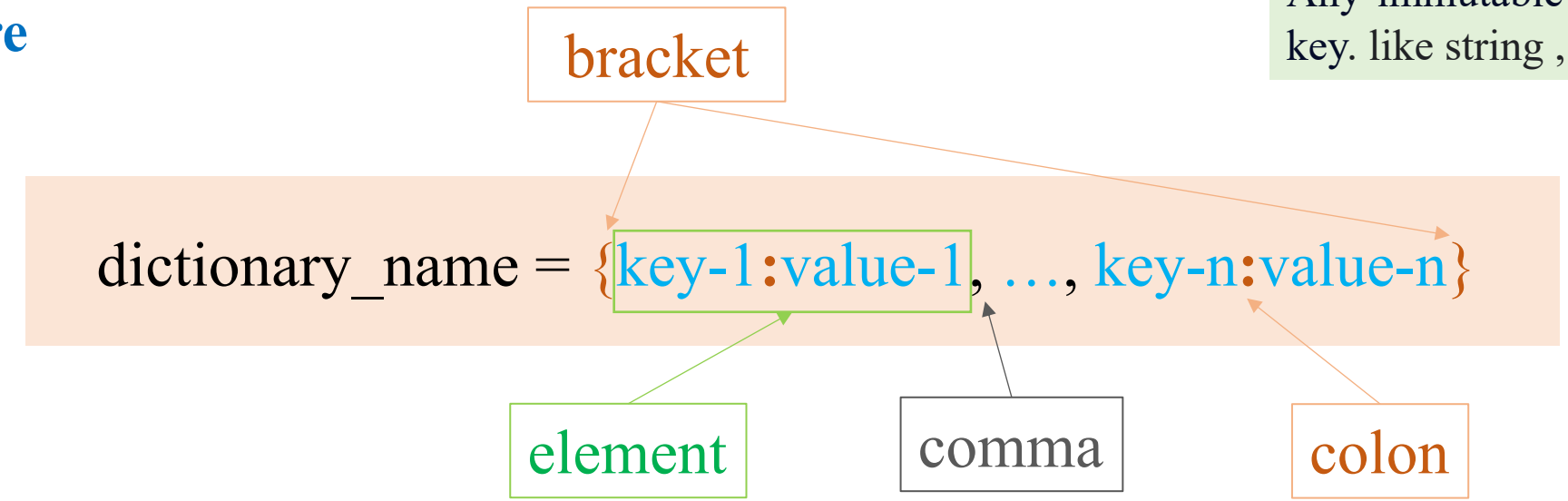
value

number_dict["One"] => 11

Dictionary

❖ Structure

Any immutable can be dictionary key. like string , number, tuple



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}
```

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

```
1  parameters = {'learning_rate': 0.1,
2                'metric': 'Accuracy'}
3
4  parameters['learning_rate'] = 0.2
5  print(parameters)
```

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

❖ Copy a dictionary

```
1  parameters = {'learning_rate': 0.1,
2                'metric': 'Accuracy'}
3
4  a_copy = parameters.copy()
5  a_copy['learning_rate'] = 0.2
6
7  print(parameters)
8  print(a_copy)
```

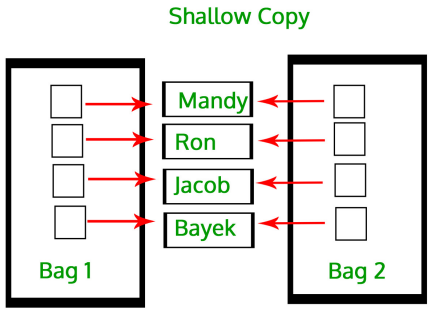
```
{'learning_rate': 0.1, 'metric': 'Accuracy'}
{'learning_rate': 0.2, 'metric': 'Accuracy'}
```

Dictionary

❖ Hàm copy() chỉ sao chép kiểu shallow

```
1.  d1 = {'a': [1,2], 'b': 5}
2.  d2 = d1.copy()
3.
4.  # thay đổi giá trị d2 sẽ ảnh hưởng đến d1
5.  d2['a'][0] = 3
6.  d2['a'][1] = 4
7.
8.  print('d1:', d1)
9.  print('d2:', d2)
```

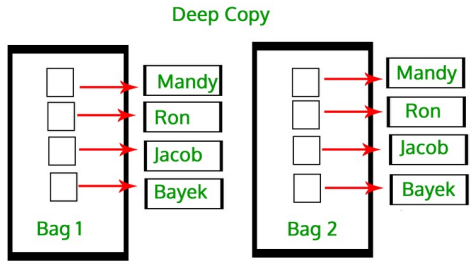
```
d1: {'a': [3, 4], 'b': 5}
d2: {'a': [3, 4], 'b': 5}
```



❖ Sử dụng hàm deepcopy() trong module copy

```
1.  import copy
2.
3.  d1 = {'a': [1,2], 'b': 5}
4.  d2 = copy.deepcopy(d1)
5.
6.  # thay đổi giá trị d2
7.  d2['a'][0] = 3
8.  d2['a'][1] = 4
9.
10. print('d1:', d1)
11. print('d2:', d2)
```

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



Dictionary

❖ Get keys and values

Get keys

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

```
learning_rate
optimizer
metric
```

Get values

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

```
0.1
Adam
Accuracy
```

Get keys

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

```
learning_rate
optimizer
metric
```

Get keys and values

```
1  parameters = {'learning_rate': 0.1,
2                'optimizer': 'Adam',
3                'metric': 'Accuracy'}
4
5  items = parameters.items()
6  for key, value in items:
7      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

```python
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

```python
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

```python
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
<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}
```

Q & A

```
1 def greet(name, age):
2     print(f"{name} is {age} years old.")
3
4 info = {"name": "Alice", "age": 30}
5 greet(**info)  #
6
```

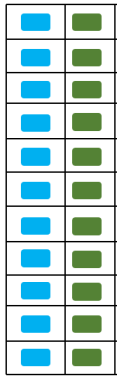
Outline

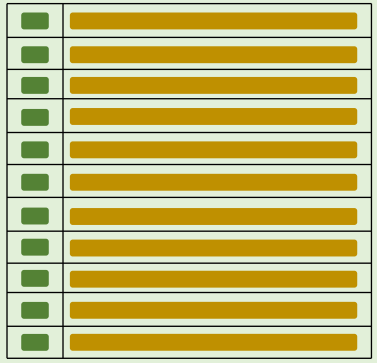
- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

AI VIET NAM
@aivietnam.edu.vn

Set in Text Classification

❖ Text classification



Text Sample → Preprocessing (standardization & vectorization) →  → Machine Learning Algorithm →  Output

”Make money” → Preprocessing → Embedding → Fully-connected layer

 → 0 1 2 4 5 6 7 4 8 → Embedding features → Support Vector Machine → Spam

AI VIET NAM
@aivietnam.edu.vn

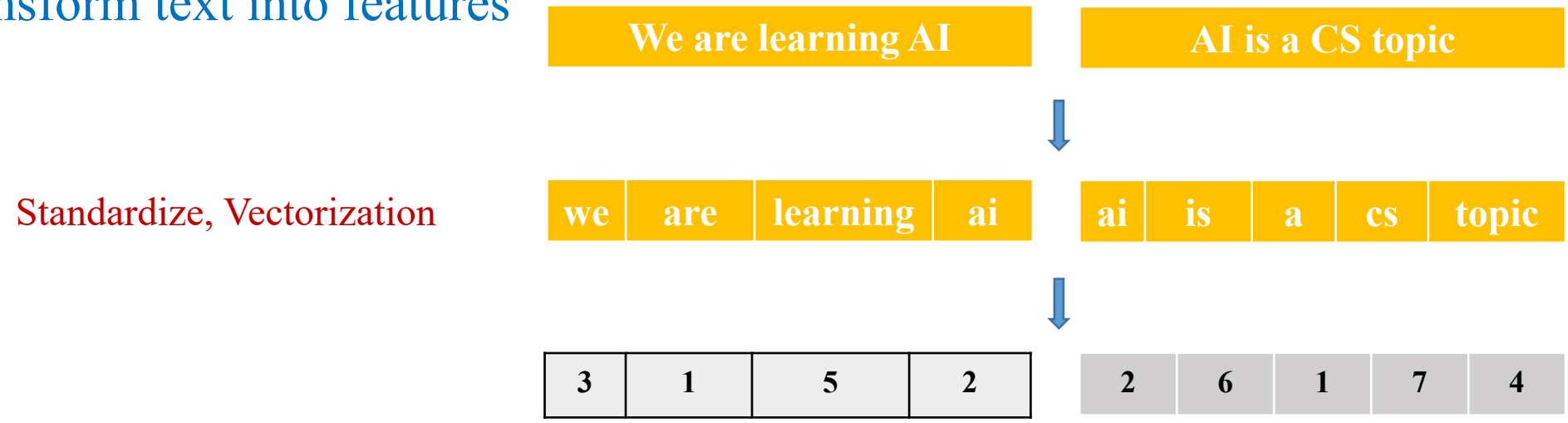
Set in Text Classification

- Example corpus
  - sample1: 'We are learning AI'
  - sample2: 'AI is a CS topic'

