AI VIETNAM
All-in-One Course

Data Structure

Tuple, Set, and Dictionary

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2022

Outline

- Common Errors (Quick Review)
- Tuple
- Set
- Dictionary
- Code Optimization

Tuple

❖ Structure

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

Create a tuple

```
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
```

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,)
```

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
```

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

❖ Immutable

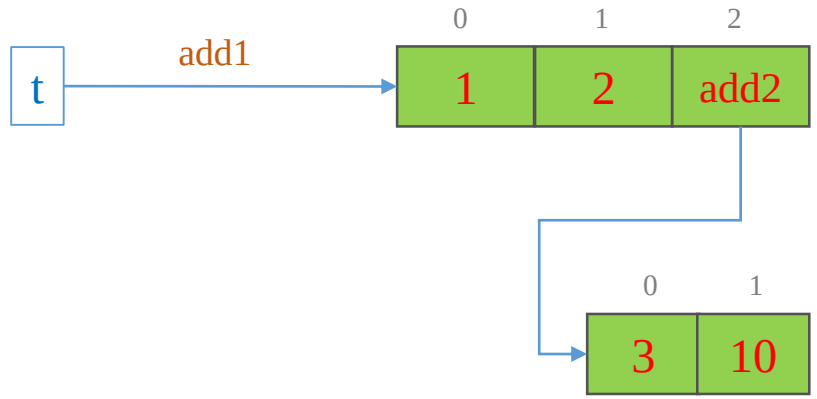
```
1. t = (1, 2, 3, 4, 5)
2. t[2] = 9
3. print(t)
```

```
---------------------------------------------------------------
TypeError                                 Traceback (most
<ipython-input-30-b1f85e7d332a> in <module>
      1 t = (1, 2, 3, 4, 5)
----> 2 t[2] = 9
      3 print(t)

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

however

```
1. t = (1, 2, [3, 10])
2. t[2][1] = 4
3. print(t)
```

```
(1, 2, [3, 4])
```



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'>
```

Tuple

❖ 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

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

Outline

- Common Errors (Quick Review)
- Tuple
- Set
- Dictionary
- Code Optimization

Set

❖ Create a set

Using curly brackets

```
# create a set
animals = {"cat", "dog", "tiger"}

print(type(animals))
print(animals)
```

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

Items with different data types

```
# create a set
a_set = {"cat", 5, True, 40.0}

print(type(a_set))
print(a_set)
```

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

Set comprehension

```
# set comprehension

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

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

Set

Access the items of a set

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

```
dog
cat
tiger
```

Copy a set

```
# copy
animals = {"cat", "dog", "tiger"}
print("Animals:", animals)

a_copy = animals.copy()
print("Copy:", a_copy)
```

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

Set

Add an item

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

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

Insert a set to another set

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

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

Join two sets

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

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

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

Not allow duplicate values

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

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

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

Set

difference function

```python
# difference

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set3 = set1.difference(set2)

print(set3)
```

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

symmetric_difference

```python
# symmetric_difference

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set3 = set1.symmetric_difference(set2)

print(set3)
```

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

difference_update function

```python
# difference_update

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set1.difference_update(set2)

print(set1)
```

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

symmetric_difference_update

```python
# symmetric_difference_update

set1 = {"apple", "banana", "cherry"}
set2 = {"pineapple", "apple"}

set1.symmetric_difference_update(set2)

print(set1)
```

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

Set

❖ Bitwise operator

```python
# AND (&)

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

print(set1 & set2)
```

```
{3}
```

```python
# XOR (^)

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

print(set1 ^ set2)
```

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

```python
# OR (|)

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

print(set1 | set2)
```

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

```python
# subtraction (-)

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

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
<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
<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))
```

???

Set

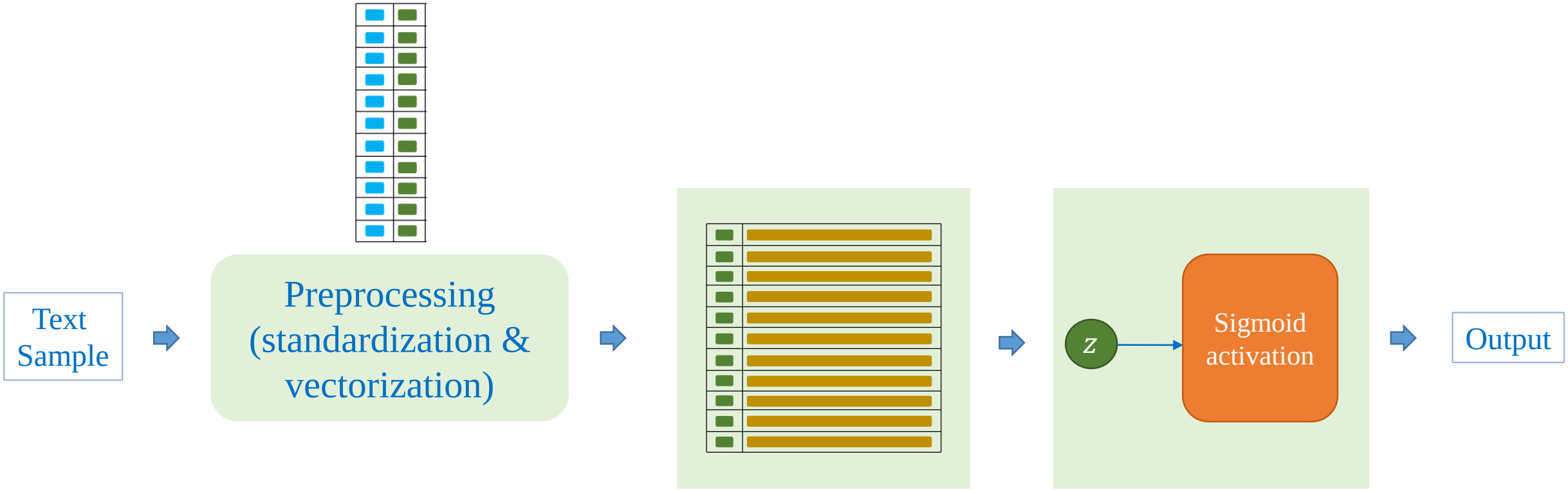
❖ Text classification

- 50,000 movie review for sentiment analysis (data)
- Consist of: + 25,000 movie review for training
  + 25,000 movie review for testing
- Label: positive – negative = 1 – 1

“A wonderful little production. The filming technique is very unassuming- very old-time-BBC fashion and gives a comforting, and sometimes discomforting, sense of realism to the entire piece…..”	positive
“This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. By 1990, the show was not really funny anymore, and it's continued its decline further to the complete waste of time it is today….”	negative
“I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air conditioned theater and watching a light-hearted comedy. The plot is simplistic, but the dialogue is witty and the characters are likable (even the well bread suspected serial killer)….”	positive
“BTW Carver gets a very annoying sidekick who makes you wanna shoot him the first three minutes he's on screen.”	negative

Set

❖ Text classification



Embedding

- Example corpus

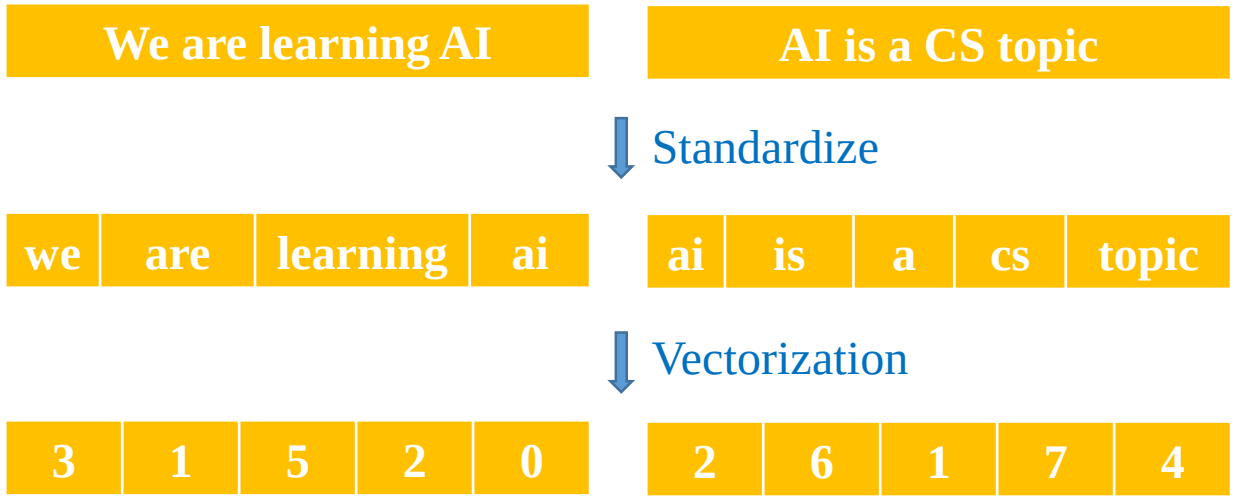
  sample1: ‘We are learning AI’

  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



Embedding Layer

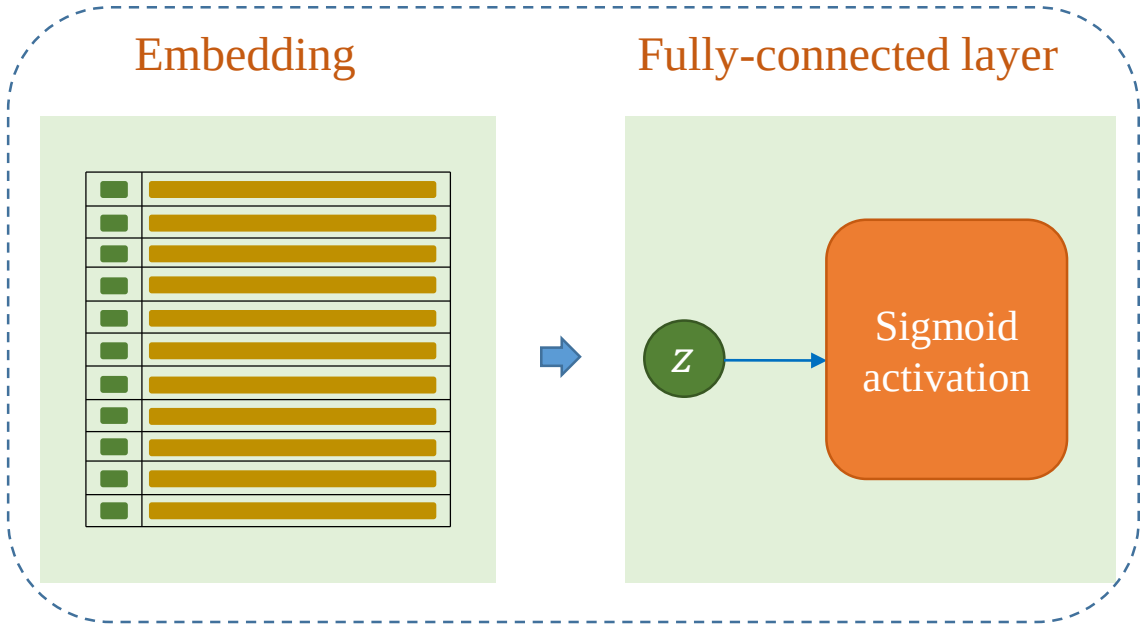
(3) Embedding layer

