AI VIET NAM
@aivietnam.edu.vn

AI VIET NAM – AI COURSE 2025

# Exercise: Object Oriented Programming

Dinh-Thang Duong và Quang-Vinh Dinh

# I. Câu hỏi tự luận

## I.1. Xây dựng class tính Sigmoid

**Sigmoid Function:** Sigmoid Function, hay còn được gọi là logistic function, là một trong những activation function cơ bản nhất trong machine learning và neural networks. Hình dạng của nó giống như chữ "S"nằm ngang. Sigmoid chuyển đổi mọi giá trị đầu vào thành một giá trị đầu ra nằm giữa 0 và 1, với một sự chuyển đổi mượt mà tại giá trị 0.

**Ứng Dụng:** Sigmoid Function thường được sử dụng trong các bài toán phân loại nhị phân, nơi mà mục tiêu là phân loại đầu vào thành một trong hai lớp.

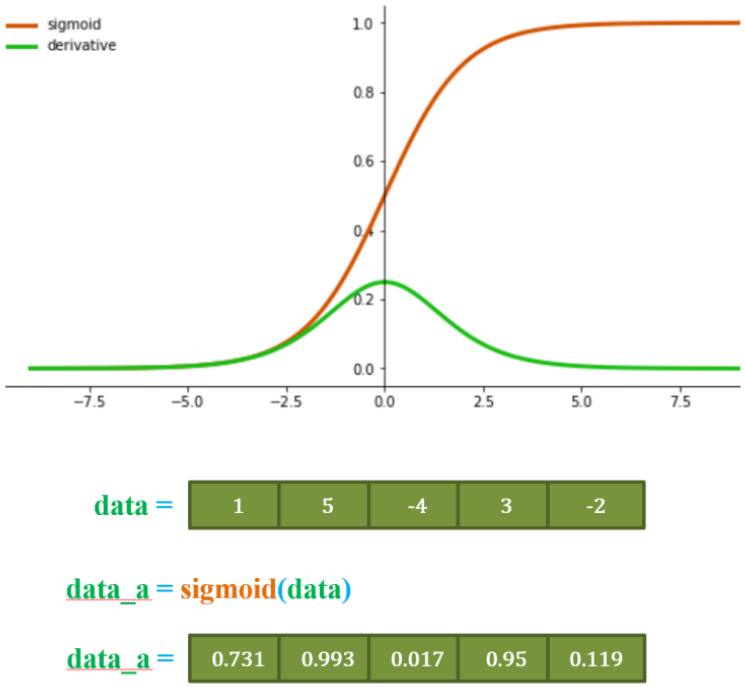
**Ưu Điểm:**

- **Dễ hiểu và triển khai:** Do tính chất đơn giản và phổ biến, Sigmoid rất được triển khai trong nhiều loại mạng neuron.
- **Đầu ra nằm trong khoảng (0,1):** Giá trị đầu ra luôn nằm trong khoảng từ 0 đến 1, giúp dễ dàng diễn giải như là xác suất.

**Nhược điểm:**

- **Vanishing gradient problem**: Khi đầu vào có giá trị lớn hoặc nhỏ, đạo hàm của Sigmoid tiệm cận đến 0, dẫn đến vấn đề vanishing gradient, làm chậm quá trình học của mạng.
- **Tâm đối xứng không nằm tại 0**: Tâm đối xứng không nằm tại điểm 0, điều này có thể gây ra vấn đề trong việc điều chỉnh trọng số trong quá trình học.

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




Hình 1: Hàm Sigmoid và đạo hàm.

```python
# Examples 1
import torch

# input data
x = torch.tensor([1.0, 5.0, -4.0])

# sigmoid function
output = torch.sigmoid(x)
print(output)
>> tensor([0.7311, 0.9933, 0.0180])
```

## I.2. Xâu dựng Ward

Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher. Một student gồm có name, yob (int) (năm sinh), và grade (string). Một teacher gồm có name, yob, và subject (string). Một doctor gồm có name, yob, và specialist (string). Lưu ý cần sử dụng a list để chứa danh sách của mọi người trong Ward.

1. Thực hiện các class student, doctor, và teacher theo mô tả trên. Thực hiện describe() method để print ra tất cả thông tin của các objects.

2. Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward.

