问题解答

Python中的特殊方法

【H3】栈/队列应用/链表操作

课堂练习：模拟仿真

数据结构与算法（Python）-08/0317

陈斌 gischen@pku.edu.cn 北京大学地球与空间科学学院

目录

问题解答

- 1.H3第5题（问题在切片、__eq__以及__iter__原理及实现）
- 2.__getitem__的用法
- 3.如何查找到python中各种类所含方法的具体实现，以及语句调用了哪些东西（比如list切片调用了getitem）
- 4.represent，slice，getitem的写法
- 5.函数内部是否可以定义类
- 6.class的定义，包括但不限于左右带下划线的是什么意思（如__getitem__）

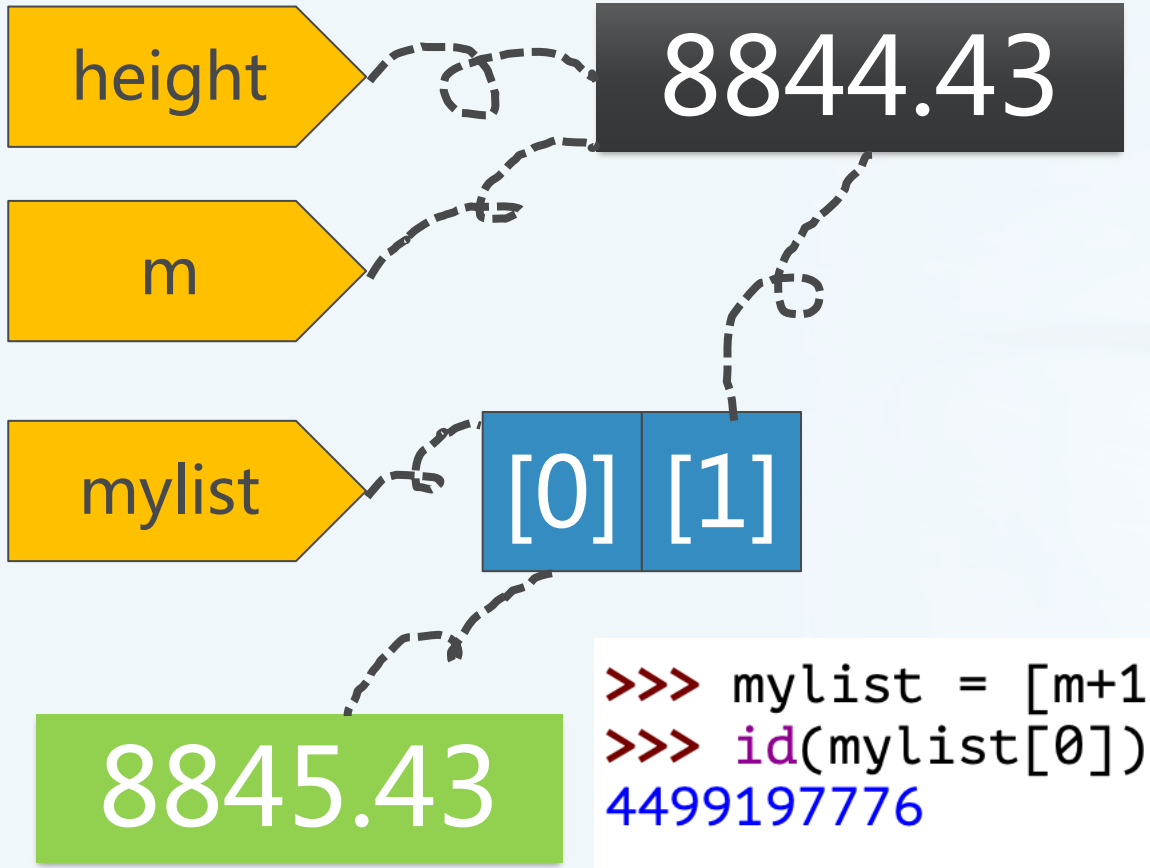
问题解答

- 7.双链表倒序检验时，为什么会出现Nonetype没有getNext函数的报错
- 8迭代器可以在哪里学

  推导式：https://www.bilibili.com/video/av87882118?p=51

  生成器：https://www.bilibili.com/video/av87882118?p=52
- 9.mooc讲的难度不够上课和作业用的
- 10.少留点作业
- 11. Python的引用和C语言的指针有什么关系

Python中的变量与对象




```
>>> height = 8844.43
>>> type(height)
<class 'float'>

>>> id(height)
4503840496

>>> m = height
>>> type(m)
<class 'float'>

>>> id(m)
4503840496
```

```
>>> mylist = [m+1, height]
>>> id(mylist[0])
4499197776

>>> id(mylist[1])
4503840496
```

```
>>> type(mylist)
<class 'list'>

>>> id(mylist)
4492984800
```

面向对象：类定义中的特殊方法

- 基本概念
- 构造与解构
- 算术运算
- 其他特殊方法

基本概念

- **特殊方法(special method)**

  也被称作魔术方法(magic method)

  在类定义中实现一些特殊方法，可以方便地使用python中一些内置操作

  所有特殊方法的名称以两个下划线(__)开始和结束

构造与解构

〉对象构造器

`__init__(self,[…)`

对象的构造器，实例化对象时调用

〉析构器

`__del__(self,[…)`

销毁对象时调用

构造与解构


```
from os.path import join

class FileObject:
    '''给文件对象进行包装从而确认在删除时文件流关闭'''

    def __init__(self, filepath='~', filename='sample.txt'):
        #读写模式打开一个文件
        self.file = open(join(filepath, filename), 'r+')

    def __del__(self):
        self.file.close()
        del self.file
```

算术运算

a+b
b.__radd__(a)

