Tài liệu hướng dẫn thực hành Buổi 3
Môn : Lý Thuyết Thông Tin

Nội dung chính : Lập trình hướng đối tượng – Giải thuật sinh mã Huffman

1. Lập trình hướng đối tượng

- Đối tượng (Object) : là những thực thể tồn tại có những hành vi nhất định
- Lớp (Class) : là một kiểu dữ liệu đặc biệt do người dùng định nghĩa, tập hợp nhiều thuộc tính đặc trưng cho mọi đối tượng được tạo ra từ lớp đó.
- Thuộc tính là các giá trị của lớp. Sau này khi các đối tượng được tạo ra từ lớp, thì thuộc tính của lớp lúc này sẽ trở thành các đặc điểm của đối tượng đó.

Trong Python, lập trình hướng đối tượng (OOP) tuân theo một số nguyên lý cơ bản:

– Tính kế thừa (Inheritance) : cho phép một lớp (class) có thể kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa.

–Cho phép một lớp kế thừa các thuộc tính và phương thức từ các lớp khác đã được định nghĩa.

–Lớp đã định nghĩa gọi là lớp cha, lớp mới phát sinh (kế thừa) gọi là lớp con

–Lớp con kế thừa các thành phần của lớp cha, đồng thời có thể mở rộng các thành phần kế thừa và bổ sung các thành phần mới

```python
# Lớp cha
class Car:

    # Constructor
    def __init__(self, hangxe, tenxe, mausac):
        # Lớp Car có 3 thuộc tính: tenxe, mausac, hang xe
        self.hangxe = hangxe
        self.tenxe = tenxe
        self.mausac = mausac

    # phương thức
    def chayxe(self):
        print ("{} đang chạy trên đường".format(self.tenxe))

    def dungxe(self, mucdich):
        print ("{} đang dừng xe để {}".format(self.tenxe, mucdich))
```

```python
# Lớp Toyota mở rộng từ lớp Car.
class Toyota(Car):

    def __init__(self, hangxe, tenxe, mausac, nguyenlieu):
 # Gọi tới constructor của lớp cha (Car)
        # để gán giá trị vào thuộc tính của lớp cha.
 super().__init__(hangxe, tenxe, mausac)

        self.nguyenlieu = nguyenlieu

    # Kế thừa phương thức cũ
    def chayxe(self):
        print ("{} đang chạy trên đường".format(self.tenxe))

    # Ghi đè (override) phương thức cùng tên của lớp cha.
    def dungxe(self, mucdich):
        print ("{} đang dừng xe để {}".format(self.tenxe, mucdich))
        print ("{} chạy bằng {}".format(self.tenxe, self.nguyenlieu))

# Bổ sung thêm thành phần mới
    def nomay(self):
        print ("{} đang nổ máy".format(self.tenxe))
```

– Tính đa hình (Polymorphism) : là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau.

–Đây là khái niệm mà hai hoặc nhiều lớp có những phương thức giống nhau nhưng có thể thực thi theo những cách thức khác nhau

–Ví dụ : Lớp Car(), Ship() và Human() đều có một phương thức là Move(). Nhưng cách thực hiện phương thức Move cho mỗi class sẽ khác nhau.

```python
class Toyota:

 def dungxe(self):
        print("Toyota dừng xe để nạp điện")

    def nomay(self):
        print("Toyota nổ máy bằng hộp số tự động")

class Porsche:

    def dungxe(self):
        print("Porsche dừng xe để bơm xăng")

    def nomay(self):
        print("Porsche nổ máy bằng hộp số cơ")

# common interface
def kiemtra_dungxe(car): car.dungxe()
```

– Tính đóng gói (Encapsulation) : là quy tắc yêu cầu trạng thái bên trong của một đối tượng được bảo vệ và tránh truy cập được từ code bên ngoài

–Ta có thể hạn chế quyền truy cập vào các thuộc tính và trạng thái bên trong của đối tượng. Điều này giúp ngăn chặn việc dữ liệu bị sữa đổi từ các tác nhân bên ngoài.