```
print(model.layers[1].weights[0])
```

```
<tf.Variable 'embedding_3/embeddings:0' shape=(8, 4) dtype=float32,
array([[-0.03296757, -0.03054714,  0.01625914, -0.02955737],
       [ 0.01487434,  0.0038103 , -0.02253381,  0.04637194],
       [-0.00092559, -0.00625734,  0.03492227, -0.01756487],
       [-0.03613882, -0.0436982 , -0.04447322, -0.02143981],
       [-0.01163145, -0.0085723 ,  0.01227676, -0.0089262 ],
       [ 0.02521226,  0.04333058, -0.02373846,  0.02480527],
       [-0.02779102, -0.03064094, -0.04173347, -0.02026683],
       [ 0.04235573,  0.00204159, -0.04987942,  0.04008349]],
      dtype=float32)>
```

index	word
0	pad
1	are/a
2	AI
3	We
4	Topic
5	Learning
6	is
7	cs

We are learning AI → 3, 1, 5, 2, 0 →



Set

❖ Convert text to numbers

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

# read content
data = a_file.read()
print(data)

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

```
data = data.replace('.', '')
print(data)
```

```
data = data.replace(',', '')
print(data)
```

```
data = data.replace('-', ' ')
print(data)
```

```
data = data.lower()
print(data)
```

```
data = data.split()
print(data)
print(len(data))
```

```
data = set(data)
print(data)
print(len(data))
```

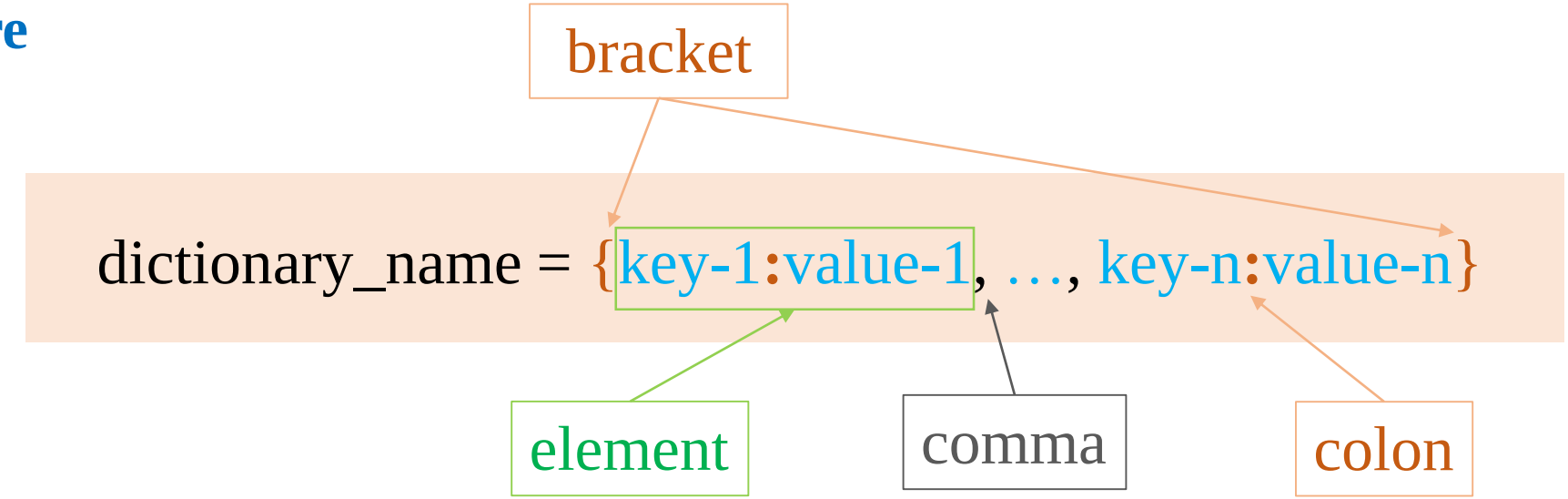
```
for index, value in enumerate(data):
    print(index, value)
```

Outline

- Common Errors (Quick Review)
- Tuple
- Set
- Dictionary
- Code Optimization

Dictionary

❖ Structure



Create a dictionary

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

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

Dictionary

❖ Create a Dictionary

