**BÀI TẬP THỰC HÀNH SỐ 2**

**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG VỚI PYTHON**

**1911127**

**Lê Nguyễn Anh**

**Bài 1: Cài đặt lớp Sinh viên và danh sách sinh viên như sau và hoàn thành các hàm còn trống (chưa có nội dung)**

# Tìm những sinh viên sinh trước 15/6/2000

    def timSVTruocNgaySinh(self, date:datetime) -> list:

        #return [sv for sv in self.dssv if sv.ngaySinh < date]

        kq = []

        for i in range(len(self.dssv)):

            if self.dssv[i].ngaySinh < date:

                kq.append(self.dssv[i])

        return kq

date = datetime.strptime("15/6/2000","%d/%m/%Y")

print(f'Danh sách sinh viên có ngày sinh trước {date} là: ')

for sv in dssv.timSVTruocNgaySinh(date):

    print(sv)

# Tìm sinh viên tên Nam

    def timSVTheoTen(self, name: str) -> list:

        #return [sv for sv in self.dssv if sv.hoTen.split(' ')[-1] == name]

        kq = []

        for i in range(len(self.dssv)):

            if self.dssv[i].hoTen.split(' ')[-1] == name:

                kq.append(self.dssv[i])

        return kq

name = "Nam"

for sv in dssv.timSVTheoTen(name):

    print(f'Danh sách sinh viên kiếm theo tên là: \n{sv}')

**Bài 2: Bổ sung phương thức :**

1. Đọc danh sách sinh viên từ tập tin (txt/csv)

f = open("D:\\Python\Lab02\dssv.txt", "r")

for i in f:

    maSo = i.split('\t')[0]

    hoTen = i.split('\t')[1]

    ngaySinh = i.split('\t')[2].strip()

    ngaySinh = datetime.strptime(ngaySinh, "%d/%m/%Y")

    sinhdiên = SinhVien(maSo, hoTen, ngaySinh)

    dssv.themSV(sinhdiên)

dssv.xuat()

1. Sắp xếp danh sách sinh viên tăng/giảm theo họ tên

#Sắp xếp dssv theo chiều tăng:

    def sortSVTang(self):

        sort = self.dssv.sort(key=lambda ht:ht.hoTen.split(" ")[-1])

        return sort

    #Sắp xếp dssv theo chiều giảm:

    def sortSVTang(self):

        sort = self.dssv.sort(key=lambda ht:ht.hoTen.split(" ")[-1],reverse=True)

        return sort

**Bài 3: Cài đặt lớp phân số có phương thức rút gọn phân số, ghi đè toán tử +,-,\*,/ như sau:**

import math

class PhanSo:

    def \_\_init\_\_(self, tu, mau):

        self.tu = tu

        self.mau = mau

        self.rutGon()

    def xuat(self):

        print(self.tu, "/", self.mau)

    def rutGon(self):

        uc = math.gcd(self.tu, self.mau)

        self.tu = self.tu // uc

        self.mau = self.mau // uc

    def \_\_add\_\_(self, other):

        result = PhanSo(0, 1)

        result.tu = self.tu\*other.mau + self.mau\*other.tu

        result.mau = self.mau\*other.mau

        result.rutGon()

        return result

    def \_\_sub\_\_(self, other):

        result = PhanSo(0, 1)

        result.tu = self.tu\*other.mau - self.mau\*other.tu

        result.mau = self.mau\*other.mau

        result.rutGon()

        return result

    def \_\_mul\_\_(self, other):

        result = PhanSo(0, 1)

        result.tu = self.tu\*other.tu

        result.mau = self.mau\*other.mau

        result.rutGon()

        return result

    def \_\_truediv\_\_(self, other):

        result = PhanSo(0, 1)

        result.tu = self.tu\*other.mau

        result.mau = self.mau\*other.tu

        result.rutGon()

        return result

**Bài 4: Cài đặt lớp danh sách phân số, bổ sung các chức năng và kiểm tra kết quả:**

1. Đếm số phân số âm trong mảng

def demPSAm(self):

        count = 0

        for ps in self.listfraction:

            if ps.tu \* ps.mau < 0:

                count += 1

        print("Số phân số âm là: ", count)

1. Tìm phân số dương nhỏ nhất

def TimPSDuongNhoNhat(self):

        min  = PhanSo(1\_000\_000\_000)

        for ps in self.listfraction:

            if ps.tu\*ps.mau > 0 and ps < min:

                min = ps

        print ("Phân số dương nhỏ nhất là: ", min)