(1) Build vocabulary from corpus

index	0	1	2	3	4	5	6	7
word	pad	are/a	ai	we	topic	learning	is	cs

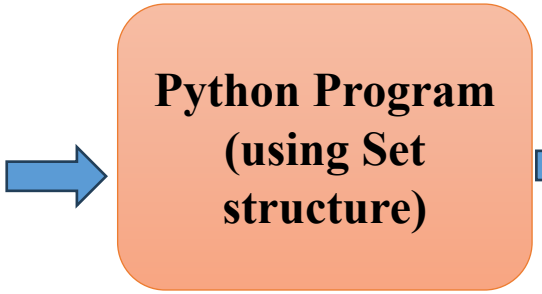
(2) Transform text into features

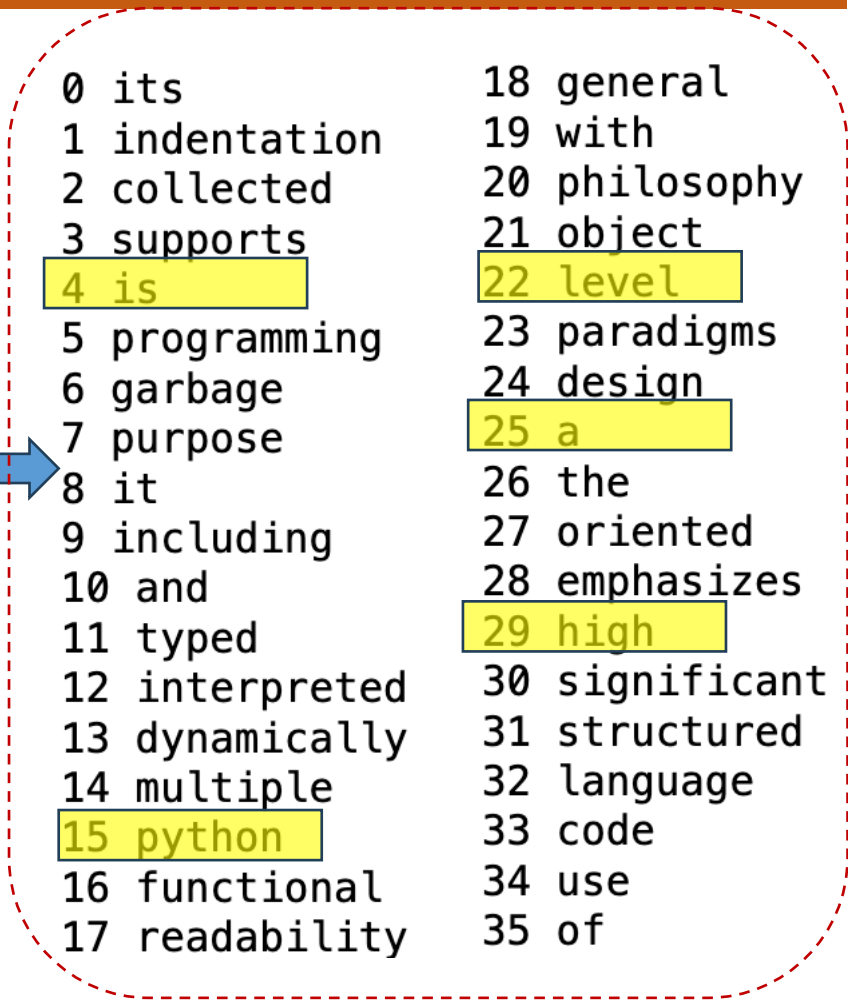


Set in Text Classification

❖Vocabulary Building

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.





Vocabulary

Input

"Python is a high-level"

