

Data Structure

Tuple, Set, and Dictionary

Quang-Vinh Dinh
Ph.D. in Computer Science

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)
```

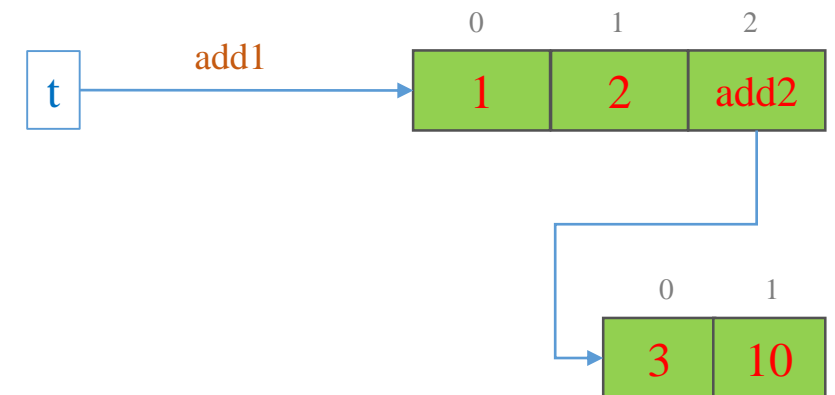
```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-30-b1f85e7d332a> in <module>
      1 t = (1, 2, 3, 4, 5)
----> 2 t[2] = 9
      3 print(t)

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

```
1 def swap(v1, v2):  
2     (v2, v1) = (v1, v2)  
3     return (v1, v2)
```

```
1 v1 = 2  
2 v2 = 3  
3 (v1, v2) = swap(v1, v2)  
4  
5 # print  
6 print(v1)  
7 print(v2)
```

3
2

Tuple slicing

Memory requirement

```
1 # memory comparison  
2 import sys  
3  
4 aList = [3, 4, 5, 6, 7]  
5 aTuple = (3, 4, 5, 6, 7)  
6  
7 print(sys.getsizeof(aList))  
8 print(sys.getsizeof(aTuple))
```

120
80

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

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

list2tuple

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

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

tuple2list

```
1 # convert from tuple to list  
2 aTuple = (3, 4, 5, 6, 7)  
3 aList = list(aList)  
4  
5 print(aList)  
6 print(type(aList))
```

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

Tuple

❖ Example: Solve quadratic equation

```

1 import math
2
3 def quadratic_equation(a, b, c):
4     '''
5     This function aims at solving the quadratic equation
6     a, b, c --- three parameters and a != 0
7     '''
8
9     # compute delta
10    delta = b*b - 4*a*c
11
12    if delta < 0:
13        return ()
14    elif delta == 0:
15        x = (-b+math.sqrt(delta))/(2*a)
16        return (x,)
17    else:
18        x1 = (-b+math.sqrt(delta))/(2*a)
19        x2 = (-b-math.sqrt(delta))/(2*a)
20        return (x1,x2)
21

```

Case 1: delta<0

```

1 result = quadratic_equation(a=5, b=0, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

<class 'tuple'>
0
()

```

Case 2: delta>0

```

1 result = quadratic_equation(a=5, b=5, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

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

```

Case 3: delta=0

```

1 result = quadratic_equation(a=4, b=4, c=1)
2 print(type(result))
3 print(len(result))
4 print(result)

```

```

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

```

Data is protected

```

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

```

```

-----
TypeError                                Traceback (most recent call last)
<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

```
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

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

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

Insert a set to another set

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

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

Join two sets

```
1 # join two sets
2 set1 = {"cat", "dog"}
3 set2 = {"duck", "tiger"}
4
5 set3 = set1.union(set2)
6 print(set3)
```

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

Not allow duplicate values

```
1 # No duplication
2 animals = {"cat", "dog", "tiger"}
3 print(animals)
4
5 animals.add("cat")
6 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'}

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

```
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'}

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

{1, 2, 3, 4, 5}

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

{1, 2, 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.

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

`{'cat', 'tiger'}`

`discard(item)`

Remove an item from the set if it is present.

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

`{'dog', 'cat'}`

Set comprehension

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

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

<https://docs.python.org/3/library/stdtypes.html?#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'}
```

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 \leftrightarrow List and Tuple

```
1 # convert from set to list
2 aSet = {1, 2, 3, 4, 5}
3
4 aList = list(aSet)
5 print(aList)
6 print(type(aList))
```

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

```
1 # convert from set to tuple
2 aSet = {1, 2, 3, 4, 5}
3
4 aTuple = tuple(aSet)
5 print(aTuple)
6 print(type(aTuple))
```

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

```
1 # convert from list to set
2 aList = [1, 2, 3, 2, 1]
3
4 aSet = set(aList)
5 print(aSet)
6 print(type(aSet))
```

???