1. Tìm tất cả vị trí của phân số x trong mảng

def TimViTriCuaPS(self, ps:PhanSo) -> int:

        for i in range(len(self.listfraction)):

            if self.listfraction[i]==ps:

                return i

        return -1

1. Tổng tất cả các phân số âm trong mảng

def TongPSAm(self):

        sum = PhanSo()

        for ps in self.listfraction:

            if ps.tu \* ps.mau < 0:

                sum = sum + ps

        print(f"{sum.tu} / {sum.mau}")

1. Xóa phân số x trong mảng

def XoaPS(self, ps):

        vt = self.TimViTriCuaPS(ps)

        if vt >= 0:

            del self.listfraction[vt]

            return True

        else:

            return False

1. Xóa tất cả phân số có tử là x

def XoaPSTheoTu(self,tu:int):

        for ps in self.listfraction:

            if ps.tu == tu:

                self.listfraction.remove(ps)

1. Sắp xếp phân số theo chiều tăng, giảm; tăng theo mẫu, tử, giảm theo mẫu tử

def SortPSTangTheoTu(self):

        sort = self.listfraction.sort(key=lambda x:x.tu)

        print(sort)

        self.xuat()

    def SortPSTangTheoMau(self):

        sort = self.listfraction.sort(key=lambda x:x.mau)

        print(sort)

        self.xuat()

    def SortPSGiamTheoTu(self):

        sort = self.listfraction.sort(key=lambda x:x.tu,reverse=True)

        print(sort)

        self.xuat()

    def SortPSGiamTheoMau(self):

        sort = self.listfraction.sort(key=lambda x:x.mau,reverse=True)

        print(sort)

        self.xuat()

**Bài 5: Sử dụng kế thừa**

Cài đặt 2 hàm còn thiếu trong danh sách sinh viên:

1. Tìm sinh viên có điểm rèn luyện từ 80 trở lên

def TimSVCoDiemRLTren80(self):

        kq = []

        for sv in self.dssv:

            if isinstance(sv, SinhVienChinhQuy):

                if sv.diemRL >= 80:

                    kq.append(sv)

        return kq

2. Tìm sinh viên có trình độ cao đẳng sinh trước 15/8/1999

def TimSVCoTrinhDoCDVaSinhTrcNgay(self):

        kq = []

        ngay = datetime.strptime("15/8/1999","%d/%m/%Y")

        trinhDo = "Cao đẳng"

        for sv in self.dssv:

            if isinstance(sv, SinhVienPhiChinhQuy):

                if sv.trinhDo == trinhDo and sv.ngaySinh < ngay:

                    kq.append(sv)

        return kq

**Bài 6: Cài đặt chương trình quản lý hình học theo mô hình lớp sau dùng Python**

Class HinhHoc:

class HinhHoc:

    def \_\_init\_\_(self, canh: float) -> None:

        self.\_\_canh = canh

    @property

    def canh(self):

        return self.\_\_canh

    def TinhDienTich(self):

        return float

    def xuat(self):

        print (f"{self.canh}")

    def \_\_str\_\_(self) -> str:

        return f"{self.canh}"

Class HinhTron:

from hinh\_hoc import HinhHoc

import math

class HinhTron(HinhHoc):

    def \_\_init\_\_(self, BanKinh: float) -> None:

        super().\_\_init\_\_(BanKinh)

        self.BanKinh = BanKinh

    def TinhDienTich(self) -> float:

        return self.BanKinh \*\* 2 \* math.pi

    def \_\_str\_\_(self) -> str:

        return f"Hình tròn bán kính {self.BanKinh} có diện tích là : {self.TinhDienTich()}"

Class HinhChuNhat:

from hinh\_hoc import HinhHoc

class HinhChuNhat(HinhHoc):

    def \_\_init\_\_(self, ChieuDai: float, ChieuRong: float) -> None:

        super().\_\_init\_\_(ChieuDai)

        self.\_\_ChieuDai = ChieuDai

        self.\_\_ChieuRong = ChieuRong

    @property

    def ChieuDai(self):

        return self.\_\_ChieuDai

    @property

    def ChieuRong(self):

        return self.\_\_ChieuRong

    def getChieuDai(self):

        return self.ChieuDai

    def getChieuRong(self):

        return self.ChieuRong

    def TinhDienTich(self):

        return self.canh \* self.ChieuRong

    def \_\_str\_\_(self) -> str:

        return f"Hình chữ nhật có chiều dài: {self.ChieuDai}, chiều rộng: {self.ChieuRong} có diện tích: {self.TinhDienTich()}"