Text data

Output

15 4 25 29 22

Numerical Data

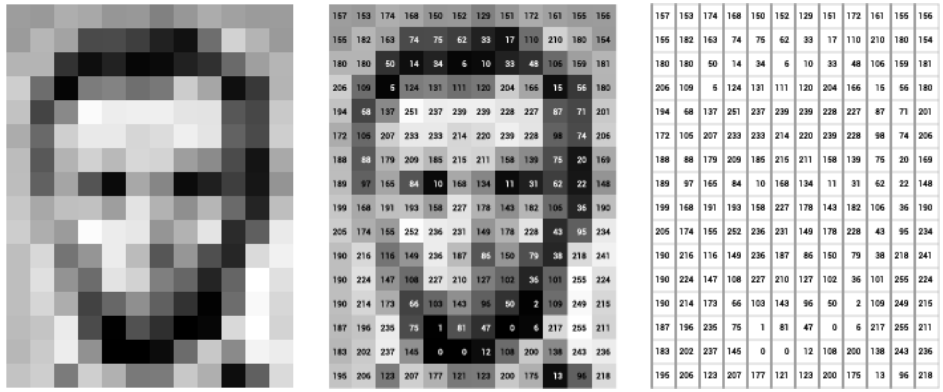
Outline

- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

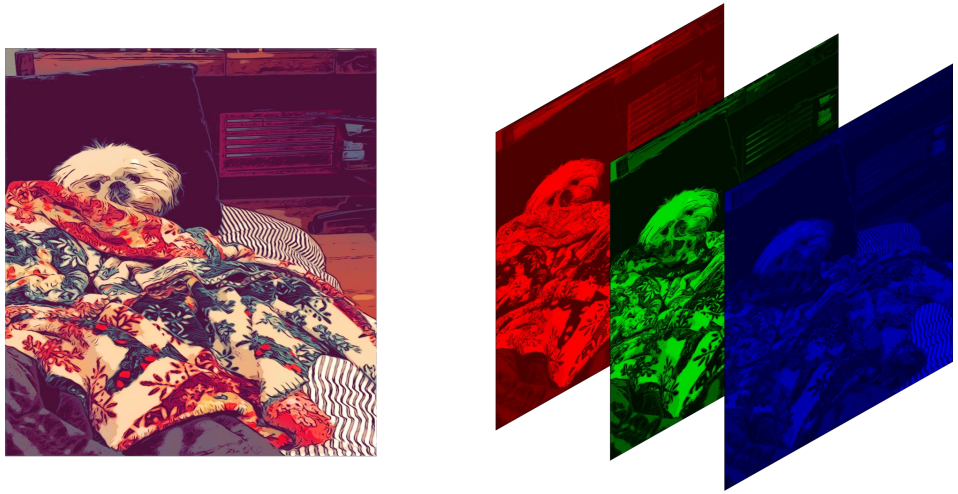
Outline

- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

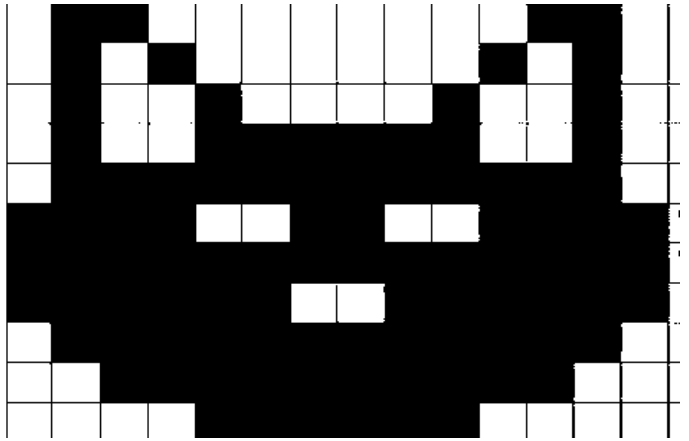
Image in Computer Vision



Grayscale image



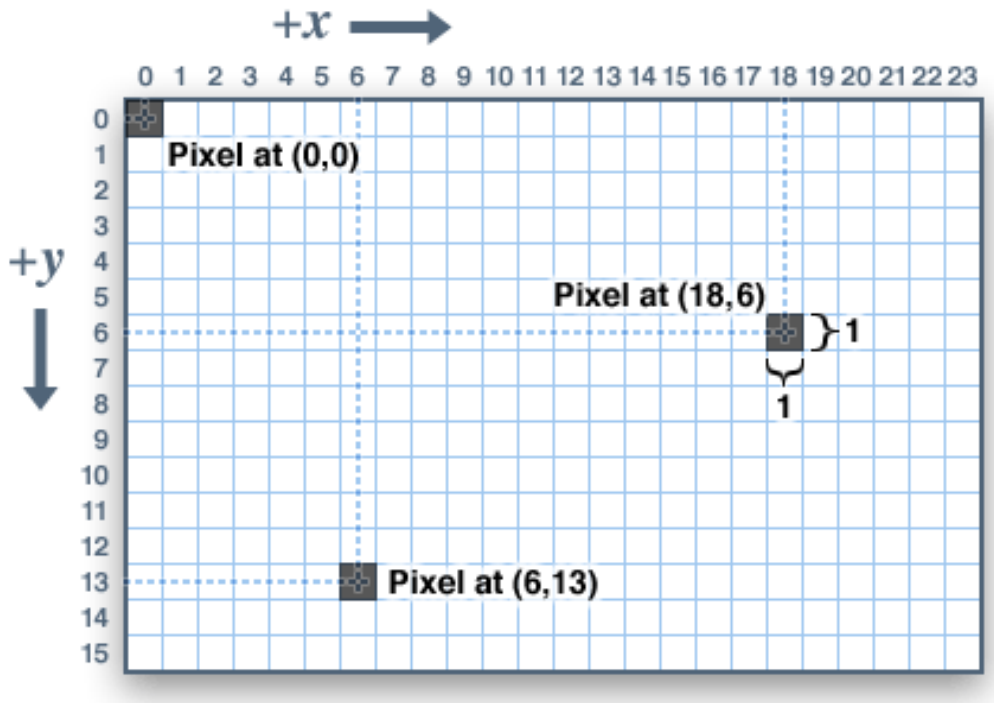
Color image

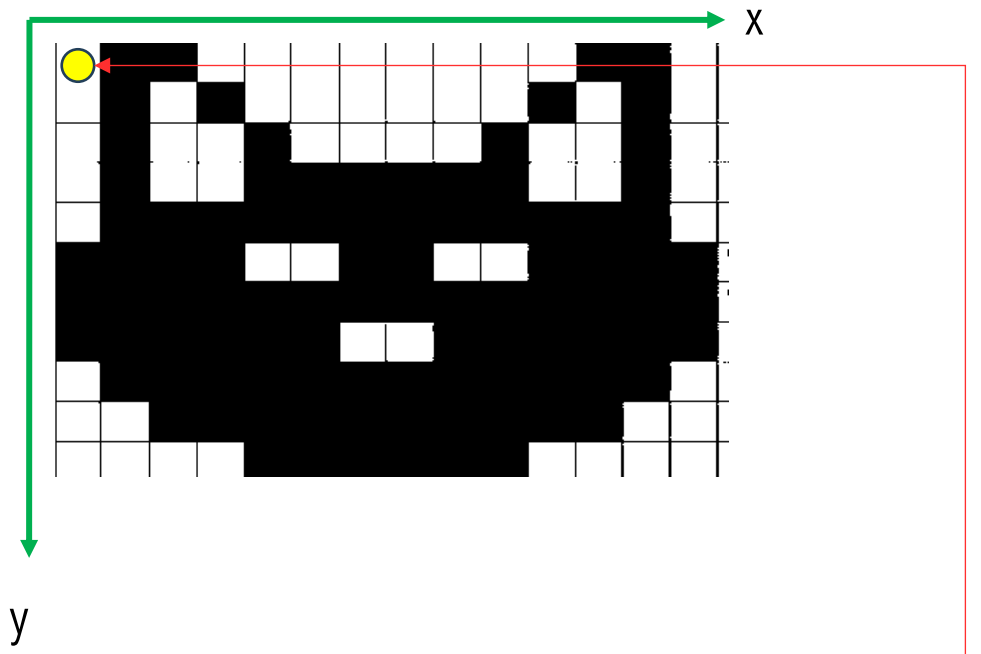


Binary Image

AI VIET NAM
@aivietnam.edu.vn

Image Coordinate System: Grayscale Image





Read image

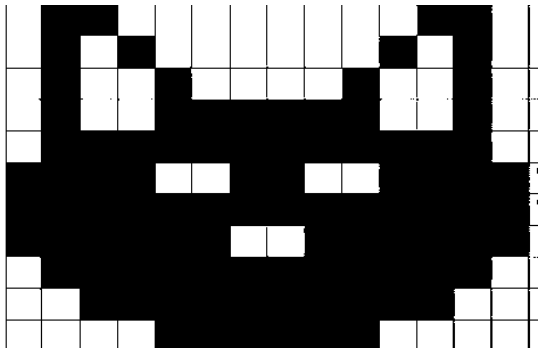
```
import cv2
from google.colab.patches import cv2_imshow

cat_image = cv2.imread("/content/binary_cat.png",0)
cv2_imshow(cat_image)
```

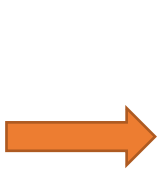
Access pixel at a specific location

```
x = 0
y = 0
pixel_value = cat_image[0][0]
print(pixel_value)
```

```
255
```

Image Histogram: Dictionary





Histogram Algorithm

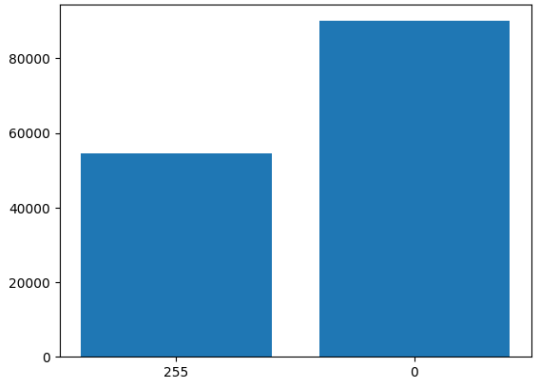
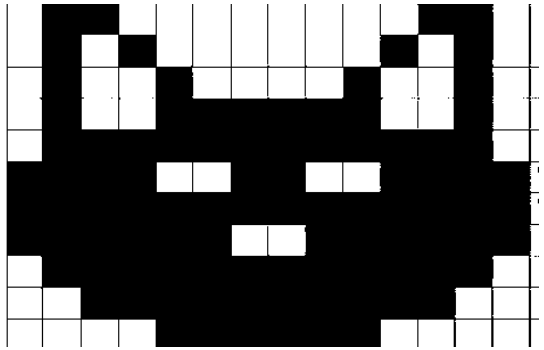
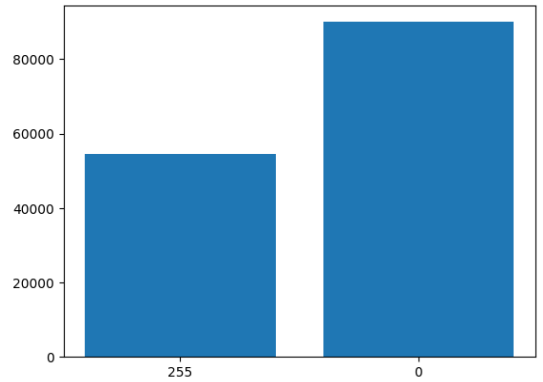


Image Histogram: Dictionary



Histogram Algorithm

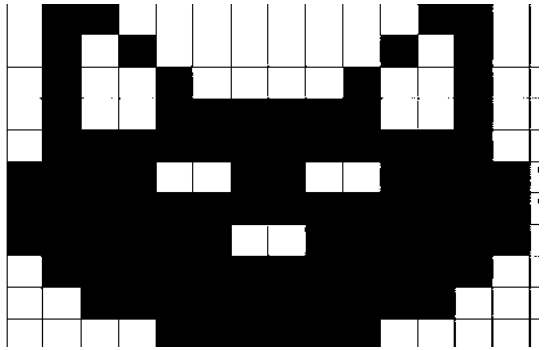


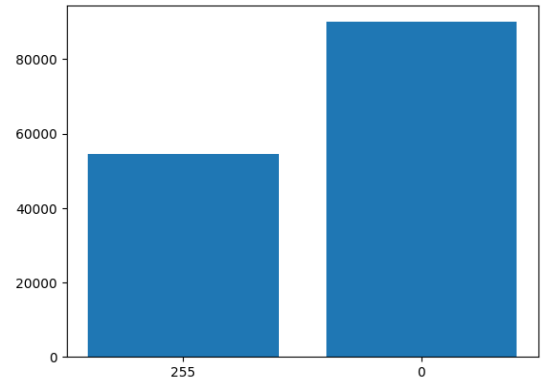
255	0	255	255	255	0	255
255	0	0	255	255	0	255
255	0	0	0	0	0	255
0	0	255	255	255	0	0
0	0	255	255	255	0	0
255	0	255	255	255	0	255
255	255	0	0	0	255	255

```
cat_image = cv2.imread("/content/binary_cat.png",0)
cv2_imshow(cat_image)
```

AI VIET NAM
@aivietnam.edu.vn

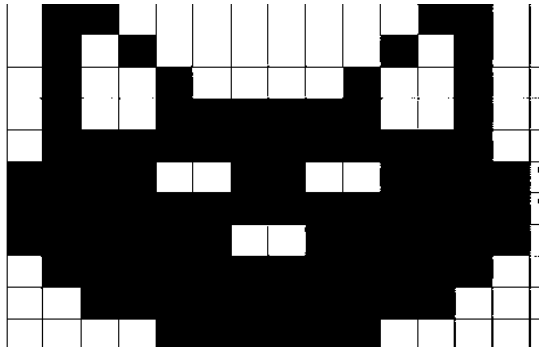
Image Histogram: Dictionary



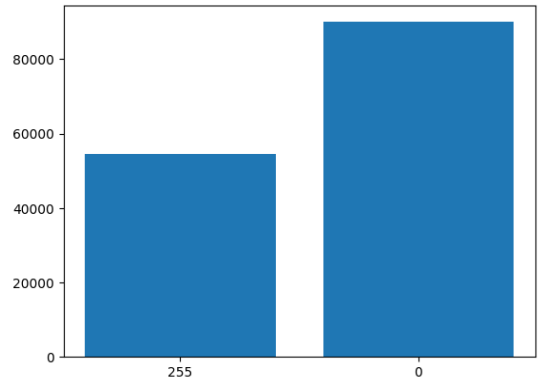



```
counts = dict()
height, width = cat_image.shape
for row in range(height):
  for col in range(width):
      counts[cat_image[row,col]] = counts.get(cat_image[row,col],0) + 1
```

Image Histogram: Dictionary



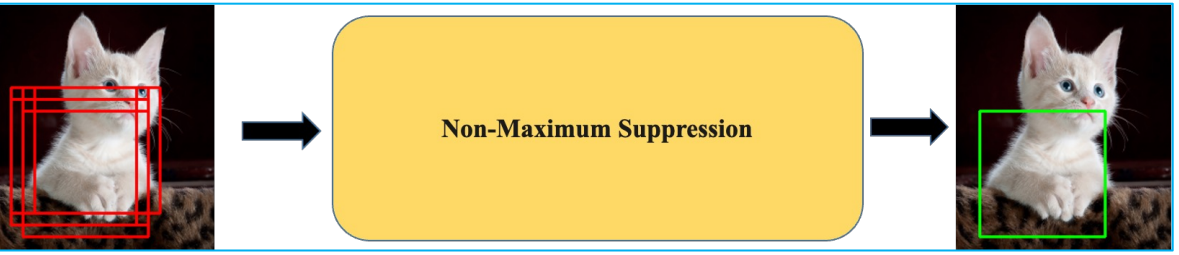
Histogram Algorithm



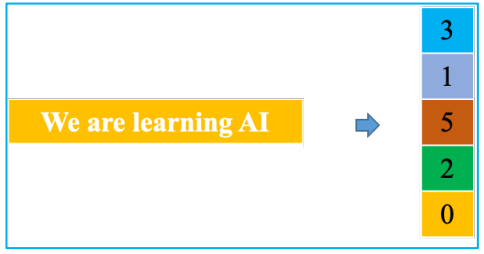

```
import matplotlib.pyplot as plt

names = list(counts.keys())
values = list(counts.values())

plt.bar(range(len(counts)), values, tick_label=names)
plt.show()
```