〉算术操作符

__add__(self,other): 使用+操作符

__sub__(self,other): 使用-操作符

__mul__(self,other): 使用*操作符

__div__(self,other): 使用/操作符

〉反运算

当左操作数不支持相应的操作时被调用

__radd__(self,other), __rsub__(self,other)

__rmul__(self,other), __rdiv__(self,other)

算术运算

〉大小比较

__eq__(self,other): 使用==操作符

__ne__(self,other): 使用!=操作符

__lt__(self,other): 使用<操作符

__gt__(self,other): 使用>操作符

__le__(self,other): 使用<=操作符

__ge__(self,other): 使用>=操作符

算术运算


```
13          __add__ = add
14
15      def __str__(self):
16          return "F<%s,%s>" % (self.fx, self.fy)
17
18      def __mul__(self, n):
19          x, y = self.fx * n, self.fy * n
20          return Force(x, y)
21
22      def __eq__(self, force2):
23          return (self.fx == force2.fx) and \
24                 (self.fy == force2.fy)
```

算术运算

```
a=Force(1,2)
a.add(b)
def add(self, b):
Force.add(a, b)
```



```
37  # 操作符使用
38  f3 = f1 + f2
39  print("Fadd=%s" % (f3,))
40  f3 = f1 * 4.5
41  print("Fmul=%s" % (f3,))
42  print("%s==%s? -> %s" % (f1, f2, f1 == f2))
```

```
Fadd=F<3,5>
Fmul=F<0.0,4.5>
F<0,1>==F<3,4>? -> False
```

其他特殊方法

len(a)
a.__len__()

字符串操作

不仅数字类型可以使用像+(__add__())和-(__sub__())的数学运算符，例如字符串类型可以使用+进行拼接，使用*进行复制

__str__(self): 自动转换为字符串

__repr__(self): 返回一个用来表示对象的字符串

__len__(self): 返回元素个数

其它特殊方法参见课程网站

http://gis4g.pku.edu.cn/python-magic-method/

Python中的特殊方法

- **视频教程：**
  https://www.bilibili.com/video/av87882118?p=46
- **文章：**
  http://gis4g.pku.edu.cn/python-magic-method-2/
  http://gis4g.pku.edu.cn/python-magic-method/

【H3】作业详解：1中缀表达式求值：回顾下

代码

```
from pythonds.basic.stack import Stack

def infixToPostfix(infixexpr):
    prec = {}
    prec["*"] = 3
    prec["/"] = 3
    prec["+"] = 2
    prec["-"] = 2
    prec["("] = 1
    opStack = Stack()
    postfixList = []
    tokenList = infixexpr.split()

    for token in tokenList:
        if token in "ABCDEFGHIJKLMNOPQRSTUVWXYZ" or token in "0123456789":
            postfixList.append(token)
        elif token == '(':
            opStack.push(token)
        elif token == ')':
            topToken = opStack.pop()
            while topToken != '(':
                postfixList.append(topToken)
                topToken = opStack.pop()
        else:
            while (not opStack.isEmpty()) and \
               (prec[opStack.peek()] >= prec[token]):
                  postfixList.append(opStack.pop())
            opStack.push(token)

    while not opStack.isEmpty():
        postfixList.append(opStack.pop())
    return " ".join(postfixList)
```

```
def postfixEval(postfixExpr):
    operandStack = Stack()
    tokenList = postfixExpr.split()

    for token in tokenList:
        if token in "0123456789":
            operandStack.push(int(token))
        else:
            operand2 = operandStack.pop()
            operand1 = operandStack.pop()
            result = doMath(token,operand1,operand2)
            operandStack.push(result)
    return operandStack.pop()
```

记录操作符优先级

解析表

操作数

(

)

操作符

操作符

操作数

操作符

合成后缀表达式字符串

【H3】作业详解：1中缀表达式求值


```
44  def calculate(s) -> float:
45      # 请在此编写你的代码（可删除pass语句）
46      def doMath(op, op1, op2):
47          if op == "*":
48              return op1 * op2
49          elif op == "/":
50              return op1 / op2
51          elif op == "+":
52              return op1 + op2
53          elif op == "-":
54              return op1 - op2
55          else: # ^
56              return op1 ** op2
57
58      def doOperand(op):
59          operand2 = operandStack.pop()
60          operand1 = operandStack.pop()
61          result = doMath(topToken, operand1, operand2)
62          operandStack.push(result)
```

```
    prec = {"^": 4, "*": 3, "/": 3, "+": 2, "-":2, "(":1}
    opStack = Stack()
    operandStack = Stack()
    tokenList = s.split()
    for token in tokenList:
        if token.isnumeric():
            operandStack.push(float(token))  # 操作数入栈
        elif token == "(":
            opStack.push(token)  # 左括号入栈
        elif token == ")":
            topToken = opStack.pop()
            while topToken != "(":  # 做计算，取下一个操作符
                doOperand(topToken)
                topToken = opStack.pop()
        else:  # 某个操作符
            while (not opStack.isEmpty()) and \
                  (prec[opStack.peek()] >= prec[token]): # 做计算
                topToken = opStack.pop()
                doOperand(topToken)
            opStack.push(token)

    while not opStack.isEmpty():
        topToken = opStack.pop()
        doOperand(topToken)
    return operandStack.pop()

    # 代码结束
```

【H3】作业详解：2基数排序


```
39  def radix_sort(s) -> list:
40      # 请在此编写你的代码（可删除pass语句）
41      # 1. 全部入队main
42      main = Queue()
43      for n in s:
44          main.enqueue(n)
45      # 2. 找最大数，以及它的位数
46      d = len(str(max(s)))
47      dstr = "%%0%dd" % d  # 前导零的模版，如"%05d"，这里的5是最大数的位数
48      # 3. 准备10个队列
49      nums= [Queue() for _ in range(10)]
50      # 4. 进行按位基数排序
51      for i in range(-1, -d-1, -1): # 一趟下标-1～-d,代表个位到最高位
52          while not main.isEmpty(): # 从main分发出去
53              n = main.dequeue()
54              dn = (dstr % n)[i] # 转成类似"00345"[-2]，这是倒数第二位
55              nums[int(dn)].enqueue(n)
56
57          for k in range(10): # 从10个队列挨个集中到main
58              while not nums[k].isEmpty():
59                  main.enqueue(nums[k].dequeue())
60      # 5. 从main导出为列表就可以了!
61      result = []
62      while not main.isEmpty():
63          result.append(main.dequeue())
64      return result
65      # 代码结束
```