```python
# dic comprehension

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

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

```python
# from zip

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

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

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

```python
# from zip

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

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

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

```python
# from zip

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

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

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

Dictionary

❖ Update a value

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

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

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

❖ Copy a dictionary

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

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

print(parameters)
print(a_copy)
```

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

Dictionary

❖ Hàm copy() chỉ sao chép kiểu shallow

```
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

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

```
learning_rate
optimizer
metric
```

Get values

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

```
0.1
Adam
Accuracy
```

Get keys

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

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

```
learning_rate
optimizer
metric
```

Get keys and values

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

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

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

Dictionary

❖ Get a value by a key

Get value using get() function

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

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

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

```
0.1

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

Get value and delete the corresponding item

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

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

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

```
0.1

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

Dictionary

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

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

item = parameters.popitem()

print(item)
print(parameters)
```

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

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

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

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

parameters.clear()

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

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

After using clear() function
{}
```

Use del keyword to delete an item

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

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

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

Dictionary

❖ Key that does not exist

Try to delete a non-existing item

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

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

KeyError: 'algorithm'
```

Try to get an item by a non-existing key

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

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

KeyError: 'algorithm'
```

Dictionary

setdefault() function

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

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

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

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

example

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

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

Result ???

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

Dictionary

❖ Get a value via a key

Method 1

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

```
4

---------------------------------------------------------------------------
KeyError                                  Traceback
<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}
```

Dictionary

❖ Common error

```
1. # aivietnam.ai
2. # Lỗi key không tồn tại
3.
4. dict = {'1':'Python', '5':'C++'}
5. print(dict['1'])
6. print(dict['5'])
7. print(dict['2'])
```

```
Python
C++


---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-11-cd160b2a5e34> in <module>
      5 print(dict['1'])
      6 print(dict['5'])
----> 7 print(dict['2'])

KeyError: '2'
```

Dictionary

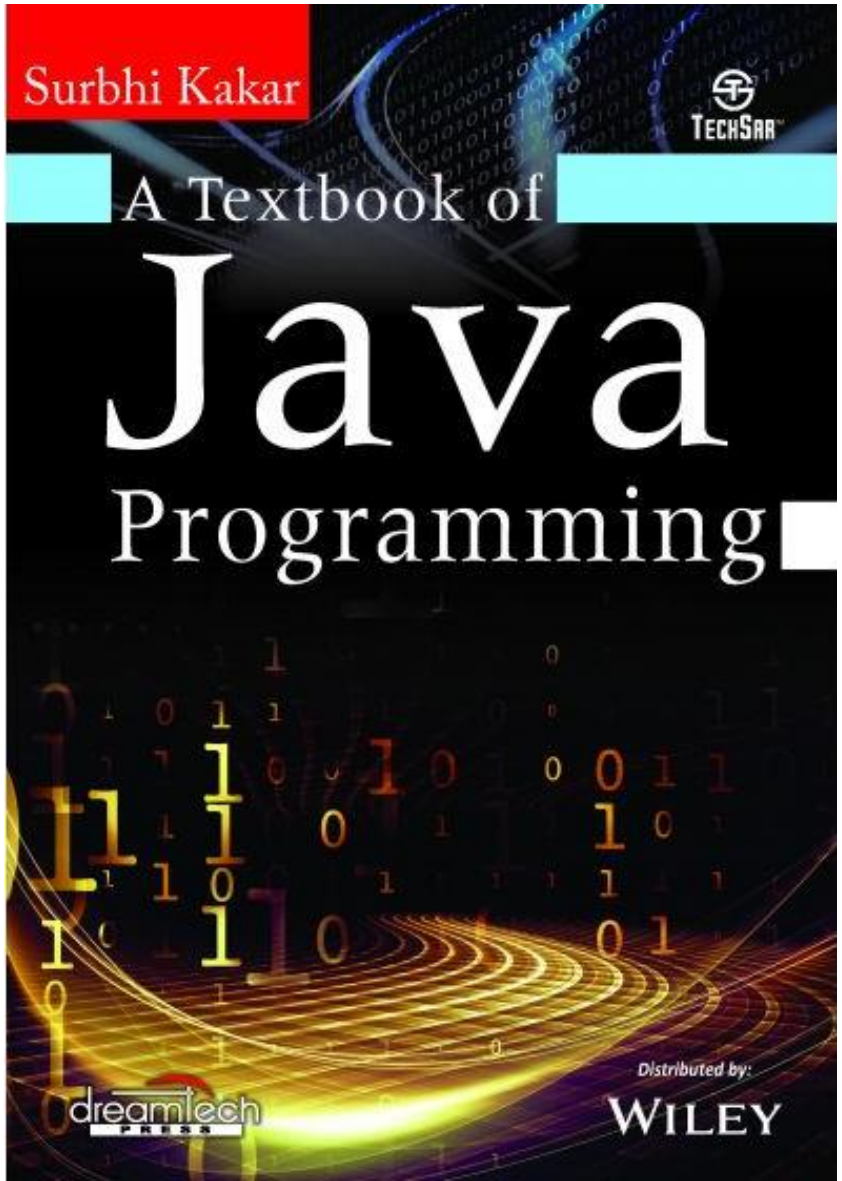
❖ Common error

```python
# aivietnam.ai
# Lỗi cố gắng lấy phần tử từ dictionary rỗng

dict = {'1':'Python', '5':'C++'}
item1 = dict.popitem()
item2 = dict.popitem()
item3 = dict.popitem()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-6-40b67d40ba0f> in <module>
      5 item1 = dict.popitem()
      6 item2 = dict.popitem()
----> 7 item3 = dict.popitem()

KeyError: 'popitem(): dictionary is empty'
```

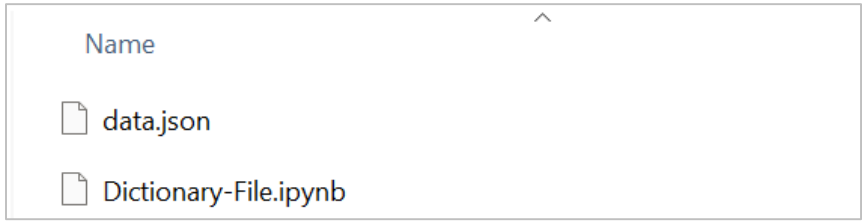
Dictionary Saving and Loading

- JSON (JavaScript Object Notation)
- Data exchange format
  - Lightweight
  - Text-based
  - Language-independent
- Designed to be both human- and machine-readable.




```
{
    "author": "Surbhi Kakar",
    "title": "Java Programming",
    "genre": "Computer",
    "price": "30.0",
    "publish_date": "2010-08-01",
    "publisher": "Dream Tech Press",
    "description": "..."
}
```

Dictionary Saving and Loading



data.json

```
{"learning_rate": 0.1,
"optimizer": "Adam",
"metric": "Accuracy"}
```

```python
import json

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

with open('data.json', 'w') as fp:
    json.dump(parameters, fp)
```

```python
with open('data.json', 'r') as fp:
    data = json.load(fp)
    print(data)
```

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

Sorting

```
sorted(iterable, key=None, reverse=False)
```

```python
# create a list
aList = [1, 5, 3, 7, 4]
print(aList)

# sort
sortedList = sorted(aList)
print(sortedList)
```

```
[1, 5, 3, 7, 4]
[1, 3, 4, 5, 7]
```

item=7

```python
# create a list
aList = [1, 5, 3, 7, 4]
print(aList)