3. Viết countDoctor() method để đếm số lượng doctor trong ward.
4. Viết sortAge() method để sort mọi người trong ward theo tuổi của họ với thứ tự tăng dần (Gợi ý: Có thể sử dụng sort của list hoặc viết thêm function đều được).
5. Viết aveTeacherYearOfBirth() method để tính trung bình năm sinh của các teachers trong ward.

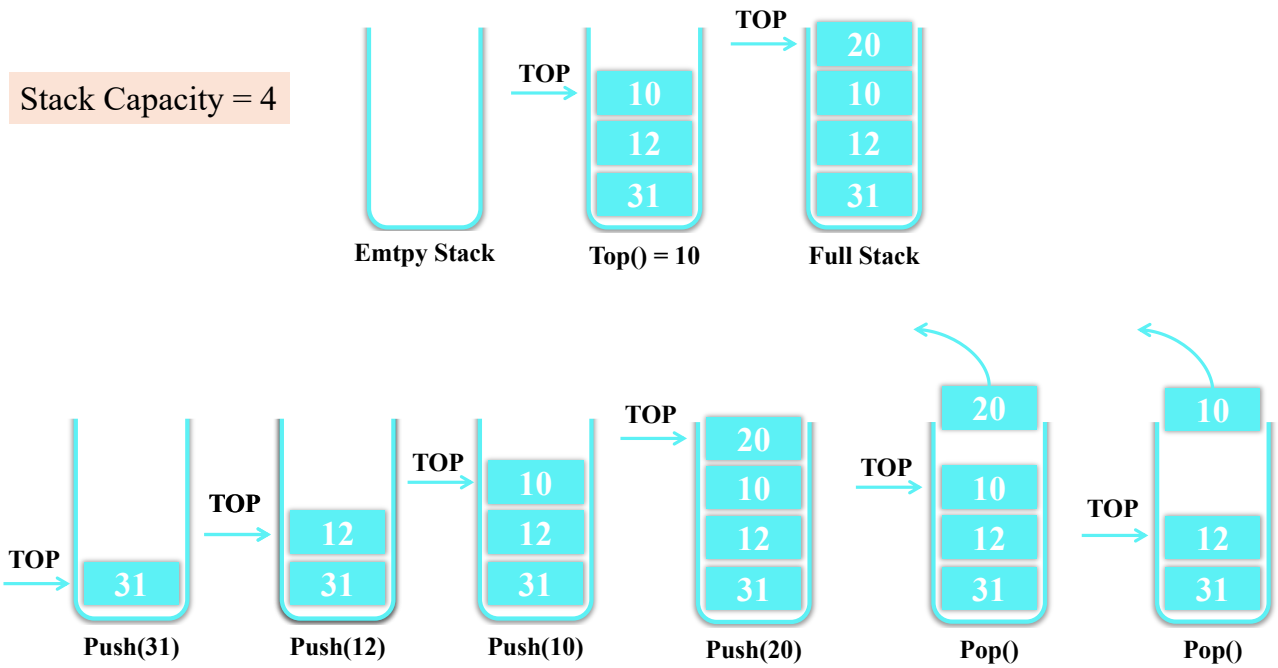
```
# Examples
# 2(a)
student1 = Student(name="studentA", yob=2010, grade="7")
student1.describe()
#output
>> Student - Name: studentA - YoB: 2010 - Grade: 7

teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
teacher1.describe()
#output
>> Teacher - Name: teacherA - YoB: 1969 - Subject: Math

doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
doctor1.describe()
#output
>> Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists


# 2(b)
print()
teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward1")
ward1.addPerson(student1)
ward1.addPerson(teacher1)
ward1.addPerson(teacher2)
ward1.addPerson(doctor1)
ward1.addPerson(doctor2)
ward1.describe()

#output
>> Ward Name: Ward1
Student - Name: studentA - YoB: 2010 - Grade: 7
Teacher - Name: teacherA - YoB: 1969 - Subject: Math
Teacher - Name: teacherB - YoB: 1995 - Subject: History
Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists
Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists

# 2(c)
print(f"\nNumber of doctors: {ward1.countDoctor()}")

#output
>> Number of doctors: 2

# 2(d)
print("\nAfter sorting Age of Ward1 people")
```

```
ward1.sortAge()
ward1.describe()

#output
>> After sorting Age of Ward1 people
Ward Name: Ward1
Student - Name: studentA - YoB: 2010 - Grade: 7
Teacher - Name: teacherB - YoB: 1995 - Subject: History
Doctor - Name: doctorB - YoB: 1975 - Specialist: Cardiologists
Teacher - Name: teacherA - YoB: 1969 - Subject: Math
Doctor - Name: doctorA - YoB: 1945 - Specialist: Endocrinologists

# 2(e)
print(f"\nAverage year of birth (teachers): {ward1.aveTeacherYearOfBirth()}")

#output
>> Average year of birth (teachers): 1982.0
```