```
1 # convert from tuple to set
2 aTuple = (1, 2, 3, 2, 1)
3
4 aSet = set(aTuple)
5 print(aSet)
6 print(type(aSet))
```

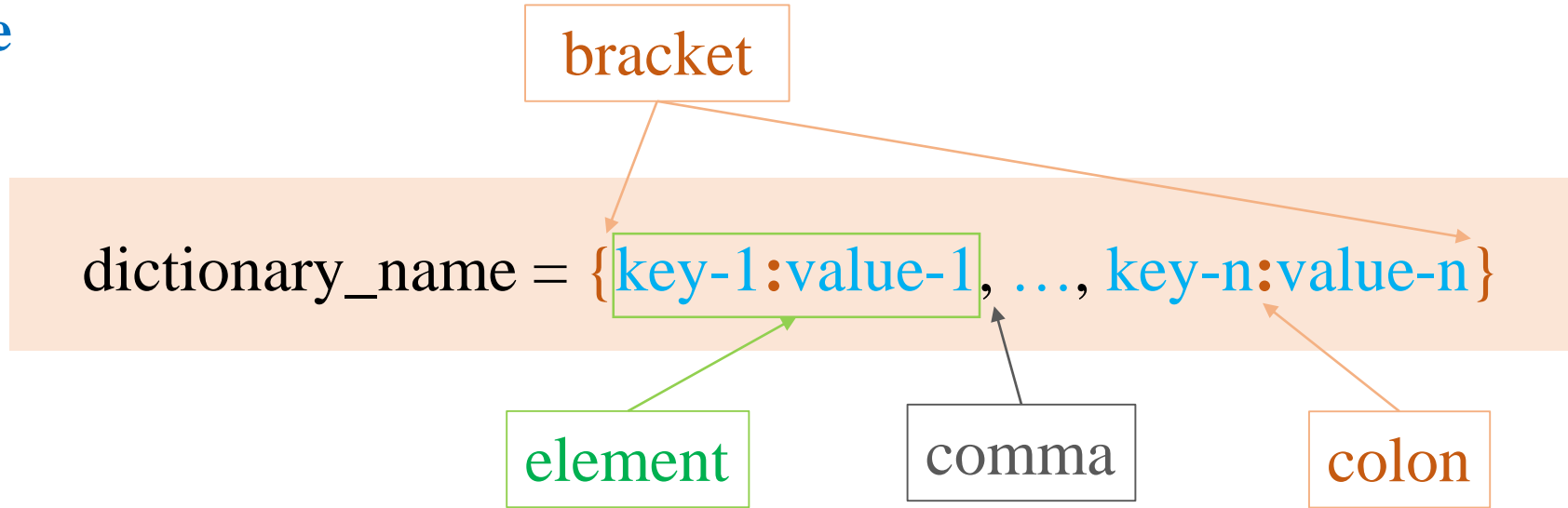
???

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

```
1 # dic comprehension
2
3 a_dict = {str(i):i for i in range(5)}
4 print(a_dict)
```

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

```
1 # from zip
2
3 tuple1 = (1, 2, 3)
4 tuple2 = (4, 5, 6)
5
6 a_dict = dict(zip(tuple1, tuple2))
7 print(type(a_dict))
8 print(a_dict)
```

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

```
1 # from zip
2
3 set1 = {1, 2, 3}
4 set2 = {4, 5, 6}
5
6 a_dict = dict(zip(set1, set2))
7 print(type(a_dict))
8 print(a_dict)
```

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

```
1 # from zip
2
3 list1 = [1, 2, 3]
4 list2 = [4, 5, 6]
5
6 a_dict = dict(zip(list1, list2))
7 print(type(a_dict))
8 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

```
1 parameters = {'learning_rate': 0.1,  
2               'optimizer': 'Adam',  
3               'metric': 'Accuracy'}  
4  
5 value = parameters.get('learning_rate')  
6 print(value)  
7  
8 print('\nAfter using get() function')  
9 print(parameters)
```

0.1

After using get() function

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

Get value and delete the corresponding item

```
1 parameters = {'learning_rate': 0.1,  
2               'optimizer': 'Adam',  
3               'metric': 'Accuracy'}  
4  
5 value = parameters.pop('learning_rate')  
6 print(value)  
7  
8 print('\nAfter using pop() function')  
9 print(parameters)
```

0.1

After using pop() function

{'optimizer': 'Adam', 'metric': 'Accuracy'}

Dictionary

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

```
1 parameters = {'learning_rate': 0.1,  
2               'optimizer': 'Adam',  
3               'metric': 'Accuracy'}  
4  
5 item = parameters.popitem()  
6  
7 print(item)  
8 print(parameters)
```

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

Use del keyword to delete an item

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

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

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

```
1 parameters = {'learning_rate': 0.1,  
2               'metric': 'Accuracy'}  
3  
4 print('Before using clear() function')  
5 print(parameters)  
6  
7 parameters.clear()  
8  
9 print('\nAfter using clear() function')  
10 print(parameters)
```

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

```
After using clear() function  
{}
```

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

```
1 # merge two dicts
2
3 fruits = {'banana': 2, 'apple': 4}
4 cereal = {'rice': 3, 'corn': 7}
5
6 result = {**fruits, **cereal}
7 print(result)
```