熟练捣腾字符串！

【H3】作业详解：3HTML标记匹配

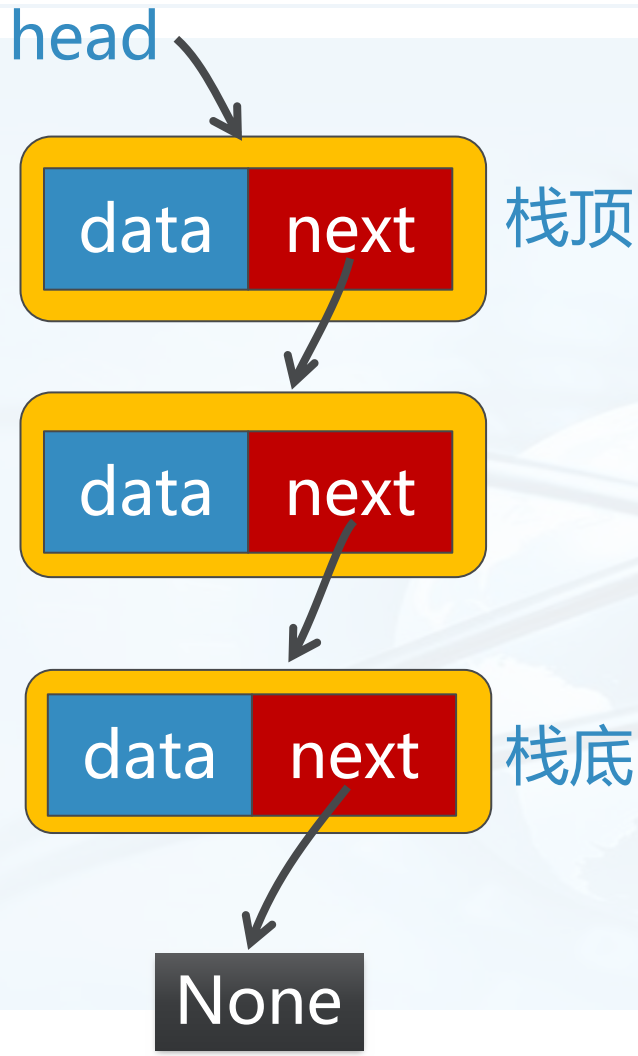

```python
def HTMLMatch(s) -> bool:
    # 请在此编写你的代码（可删除pass语句）
    def isOpenTag(tag):  # 判断是否开标记
        return tag[1]!= "/"

    def matches(open, close):  # 判断两个是否配对
        return open==close.replace("/", "")

    def getTag(s, i): # 从i位置"<"开始找到一个标记，<tag>或者</tag>
        t = ""
        while s[i] != ">":
            t+= s[i]
            i += 1
        t += ">"
        return t, i  # 返回标记和结束位置
```

【H3】作业详解：3HTML标记匹配

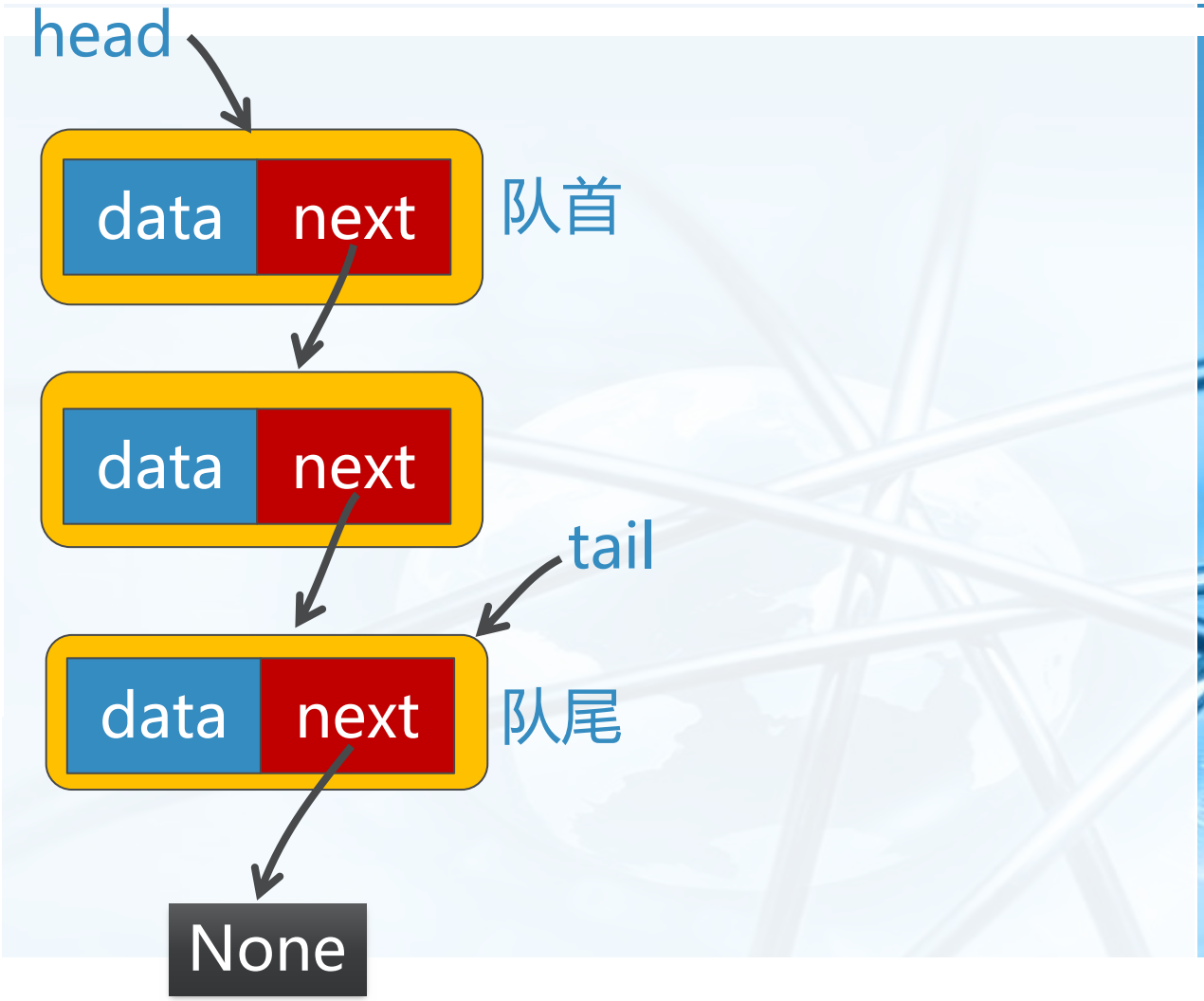

```
    st = Stack()
    balanced = True
    index = 0
    while index < len(s) and balanced:
        symbol = s[index]
        if symbol == "<":
            tag, index = getTag(s, index)
        if isOpenTag(tag):
            st.push(tag)
        else:
            if st.isEmpty():
                balanced = False
            else:
                top = st.pop()
                if not matches(top, tag):
                    balanced = False
        # 忽略所有非标记内容
        while index < len(s) and s[index] != "<":
            index += 1
    if balanced and st.isEmpty():
        return True
    else:
        return False
    # 代码结束
```