–Trong python ta khai báo thuộc tính private bằng cách sử dụng dấu gạch dưới làm tiền tố ’’_’’ hoặc ’’__’’

```
3  class Computer:
4
5      def __init__(self):
6          # Thuộc tính private ngăn chặn sửa đổi trực tiếp
7          self.__maxprice = 900
8
9      def sell(self):
10         print("Giá bán sản phẩm: {}".format(self.__maxprice))
11
12     def setMaxPrice(self, price):
13         self.__maxprice = price
```

```
15 c = Computer()
16 c.sell()
17
18 # thay đổi giá bán ( không thành công)
19 c.__maxprice = 1000
20 c.sell()
21
22 # sử dụng hàm setter để thay đổi giá. ( thành công)
23 c.setMaxPrice(1000)
24 c.sell()
```

2. Xử lý ngoại lệ

Ngoại lệ (Exception) là lỗi xảy ra trong quá trình thực thi một chương trình. Khi nó xảy ra, Python tạo ra một exception để xử lý vấn đề đó tránh cho ứng dụng hay server bị crash.

Ngoại lệ có thể là bất kỳ điều kiện bất thường nào trong chương trình phá vỡ luồng thực thi chương trình đó. Bất cứ khi nào một ngoại lệ xuất hiện, chương trình sẽ ngừng thực thi, chuyển qua quá trình gọi và in ra lỗi đến khi nó được xử lý.

Trong python, các ngoại lệ được xử lý bằng khối lệnh try ….. except….

```
try:
        # lệnh...
except:
        # lệnh...
```

Phần thân của try sẽ gồm code có thể tạo ra exception, nếu một exception được tạo ra, tất cả câu lệnh trong khối sẽ bị bỏ qua.

Mặt khác, phần thân của except được gọi bởi exception handler, vì nó được dùng để bắt lỗi. Khối except sẽ thực hiện khi lỗi được tạo ra, không thì sẽ được bỏ qua.

```
import sys

randomList = ['a', 0, 2.0]

for i in randomList:
 try:
    print("Iterative :", i)
    r = (1.0/(i))
    break
 except:
    print("Exception: ",sys.exc_info()[0],".")
    print("Next Iterative : ")
    print()

print("Result with iterative ",i," is:",r)
```

Try …. Finally : đây là một cách khác để sử dụng try trong python.

- Khối lệnh trong block Finally sẽ luôn được thực thi dù chương trình có lỗi hay không.
- Thường thì những lệnh trong Finally sẽ dùng để giải phóng tài nguyên

Định nghĩa một exception : Ta cần kế thừa lớp Exception của Python.
Sau đó ta dùng raise để gọi exception vừa định nghĩa

```
class MyPersonalException(Exception):
    def __init__(self, msg):
        print("Error : " + msg)
```

3. **Bài tập**

Sinh viên tạo file <<Hoten>>_THBuoi3.py (vd : tranvanan_THBuoi3.py), Nhớ chú thích họ tên và MSSV lên đầu file.

Câu 1 : Viết chương trình khởi tạo lớp human gồm có những thông tin sau : họ tên, năm sinh, quê quán, nghề nghiệp. Viết phương thức :

- live(self, noicutru) : in ra màn hình người đó tên gì đang sống ở đâu
- work(self, diachicoquan) : in ra màn hình người đó tên gì đang làm nghề gì tại cơ quan nào (địa chỉ cơ quan)

Câu 2 : Viết chương trình cho phép người dùng nhập vào số lượng người muốn nhập vào hệ thống. Tạo ra số lượng đối tượng thuộc lớp human tương ứng. Sau đó tiến hành nhập thông tin, từng người (họ tên, năm sinh, ….) và lưu trong các biến thuộc lớp human.

Ví dụ :

Nhập số lượng người : 2
Nhập họ tên người thứ 1 : …..
Nhập năm sinh người thứ 1 : …
Nhập quê quán người thứ 1: …
Nhập nghề nghiệp người thứ 1: …
Nhập họ tên người thứ 2: …
…

Sau đó in thông tin của các người dùng đó ra màn hình và sử dụng hàm work() để cho biết thông tin về nơi làm việc của những người đó với diachicoquan của từng người do sinh viên tự nhập vào.

Câu 3 : Khai báo lớp Student là lớp con được kế thừa từ lớp Human với các thuộc tính được kế thừa từ lớp cha (thuộc tính nghề nghiệp được set mặc định thành "student"). Ngoài ra lớp student còn có các thuộc tính riêng là MSSV, ngành học và điểm trung bình (điểm trung bình sử dụng thang điểm 10). Các phương thức :

- Study(self, class) : cho biết sinh viên {tên} có {mssv} thuộc {ngành} đang học tại phòng học {class} nào
- Rank(self) : Cho biết thứ hạng của sv dựa vào điểm trung bình (ĐTB).

