BÀI TẬP THỰC HÀNH
MÔN TRÍ TUỆ NHÂN TẠO
LAB 3: HƯỚNG ĐỐI TƯỢNG TRONG PYTHON

A. Lý thuyết và ví dụ:

Phạm vi, tầm vực:

Từ khóa global để đưa biến cục bộ thành toàn cục trong hàm:

```
x = 1
def increase():
    x = 1
    x = x + 1
    print(x)      # Displays 2

increase()
print(x)         # Displays 1
```

```
x = 1
def increase():
    global x
    x = x + 1
    print(x)      # Displays 2

increase()
print(x)         # Displays 2
```

Khái niệm hướng đối tượng:

- Những đối tượng (object) thuộc về cùng một loại thường được định nghĩa bằng cách dùng một **lớp** (class) chung.
- Một lớp trong ngôn ngữ Python dùng các biến để lưu các **thành phần dữ liệu** của lớp và định nghĩa các **phương thức** (method) để thực hiện các tác vụ.
- Một đối tượng là một *thể hiện* (instance) của một lớp và chúng ta có thể tạo ra nhiều thể hiện của một lớp. Việc tạo ra một thể hiện của một lớp được gọi là sự *hiện thể hóa* (instantiation).

Định nghĩa lớp

- Python dùng cú pháp như sau để định nghĩa một lớp mới:

```
class ClassName:
    initializer
    methods
```

Thí dụ: Kiểu dữ liệu Stack

Kiểu dữ liệu stack được định nghĩa bằng cấu trúc và các tác vụ sau đây. Một stack được cấu trúc như là một tập phần tử có thứ tự mà khi các phần tử được thêm vào hay gỡ bỏ thì đang ở vị trí trên cùng (top) của stack. Stack được sắp thứ tự theo kiểu LIFO. Các tác vụ của lớp Stack như sau:

- **Stack()** tạo ra một stack mới và rỗng. Tác vụ này không đòi hỏi tham số và trả về một stack rỗng.
- **push(item)** thêm một phần tử mới vào vị trí top của stack. Tác vụ này cần một phần tử và không trả về giá trị nào cả.
- **pop()** gỡ bỏ phần tử ở vị trí top của stack. Tác vụ này không đòi hỏi tham số nào cả và trả về giá trị phần tử được gỡ bỏ. Stack bị thay đổi.
- **peek()** trả về phần tử ở vị trí top của stack nhưng không gỡ bỏ nó. Tác vụ này không đòi hỏi tham số nào cả . Stack không bị thay đổi.
- **is_empty()** kiểm tra xem stack có rỗng hay không. Tác vụ này không đòi hỏi tham số nào cả và trả về một giá trị bool.
- **size()** trả về số phần tử của stack. Tác vụ này không đòi hỏi tham số nào cả và trả về một giá trị nguyên.

```
# Sự hiện thực đầy đủ của kiểu dữ liệu stack như là một lớp
class Stack:
    def __init__(self):    # phương thức initializer
        self.items = []     # items của một đối tượng stack là một list rỗng
    def is_empty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def peek(self):
        return self.items[len(self.items)-1]
    def size(self):
        return len(self.items)
```

Chú ý: Phương thức **init** cần có hai ký hiệu underscore đi trước và đi sau.

Ví dụ: sử dụng lớp Stack:

```
s1  = Stack()     # tạo ra một đối tượng Stack rỗng
s1 = s1.push(5)   # đưa 5 vào stack s1
s1 = s1.push(6)      # đưa 6 vào stack s1
s1= s1.push(8)      # đưa 8 vào stack s1
s1= s1.pop()          # gỡ phần tử cuối cùng ra khỏi s1
print(s1)
print(s1.peek())      # in ra phần tử 6
s1= s1.pop()          # gỡ phần tử 6 ra khỏi s1
s1= s1.pop()            # gỡ phần tử 5 ra khỏi s1
print(s1.is_empty())      # in ra trị True
```