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

Check if a key exists

```
1 # check if a key exists
2
3 fruits = {'banana': 2, 'apple': 4}
4
5 print('apple' in fruits)
6 print('corn' in fruits)
```

True
False

Remove empty items

```
1 # remove empty items
2
3 fruits = {'banana': 2, 'apple': None}
4
5 dict1 = {key:value for (key, value)
6           in fruits.items()
7           if value is not None}
8 print(dict1)
```

{'banana': 2}

Common Errors

```
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'
```


Common Errors

```
1. # aivietnam.ai
2. # Lỗi cố gắng lấy phần tử từ dictionary rỗng
3.
4. dict = {'1': 'Python', '5': 'C++'}
5. item1 = dict.popitem()
6. item2 = dict.popitem()
7. 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

```
data.json x
1 {"learning_rate": 0.1,
2  "optimizer": "Adam",
3  "metric": "Accuracy"}
```

```
1 import json
2
3 parameters = {'learning_rate': 0.1,
4               'optimizer': 'Adam',
5               'metric': 'Accuracy'}
6
7 with open('data.json', 'w') as fp:
8     json.dump(parameters, fp)
```

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

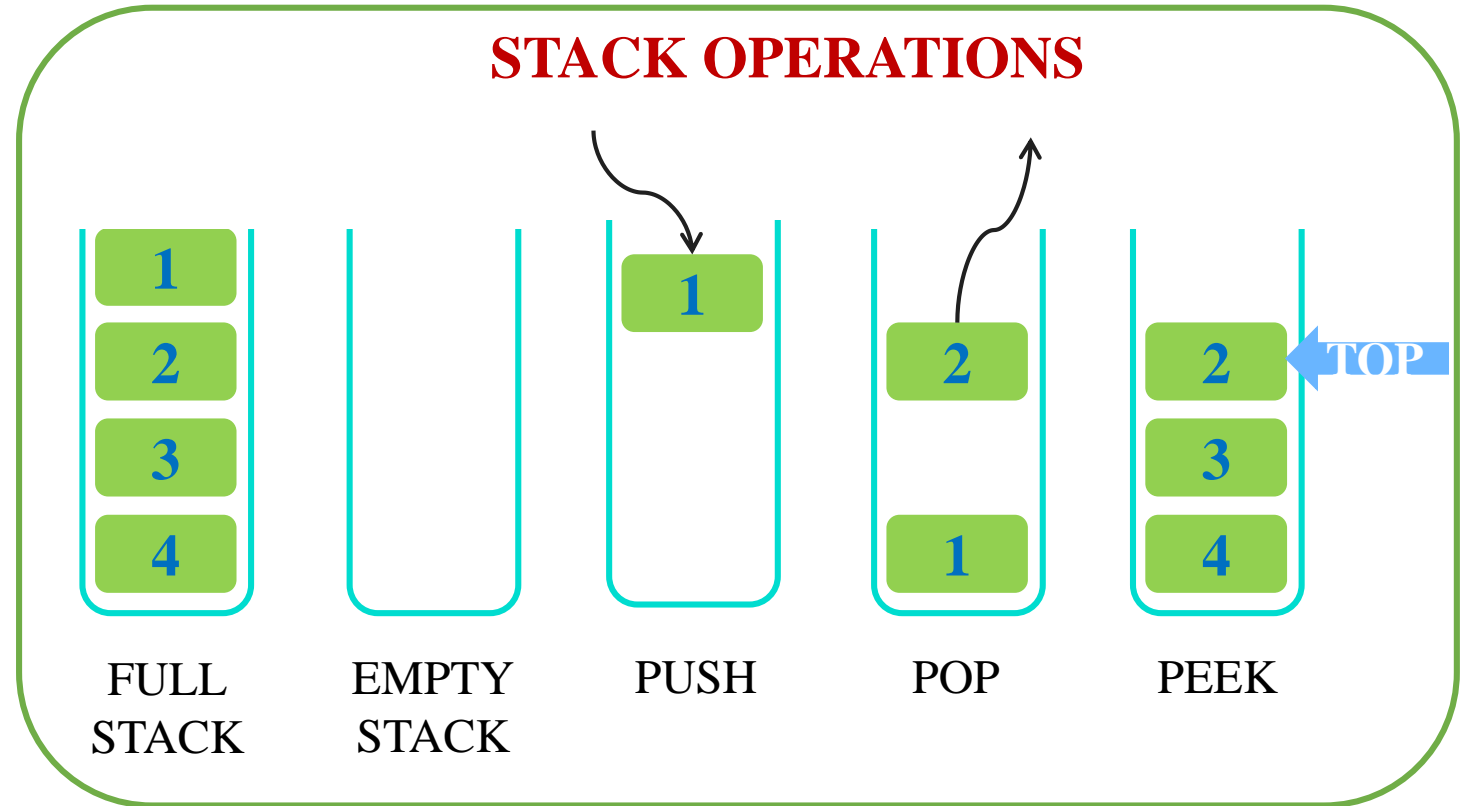
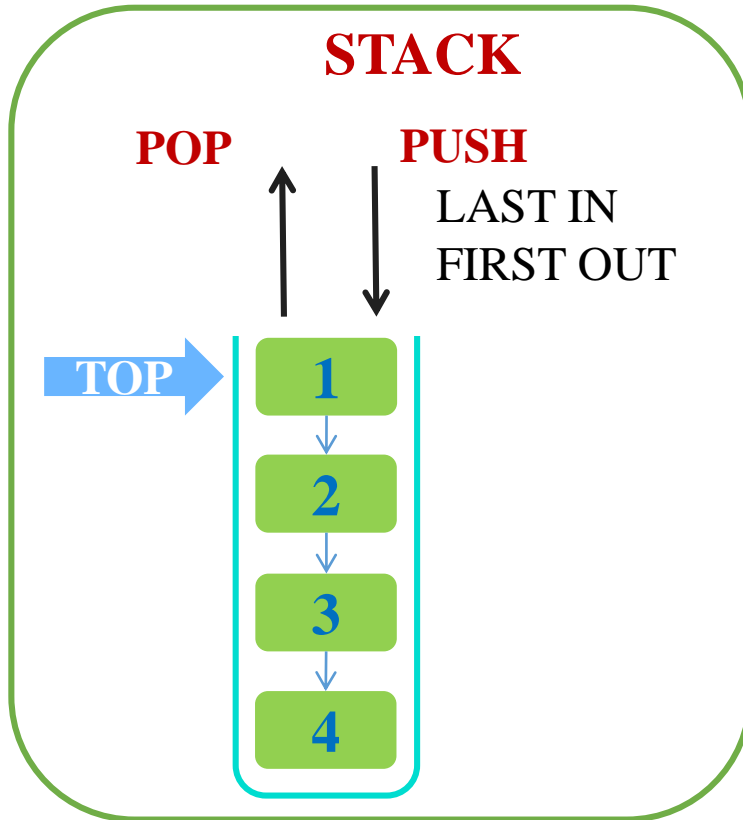
```
{'learning_rate': 0.1, 'optimizer': 'Adam', 'metric': 'Accuracy'}
```

Outline

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

Stack

❖ Introduction



Stack

Simple Implementation