# function for sorting
def compare(item):
    return item

# sort
sortedList = sorted(aList, key=compare)
print(sortedList)
```

```
[1, 5, 3, 7, 4]
[1, 3, 4, 5, 7]
```

Sorting

```python
# data
list1 = ['a', 'g', 'e', 'h', 'b']
list2 = [16, 13, 18, 11, 15]

# create a dictionary
list3 = list(zip(list1, list2))
print(list3)
```

```
[('a', 16), ('g', 13), ('e', 18), ('h', 11), ('b', 15)]
```

```python
list4 = sorted(list3)
print(list4)
```

```
[('a', 16), ('b', 15), ('e', 18), ('g', 13), ('h', 11)]
```

```python
# data
list1 = ['a', 'g', 'e', 'h', 'b']
list2 = [16, 13, 18, 11, 15]

# function for sorting
def compare(item):
    return item[0]

# create a dictionary
list3 = list(zip(list1, list2))
print(list3)
```

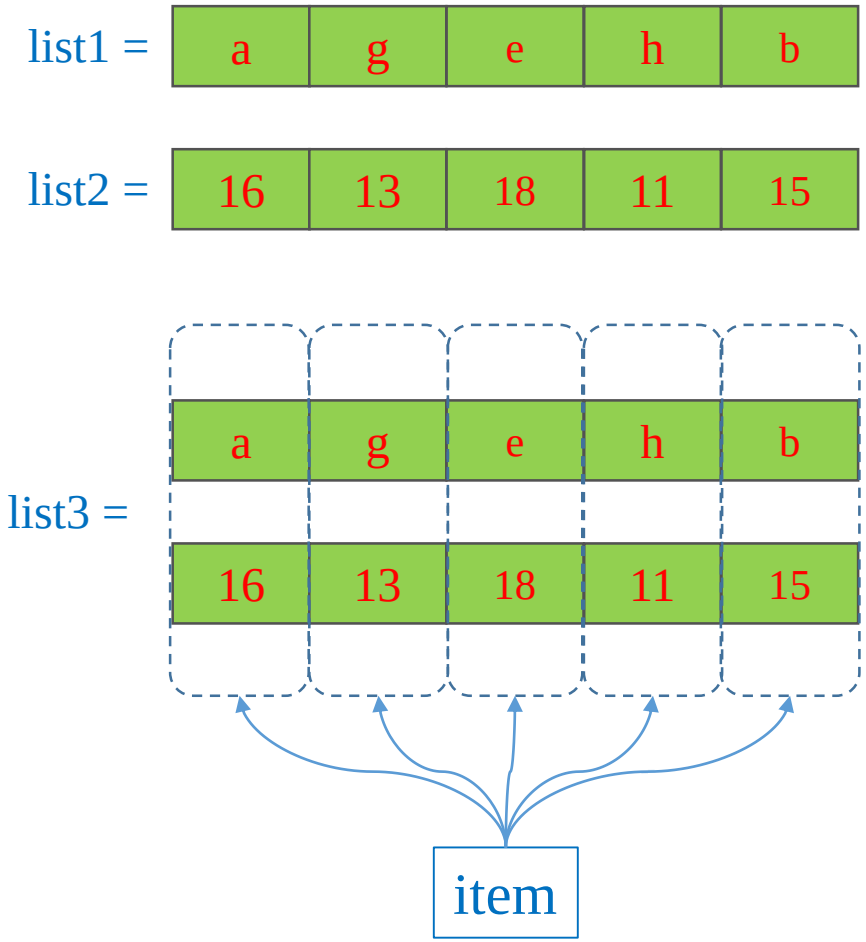
item=('g', 13)

```
[('a', 16), ('g', 13), ('e', 18), ('h', 11), ('b', 15)]
```

```python
list4 = sorted(list3, key=compare)
print(list4)
```

```
[('a', 16), ('b', 15), ('e', 18), ('g', 13), ('h', 11)]
```

Sorting




```
1  # data
2  list1 = ['a', 'g', 'e', 'h', 'b']
3  list2 = [16, 13, 18, 11, 15]
4
5  # function for sorting
6  def compare(item):
7      return item[1]
8
9  # create a dictionary
10 list3 = list(zip(list1, list2))
11 print(list3)
```

item=('g', 13)

```
[('a', 16), ('g', 13), ('e', 18), ('h', 11), ('b', 15)]
```

```
1  list4 = sorted(list3, key=compare)
2  print(list4)
```

```
[('h', 11), ('g', 13), ('b', 15), ('a', 16), ('e', 18)]
```

Lambda function

- Take any number of arguments
- Can only have one expression

Syntax

lambda arguments : expression

```
1  # lambda function
2  a_lfunction = lambda v: v + 10
3  print(a_lfunction(5))
```

```
15
```

```
1  # lambda function
2  a_lfunction = lambda v1, v2: v1+v2
3  print(a_lfunction(3, 4))
```

```
7
```

AI VIETNAM
All-in-One Course

Sorting

Using lambda function

```
1  # data
2  list1 = ['a', 'g', 'e', 'h', 'b']
3  list2 = [16, 13, 18, 11, 15]
4
5  # create a dictionary
6  list3 = list(zip(list1, list2))
7  print(list3)
```

```
[('a', 16), ('g', 13), ('e', 18), ('h', 11), ('b', 15)]
```

```
1  list4 = sorted(list3, key=lambda item: item[1])
2  print(list4)
```

```
[('h', 11), ('g', 13), ('b', 15), ('a', 16), ('e', 18)]
```

```
1  # data
2  list1 = ['a', 'g', 'e', 'h', 'b']
3  list2 = [16, 13, 18, 11, 15]
4
5  # function for sorting
6  def compare(item):
7      return item[1]
8
9  # create a dictionary
10 list3 = list(zip(list1, list2))
11 print(list3)
```

item=('g', 13)

```
[('a', 16), ('g', 13), ('e', 18), ('h', 11), ('b', 15)]
```

```
1  list4 = sorted(list3, key=compare)
2  print(list4)
```

```
[('h', 11), ('g', 13), ('b', 15), ('a', 16), ('e', 18)]
```

Outline

- Common Errors (Quick Review)
- Tuple
- Set
- Dictionary
- Code Optimization

Code Optimization (1)

❖ Use built-in functions

Tìm các phần tử duy nhất trong list

```python
import random

random.seed(42)
my_list = [random.randint(0, 100) for i in range(100000)]
```

```python
%%time

# slower
list_unique = []
for num in my_list:
    if num not in list_unique:
        list_unique.append(num)
```

```
Wall time: 132 ms
```

```python
%%time

# faster
list_unique = list(set(my_list))
```

```
Wall time: 2.23 ms
```

Tính tổng các phần tử trong list

```python
import random

random.seed(42)
my_list = [random.randint(0, 100) for i in range(100000)]
```

```python
%%time

# slower
result = 0
for num in my_list:
    result += num

print(result)
```

```
5007791
Wall time: 7.98 ms
```

```python
%%time

# faster
result = sum(my_list)
print(result)
```

```
5007791
Wall time: 998 µs
```

Code Optimization (2)

❖ Use built-in functions

Dùng list comprehension

```python
import random

random.seed(42)
my_list = [random.randint(0, 100) for i in range(100000)]
```

```python
%%time

# slower
new_list = []
for num in my_list:
    new_list.append(num*2)

#print(new_list)
```

```
Wall time: 8.98 ms
```

```python
%%time

# faster
new_list = [num*2 for num in my_list]
#print(new_list)
```

```
Wall time: 3.99 ms
```

Đếm số phần tử trong list dùng Counter

```python
import random

random.seed(42)
my_list = [random.randint(0, 10) for i in range(100000)]
```

```python
%%time

# slower
my_dict = {}

for char in my_list:
    if char not in my_dict:
        my_dict[char] = 1

    my_dict[char] += 1

#print(my_dict)
```

```
Wall time: 21.9 ms
```

```python
%%time

from collections import Counter

my_dict = Counter(my_list)
#print(my_dict)
```

```
Wall time: 7.99 ms
```

Bad Codes

```python
# bad code 1

fruits= [ 'apple' ,"lemon",'banana']
```

```python
# bad code 2

x = 1
y = 2
z = 3

f1 = x + y - z
f2 = x/y + z
f3 = x - y*z
f4 = (x/y) * z
```

```python
# bad code 3

weights_1 = [1.0, 2.5, 3.7]
weights_2 = [1.0,2.5,3.7]
weights_3 = [1.0 , 2.5 , 3.7]
```