## I.3. Xây dựng class Stack

Thực hiện xây dựng class Stack với các chức năng (method) sau đây:

- **initialization** method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa.
- **.isEmpty()**: kiểm tra stack có đang rỗng.
- **.isFull()**: kiểm tra stack đã full chưa.
- **.pop()**: loại bỏ top element và trả về giá trị đó.
- **.push(value)** add thêm value vào trong stack.
- **.top()** lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó.
- **Lưu ý:** Không cần thiết phải thực hiện với pointer như trong hình minh họa số 2.




Hình 2: Hình ảnh minh họa về các Stack và một số phương thức trên Stack.

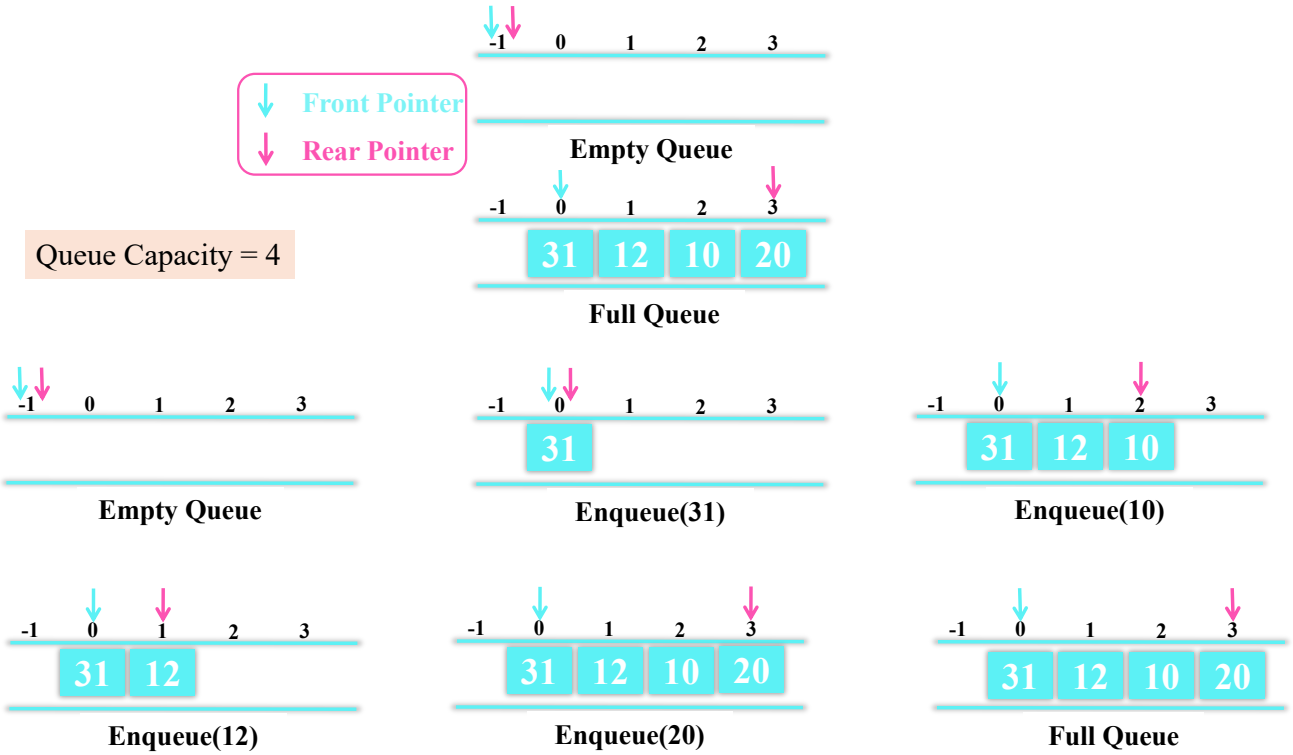
```
stack1 = MyStack(capacity=5)

stack1.push(1)

stack1.push(2)

print(stack1.isFull())
>> False

print(stack1.top())
>>2

print(stack1.pop())
>> 2

print(stack1.top())
>> 1

print(stack1.pop())
>> 1

print(stack1.isEmpty())
>> True
```