```
1  # Stack implementation using list
2
3  # create an empty stack
4  stack = []
5
6  # push elements to the end of the list
7  stack.append(12)
8  stack.append(8)
9  stack.append(21)
10 print(stack)
11
12 # get elements from the end of the list
13 print(stack.pop(-1))
14 print(stack.pop(-1))
15 print(stack.pop(-1))
```

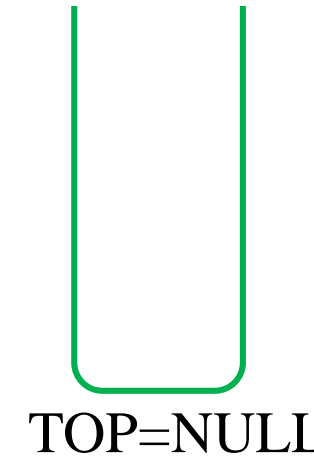
[12, 8, 21]

21

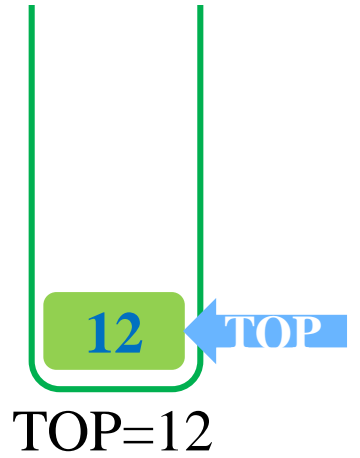
8

12

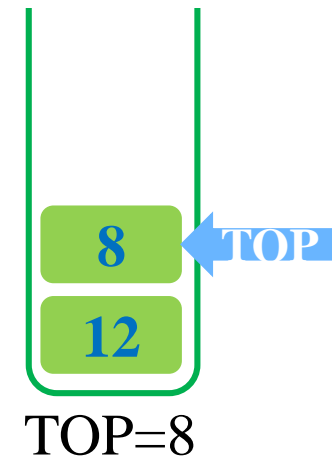
STEP1
Create Empty Stack



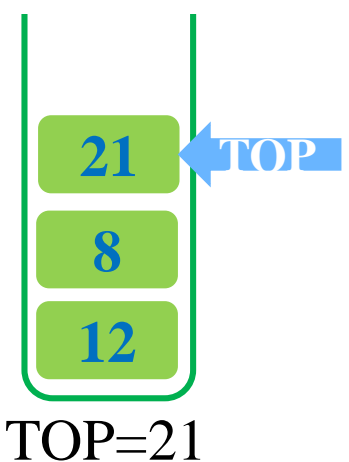
STEP2
PUSH(12)



STEP3
PUSH(8)



STEP3
PUSH(21)



Stack