Summary

Data Structure	Ordered	Mutable	Constructor	Example
List	Yes	Yes	[] or list()	[5. 7, ‘yes’, 5.7]
Tuple	Yes	No	() or tuple()	(5.7, ‘yes’, 5.7)
Set	No	Yes	{} or set()	{5.7, ‘yes’ }
Dictionary	No	Yes	{} or dict{}	{‘key’: value}



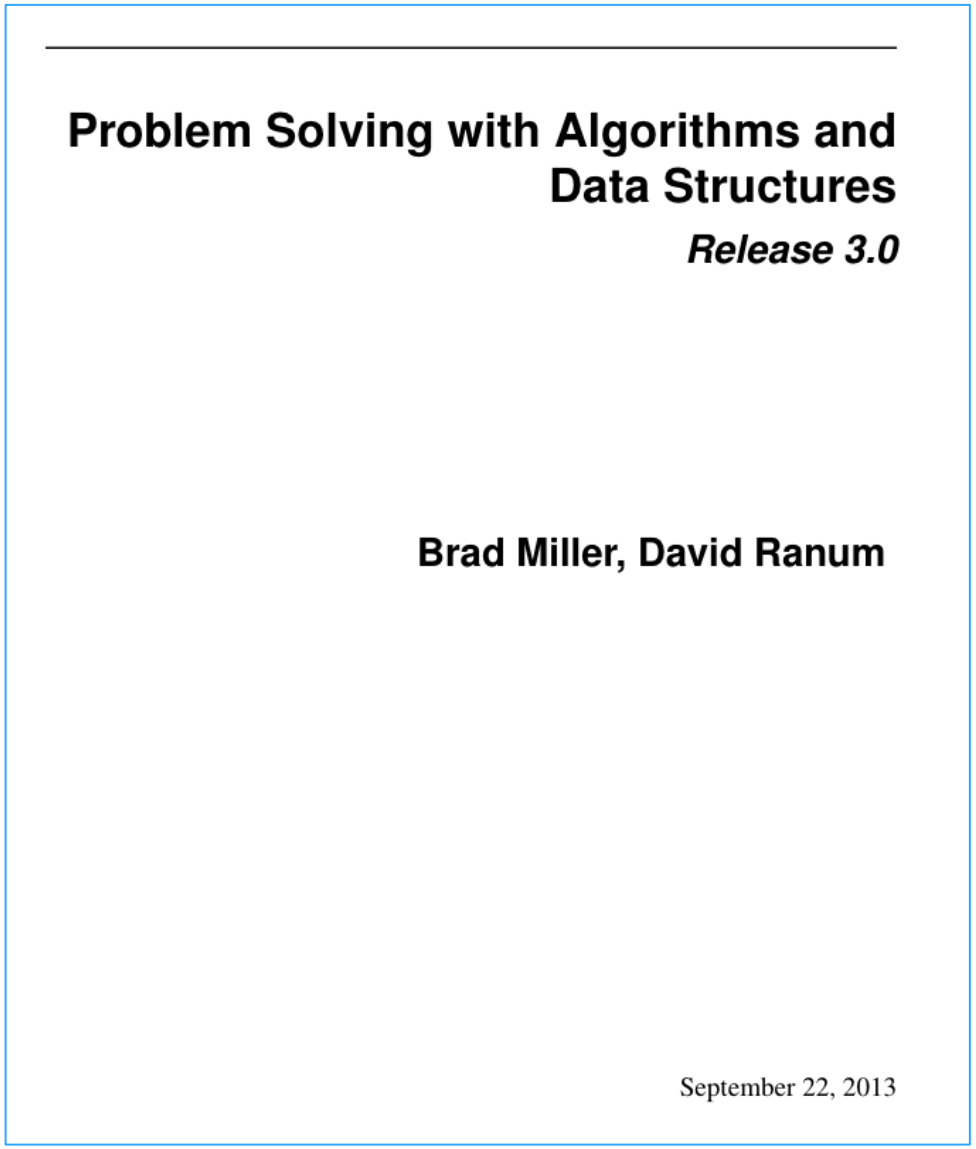
IOU, NMS using Tuple

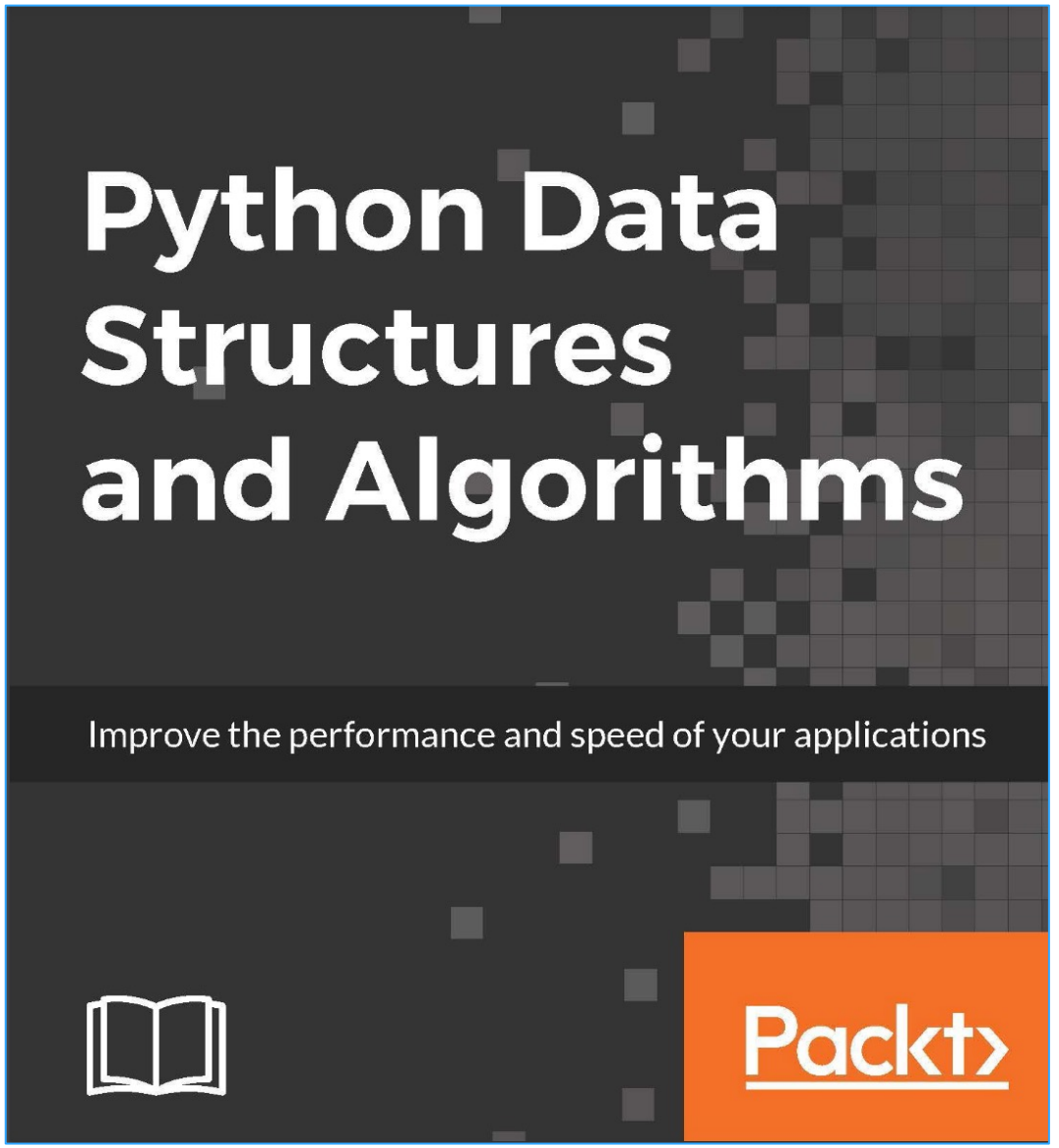


Text embedding using Set

Histogram using Dictionary

References

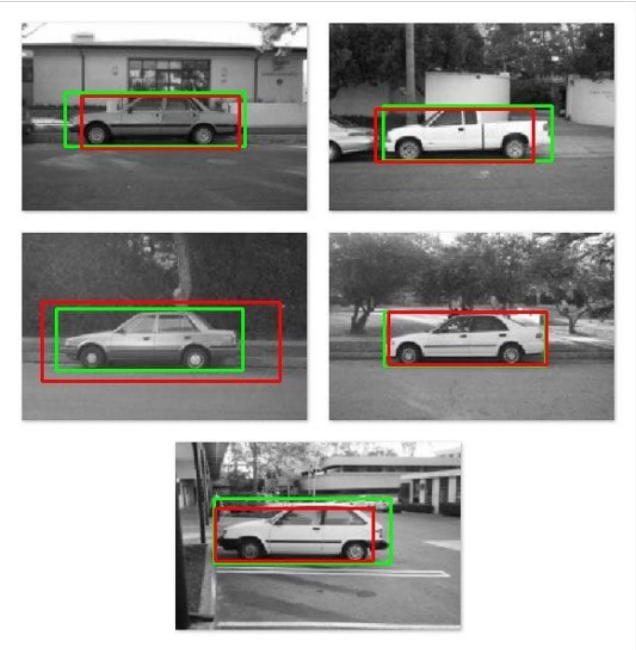

Problem Solving with Algorithms and Data Structures
Release 3.0
Brad Miller, David Ranum
September 22, 2013