```python
# bad code 4

'aivietnam' . upper()
```

```python
# bad code 5

print ('aivietnam')
```

```python
# bad code 6

fruits = ['apple', 'lemon', 'banana']
apple = fruits [0]
sub_set = fruits   [:2]
```

Bad Codes

```python
# bad code 7

weights_1 = [1.0, 2.5, 3.7]
weights_2 = [ 1.0, 2.5, 3.7 ]
```

```python
# bad code 8

print('AI VIETNAM')  # a comment
print('AI VIETNAM')# a comment
```

```python
# bad code 9

class AdamOptimizer:
    def exampleMethod1():
        # ...
    def exampleMethod2():
        #...
def train():
    #...
```

```python
# bad code 10

import math, os, sys

#  -- or --

import math
import os
import sys
```

Bad Codes

```python
# bad code 11 - style

fruits = ['apple', 'lemon', 'banana']
apple = fruits[0]
sub_set = fruit[:2]

#  -- or --

fruits  = ['apple', 'lemon', 'banana']
apple   = fruits[0]
sub_set = fruit[:2]
```

```python
# bad code 12

fruits = ['apple', 'lemon', 'banana']
number_of_fruits = len(fruits)
print(number_of_fruits)

len = 3
print(len(fruits))
```

```python
# bad code 13 - magic number

radius = 4
area = radius*radius*3.14159
```

```python
# bad code 14 - one entry, one exit

def ReLU(number):
    if (number <= 0):
        return 0
    else:
        return number

print(ReLU(-8))
```

Bad Codes

```python
# bad code 15 - dead code

def ReLU(number):
    if (number <= 0):
        return 0
    else:
        return number

    return 'Input is not a number!'

print(ReLU(-8))
```

```python
# bad code 16 - comment

def flip(times):
    number_of_heads = 0
    number_of_tails = 0

    for _ in range(times):
        number = random.randint(0, 1)
        if (number == 1):
            number_of_heads = number_of_heads + 1
        else:
            number_of_tails = number_of_tails + 1

    return number_of_heads, number_of_tails

number_of_heads, number_of_tails = flip(1000)
print(number_of_heads)
print(number_of_tails)
```

Bad Codes

```python
# bad code 17 - global

number_of_heads = 0
number_of_tails = 0

def flip(times):
    global number_of_heads
    global number_of_tails

    for _ in range(times):
        number = random.randint(0, 1)
        if (number == 1):
            number_of_heads = number_of_heads + 1
        else:
            number_of_tails = number_of_tails + 1

flip(1000)
print(number_of_heads)
print(number_of_tails)
```

```python
# bad code 18 - make thing complicated

import random
class Dice:
    def __init__(self, sides=6):
        self.sides = sides
    def roll(self):
        return random.randint(1, self.sides)

d = Dice()
print('You rolled a', d.roll())
```

Bad Codes

```python
# bad code 19

numbers = []

for i in range(1, 100):
    if (i%5 == 0):
        numbers.append(i)

print(numbers)
```

```python
numbers = [i for i in range(1, 100) if (i%5 == 0)]
print(numbers)
```

```python
# bad code 21

# open a file
a_file = open('hello_world.txt', 'w')

# write data to file
text3 = 'writing line \n'
a_file.write(text3)
```

```python
# bad code 20

try:
    num = input('Enter a number: ')
    num = int(num)
except ValueError:
    pass # do nothing
```

Bad Codes

```python
# bad code 22 - Unpythonic

path = 'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
print(path)

with open(path, 'r') as file:
    lines = file.readlines()
    print(lines)
```

```python
path = 'E:\\Data\\AICourse-2021\\1.BasicPython\\file\\hello_world.txt'
print(path)

with open(path, 'r') as file:
    lines = file.readlines()
    print(lines)
```

```python
path = r'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
print(path)

with open(path, 'r') as file:
    lines = file.readlines()
    print(lines)
```

Bad Codes

```python
# bad code 23

name = 'John'
age  = 26

print('Hello ' + name + '. Are you ' + str(age) + '?')
```

```python
name = 'John'
age  = 26

print(f'Hello {name}. Are you {age}?')
```

```python
# bad code 24 - Unpythonic

fruits = {'banana': 2}
if 'apple' not in fruits:
    fruits['apple'] = 0

fruits['apple'] += 10
print(fruits)
```

```python
fruits = {'banana': 2}
fruits.setdefault('apple', 0)

fruits['apple'] += 10
print(fruits)
```

```python
# bad code 25

def get_salary_rate(employee_class):
    if employee_class == 'level-1':
        result = 5.7
    elif employee_class == 'level-2':
        result = 4.2
    elif employee_class == 'level-3':
        result = 3.8
    elif employee_class == 'level-4':
        result = 3.3
    else:
        result = 2.9

    return result

print(get_salary_rate('level-1'))
print(get_salary_rate('level-4'))
```

```python
salary_rates = {'level-1': 5.7,
                'level-2': 4.2,
                'level-3': 3.8,
                'level-4': 3.3,
                'level-5': 2.9}

print(salary_rates['level-1'])
print(salary_rates['level-4'])
```

Bad Codes

```python
# bad code 26 - Unpythonic

def a_function(value):

    # do something

    if 1 < value and value < 10:
        print('code inside if')
    else:
        print('code inside else')

    # do something and return something

a_function(4)
a_function(40)
```

```python
def a_function(value):

    # do something

    if 1 < value < 10:
        print('code inside if')
    else:
        print('code inside else')

    # do something and return something

a_function(4)
a_function(40)
```

```python
# bad code 27

def contain_fruit(a_fruit):
    if (a_fruit == 'banana'):
        result = True
    elif (a_fruit == 'apple'):
        result = True
    elif (a_fruit == 'peach'):
        result = True
    else:
        result = False

    return result

print(contain_fruit('banana'))
print(contain_fruit('pineapple'))
```

```python
def contain_fruit(a_fruit):
    result = a_fruit in ('banana',
                         'apple',
                         'peach')

    return result

print(contain_fruit('banana'))
print(contain_fruit('pineapple'))
```

Bad Codes

```python
# super bad code 28

value = True + True + False + True
char1 = 'aivietnam'[False]
char2 = 'aivietnam'[True]
char3 = 'aivietnam'[-True]

print(value)
print(char1)
print(char2)
print(char3)
```

```python
# bad code 29

def add_fruit(a_fruit, fruits=['apple']):
    fruits.append(a_fruit)

    return fruits
```

```python
fruits = add_fruit('banana')
print(fruits)

fruits = add_fruit('banana')
print(fruits)
```

```
['apple', 'banana']
['apple', 'banana', 'banana']
```

Bad Codes

```python
# bad code 30

import math

def quadratic_equation(a, b, c):
    # compute delta
    delta = b*b - 4*a*c

    if delta < 0:
        result = 'The equation has no solution'
    elif delta == 0:
        x = (-b+math.sqrt(delta))/2*a
        result = (x,)
    else:
        x1 = (-b+math.sqrt(delta))/(2*a)
        x2 = (-b-math.sqrt(delta))/(2*a)
        result = (x1,x2)

    return result

print(quadratic_equation(3, 2, 1))
print(quadratic_equation(1, 2, 1))
```

Good Code