## I.4. Xây dựng class Queue

Thực hiện xây dựng class Queue với các chức năng (method) sau đây:

- **initialization** method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa
- **.isEmpty()**: kiểm tra queue có đang rỗng
- **.isFull()**: kiểm tra queue đã full chưa
- **.dequeue()**: loại bỏ first element và trả về giá trị đó
- **.enqueue(value)** add thêm value vào trong queue
- **.front()** lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó
- **Không cần thiết phải thực hiện với các pointers như trong hình minh họa**

```
queue1 = MyQueue(capacity=5)

queue1.enqueue(1)

queue1.enqueue(2)

print(queue1.isFull())
>> False

print(queue1.front())
>> 1

print(queue1.dequeue())
>> 1

print(queue1.front())
>> 2

print(queue1.dequeue())
>> 2

print(queue1.isEmpty())
>> True
```




Hình 3: Hình ảnh minh họa về các Queue và một số phương thức trên Queue.

# II. Câu hỏi trắc nghiệm

1. (Repeat PyTorch code) Kết quả của đoạn code dưới đây là gì?

```python
import torch

# input data
x = torch.tensor([5.0, 3.0])

# sigmoid function
output = torch.sigmoid(x)
print(output)
```

   (a) `[0.0900, 0.9526]`.

   (b) `[0.9933, 0.6652]`.

   (c) `[0.9933, 0.9526]`.

   (d) `[0.1900, 0.2447]`.

2. Hoàn thành đoạn code sau đây theo công thức tính sigmoid và cho biết kết quả in ra màn hình là gì?

```python
import torch
import torch.nn as nn

class MySigmoid(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        ### Your Code Here

        ### End Code Here

data = torch.Tensor([3.0, -2.0])
my_sigmoid = MySigmoid()
output = my_sigmoid(data)
output
```

   (a) `[0.1192, 0.9526]`.

   (b) `[0.9526, 0.1192]`.

   (c) `[0.1192, 0.2447]`.

   (d) `[0.7054, 0.1192]`.

3. Một người (person) có thể là student, doctor, hoặc teacher. Một student gồm có name (string), yob (int) (năm sinh), và grade (string). Các bạn thực hiện viết class Student theo mô tả trên

(Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object). Theo đó, kết quả đầu ra khi thực hiện lời gọi method describe() là gì?

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Student(Person):
    def __init__(self, name:str, yob:int, grade:str):
        ### Your Code Here

        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

student1 = Student(name="studentZ2023", yob=2011, grade="6") student1.describe()
```

(a) Student - Name: studentZ2023 - YoB: 2011 - Grade: 6.

(b) Student - Name: studentZ2023 - YoB: 6 - Grade: 2011.

(c) Student - Name: 6 - YoB: studentZ2023 - Grade: 2011.

(d) Tất cả đều sai.

4. Một người (person) có thể là student, doctor, hoặc teacher. Một teacher gồm có name (string), yob (int), và subject (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để in ra mànhình tất cả thông tin của object). Theo đó, kết quả đầu ra khi thực hiện lời gọi method describe() là gì?

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
```

```
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Teacher(Person):
    def __init__(self, name:str, yob:int, subject:str):
        ### Your Code Here

        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

teacher1 = Teacher(name="teacherZ2023", yob=1991, subject="History") teacher1.
                                                describe()
```

(a) Teacher - Name: 1991 - YoB: teacherZ2023 - Subject: History.

(b) Teacher - Name: teacherZ2023 - YoB: 1991 - Subject: History.

(c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991.

(d) Tất cả đều sai.

5. Một người (person) có thể là student, doctor, hoặc teacher. Một doctor gồm có name (string), yob (string), và specialist (string). Các bạn thực hiện viết class Teacher theo mô tả trên (Các bạn sẽ viết thêm describe() method để print ra tất cả thông tin của object). Theo đó, kết quả đầu ra khi thực hiện lời gọi method describe() là gì?

```
from abc import ABC, abstractmethod

class Person(ABC):
    def __init__(self, name:str, yob:int):
        self._name = name
        self._yob = yob

    def getYoB(self):
        return self._yob

    @abstractmethod
    def describe(self):
        pass


class Doctor(Person):
    def __init__(self, name:str, yob:int, specialist:str):
        ### Your Code Here

```

```
        ### End Code Here

    def describe(self):
        ### Your Code Here

        ### End Code Here

doctor1 = Doctor(name="doctorZ2023", yob=1981, specialist="Endocrinologists")
                                                doctor1.describe()
```

(a) Doctor - Name: doctorZ2023 - YoB: 1981 - Specialist: Endocrinologists.