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

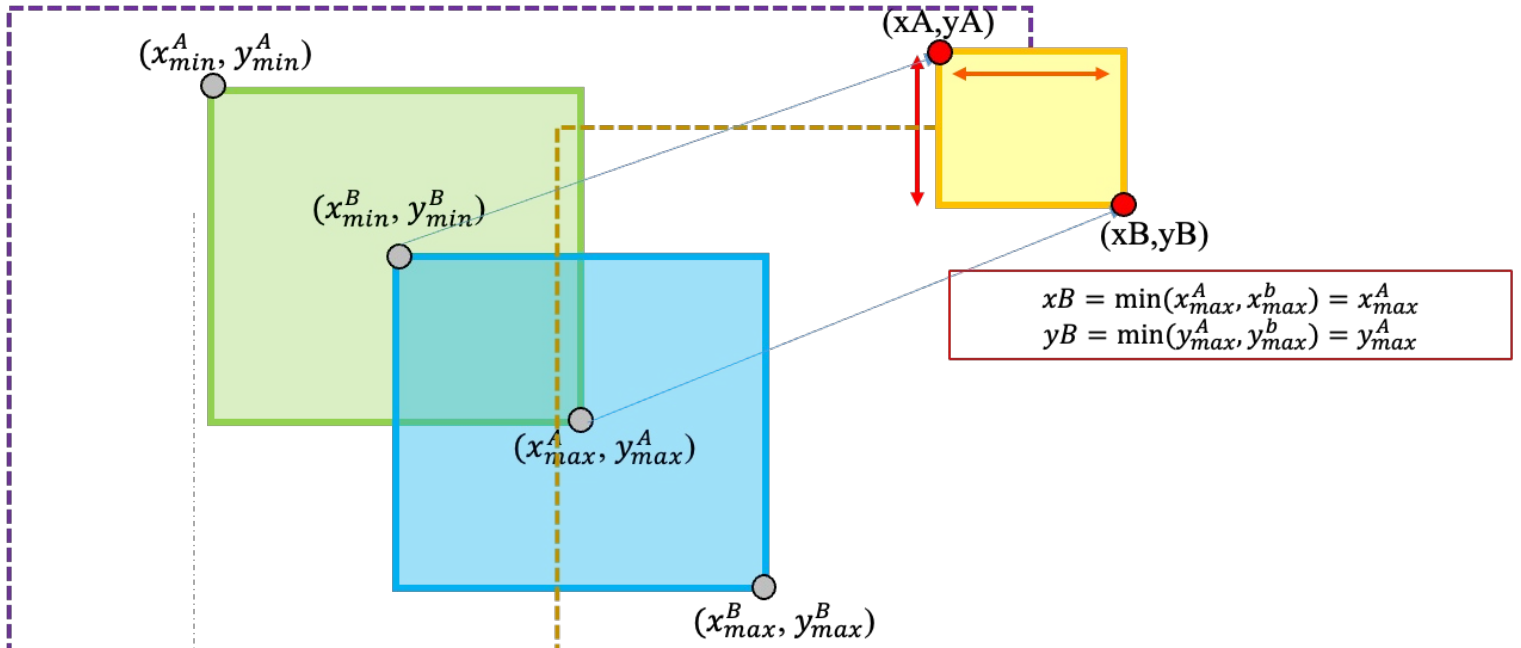
Outline

- What is a Tuple in Python
- What is a Set in Python
- What is a Dictionary in Python
- Quizz
- Text Classification using Set
- Image Histogram using Dictionary
- Further Reading

IOU Calculation Using Tuple



$$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$$
$$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$$

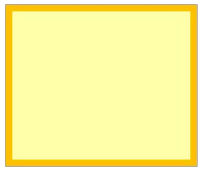


$$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$$
$$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$$