```python
# swap

x, y = 3, 4
print(x, y)

x, y = y, x
print(x, y)
```

```
3 4
4 3
```

```python
# reverse a string

name = "ai vietname"
print(name)
print(name[::-1])
```

```
ai vietname
emanteiv ia
```

```python
# condition

n = 8
result = 1 < n < 10
print(result)
```

```
True
```

```python
# join

a = ["Hi", "AI", "VIETNAM"]
print(" ".join(a))
```

```
Hi AI VIETNAM
```

Good Code

```python
# unpacking

a_list = [1, 2, 3]
x, y, z = a_list

print(x, y, z)
```

```
1 2 3
```

```python
# check if contained
m = 1

if m in [1, 3 ,5, 7]:
    print('Contained!')
```

```
Contained!
```

```python
# enumerate

a_list = [4, 5, 6]
for i, value in enumerate(a_list):
    print(i, ': ', value)
```

```
0 :  4
1 :  5
2 :  6
```

Good Code

```python
# Unpacking operator

def print_data(x, y, z):
    print(x, y, z)

a_dict = {'x': 1, 'y': 2, 'z': 3}
a_list = [3, 4, 5]

print_data(*a_dict)
print_data(**a_dict)
print_data(*a_list)
```

```
x y z
1 2 3
3 4 5
```

```python
# Use a dictionary to store a switch

ops = {
        'addition': lambda x, y: x + y,
        'subtraction': lambda x, y: x - y,
        'multiplication': lambda x, y: x * y,
        'division': lambda x, y: x / y
      }

print(ops['addition'](4, 6))
print(ops['multiplication'](4, 6))
```

```
10
24
```

Good Code

```python
# get the most frequent element

test = [1, 2, 3, 4, 2]

print(max(test))
print(max(test, key=test.count))
```

```
4
2
```

```python
# create dict from two tuples

t1 = (1, 2, 3)
t2 = (10, 20, 30)

a_dict = dict(zip(t1,t2))
print(a_dict)
```

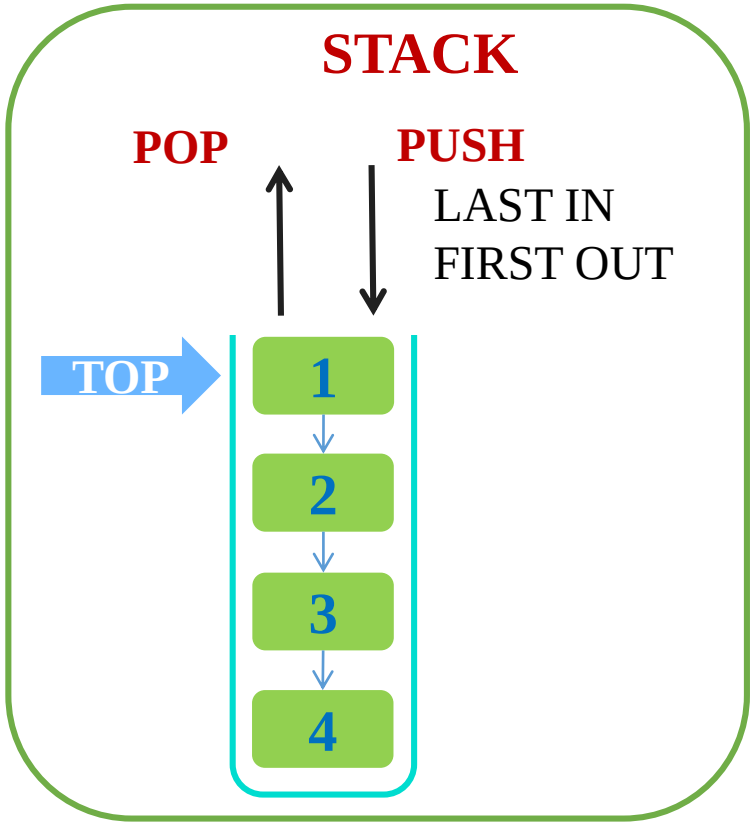
```
{1: 10, 2: 20, 3: 30}
```

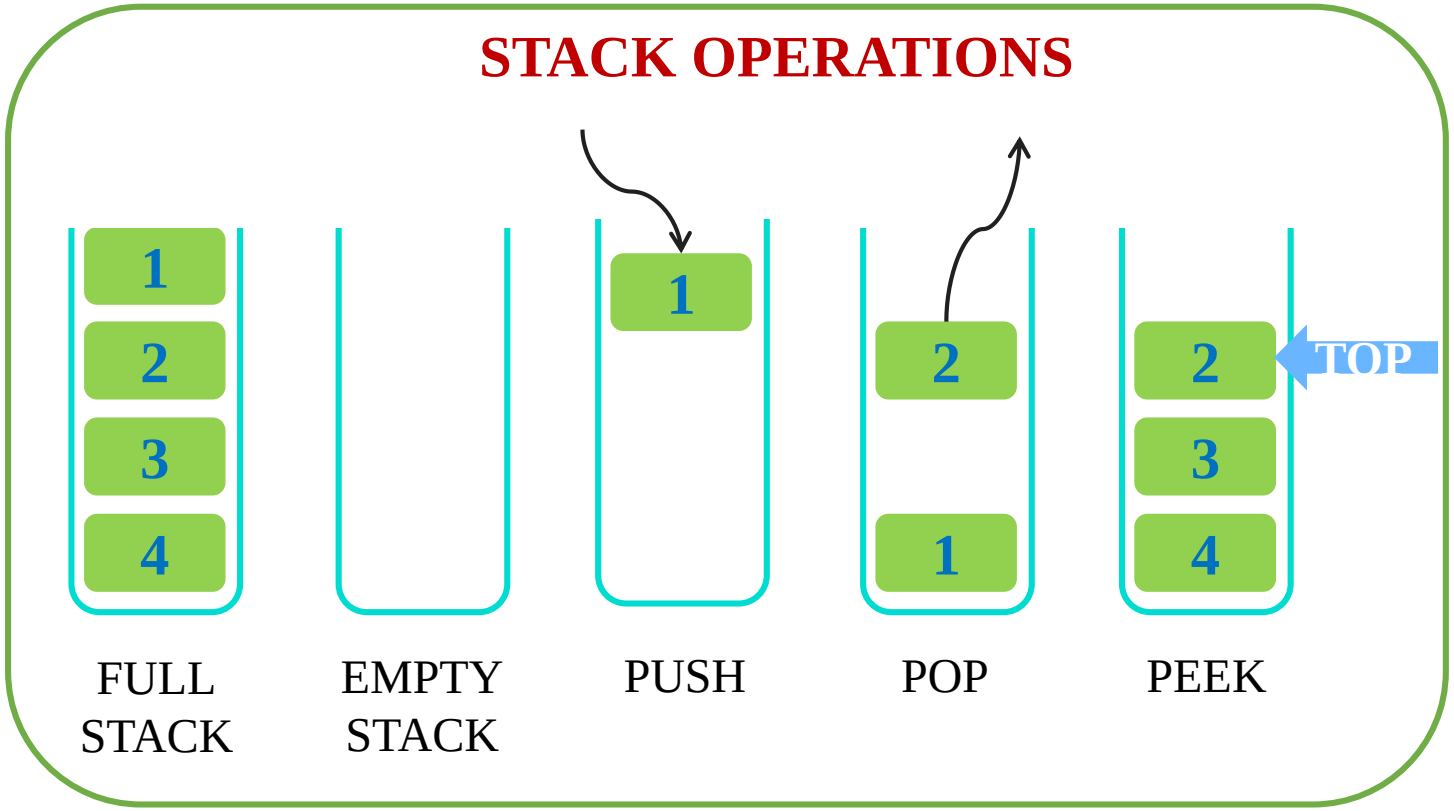
Outline

- Common Errors (Quick Review)
- Tuple
- Set
- Dictionary
- Code Optimization

Stack

❖ Introduction



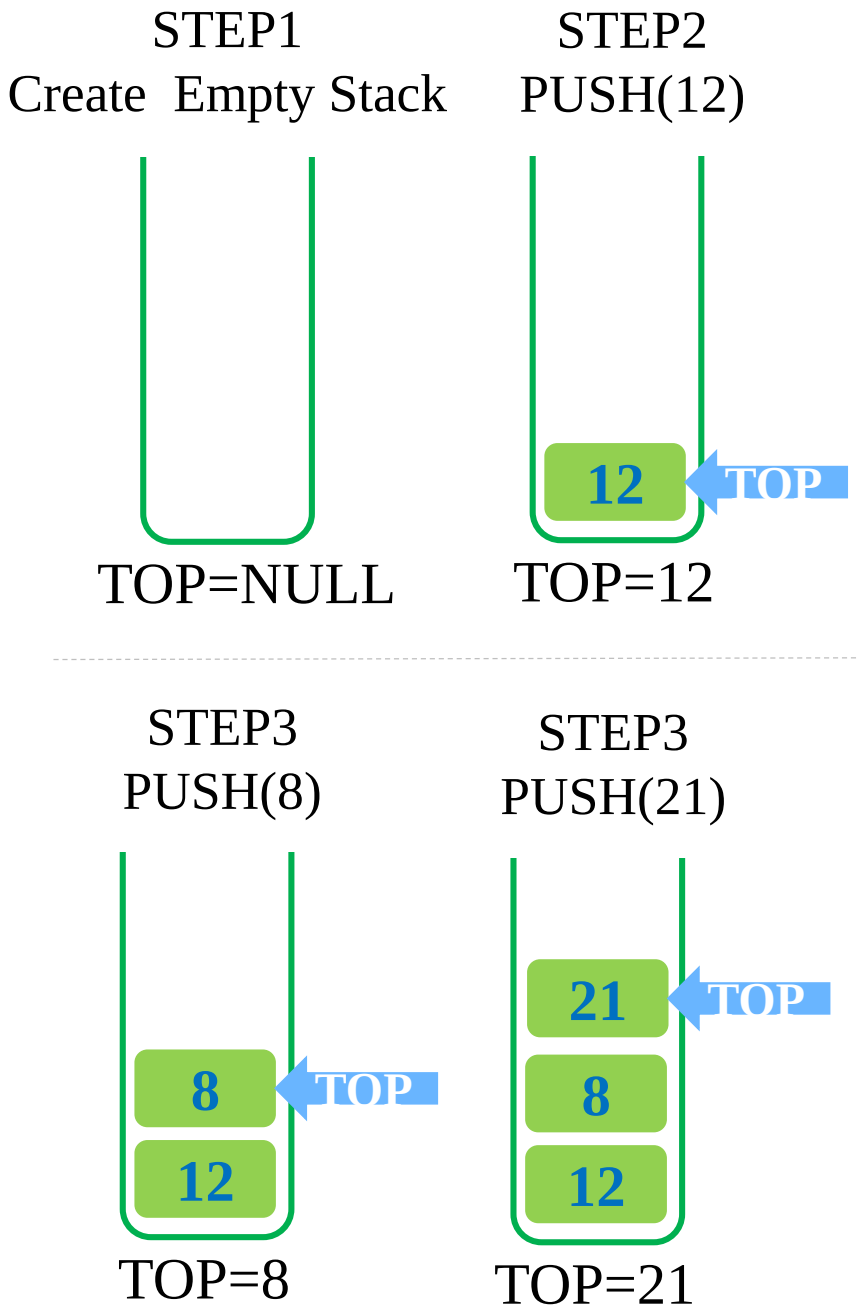


Stack

Simple Implementation