  ĐTB < 4 : Loại Yếu

  4 <= ĐTB < 6 : Loại Trung bình

  6 <= ĐTB < 8 : Loại Khá

  8 <= ĐTB < 10 : Loại Giỏi

  In ra màn hình : Sinh viên {tên} có {MSSV} với điểm trung bình { ĐTB } được xếp loại {loại}

Câu 4 : Khai báo thông tin 2 sinh viên như sau :

- SV1 : Tên : Lê Văn An, sinh năm 2005, quê Vĩnh Long, nghề nghiệp mặc định là sinh viên, MSSV = 12345, ngành CNTT, ĐTB = 7.6
- SV2 : Tên : Trần Văn Bình, sinh năm 2007, quê Trà Vinh, MSSV = 56789, Ngành Tài chính ngân hàng, ĐTB = 4.5

Hãy in thông tin 2 sinh viên này ra màn hình và cho biết thứ hạng của 2 sinh viên này dựa trên điểm trung bình của họ.

Câu 5 : Viết Hàm cho phép người dùng nhập vào thông tin của **MỘT** sinh viên với những điều kiện như sau :

- Họ tên có độ dài không quá 25 ký tự, Nếu dài hơn thì hiện ra thông báo "Ho ten khong vuot qua 25 ky tu !" và yêu cầu người dùng nhập lại
- Năm sinh có độ dài đúng 4 ký tự số, Nếu có ký tự không phải số thì yêu cầu người dùng nhập lại. Nếu dài hơn 4 ký tự thì hiển thị thông báo "Nam sinh phai co 4 chu so " và cũng yêu cầu người dùng nhập lại
- Quê quán có độ dài không quá 50 ký tự, Nếu dài hơn thì hiện ra thông báo "Que quan khong vuot qua 50 ky tu !" và yêu cầu người dùng nhập lại
- Nghề nghiệp mặc định là sinh viên (Khai báo trực tiếp lúc khai báo lớp Sinh viên) nên không cần phải nhập
- MSSV chứa tối đa 5 ký tự số, không chấp nhận các ký tự chữ cái. Nếu số ký tự dài hơn 5 thì yêu cầu người dùng nhập lại. Nếu có ký tự không phải số thì hiển thị thông báo : "MSSV chi chua cac ky tu so ! " và yeu cầu người dùng nhập lại.
- Ngành học có độ dài không quá 40 ký tự, Nếu dài hơn thì hiện ra thông báo "nganh hoc khong vuot qua 40 ky tu !" và yêu cầu người dùng nhập lại
- Điểm trung bình phải lớn hơn 0 và nhỏ hơn 10. Nếu ĐTB nhỏ hơn 0 hoặc lớn hơn 10 thì yêu cầu người dùng nhập lại

Câu 6 : Viết hàm cho phép người dùng nhập vào số lượng sinh viên cần Nhập thông tin (ví dụ : 3). Sau đó sử dụng **hàm đã viết ở câu 5** để cho người dùng nhập vào thông tin của từng sinh viên. Sau đó in thông tin các sinh viên ra màn hình theo định dạng như sau :

Sinh vien 1

Ho ten : …….
Nam sinh : ……
Que quan : ……
MSSV : ……
Nganh Hoc : ……
Diem trung binh : ……
Xep Loai : ……

Sinh vien 2

Ho ten : …….
Nam sinh : ……
Que quan : ……
MSSV : ……
Nganh Hoc : ……
Diem trung binh : ……
Xep Loai : ……

Sinh vien 3

Ho ten : …….
Nam sinh : ……
Que quan : ……
MSSV : ……
Nganh Hoc : ……
Diem trung binh : ……
Xep Loai : ……

Câu 7 : Hãy khai báo một lớp Nodes gồm có 4 thuộc tính : probability, symbol, left, right (mặc định của left và right là None) do người dùng truyền vào và một thuộc tính code với giá trị mặc định là chuỗi rỗng

Câu 8 : Hãy viết một hàm CalculateProbability(data) để đếm số lượng ký tự xuất hiện và lưu trữ vào một từ điển. ví dụ : data = “ABBCCCDDDD” thì ta lưu vào từ điển dict1 như sau :