Class HinhVuong:

from hinh\_chu\_nhat import HinhChuNhat

class HinhVuong(HinhChuNhat):

    def \_\_init\_\_(self, canh: float) -> None:

        super().\_\_init\_\_(canh, canh)

    def TinhDienTich(self):

        return self.canh \* self.canh

    def \_\_str\_\_(self) -> str:

        return f"Hình vuông cạnh: {self.canh} có diện tích: {self.TinhDienTich()}"

Class DanhSachHinhHoc:

from hinh\_chu\_nhat import HinhChuNhat

from hinh\_hoc import HinhHoc

from hinh\_tron import HinhTron

from hinh\_vuong import HinhVuong

class DanhSachHinhHoc:

    def \_\_init\_\_(self) -> None:

        self.dshh = []

    def themHinh(self, hh: HinhHoc):

        self.dshh.append(hh)

    def xuat(self):

        for hh in self.dshh:

            print(hh)

Class KieuHinh:

from enum import Enum

class KieuHinh(Enum):

    tat\_Ca = 1

    hinh\_Tron = 2

    hinh\_Vuong = 3

    hinh\_Chu\_Nhat = 4

* Tìm hình có diện tích lớn nhất

def TimHinhCoDienTichLonNhat(self):

        hinhHoc = HinhHoc()

        max = 0

        for hh in self.dshh:

            if hh.TinhDienTich() > max:

                max = hh.TinhDienTich()

                hinhHoc = hh

        print(hinhHoc)

* Tìm hình có diện tích nhỏ nhất

def TimHinhCoDienTichNhoNhat(self):

        hinhHoc = HinhHoc()

        min = 0

        for hh in self.dshh:

            if hh.TinhDienTich() < min:

                min = hh.TinhDienTich()

                hinhHoc = hh

        print(hinhHoc)

* Tìm hình tròn có diện tích lớn nhất

def TimHinhTronCoDienTichLonNhat(self):

        hinhHoc = HinhHoc()

        max = 0

        for hh in self.dshh:

            if isinstance(hh, HinhTron):

                if hh.TinhDienTich() > max:

                    max = hh.TinhDienTich()

                    hinhHoc = hh

        print(hinhHoc)

* Sắp các hình giảm dần theo diện tích

def SapGiamTheoDienTich(self):

        return self.dshh.sort(key=lambda x:x.TinhDienTich(), reverse=True)

- Đếm số lượng hình theo loại

#kieuHinh = "hinh\_Tron" or "hinh\_Chu\_Nhat" or "hinh\_Vuong" or "tat\_Ca"

def DemSoLuongKieuHinh(self, kieuHinh: str):

        kq = []

        for hh in self.dshh:

            if kieuHinh == KieuHinh.hinh\_Tron.name:

                if isinstance(hh, HinhTron):

                    kq.append(hh)

            elif kieuHinh == KieuHinh.hinh\_Chu\_Nhat.name:

                if isinstance(hh, HinhChuNhat):

                    kq.append(hh)

            elif kieuHinh == KieuHinh.hinh\_Vuong.name:

                if isinstance(hh, HinhVuong):

                    kq.append(hh)

            elif kieuHinh == KieuHinh.tat\_Ca.name:

                kq.append(hh)

        print("Số lượng hình theo loại hình là: ", len(kq))

* Tính tổng diện tích các hình

def TinhTongDienTich(self):

        sum = 0

        for hh in self.dshh:

            sum += hh.TinhDienTich()

        return sum

* Tìm hình có diện tích lớn nhất theo loại hình học cho trước

#kieuHinh = "hinh\_Tron" or "hinh\_Chu\_Nhat" or "hinh\_Vuong" or "tat\_Ca"

def TimHinhCoDienTichLonNhatTheoKieuHinh(self, kieuHinh: str):

        max = 0

        hinhHoc = HinhHoc()

        for hh in self.dshh:

            if kieuHinh == KieuHinh.hinh\_Tron.name and isinstance(hh,HinhTron):

                if hh.TinhDienTich() > max:

                    max = hh.TinhDienTich()

                    hinhHoc = hh

            elif kieuHinh == KieuHinh.hinh\_Vuong.name and isinstance(hh,HinhVuong):

                if hh.TinhDienTich() > max:

                    max = hh.TinhDienTich()

                    hinhHoc = hh

            elif kieuHinh == KieuHinh.hinh\_Chu\_Nhat.name and isinstance(hh,HinhChuNhat):

                if hh.TinhDienTich() > max:

                    max = hh.TinhDienTich()

                    hinhHoc = hh

            elif kieuHinh == KieuHinh.tat\_Ca.name:

                if hh.TinhDienTich() > max:

                    max = hh.TinhDienTich()

                    hinhHoc = hh

        return hinhHoc

* Tìm vị trí của hình h trong danh sách
* Xóa một hình tại vị trí cho trước

def XoaTaiViTri(self, viTri: int):

        for i in range(len(self.dshh)):

            if i == viTri:

                self.dshh.remove(self.dshh[i])

                return True

            else:

                return False

* Tìm hình theo diện tích

def TimHinhTheoDTich(self, dt:float):

        hinhHoc = HinhHoc()

        for hh in self.dshh:

            if hh.TinhDienTich() == dt:

                hinhHoc = hh

        return hinhHoc

* Xóa một hình học khỏi danh sách

def XoaHinh(self, hh:HinhHoc):

        for h in self.dshh:

            if h == hh:

                self.dshh.remove(h)

                return True

            else:

                return False

* Xóa tất cả các hình theo loại cho trước

#kieuHinh = "hinh\_Tron" or "hinh\_Chu\_Nhat" or "hinh\_Vuong" or "tat\_Ca"

def XoaHinhTheoLoai(self, kieuHinh:str):

        for hh in self.dshh:

            if kieuHinh == KieuHinh.hinh\_Tron.name:

                if isinstance(hh,HinhTron):

                    self.dshh.remove(hh)

            elif kieuHinh == KieuHinh.hinh\_Chu\_Nhat.name:

                if isinstance(hh,HinhChuNhat):

                    self.dshh.remove(hh)

            elif kieuHinh == KieuHinh.hinh\_Vuong.name:

                if isinstance(hh,HinhVuong):

                    self.dshh.remove(hh)

            else:

                self.dshh.remove(hh)

* Xuất danh sách hình theo loại cho trước và sắp tăng hoặc giảm

#kieuHinh = "hinh\_Tron" or "hinh\_Chu\_Nhat" or "hinh\_Vuong" or "tat\_Ca"

def XuatHinhTheoChieuTangGiam(self, kieuHinh:str, tang:bool):

        kq = []

        for hh in self.dshh:

            if kieuHinh==KieuHinh.hinh\_Tron.name:

                if isinstance(hh,HinhTron):

                    kq.append(hh)

            elif kieuHinh==KieuHinh.hinh\_Chu\_Nhat.name:

                if isinstance(hh,HinhChuNhat):

                    kq.append(hh)

            elif kieuHinh==KieuHinh.hinh\_Vuong.name:

                if isinstance(hh,HinhVuong):

                    kq.append(hh)

            elif kieuHinh==KieuHinh.tat\_Ca.name:

                kq.append(hh)

        if tang==False:

            kq.sort(key=lambda x:x.TinhDienTich(),reverse=False)

        else:

            kq.sort(key=lambda x:x.TinhDienTich(),reverse=True)

        return kq

* Tính tổng diện tích các hình theo loại

#kieuHinh = "hinh\_Tron" or "hinh\_Chu\_Nhat" or "hinh\_Vuong" or "tat\_Ca"

    def TinhTongDTTheoKieuHinh(self, kieuHinh:str):

        sum =0

        for hh in self.dshh:

            if kieuHinh == KieuHinh.hinh\_Tron.name:

                if isinstance(hh,HinhTron):

                    sum += hh.TinhDienTich()

            elif kieuHinh == KieuHinh.hinh\_Chu\_Nhat.name:

                if isinstance(hh,HinhChuNhat):

                    sum += hh.TinhDienTich()

            elif kieuHinh == KieuHinh.hinh\_Vuong.name:

                if isinstance(hh,HinhVuong):

                    sum += hh.TinhDienTich()

            elif kieuHinh == KieuHinh.tat\_Ca.name:

                sum += hh.TinhDienTich()

        return sum