【H3】作业详解：4链表实现栈和队列


```
28  class LinkStack():
29      # 请在此编写你的代码（可删除pass语句）
30      def __init__(self):
31          self.head = None # 链表的头作为栈顶
32          self.length = 0
33
34      def isEmpty(self):
35          return self.head is None
36
37      def size(self):
38          return self.length
39
40      def peek(self):
41          return self.head.getData()
42
43      def push(self, item):
44          top = Node(item)
45          top.setNext(self.head)
46          self.head = top
47          self.length += 1
48
49      def pop(self):
50          top_item = self.head.getData()
51          self.head = self.head.getNext()
52          self.length -= 1
53          return top_item
54
55      # 代码结束
```

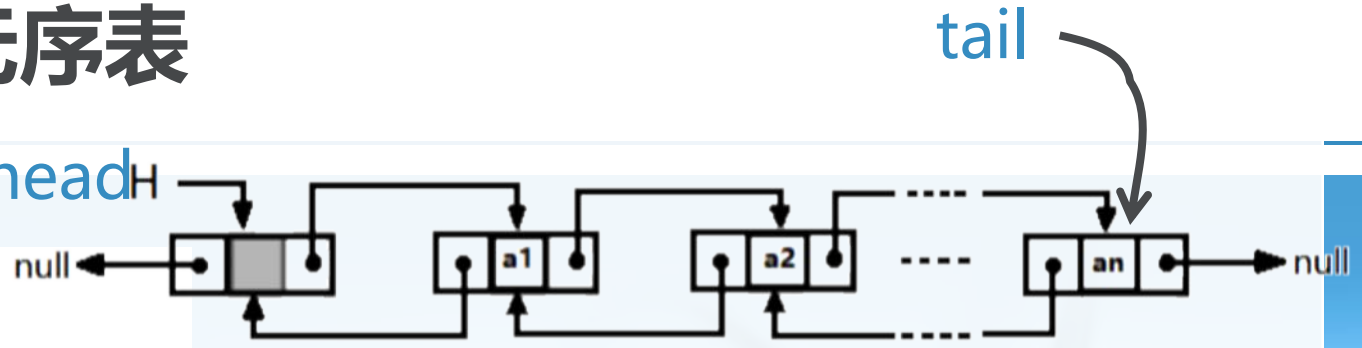


```python
class LinkQueue():
    # 请在此编写你的代码（可删除pass语句）
    def __init__(self):
        self.head = None # 链表头作为队列的头
        self.tail = None # 链表尾作为队列的尾
        self.length = 0

    def isEmpty(self):
        return self.head is None

    def size(self):
        return self.length

    def enqueue(self, item):
        tail = Node(item)
        if self.head is None:
            # 第一个入队
            self.head = self.tail = tail
        else:
            self.tail.setNext(tail)
            self.tail = self.tail.getNext()
        self.length += 1
    def dequeue(self):
        top_item = self.head.getData()
        if self.head == self.tail:
            # 唯一一个出队
            self.head = self.tail = None
        else:
            self.head = self.head.getNext()
        self.length -= 1
        return top_item