dict1[A] = 1
dict1[B] = 2
dict1[C] = 3
dict1[D] = 4

Câu 9 : Hãy khai báo các hàm hỗ trợ sau :

```
def CalculateCodes(node, value = ''):
    # a huffman code for current node
    newValue = value + str(node.code)

    if(node.left):
        CalculateCodes(node.left, newValue)
    if(node.right):
        CalculateCodes(node.right, newValue)

    if(not node.left and not node.right):
        the_codes[node.symbol] = newValue

    return the_codes

""" A supporting function in order to get the encoded result """
def OutputEncoded(the_data, coding):
    encodingOutput = []
    for element in the_data:
        # print(coding[element], end = '')
        encodingOutput.append(coding[element])

    the_string = ''.join([str(item) for item in encodingOutput])
    return the_string

""" A supporting function in order to calculate the space difference between compressed and non
    compressed data"""
def TotalGain(the_data, coding):
    # total bit space to store the data before compression
    beforeCompression = len(the_data) * 8
    afterCompression = 0
    the_symbols = coding.keys()
    for symbol in the_symbols:
        the_count = the_data.count(symbol)
        # calculating how many bit is required for that symbol in total
        afterCompression += the_count * len(coding[symbol])
    print("Space usage before compression (in bits):", beforeCompression)
    print("Space usage after compression (in bits):",  afterCompression)
```

Câu 10 : Hãy khai báo hàm dùng để xây dựng bộ mã Huffman như sau :

```
def HuffmanEncoding(the_data):
    symbolWithProbs = CalculateProbability(the_data)
    the_symbols = symbolWithProbs.keys()
    the_probabilities = symbolWithProbs.values()
    print("symbols: ", the_symbols)
    print("probabilities: ", the_probabilities)

    the_nodes = []

    # converting symbols and probabilities into huffman tree nodes
    for symbol in the_symbols:
        the_nodes.append(Nodes(symbolWithProbs.get(symbol), symbol))

    while len(the_nodes) > 1:
        # sorting all the nodes in ascending order based on their probability
        the_nodes = sorted(the_nodes, key = lambda x: x.probability)
        # for node in nodes:
        #      print(node.symbol, node.prob)

        # picking two smallest nodes
        right = the_nodes[0]
        left = the_nodes[1]

        left.code = 0
        right.code = 1

        # combining the 2 smallest nodes to create new node
        newNode = Nodes(left.probability + right.probability, left.symbol + right.symbol, left,
            right)

        the_nodes.remove(left)
        the_nodes.remove(right)
        the_nodes.append(newNode)

    huffmanEncoding = CalculateCodes(the_nodes[0])
    print("symbols with codes", huffmanEncoding)
    TotalGain(the_data, huffmanEncoding)
    encodedOutput = OutputEncoded(the_data,huffmanEncoding)
    return encodedOutput, the_nodes[0]
```

Câu 11 : Hãy khai báo hàm dùng để giải mã bộ mã Huffman như sau :

```
def HuffmanDecoding(encodedData, huffmanTree):
    treeHead = huffmanTree
    decodedOutput = []
    for x in encodedData:
        if x == '1':
            huffmanTree = huffmanTree.right
        elif x == '0':
            huffmanTree = huffmanTree.left
        try:
            if huffmanTree.left.symbol == None and huffmanTree.right.symbol == None:
                pass
        except AttributeError:
            decodedOutput.append(huffmanTree.symbol)
            huffmanTree = treeHead

    string = ''.join([str(item) for item in decodedOutput])
    return string
```

Câu 12 : Hãy thử dùng những hàm đã viết ở trên để mã hóa đoạn Văn bản sau với giải thuật Huffman: “AAAAAAABBCCCCCCDDDEEEEEEEEE”

Câu 13 : Hãy thử nghiệm lại kết quả trên với những đoạn văn bản khác.

Phụ Lục

Bài giải cho giải thuật kiểm tra tính tách được của bảng mã bài Thực hành 02 :

```
def isempty(a):
    return len(a) == 0
def getsuffix(a,b):
    s1 = set()
    for i in a:
        for j in b:
            if(i.startswith(j) and i!=j):
                suf = i.rsplit(j,1)[1]
                s1.add(suf)
    return s1
```

```
def kiemtrabangma(a):
    s0 = a
    l1 = []
    flag = True
    s1 = getsuffix(a,a)
    print(s1)
    l1.append(s0)
    l1.append(s1)
    i = 1
    t = 0
    while (not isempty(s1)):
        s2 = set()
        i += 1

        s2 = getsuffix(list(s0),list(s1))
        print(s2)
        stemp = s2.intersection(s0)
        for j in l1:
            if(s2 == j):
                flag = False
                t = l1.index(j)
                break
        if(isempty(s2)):
            print("Day la bang ma tach duoc")
            break
        elif( not isempty(stemp) ) :
            print("Day la bang ma khong tach duoc do S{} giao voi S0 = {}".format(i,stemp))
            break
        elif(flag == False):
            print("Day la bang ma khong tach duoc do S{} = S{}".format(i,t))
            break
        else:
                l1.append(s2)
                s1 = s2
```