```
1 # Stack implementation using list
2
3 # create an empty stack
4 stack = []
5
6 # push elements to the end of the list
7 stack.append(12)
8 stack.append(8)
9 stack.append(21)
10 print(stack)
11
12 # get elements from the end of the list
13 print(stack.pop(-1))
14 print(stack.pop(-1))
15 print(stack.pop(-1))
```

[12, 8, 21]

21

8

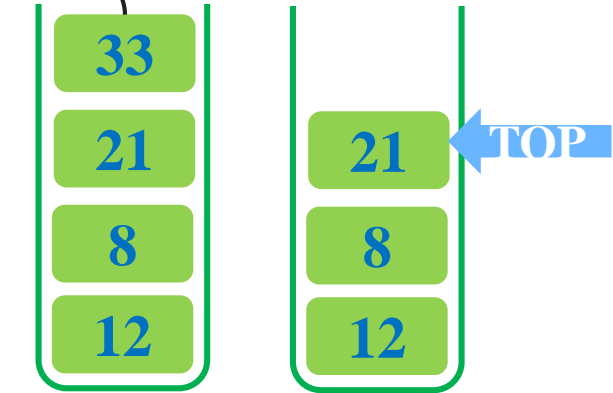
12

STEP1
Create Stack



TOP=33

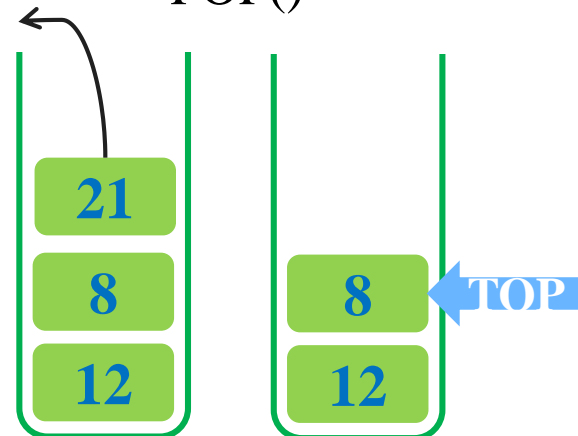
STEP2
POP()



TOP=33

TOP=21

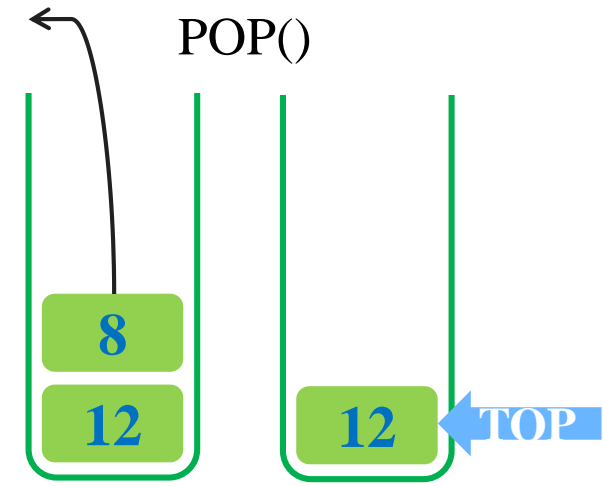
STEP3
POP()



TOP=21

TOP=8

STEP4
POP()



TOP=8

TOP=12

Queue

❖ Introduction

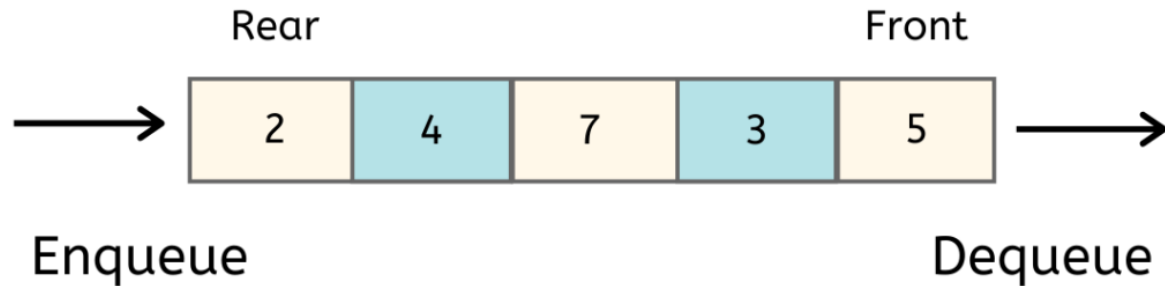
Enqueue

Add an element to the end of the queue

Dequeue

Remove an element from the front of the queue

Queue



First in First out
(FIFO)

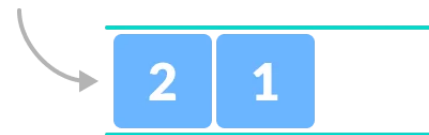


full queue

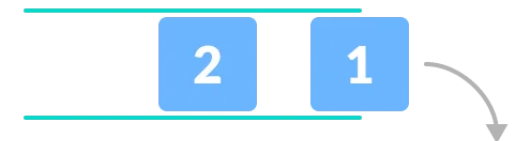


empty queue

enqueue



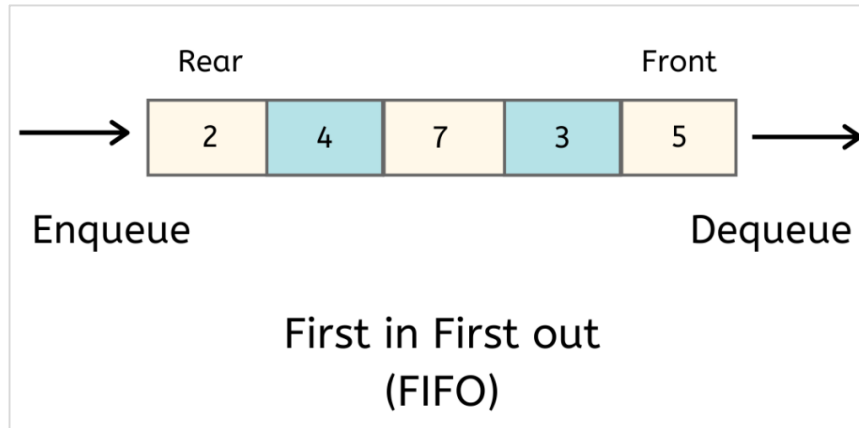
enqueue



dequeue

Queue

Simple Implementation