```



【H3】作业详解：5双链无序表

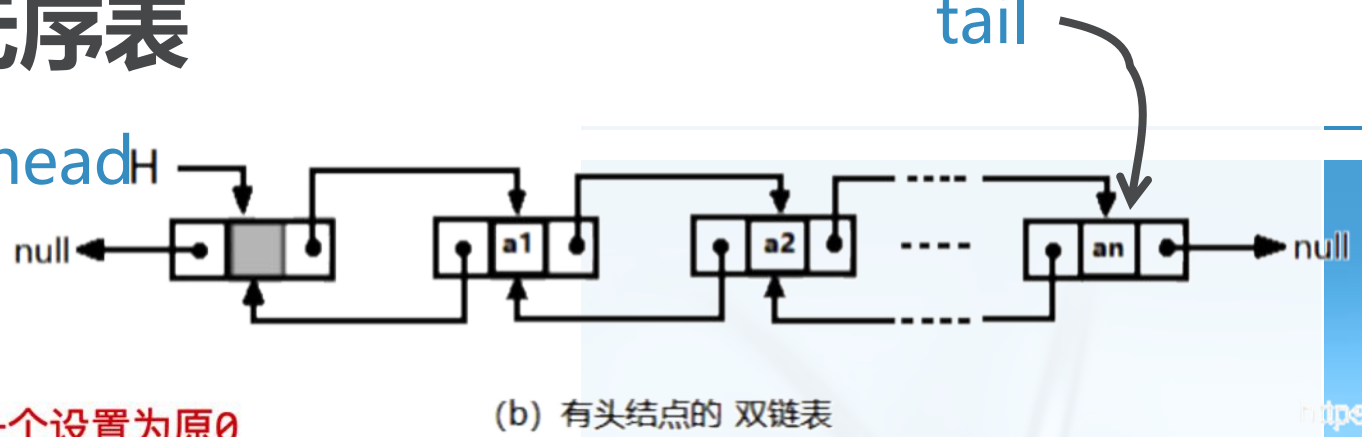
```
class DoublyLinkedList():
    # 请在此编写你的代码（可删除pass语句）
    def __init__(self, it= None):
        self.head = None # 表头作为index(0)
        self.tail = None # 表尾作为index(-1)
        self.length = 0
        if it is not None:
            for d in it:
                self.append(d)

    def isEmpty(self):
        return self.head is None

    def size(self):
        return self.length
    __len__= size

    def getTail(self):
        return self.tail

```



(b) 有头结点的 双链表

【H3】作业详解：5双链无序表

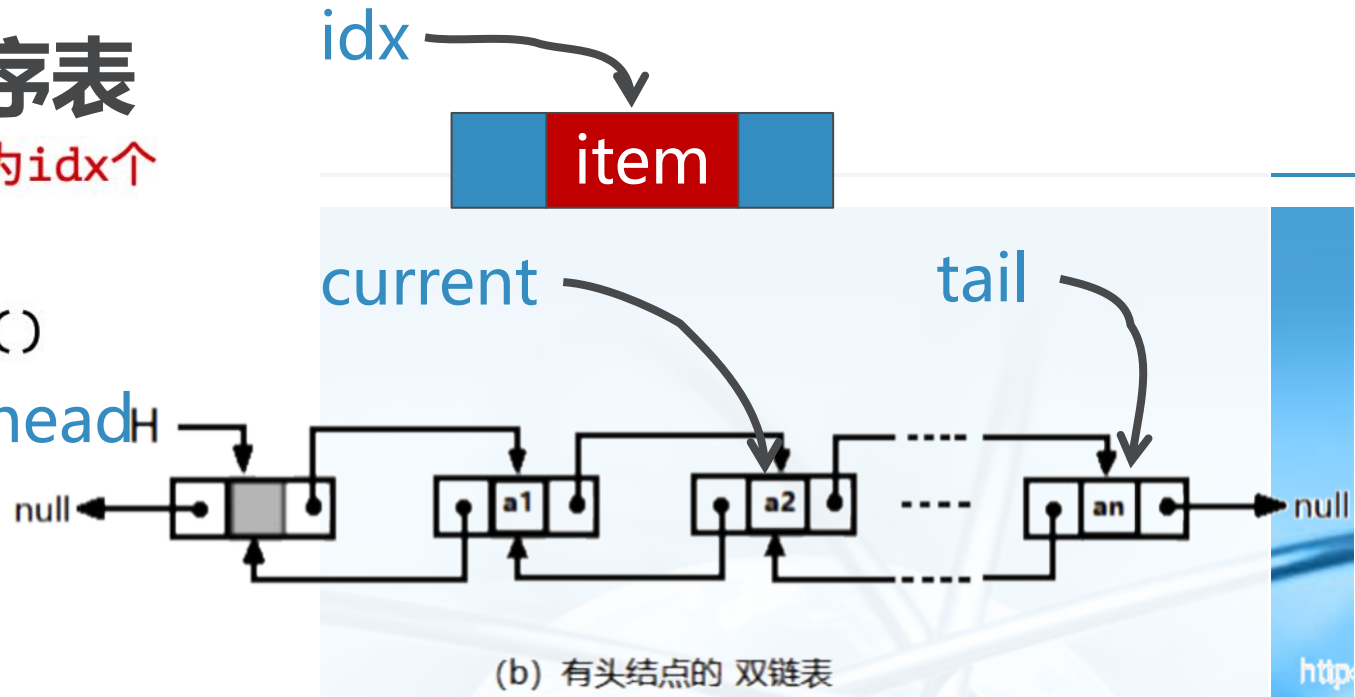
```
54      def add(self, item): # 加为第一个
55          idx0 = Node(item)
56          if self.head is None:
57              # 第一个加入列表
58              self.head = self.tail = idx0
59          else:
60              # 原0的前一个设为idx0，idx0的后一个设置为原0
61              self.head.setPrev(idx0)
62              idx0.setNext(self.head)
63              # 头指向idx0
64              self.head = idx0
65          self.length += 1
66
67      def append(self, item): # 加为最后一个
68          idx_1 = Node(item)
69          if self.head is None:
70              # 第一个添加到列表
71              self.head = self.tail = idx_1
72          else:
73              # 原最后一个的后一个设为idx_1，idx_1的前一个设置为原最后一个
74              self.tail.setNext(idx_1)
75              idx_1.setPrev(self.tail)
76              # 尾指向idx_1
77              self.tail = idx_1
78          self.length += 1
```