Map và lamda

- Python có một cách đặc biệt để gọi hàm theo kiểu lặp mà không cần tạo vòng lặp. Nếu ta muốn áp dụng cùng một hàm lên mọi phần tử của một list, ta có thể dùng lệnh *map* của Python với cú pháp *map*(*function*, *list*). Lệnh này sẽ áp dụng hàm được nêu lên mọi phần tử của một list.

```
>>> items = [1,2,3,4,5]
>>> def inc(x): return x+1
...
>>> list(map(inc,items))
[2, 3, 4, 5, 6]
```

- Có một mánh lới khi hàm là một hàm vô danh là dùng lệnh **lambda** với cú pháp **lambda** *args* : *command*
- Hàm lambda chỉ thực thi trên một phát biểu, nhưng nó tạo điều kiện cho ta viết câu lệnh ngắn để thực hiện những việc phức tạp.
- Ví dụ, câu lệnh sau đây sẽ tiếp nhận một list và tính lập phương mỗi phần tử trong list và nhân với 7.

map(lambda x**:pow(**x,3)+7**,list)**

filter

Một cách khác là lệnh **lambda** được dùng kết hợp với lệnh **filter**. Cách này trả về các phần tử của list mà thỏa điều kiện được nêu trong câu lệnh nằm trong phát biểu lambda.
Thí dụ

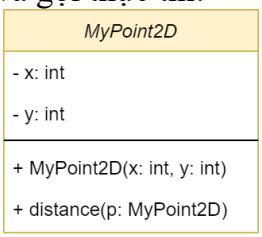
filter(lambda x**:**x>=2**,list)**

Trả về những phần tử trong list mà thỏa điều kiện lớn hơn hay bằng 2.

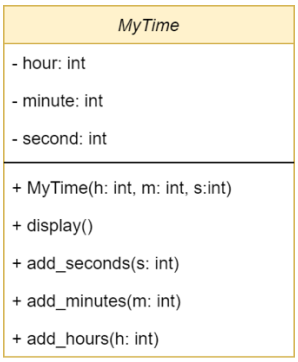
B. Website tham khảo: https://www.w3schools.com/python/python_classes.asp

C. Bài tập:

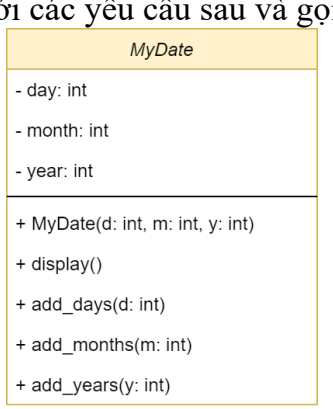
Bài 1. Xây dựng class MyPoint2D và gọi thực thi:



Bài 2. Xây dựng class MyTime và gọi thực thi:



Bài 3. Xây dựng class MyDate với các yêu cầu sau và gọi thực thi:



Bài 4. Tìm hiểu tính kế thừa trong OOP của python:
https://www.w3schools.com/python/python_inheritance.asp
Bài 5. Tìm hiểu Json với python:
https://www.w3schools.com/python/python_json.asp
Bài 6. Tìm hiểu đọc và ghi file với python:
https://www.w3schools.com/python/python_file_handling.asp
Bài 7. Vận dụng các nội dung của bài 1, 2, 3, 4, 5, 6 để thực hiện yêu cầu tổng hợp sau:
Xây dựng 1 class mang tên InputData gồm các phương thức:
+ Đọc các phần tử trong danh sách từ file BT.txt, biết rằng số phần tử được đặt ở dòng đầu tiên trong file.
+ Chuyển toàn bộ các phần tử từ danh sách ở câu trên thành dictionary theo dạng: “element 1”: value
+ Ghi toàn bộ dictionary sang thành chuỗi Json.
+ Ghi chuỗi Json thành file.
+ class InputSpecialData kế thừa từ class InputData. Bổ sung thêm phương thức ThongKe để trả về dictionary các phần tử có value là số chẵn.

---HẾT---