```python
# Stack implementation using list

# create an empty stack
stack = []

# push elements to the end of the list
stack.append(12)
stack.append(8)
stack.append(21)
print(stack)

# get elements from the end of the list
print(stack.pop(-1))
print(stack.pop(-1))
print(stack.pop(-1))
```

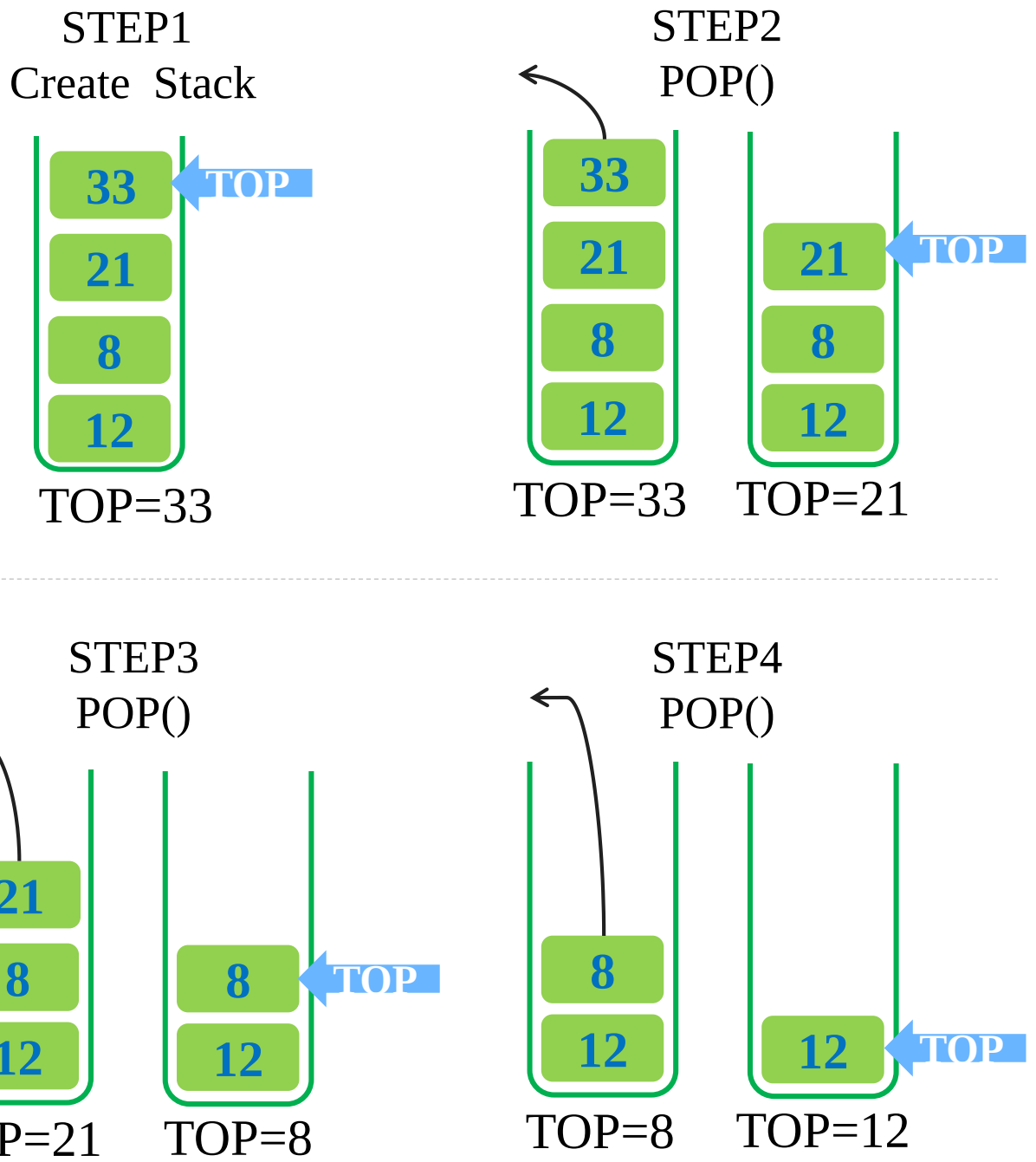
```
[12, 8, 21]
21
8
12
```



Stack