(b) 有头结点的 双链表

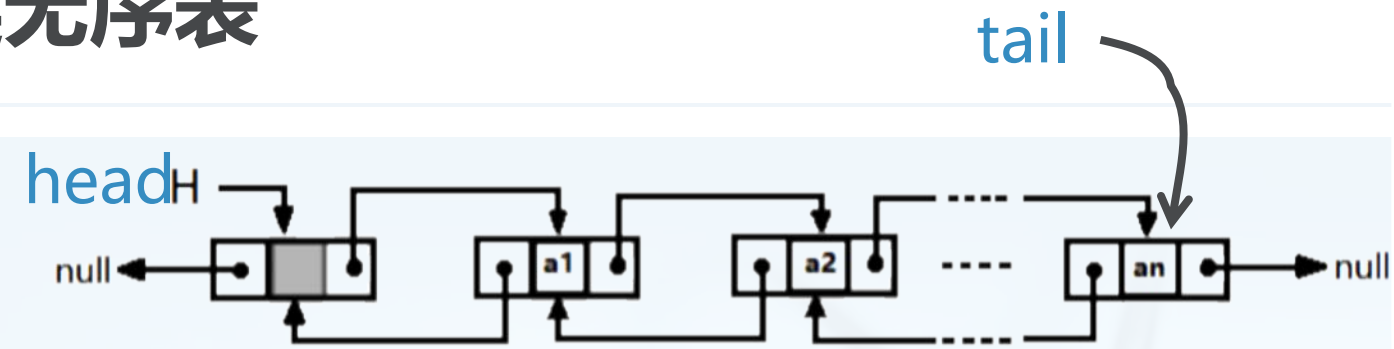
【H3】作业详解：5双链无序表

```
80      def insert(self, idx, item): # 加为idx个
81          current, n = self.head, 0
82          while n< idx:
83              current = current.getNext()
84              n += 1
85          # 插入在current的前面
86          if current is None:
87              if self.head is None:
88                  # 这是第一个插入到列表的
89                  self.add(item)
90              else:
91                  # 这是加为最后一个
92                  self.append(item)
93          else:
94              # 加在中间，idx节点添加到current的前一个
95              idx = Node(item)
96              idx.setNext(current)
97              idx.setPrev(current.getPrev())
98              if idx.getPrev() is not None:
99                  idx.getPrev().setNext(idx)
100             current.setPrev(idx)
101         self.length += 1
```



(b) 有头结点的 双链表

【H3】作业详解：5双链无序表

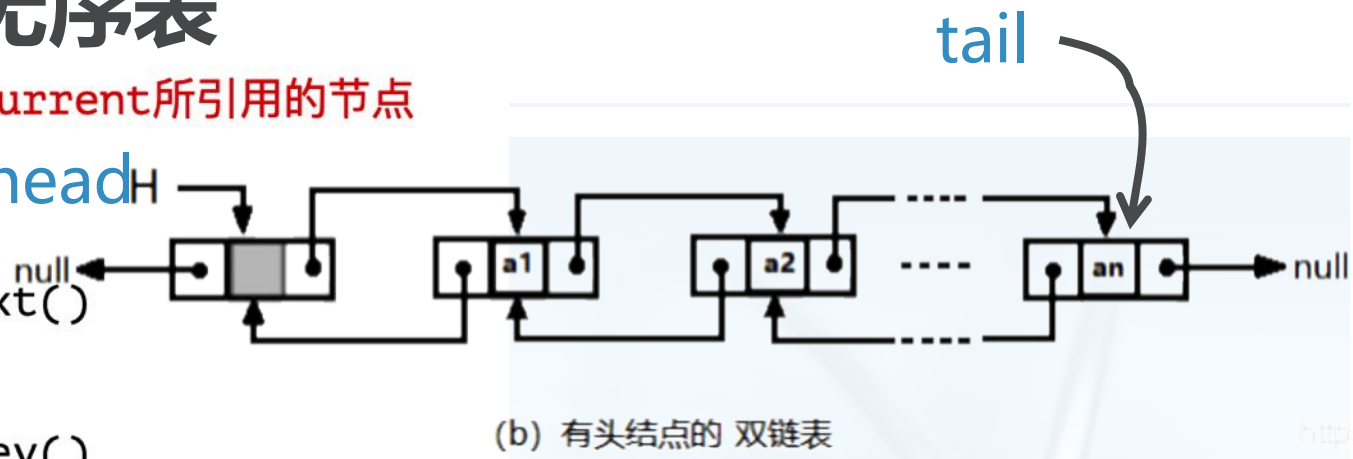


(b) 有头结点的 双链表

```
103     def index(self, item):
104         current, n = self.head, 0
105         while current is not None:
106             if current.getData() == item:
107                 break
108             current = current.getNext()
109             n += 1
110         else:
111             return None
112         return n
113
114     def search(self, item):
115         return self.index(item) is not None
```

【H3】作业详解：5双链无序表

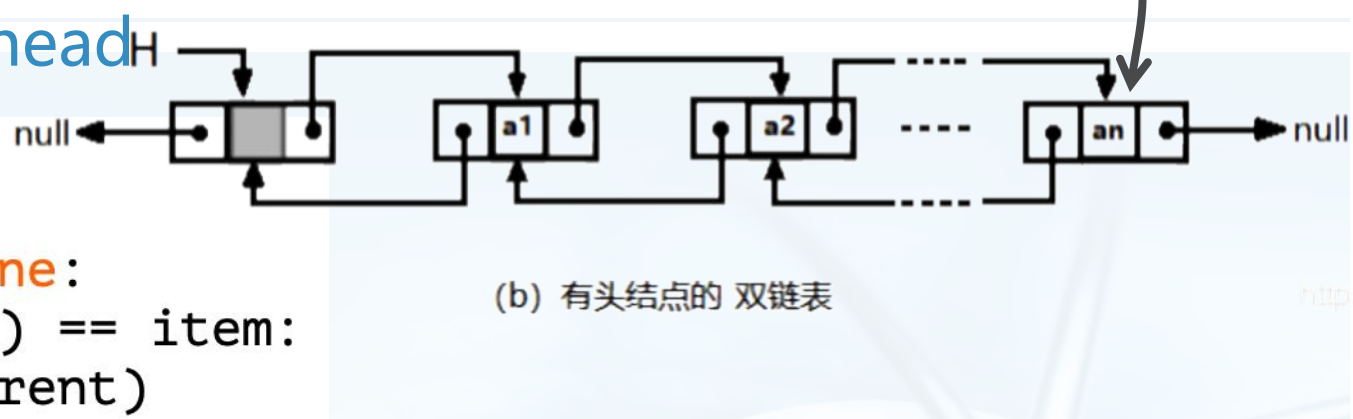
```
117     def delete(self, current): # 删除current所引用的节点
118         # 删除本节点
119         if self.head == current:
120             # 删除了第一个节点
121             self.head = current.getNext()
122         if self.tail == current:
123             # 删除了最后一个节点
124             self.tail = current.getPrev()
125         if current.getPrev() is not None:
126             current.getPrev().setNext(current.getNext())
127         if current.getNext() is not None:
128             current.getNext().setPrev(current.getPrev())
129         self.length -= 1
130 
131     def remove(self, item):
132         current = self.head
133         while current is not None:
134             if current.getData() == item:
135                 self.delete(current)
136                 break
137             current = current.getNext()
```