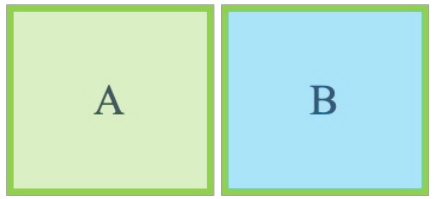
$W = xB - xA$

$H = yB - yA$

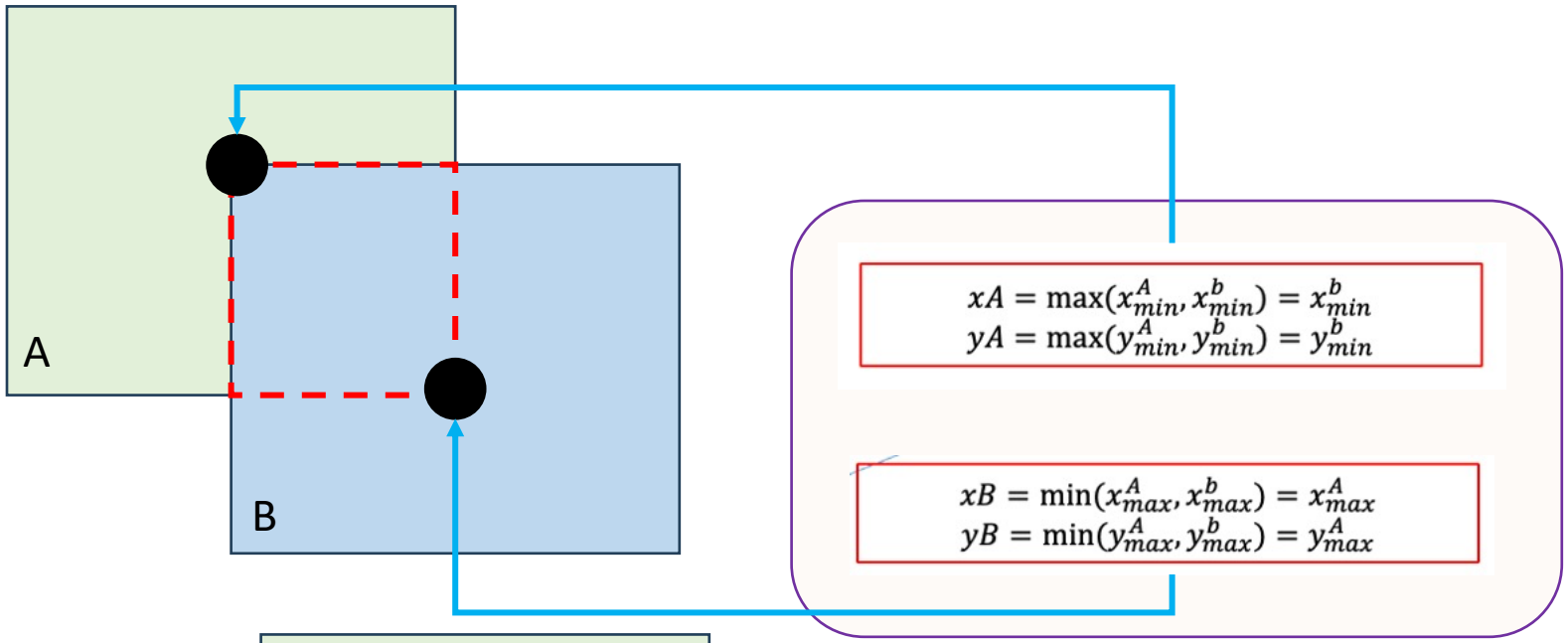
$A \cap B = W * H$

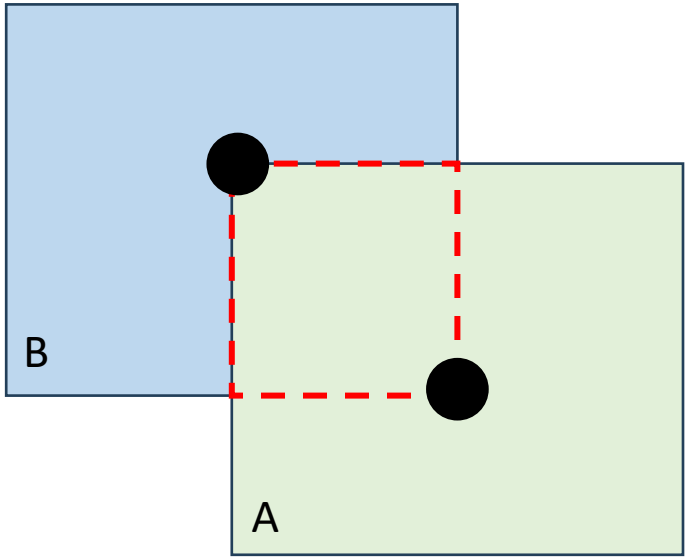


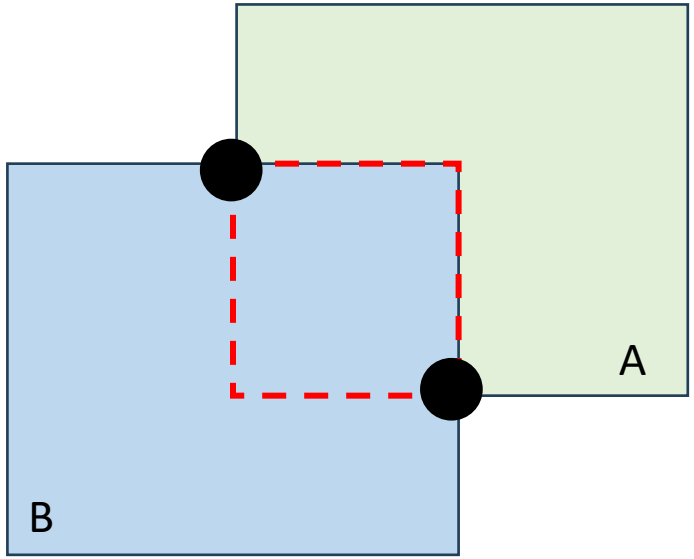
Area

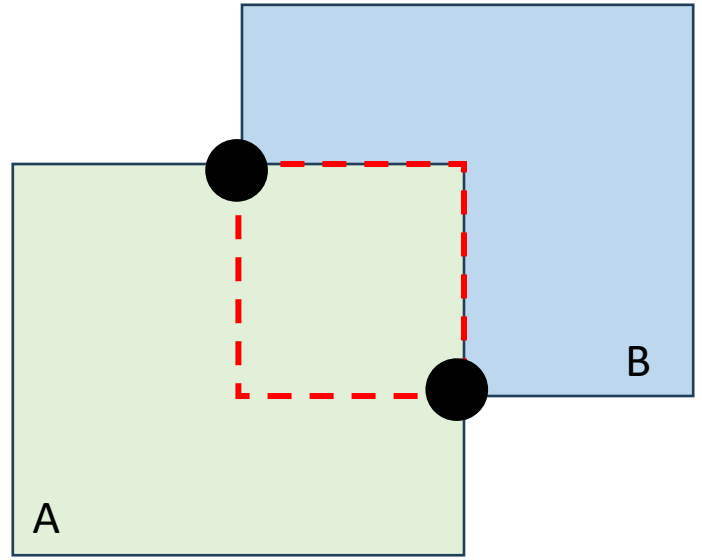


IOU Calculation Using Tuple



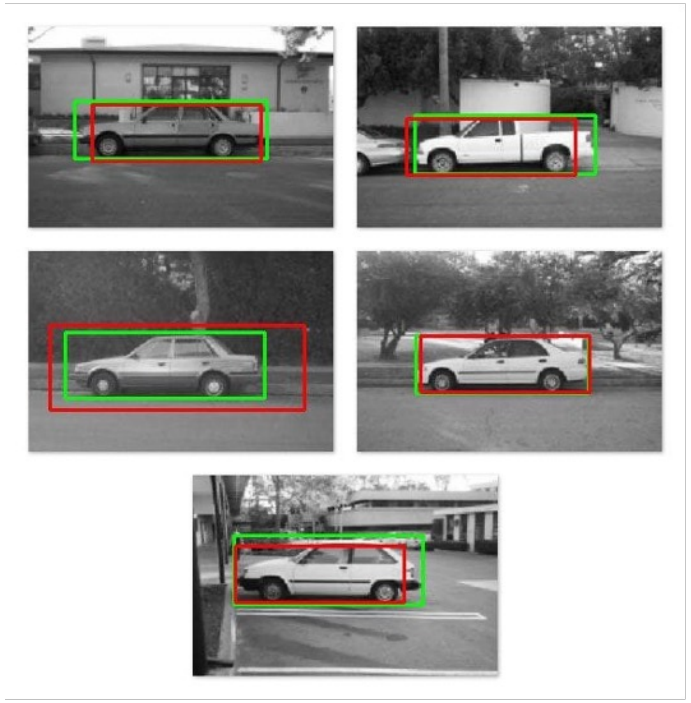






AI VIET NAM
@aivietnam.edu.vn

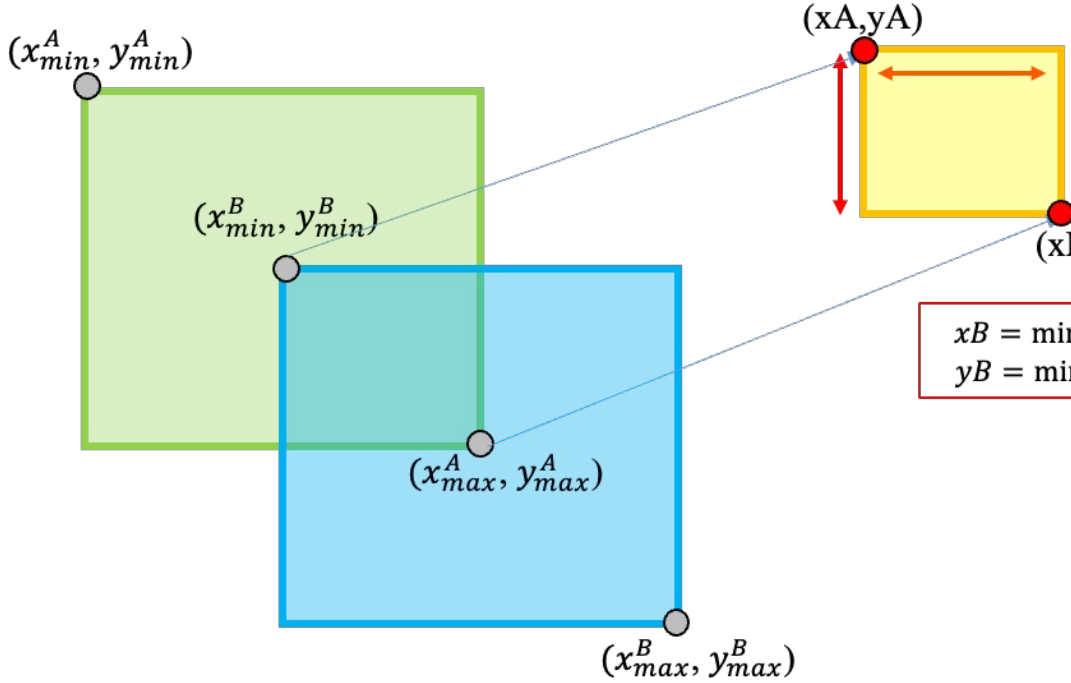
IOU Calculation Using Tuple



Giả sử ta có 2 boxes với thông tin như sau:

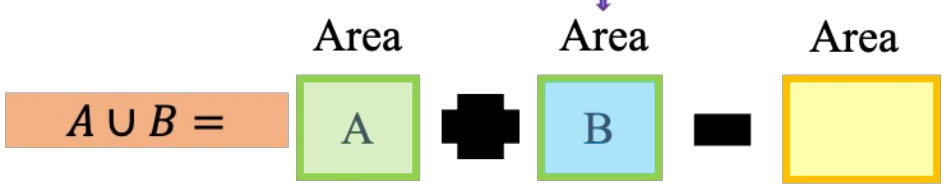
- **Box A** có tọa độ $(x^A_{min}, y^A_{min}, x^A_{max}, y^A_{max})$ => Tuple
- **Box B** có tọa độ $(x^B_{min}, y^B_{min}, x^B_{max}, y^B_{max})$ => Tuple

$$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$$
$$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$$



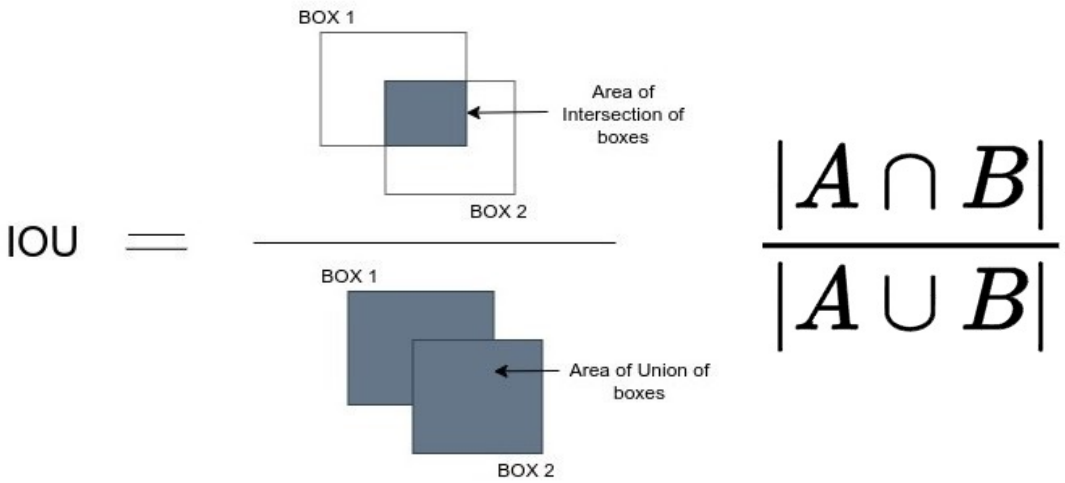
$$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$$
$$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$$

$(x^B_{max} - x^B_{min}) * (y^B_{max} - y^B_{min})$



$(x^A_{max} - x^A_{min}) * (y^A_{max} - y^A_{min})$

IOU Calculation Using Tuple



Tính diện tích của 2 box A, B

area1 = $(x^A_{max} - x^A_{min}) * (y^A_{max} - y^A_{min})$

area2 = $(x^B_{max} - x^B_{min}) * (y^B_{max} - y^B_{min})$

Tìm tọa đọ của vùng giao nhau (Intersection)

$xA = \max(x^A_{min}, x^b_{min}) = x^b_{min}$
$yA = \max(y^A_{min}, y^b_{min}) = y^b_{min}$

$xB = \min(x^A_{max}, x^b_{max}) = x^A_{max}$
$yB = \min(y^A_{max}, y^b_{max}) = y^A_{max}$

Tính diện tích vùng giao nhau

$W = xB - xA$

H = yB – yA

intersection_area = w*h

Tính diện tích phần hợp nhau

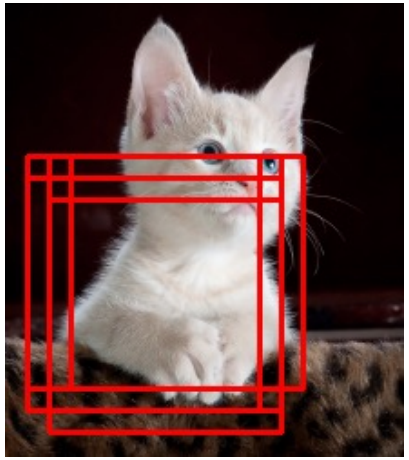
union_area = area1 + area2 - intersection_area