```python
# Stack implementation using list

# create an empty stack
stack = []

# push elements to the end of the list
stack.append(12)
stack.append(8)
stack.append(21)
print(stack)

# get elements from the end of the list
print(stack.pop(-1))
print(stack.pop(-1))
print(stack.pop(-1))
```

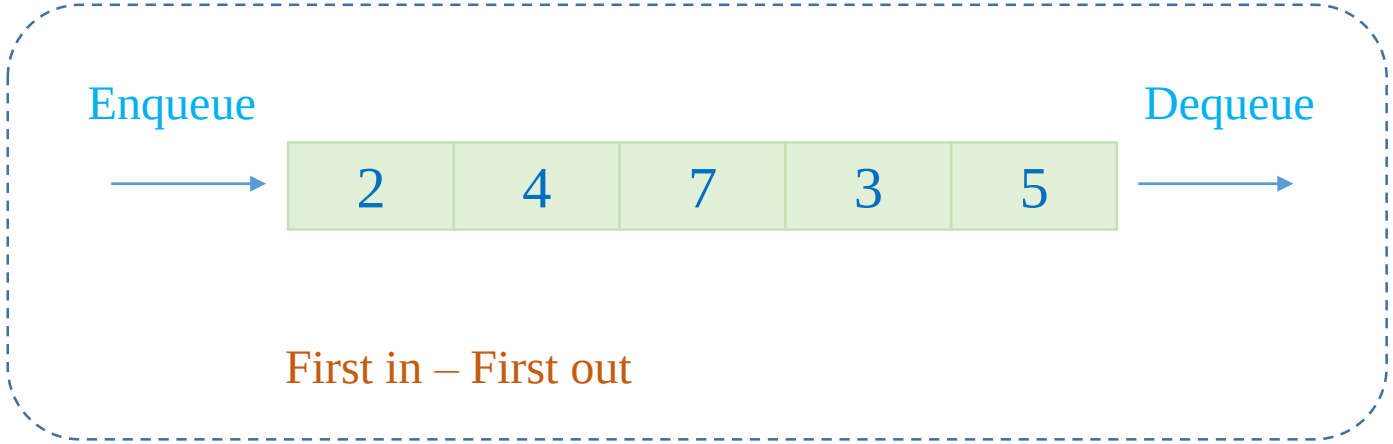
```
[12, 8, 21]
21
8
12
```

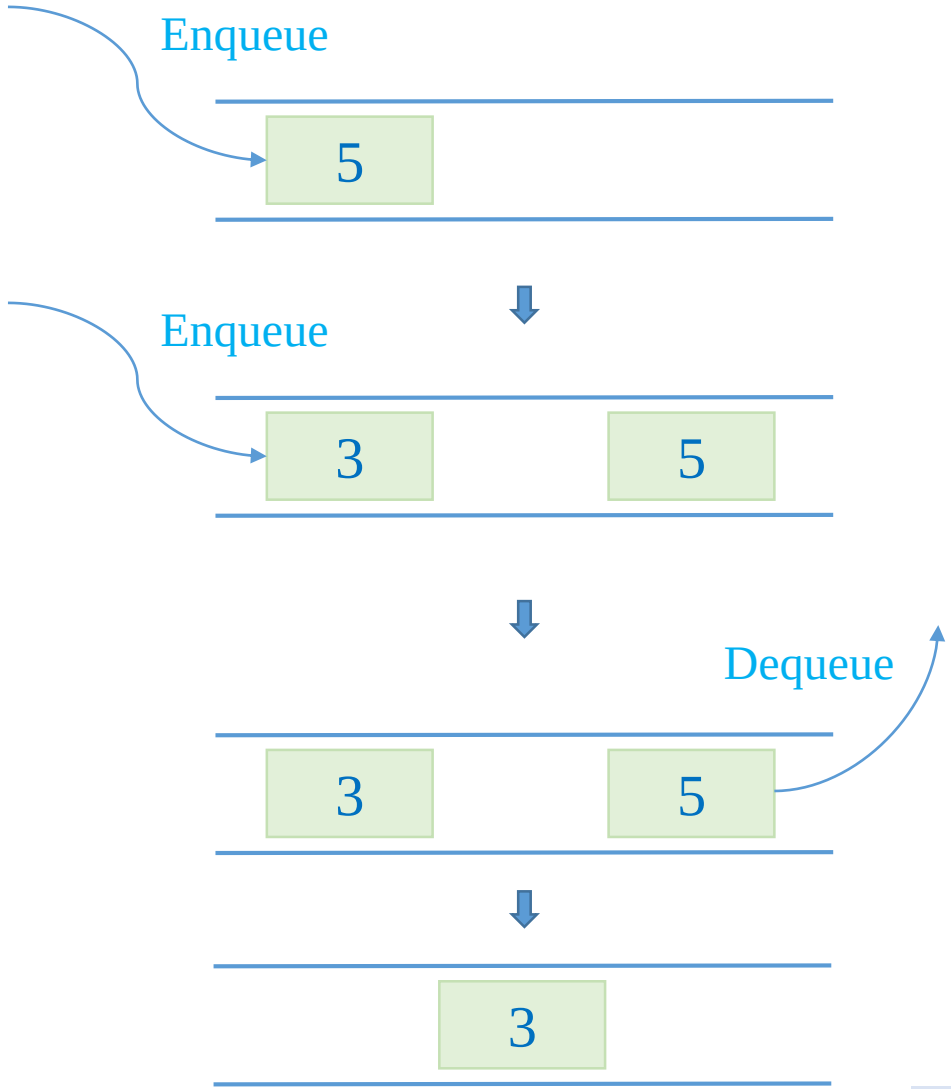


Queue

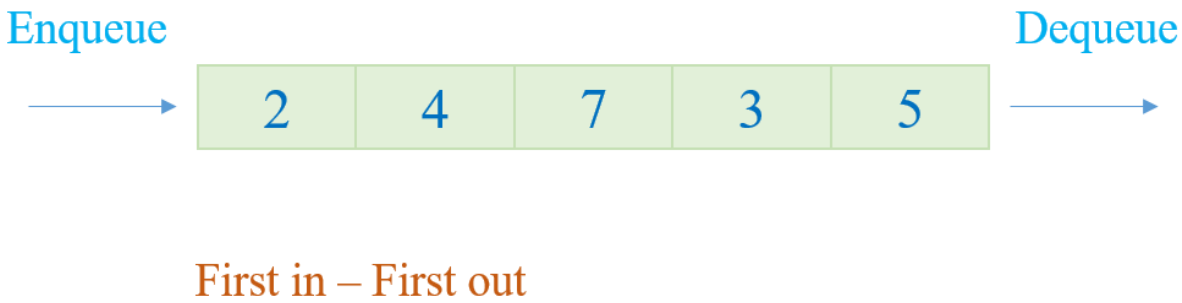
❖ Introduction

Enqueue	Add an element to the end of the queue
Dequeue	Remove an element from the front of the queue





Queue



Simple Implementation

```python
# get elements from the beginning
print(queue.pop(0))
print(queue.pop(0))
print(queue.pop(0))

# print the queue
print(queue)
```

```
5
3
7
[4, 2]
```

```python
# Queue implementation using list

# Initializing a queue
queue = []

# push elements to the end of the list
queue.append(5)
queue.append(3)
queue.append(7)
queue.append(4)
queue.append(2)

# print the queue
print(queue)
```

```
[5, 3, 7, 4, 2]
```

Further Reading