```
1 # get elements from the beginning
2 print(queue.pop(0))
3 print(queue.pop(0))
4 print(queue.pop(0))
5
6 # print the queue
7 print(queue)
```

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

```
1 # Queue implementation using list
2
3 # Initializing a queue
4 queue = []
5
6 # push elements to the end of the list
7 queue.append(2)
8 queue.append(4)
9 queue.append(7)
10 queue.append(3)
11 queue.append(5)
12
13 # print the queue
14 print(queue)
```

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


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

```
1 import random
2
3 random.seed(42)
4 my_list = [random.randint(0, 100) for i in range(100000)]
```

```
1 %%time
2
3 # slower
4 list_unique = []
5 for num in my_list:
6     if num not in list_unique:
7         list_unique.append(num)
```

Wall time: 132 ms

```
1 %%time
2
3 # faster
4 list_unique = list(set(my_list))
```

Wall time: 2.23 ms

Tính tổng các phần tử trong list

```
1 import random
2
3 random.seed(42)
4 my_list = [random.randint(0, 100) for i in range(100000)]
```

```
1 %%time
2
3 # slower
4 result = 0
5 for num in my_list:
6     result += num
7
8 print(result)
```

5007791
Wall time: 7.98 ms

```
1 %%time
2
3 # faster
4 result = sum(my_list)
5 print(result)
```

5007791
Wall time: 998 µs

Code Optimization (2)

❖ Use built-in functions

Dùng list comprehension

```
1 import random
2
3 random.seed(42)
4 my_list = [random.randint(0, 100) for i in range(100000)]
```

```
1 %%time
2
3 # slower
4 new_list = []
5 for num in my_list:
6     new_list.append(num*2)
7
8 #print(new_list)
```

Wall time: 8.98 ms

```
1 %%time
2
3 # faster
4 new_list = [num*2 for num in my_list]
5 #print(new_list)
```

Wall time: 3.99 ms

Đếm số phần tử trong list dùng Counter

```
1 import random
2
3 random.seed(42)
4 my_list = [random.randint(0, 10) for i in range(100000)]
```

```
1 %%time
2
3 # slower
4 my_dict = {}
5
6 for char in my_list:
7     if char not in my_dict:
8         my_dict[char] = 1
9
10    my_dict[char] += 1
11
12 #print(my_dict)
```

Wall time: 21.9 ms

```
1 %%time
2
3 from collections import Counter
4
5 my_dict = Counter(my_list)
6 #print(my_dict)
```

Wall time: 7.99 ms

Bad Codes

```
1 # bad code 1
2
3 fruits= [ 'apple' , "lemon", 'banana']
```

```
1 # bad code 2
2
3 x = 1
4 y = 2
5 z = 3
6
7 f1 = x + y - z
8 f2 = x/y + z
9 f3 = x - y*z
10 f4 = (x/y) * z
```

```
1 # bad code 3
2
3 weights_1 = [1.0, 2.5, 3.7]
4 weights_2 = [1.0, 2.5, 3.7]
5 weights_3 = [1.0 , 2.5 , 3.7]
```

```
1 # bad code 4
2
3 'aivietnam' . upper()
```

```
1 # bad code 5
2
3 print ('aivietnam')
```

```
1 # bad code 6
2
3 fruits = ['apple', 'lemon', 'banana']
4 apple = fruits [0]
5 sub_set = fruits [:2]
```

Bad Codes

```
1 # bad code 7
2
3 weights_1 = [1.0, 2.5, 3.7]
4 weights_2 = [ 1.0, 2.5, 3.7 ]
```