(b) 有头结点的 双链表

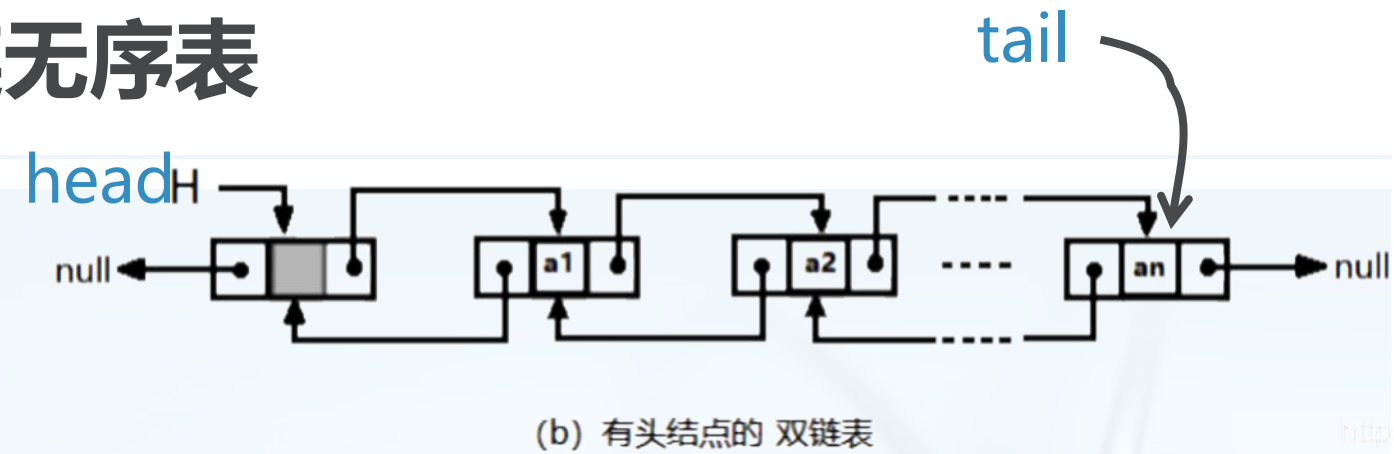
【H3】作业详解：5双链无序表

```
131     def remove(self, item):
132         current = self.head
133         while current is not None:
134             if current.getData() == item:
135                 self.delete(current)
136                 break
137             current = current.getNext()
138 
139     def pop(self, n = None):
140         if n is None:
141             n = self.length - 1
142         current, i = self.head, 0
143         while i < n:
144             current= current.getNext()
145             i += 1
146         dat = current.getData()
147         self.delete(current)
148         return dat
```



(b) 有头结点的 双链表

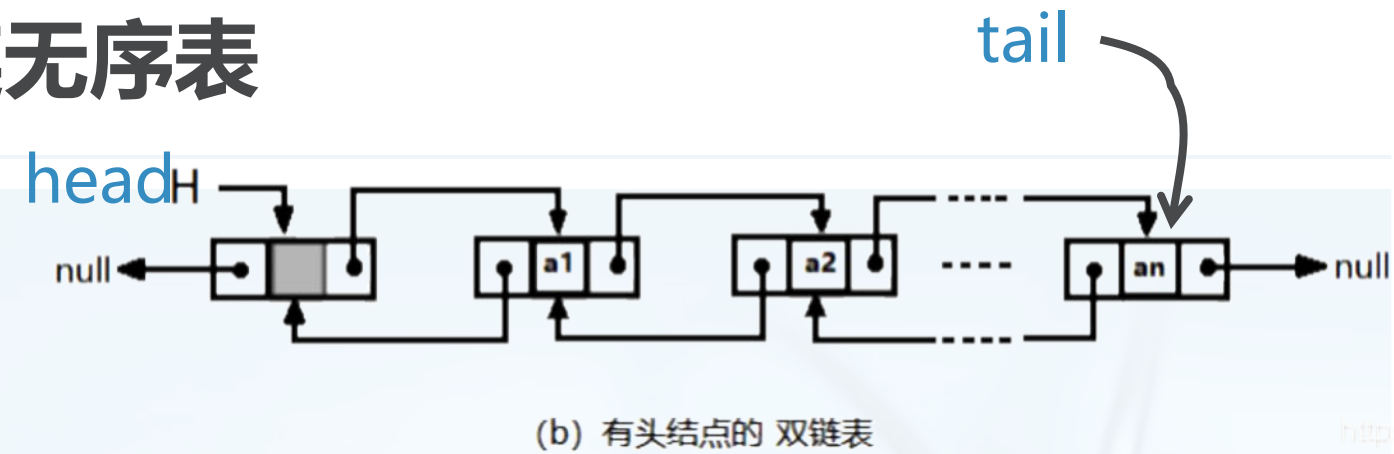
【H3】作业详解：5双链无序表



(b) 有头结点的 双链表

```
150     def __str__(self):
151         tlist = []
152         current = self.head
153         while current is not None:
154             tlist.append(current.getData())
155             current= current.getNext()
156         return str(tlist)
157     __repr__ = __str__
```