(b) Doctor - Name: 1981 - YoB: doctorZ2023 - Specialist: Endocrinologists.

(c) Teacher - Name: History - YoB: teacherZ2023 - Subject: 1991.

(d) Tất cả đều sai.

6. Một Ward gồm có name (string) và danh sách của mọi người trong Ward. Một người person có thể là student, doctor, hoặc teacher và cần sử dụng một list để chứa danh sách của mọi người trong Ward. Viết addPerson(person) method trong Ward class để add thêm một người mới với nghề nghiệp bất kỳ (student, teacher, doctor) vào danh sách người của ward. Tạo ra một ward object, và thêm vào 1 student, 2 teacher, và 2 doctor. Thực hiện describe() method để in ra tên ward (name) và toàn bộ thông tin của từng người trong ward. Như vậy, đáp án đúng nhất cho hàm đếm số lượng doctor là gì?

```
class Ward:
    def __init__(self, name:str):
        self.__name = name
        self.__listPeople = list()

    def addPerson(self, person:Person):
        self.__listPeople.append(person)

    def describe(self):
        print(f"Ward Name: {self.__name}")
        for p in self.__listPeople:
            p.describe()

    def countDoctor(self):
        ### Your Code Here

        ### End Code Here

student1 = Student(name="studentA", yob=2010, grade="7")
teacher1 = Teacher(name="teacherA", yob=1969, subject="Math")
teacher2 = Teacher(name="teacherB", yob=1995, subject="History")
doctor1 = Doctor(name="doctorA", yob=1945, specialist="Endocrinologists")
doctor2 = Doctor(name="doctorB", yob=1975, specialist="Cardiologists")
ward1 = Ward(name="Ward1")
ward1.addPerson(student1)
ward1.addPerson(teacher1)
```

```
ward1.addPerson(teacher2)
ward1.addPerson(doctor1)
ward1.addPerson(doctor2)
ward1.countDoctor()
```

(a) 4.

(b) 3.

(c) 2.

(d) 1.

7. Thực hiện xây dựng class Stack với các chức năng (method) sau đây:

   - init() method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa.

   - isFull(): kiểm tra stack đã đầy chưa.

   - push(value): thêm value vào trong stack.

   Kết quả đầu ra là gì?

```
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ### Your Code Here

        ### End Code Here

stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.isFull())
```

(a) True.

(b) False.

(c) None.

(d) Raise an error.

8. Thực hiện xây dựng class Stack với các chức năng (method) sau đây:

   - init() method nhận một input "capacity": dùng để khởi tạo stack với capacity là số lượng element mà stack có thể chứa.

- isEmpty(): kiểm tra stack có đang rỗng.
- isFull(): kiểm tra stack đã đầy chưa.
- push(value): thêm value vào trong stack.
- top(): lấy giá trị top element hiện tại của stack, nhưng không loại bỏ giá trị đó.

Kết quả đầu ra là gì?

```python
class MyStack:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__stack = []

    def isFull(self):
        return len(self.__stack) == self.__capacity

    def push(self, value):
        ### Your Code Here

        ### End Code Here

    def top(self):
        ### Your Code Here

        # End Code Here

stack1 = MyStack(capacity=5)
stack1.push(1)
stack1.push(2)
print(stack1.top())
```

(a) 1.

(b) 2.

(c) None.

(d) Raise an error.

9. Thực hiện xây dựng class Queue với các chức năng (method) sau đây

- init() method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa.
- isFull(): kiểm tra queue đã đầy chưa.
- enqueue(value): thêm value vào trong queue.

Kết quả đầu ra là gì?

```python
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []
```

```
    def isFull(self):
        return len(self.__queue) == self.__capacity

    def push(self, value):
        ### Your Code Here

        ### End Code Here

queue1 = MyQueue(capacity=5)
queue1.push(1)
queue1.push(2)
print(queue1.isFull())
```

(a) False.

(b) True.

(c) None.

(d) Raise an error.