```
1 # bad code 8
2
3 print('AI VIETNAM') # a comment
4 print('AI VIETNAM') # a comment
```

```
1 # bad code 9
2
3 class AdamOptimizer:
4     def exampleMethod1():
5         # ...
6     def exampleMethod2():
7         #...
8 def train():
9     #...
```

```
1 # bad code 10
2
3 import math, os, sys
4
5 # -- or --
6
7 import math
8 import os
9 import sys
```

Bad Codes

```
1 # bad code 11 - style
2
3 fruits = ['apple', 'lemon', 'banana']
4 apple = fruits[0]
5 sub_set = fruit[:2]
6
7 # -- or --
8
9 fruits = ['apple', 'lemon', 'banana']
10 apple = fruits[0]
11 sub_set = fruit[:2]
```

```
1 # bad code 12
2
3 fruits = ['apple', 'lemon', 'banana']
4 number_of_fruits = len(fruits)
5 print(number_of_fruits)
6
7 len = 3
8 print(len(fruits))
```

```
1 # bad code 13 - magic number
2
3 radius = 4
4 area = radius*radius*3.14159
```

```
1 # bad code 14 - one entry, one exit
2
3 def ReLU(number):
4     if (number <= 0):
5         return 0
6     else:
7         return number
8
9 print(ReLU(-8))
```

Bad Codes

```
1  # bad code 15 - dead code
2
3  def ReLU(number):
4      if (number <= 0):
5          return 0
6      else:
7          return number
8
9      return 'Input is not a number!'
10
11 print(ReLU(-8))
```

```
1  # bad code 16 - comment
2
3  def flip(times):
4      number_of_heads = 0
5      number_of_tails = 0
6
7      for _ in range(times):
8          number = random.randint(0, 1)
9          if (number == 1):
10             number_of_heads = number_of_heads + 1
11          else:
12             number_of_tails = number_of_tails + 1
13
14      return number_of_heads, number_of_tails
15
16 number_of_heads, number_of_tails = flip(1000)
17 print(number_of_heads)
18 print(number_of_tails)
```

Bad Codes

```
1  # bad code 17 - global
2
3  number_of_heads = 0
4  number_of_tails = 0
5
6  def flip(times):
7      global number_of_heads
8      global number_of_tails
9
10     for _ in range(times):
11         number = random.randint(0, 1)
12         if (number == 1):
13             number_of_heads = number_of_heads + 1
14         else:
15             number_of_tails = number_of_tails + 1
16
17 flip(1000)
18 print(number_of_heads)
19 print(number_of_tails)
```

```
1  # bad code 18 - make thing complicated
2
3  import random
4  class Dice:
5      def __init__(self, sides=6):
6          self.sides = sides
7      def roll(self):
8          return random.randint(1, self.sides)
9
10 d = Dice()
11 print('You rolled a', d.roll())
```


Bad Codes

```
1 # bad code 19
2
3 numbers = []
4
5 for i in range(1, 100):
6     if (i%5 == 0):
7         numbers.append(i)
8
9 print(numbers)
```

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

```
1 # bad code 21
2
3 # open a file
4 a_file = open('hello_world.txt', 'w')
5
6 # write data to file
7 text3 = 'writing line \n'
8 a_file.write(text3)
```

```
1 # bad code 20
2
3 try:
4     num = input('Enter a number: ')
5     num = int(num)
6 except ValueError:
7     pass # do nothing
```

Bad Codes

```
1  # bad code 22 - Unpythonic
2
3  path = 'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
4  print(path)
5
6  with open(path, 'r') as file:
7      lines = file.readlines()
8      print(lines)
```

```
1  path = 'E:\\Data\\AICourse-2021\\1.BasicPython\\file\\hello_world.txt'
2  print(path)
3
4  with open(path, 'r') as file:
5      lines = file.readlines()
6      print(lines)
```

```
1  path = r'E:\Data\AICourse-2021\1.BasicPython\file\hello_world.txt'
2  print(path)
3
4  with open(path, 'r') as file:
5      lines = file.readlines()
6      print(lines)
```

Bad Codes

```
1 # bad code 23
2
3 name = 'John'
4 age  = 26
5
6 print('Hello ' + name + '. Are you ' + str(age) + '?')
```

```
1 name = 'John'
2 age  = 26
3
4 print(f'Hello {name}. Are you {age}?')
```

```
1 # bad code 24 - Unpythonic
2
3 fruits = {'banana': 2}
4 if 'apple' not in fruits:
5     fruits['apple'] = 0
6
7 fruits['apple'] += 10
8 print(fruits)
```

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

```
1 # bad code 25
2
3 def get_salary_rate(employee_class):
4     if employee_class == 'level-1':
5         result = 5.7
6     elif employee_class == 'level-2':
7         result = 4.2
8     elif employee_class == 'level-3':
9         result = 3.8
10    elif employee_class == 'level-4':
11        result = 3.3
12    else:
13        result = 2.9
14
15    return result
16
17 print(get_salary_rate('level-1'))
18 print(get_salary_rate('level-4'))
```