【H3】作业详解：5双链无序表



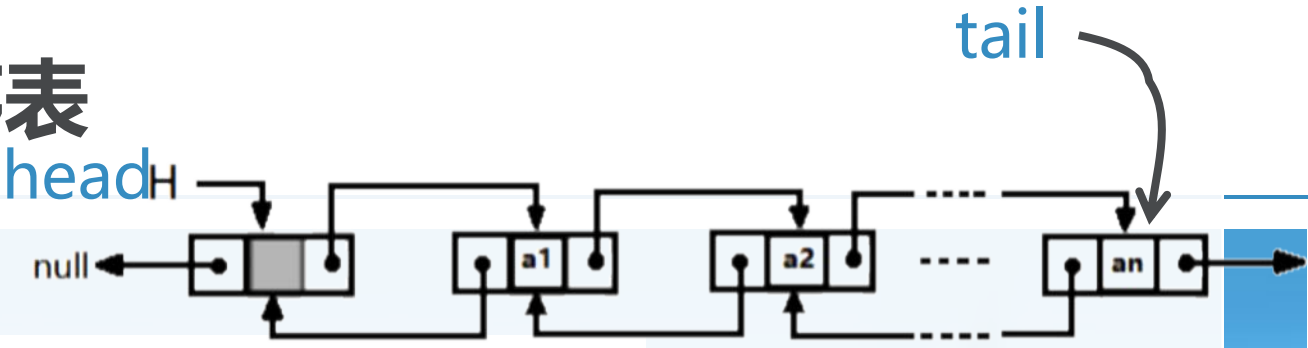
(b) 有头结点的 双链表

lst[9]
lst[1:3:2]

```
159     def __getitem__(self, key):
160         if isinstance(key, int):
161             # 按照下标
162             current, i = self.head, 0
163             while i < key:
164                 current= current.getNext()
165                 i += 1
166             if current is not None:
167                 return current.getData()
168             else:
169                 raise StopIteration
```

https://www.jb51.net/article/87444.htm

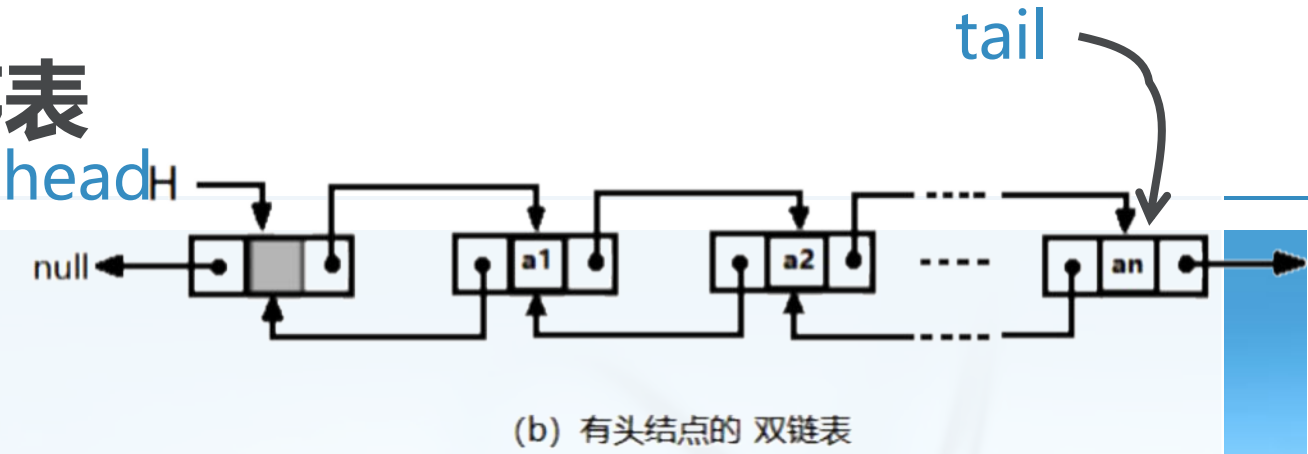
【H3】作业详解：5双链无序表



(b) 有头结点的 双链表

```
170         elif isinstance(key, slice):
171             start = 0 if key.start is None else key.start
172             stop = self.length if key.stop is None else key.stop
173             step = 1 if key.step is None else key.step
174             current, i = self.head, 0
175             # 定位到start
176             while i< start:
177                 current = current.getNext()
178                 i += 1
179             # 开始拷贝
180             dcopy = DoublyLinkedList()
181             while i< stop:
182                 dcopy.append(current.getData())
183                 s = step
184                 while current is not None and s> 0:
185                     current= current.getNext()
186                     s -= 1
187                 i += step
188             # 返回拷贝
189             return dcopy
```

【H3】作业详解：5双链无序表



(b) 有头结点的 双链表

```
191     def __eq__(self, other):
192         if other is None or not isinstance(other, DoublyLinkedList):
193             return False
194         if len(self) != len(other):
195             return False
196         for s, o in zip(self, other):
197             if s != o:
198                 return False
199         else:
200             return True
201
202     # 代码结束
```

课堂练习【K04】模拟仿真支持决策

- 考虑现实生活中的一个能够优化的业务场景，尽量完整定义问题，并设计一个模拟仿真来解决优化决策的问题。
- 仿照课程中的打印机决策优化问题来组织你的作业，不需要编程。
- 请说明你做的各种假设和参数，并指出需要用到的队列和各种概率数据。例如：

  排队等待洗车；

  超市等待结账；

  航班起飞降落；

  银行出纳柜台等等。

- 请将作业做成文档提交。

  （pdf，doc，docx，ppt，pptx）