选23、线性表的类实现

1、通过编写Python类来实现线性表的抽象数据类型（数组实现）

```
class array:
    def __init__(self,max_length):
        self.array=[None]*max_length
        self.length=0
    def data(self):   #获取当前数组中的有效数据
        return self.array[:self.length]
    def insert(self,n,pos=None):   #插入数据，默认插入到最后面
        if pos==None:
            pos=self.length
        for i in range(self.length,pos,-1):
            self.array[i]=self.array[i-1]
        self.array[pos]=n
        self.length+=1
    def pop(self,pos=None):   #吐出数据，默认吐出最后面的数据
        if pos==None:
            pos=self.length-1
        res=self.array[pos]
        for i in range(pos,self.length-1):
            self.array[i]=self.array[i+1]
        self.array[self.length-1]=None
        self.length-=1
        return res
    def sort(self,ascending=True):  #得到排序结果
        arr=self.array[:self.length]
        for i in range(self.length-1):
            for j in range(self.length-1,i,-1):
                if ascending:
                    if arr[j]<arr[j-1]:
                        arr[j],arr[j-1]=arr[j-1],arr[j]
                else:
                    if arr[j]>arr[j-1]:
                        arr[j],arr[j-1]=arr[j-1],arr[j]
        return arr
    def search(self,key):   #数据查找
        for i in range(self.length):
            if key==self.array[i]:
                return i
        return None
```

调用：

```
a=array(10)
a.insert(10)
a.insert(2,0)
a.insert(3,1)
a.insert(4,1)
a.insert(1,0)
print("数组数据为： ",a.data())
print("数组降序排序后结果为： ",a.sort(ascending=False))
print("吐出最后一个元素:",a.pop())
print("吐出元素后数组数据:",a.data())
print("查找元素4所在位置:",a.search(4))
```

```
数组数据为： [1, 2, 4, 3, 10]
数组降序排序后结果为： [10, 4, 3, 2, 1]
吐出最后一个元素： 10
吐出元素后数组数据： [1, 2, 4, 3]
查找元素4所在位置： 2
```

2、通过继承array类来实现队列的抽象数据类型

类的继承是面相对象程序设计的精髓之一，通过继承类来实现在避免重复造轮子的情况下改造轮子

```
#队列
class Queue(array):
    queue=None
    def __init__(self,maxlength):
        self.queue=array(maxlength)
    def dataIn(self,data):
        self.queue.insert(data)
    def dataOUT(self):
        return self.queue.pop(0)
    def dataLength(self):
        return self.queue.length
```

```
q=Queue(10)
q.dataIn(10)
q.dataIn(2)
q.dataIn(3)
print(q.dataOUT())
print(q.dataLength())
```

```
10
2
```

3、请同学们探索stack（栈）类的实现

参考：

```
#栈
class Stack(array):
    stack=None
    def __init__(self,maxdepth):
        self.stack=array(maxdepth)
    def push(self,data):
        self.stack.insert(data)
    def pop(self):
        return self.stack.pop()
    def depth(self):
        return self.stack.length
```

```
st=Stack(10)
print(st.depth())
st.push("A")
st.push("B")
st.push("C")
print(st.depth())
print(st.pop())
```

```
0
3
C
```