```
1 salary_rates = {'level-1': 5.7,
2                  'level-2': 4.2,
3                  'level-3': 3.8,
4                  'level-4': 3.3,
5                  'level-5': 2.9}
6
7 print(salary_rates['level-1'])
8 print(salary_rates['level-4'])
```

Bad Codes

```
1 # bad code 26 - Unpythonic
2
3 def a_function(value):
4     # do something
5
6     if 1 < value and value < 10:
7         print('code inside if')
8     else:
9         print('code inside else')
10
11     # do something and return something
12
13
14 a_function(4)
15 a_function(40)
```

```
1 def a_function(value):
2
3     # do something
4
5     if 1 < value < 10:
6         print('code inside if')
7     else:
8         print('code inside else')
9
10     # do something and return something
11
12
13 a_function(4)
14 a_function(40)
```

```
1 # bad code 27
2
3 def contain_fruit(a_fruit):
4     if (a_fruit == 'banana'):
5         result = True
6     elif (a_fruit == 'apple'):
7         result = True
8     elif (a_fruit == 'peach'):
9         result = True
10    else:
11        result = False
12
13    return result
14
15 print(contain_fruit('banana'))
16 print(contain_fruit('pineapple'))
```

```
1 def contain_fruit(a_fruit):
2     result = a_fruit in ('banana',
3                           'apple',
4                           'peach')
5
6     return result
7
8 print(contain_fruit('banana'))
9 print(contain_fruit('pineapple'))
```

Bad Codes

```
1 # super bad code 28
2
3 value = True + True + False + True
4 char1 = 'aivietnam'[False]
5 char2 = 'aivietnam'[True]
6 char3 = 'aivietnam'[-True]
7
8 print(value)
9 print(char1)
10 print(char2)
11 print(char3)
```

```
1 # bad code 29
2
3 def add_fruit(a_fruit, fruits=['apple']):
4     fruits.append(a_fruit)
5
6     return fruits
```

```
1 fruits = add_fruit('banana')
2 print(fruits)
3
4 fruits = add_fruit('banana')
5 print(fruits)
```

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

Bad Codes

```
1  # bad code 30
2
3  import math
4
5  def quadratic_equation(a, b, c):
6      # compute delta
7      delta = b*b - 4*a*c
8
9      if delta < 0:
10         result = 'The equation has no solution'
11     elif delta == 0:
12         x = (-b+math.sqrt(delta))/2*a
13         result = (x,)
14     else:
15         x1 = (-b+math.sqrt(delta))/(2*a)
16         x2 = (-b-math.sqrt(delta))/(2*a)
17         result = (x1,x2)
18
19     return result
20
21 print(quadratic_equation(3, 2, 1))
22 print(quadratic_equation(1, 2, 1))
```

Good Code

```
1  # swap
2
3  x, y = 3, 4
4  print(x, y)
5
6  x, y = y, x
7  print(x, y)
```

3 4
4 3

```
1  # condition
2
3  n = 8
4  result = 1 < n < 10
5  print(result)
```

True

```
1  # reverse a string
2
3  name = "ai vietname"
4  print(name)
5  print(name[::-1])
```

ai vietname
emanteiv ia

```
1  # join
2
3  a = ["Hi", "AI", "VIETNAM"]
4  print(" ".join(a))
```

Hi AI VIETNAM

Good Code

```
1 # unpacking
2
3 a_list = [1, 2, 3]
4 x, y, z = a_list
5
6 print(x, y, z)
```

1 2 3

```
1 # check if contained
2 m = 1
3
4 if m in [1, 3, 5, 7]:
5     print('Contained!')
```

Contained!

```
1 # enumerate
2
3 a_list = [4, 5, 6]
4 for i, value in enumerate(a_list):
5     print(i, ': ', value)
```

0 : 4
1 : 5
2 : 6

Good Code

```
1  # Unpacking operator
2
3  def print_data(x, y, z):
4      print(x, y, z)
5
6  a_dict = {'x': 1, 'y': 2, 'z': 3}
7  a_list = [3, 4, 5]
8
9  print_data(*a_dict)
10 print_data(**a_dict)
11 print_data(*a_list)
```

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

```
1  # Use a dictionary to store a switch
2
3  ops = {
4      'addition': lambda x, y: x + y,
5      'subtraction': lambda x, y: x - y,
6      'multiplication': lambda x, y: x * y,
7      'division': lambda x, y: x / y
8  }
9
10 print(ops['addition'](4, 6))
11 print(ops['multiplication'](4, 6))
```

```
10
24
```

Good Code

```
1 # get the most frequent element
2
3 test = [1, 2, 3, 4, 2]
4
5 print(max(test))
6 print(max(test, key=test.count))
```

4

2

```
1 # create dict from two tuples
2
3 t1 = (1, 2, 3)
4 t2 = (10, 20, 30)
5
6 a_dict = dict(zip(t1,t2))
7 print(a_dict)
```

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

Further Reading