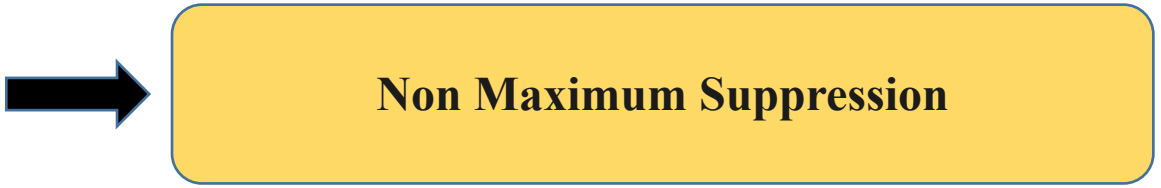
Dựa trên phần giao và phần hợp để tính IoU

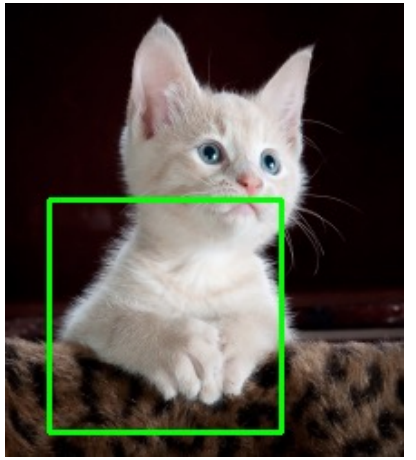
IoU = intersection_area / union_area

AI VIET NAM
@aivietnam.edu.vn

Non-Maxima Suppression: List & Tuple







List

Tuple

```
# import the necessary packages
import numpy as np
import cv2
from google.colab.patches import cv2_imshow

boundingBoxes = np.array([
  (12, 84, 140, 212, 0.3),
  (24, 84, 152, 212, 0.4),
  (36, 84, 164, 212, 0.5),
  (12, 96, 140, 224, 0.6),
  (24, 96, 152, 224, 0.7),
  (24, 108, 152, 236, 0.8)])
```

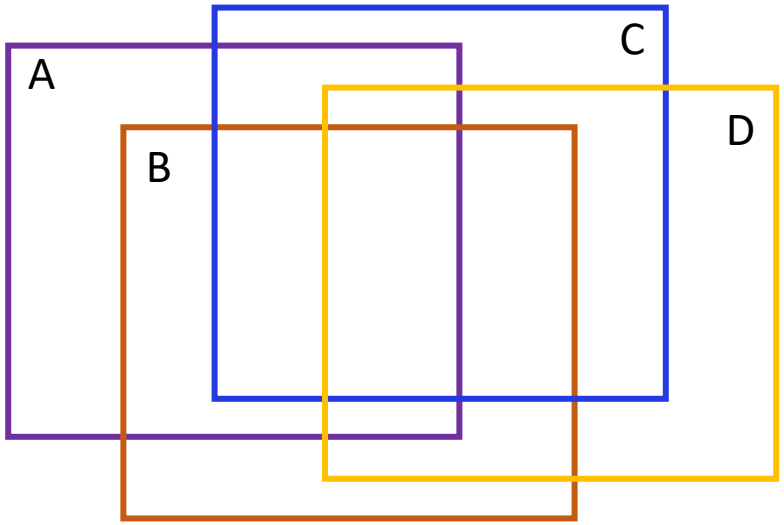
Algorithm

Step 1 : Select the prediction **S** with highest confidence score and remove it from **P** and add it to the final prediction list **keep**. (**keep** is empty initially).

Step 2 : Now compare this prediction **S** with all the predictions present in **P**. Calculate the IoU of this prediction **S** with every other predictions in **P**. If the IoU is greater than the threshold **thresh_iou** for any prediction **T** present in **P**, remove prediction **T** from **P**.

Step 3 : If there are still predictions left in **P**, then go to **Step 1** again, else return the list **keep** containing the filtered predictions.

Non-Maxima Suppression: List & Tuple

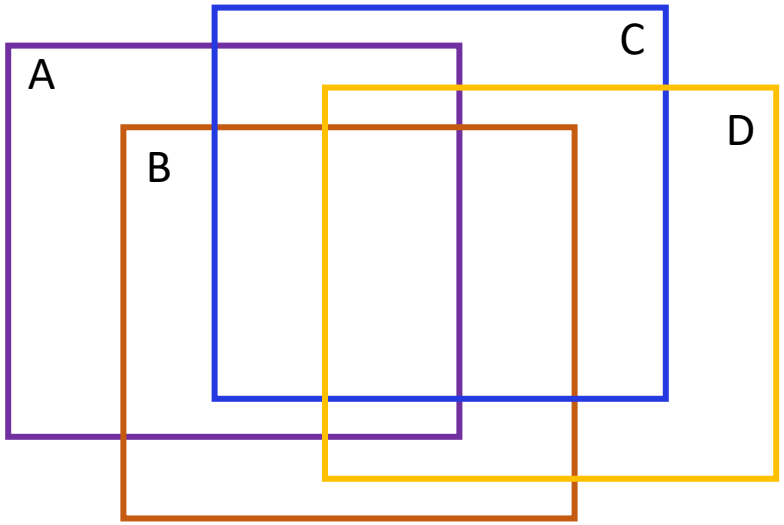


Box	Confidence	Xmin	Ymin	Xmax	Ymax
A	0.9	-	-	-	-
B	0.8	-	-	-	-
C	0.7	-	-	-	-
D	0.6	-	-	-	-

AI VIET NAM
@aivietnam.edu.vn

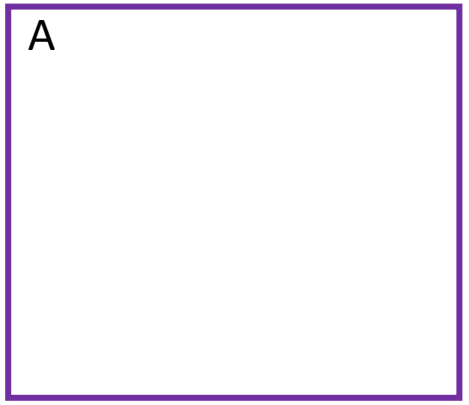
Non-Maxima Suppression: List & Tuple

Step 1 : Select the prediction **S** with highest confidence score and remove it from **P** and add it to the final prediction list **keep**. (**keep** is empty initially).



Step 1

Prediction: B,C, D
Result: {A}



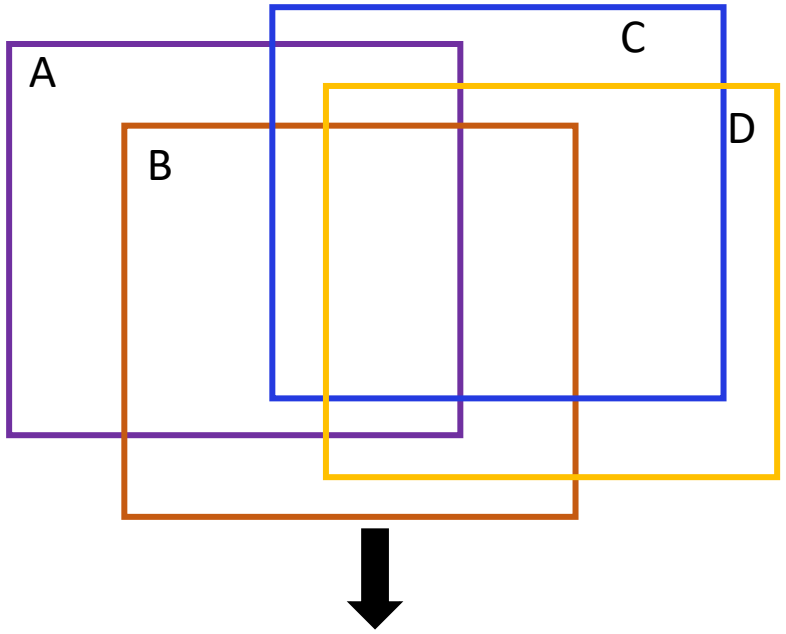
Prediction: A,B,C, D
Result: {}

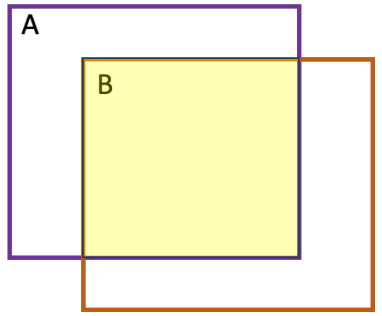
Box	Confidence	Xmin	Ymin	Xmax	Ymax
A	0.9	-	-	-	-
B	0.8	-	-	-	-
C	0.7	-	-	-	-
D	0.6	-	-	-	-

AI VIET NAM
@aivietnam.edu.vn

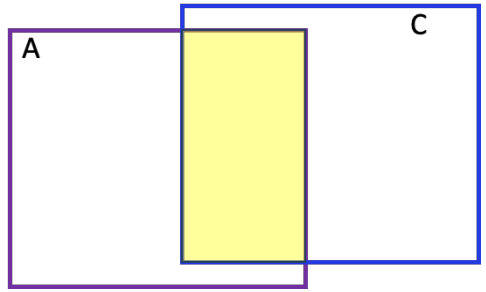
Non-Maxima Suppression: List & Tuple

Step 2 : Now compare this prediction **S** with all the predictions present in **P**. Calculate the IoU of this prediction **S** with every other predictions in **P**. If the IoU is greater than the threshold **thresh_iou** for any prediction **T** present in **P**, remove prediction **T** from **P**.