10. Thực hiện xây dựng class Queue với các chức năng (method) sau đây

- init() method nhận một input "capacity": dùng để khởi tạo queue với capacity là số lượng element mà queue có thể chứa.
- isFull(): kiểm tra queue đã đầy chưa.
- push(value): thêm value vào trong queue.
- front(): lấy giá trị first element hiện tại của queue, nhưng không loại bỏ giá trị đó.

Kết quả đầu ra là gì?

```
class MyQueue:
    def __init__(self, capacity):
        self.__capacity = capacity
        self.__queue = []

    def isEmpty(self):
        return len(self.__queue) == 0

    def isFull(self):
        return len(self.__queue) == self.__capacity

    def dequeue(self):

    def enqueue(self, value):

    def front(self):
        ### Your Code Here

        ### End Code Here

```

```
queue1 = MyQueue(capacity=5)
queue1.enqueue(1)
assert queue1.is_full() == False
queue1.enqueue(2)
print(queue1.front())
```

(a) 4.

(b) 3.

(c) 2.

(d) 1.

# Phụ lục

1. **Hint:** Các file code gợi ý có thể được tải tại đây.

2. **Solution:** Các file code cài đặt hoàn chỉnh và phần trả lời nội dung trắc nghiệm có thể được tải tại đây (Lưu ý: Sáng thứ 3 khi hết deadline phần bài tập, ad mới copy các nội dung bài giải nêu trên vào đường dẫn).

3. **Rubric:**

| Mục | Kiến Thức | Đánh Giá |
|---|---|---|
| 1. | - Làm quen với Pytorch.<br>- Hiểu cơ bản về nn.Module trong Pytorch.<br>- Tạo Class mới từ Module. | - Khai báo các tham số.<br>- Ghi đè phương thức forward.<br>- Áp dụng cho hàm Sigmoid. |
| 2. | - Khái niệm Abstract Class trong Python.<br>- Khái niệm Inheritance trong Python.<br>- Khái niệm Polymorphism trong Python.<br>- Khái nhiệm Encapsulation trong Python.<br>- Python Access Modifiers: public, protected, và private members của class.<br>- Sử dụng nhiều method với các yêu cầu khác nhau trong Python.<br>- Đếm các object cùng class trong 1 list các object.<br>- Sort 1 list các object.<br>- Lọc các object cùng loại và tính toán dựa trên các data member. | - Hiểu và thực hiện được các khái niệm Abstract class, Inheritance, Polymorphism, Encapsulation.<br>- Sử dụng được các Python Access Modifiers.<br>- Khởi tạo được các method khác nhau trong class theo yêu cầu.<br>- Biết cách lọc và đếm các object cùng class.<br>- Thực hiện được xếp các object dựa trên các attribute. |
| 3. | - Hiểu rõ về các thành phần và cơ chế hoạt động của stack (kiểm tra full hay empty, pop, push, lấy top element của stack).<br>- Sử dụng các kiểu data cần thiết và dùng class để tạo thành 1 stack class.<br>- Hiểu được rằng stack có thể được thực hiện từ các kiểu dữ liệu khác nhau chỉ cần nắm rõ nguyên tắc hoạt động. | - Cơ bản biết kết hợp các kiểu data khác nhau để tạo ra một kiểu mà mình mong muốn.<br>- Đã có khả năng thiết kế stack dựa trên các kiểu data và kết hợp với class để tạo stack class hoàn chỉnh.<br>- Hiểu được cơ bản cách thiết kế 1 data structure (cần tìm hiểu nắm rõ được cơ chế hoạt động và các hành vi của data structure). |

| | | |
|---|---|---|
| 4. | - Hiểu rõ về các thành phần và cơ chế hoạt động của queue (kiểm tra full hay empty, push, pop, lấy first element của queue).<br>- Sử dụng các kiểu data cần thiết và dùng class để tạo thành 1 queue class.<br>- Hiểu được rằng queue có thể được thực hiện từ các kiểu dữ liệu khác nhau chỉ cần nắm rõ nguyên tắc hoạt động. | - Cơ bản biết kết hợp các kiểu data khác nhau để tạo ra một kiểu mà mình mong muốn.<br>- Đã có khả năng thiết kế queue dựa trên các kiểu data và kết hợp với class để tạo queue class hoàn chỉnh.<br>- Hiểu được cơ bản cách thiết kế 1 data structure (cần tìm hiểu nắm rõ được cơ chế hoạt động và các hành vi của data structure). |