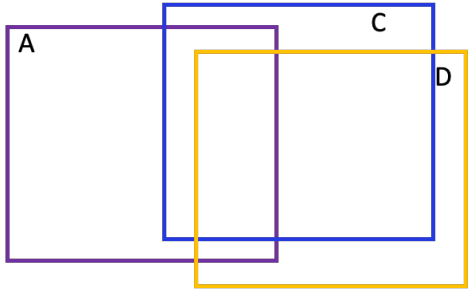


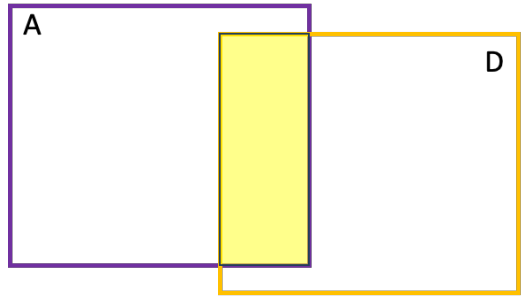
IOU > thresh_iou Remove B



IOU < thresh_iou Keep C

Prediction: C,D
Result: {A}



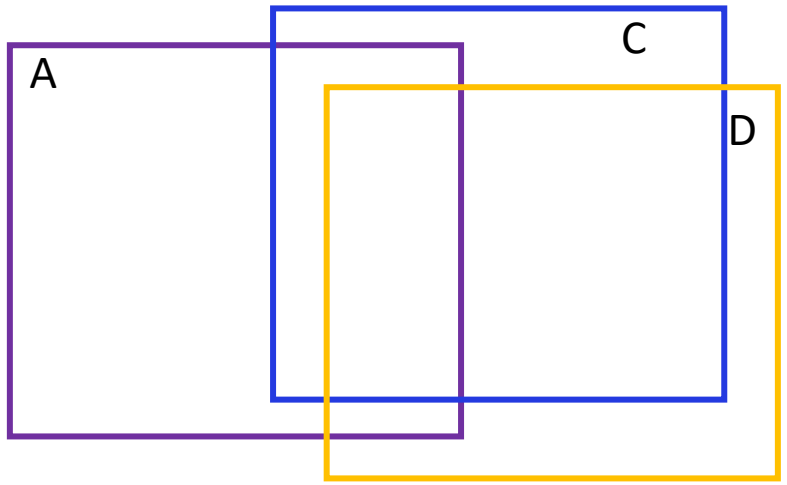


IOU < thresh_iou Keep D

Non-Maxima Suppression: List & Tuple

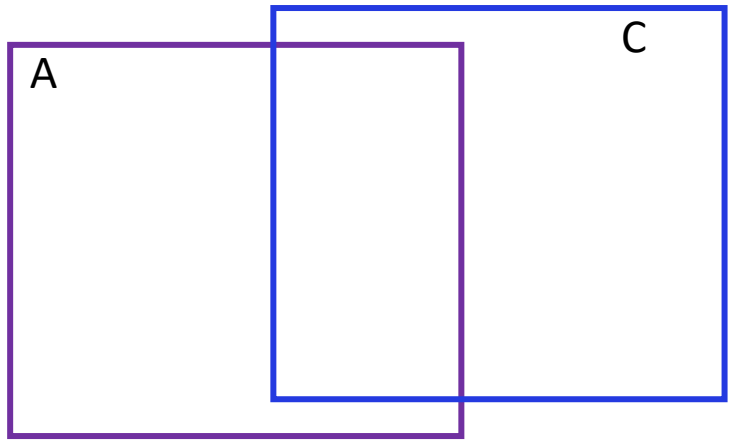
Step 3 : If there are still predictions left in **P**, then go to **Step 1** again, else return the list **keep** containing the filtered predictions.

Prediction: C,D
Result: {A}



Step 3

Prediction: D
Result: {A,C}

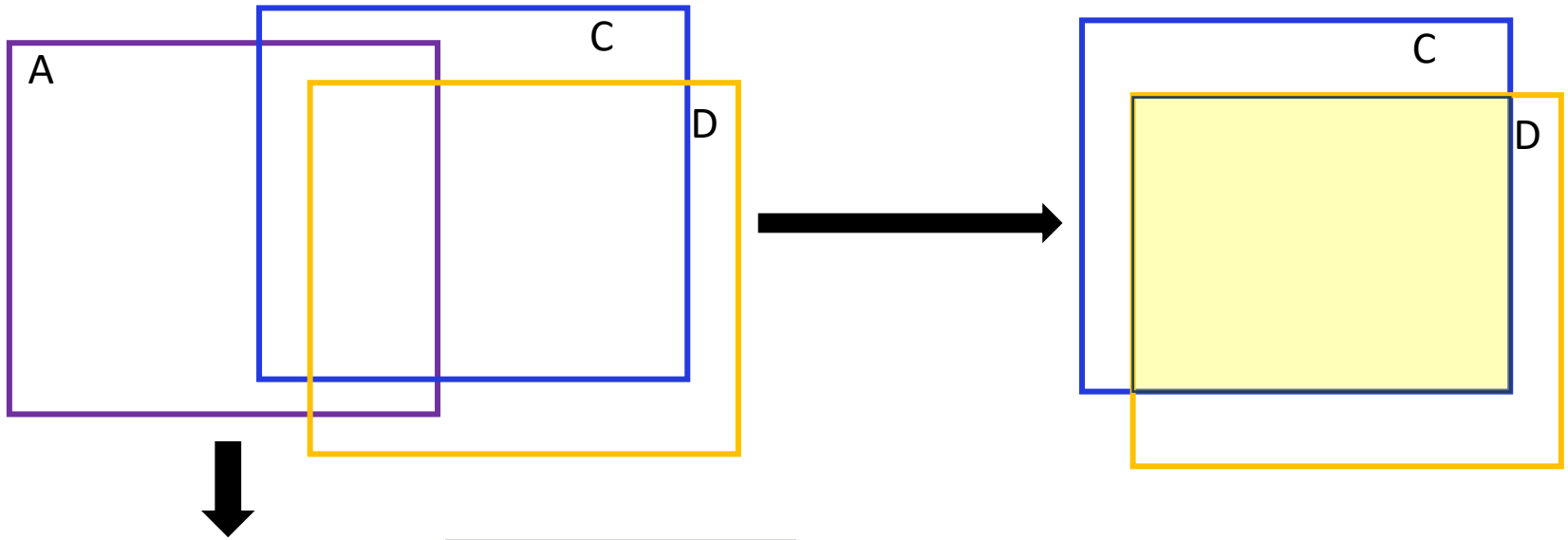


AI VIET NAM
@aivietnam.edu.vn

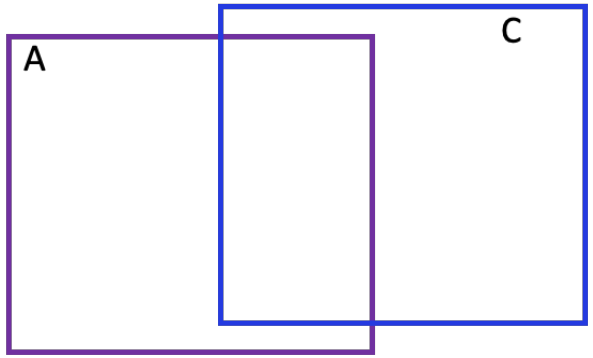
Non-Maxima Suppression: List & Tuple

Step 2 : Now compare this prediction **S** with all the predictions present in **P**. Calculate the IoU of this prediction **S** with every other predictions in **P**. If the IoU is greater than the threshold **thresh_iou** for any prediction **T** present in **P**, remove prediction **T** from **P**.

Prediction: C,D
Result: {A}

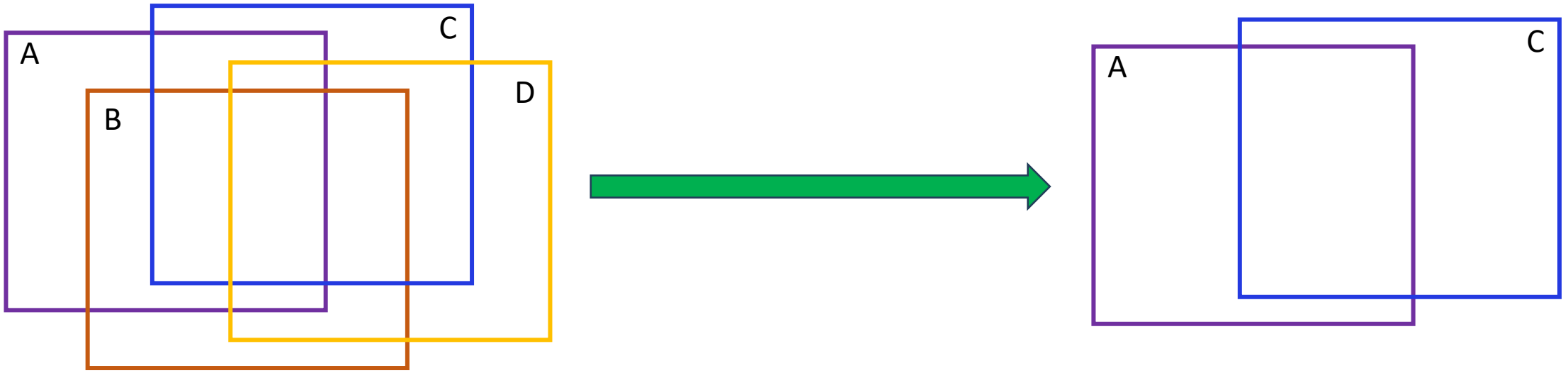


IOU > thresh_iou **Remove D**



Prediction: {}
Result: {A,C}

Non-Maxima Suppression: List & Tuple



Box	Confidence	Xmin	Ymin	Xmax	Ymax
A	0.9	-	-	-	-
B	0.8	-	-	-	-
C	0.7	-	-	-	-
D	0.6	-	-	-	-