

桂林电子科技大学

实验 2 文件操作和异常处理

实验报告

实验名称	文件操作和异常处理				辅导员意见： 成绩 辅导员 签 名
院 系	计算机与信息安全学院	专业	软件工程		
学 号	2200350204	姓名	李禹佳		
实验日期	2023	年	12	月 8 日	

一、实验目的

1. 掌握不同文件类型的读写和异常处理；
2. 掌握目录遍历的方法；
3. 掌握 json 解析器的设计与实现；
4. 掌握装饰器的使用；
5. 掌握类的设计和定制

二、实验内容

1. 读取 txt 文件内容；
2. 素数文件写入文件；
3. 遍历输出文件目录；
4. 读写 json 文件；
5. 工资结算计算；
6. Json 解析器；
7. 类的设计；
8. 类的定制。

三、实验环境

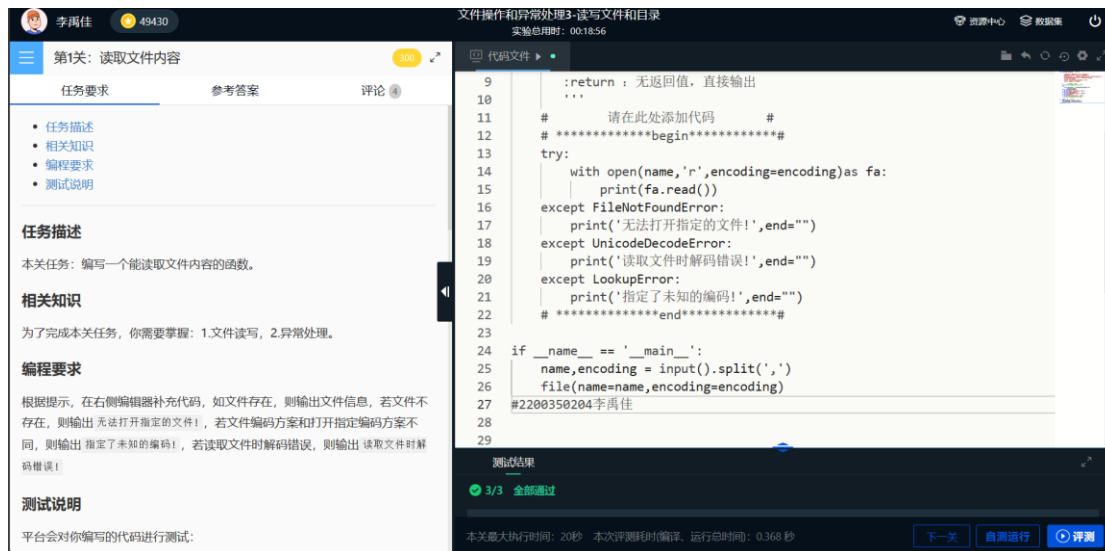
在 Educoder 平台进行实验

四、实验要求

根据每个实训的每个关卡要求完成代码提交和测评

五、实验步骤

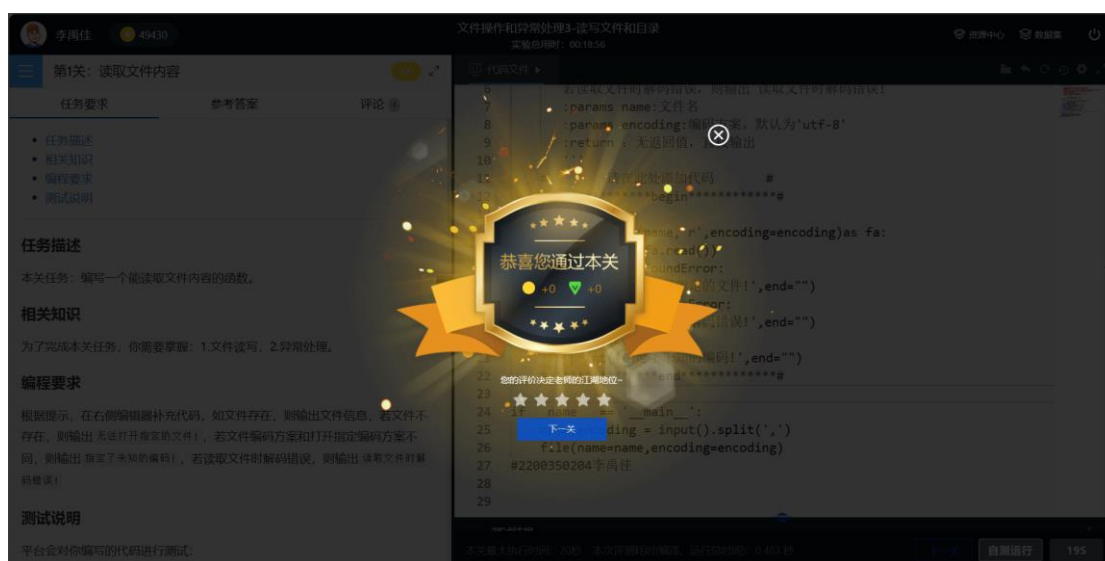
1. 读取 txt 文件内容
代码截图：



在代码中我对异常情况进行了处理，情况分别为文件不存在，文件解码错误，文件编码错误。异常捕获为：FileNotFoundError，UnicodeError，LookupError 如果没有上述异常错误，则打印目标文件内容



通关



- 素数文件写入文件
- 代码截图：

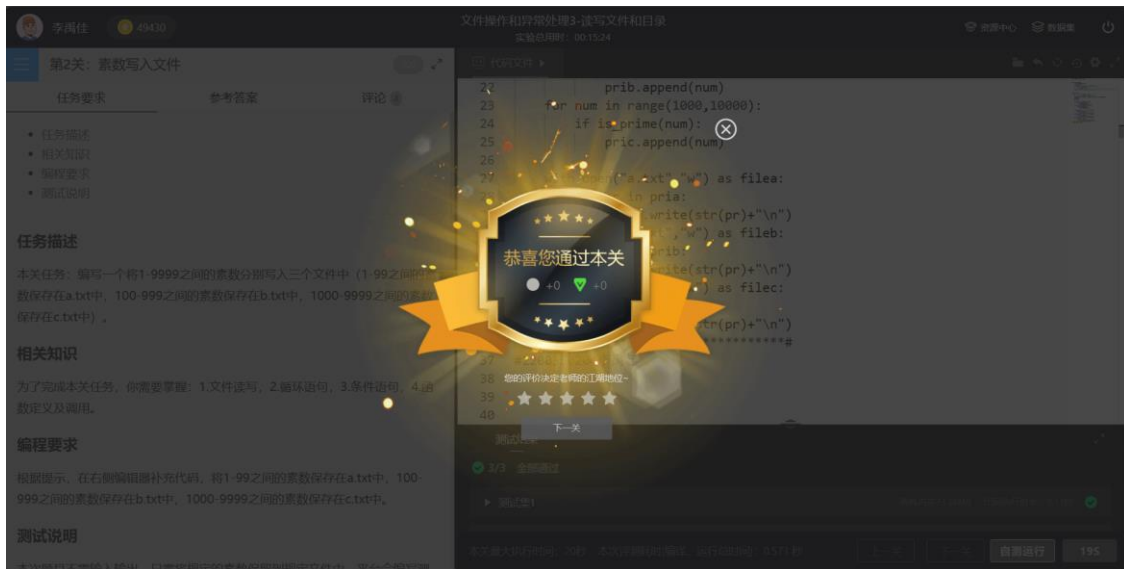
```
1 from math import sqrt
2
3
4 def is_prime(n):
5     """判断素数的函数"""
6     assert n > 0
7     for factor in range(2, int(sqrt(n)) + 1):
8         if n % factor == 0:
9             return False
10    return True if n != 1 else False
11
12
13 def store():
14    # 请在此处添加代码 #
15    # *****begin*****#
16    pria=[num for num in range(1,100) if is_prime(num)]
17    prib=[]
18    pric=[]
19
20    for num in range(100,1000):
21        if is_prime(num):
22            prib.append(num)
23    for num in range(1000,10000):
24        if is_prime(num):
25            pric.append(num)
```

```
17    prib=[]
18    pric=[]
19
20    for num in range(100,1000):
21        if is_prime(num):
22            prib.append(num)
23    for num in range(1000,10000):
24        if is_prime(num):
25            pric.append(num)
26
27    with open("a.txt","w") as filea:
28        for pr in pria:
29            filea.write(str(pr)+"\n")
30    with open("b.txt","w") as fileb:
31        for pr in prib:
32            fileb.write(str(pr)+"\n")
33    with open("c.txt","w") as filec:
34        for pr in pric:
35            filec.write(str(pr)+"\n")
36    # *****end*****#
37    #2200350204李禹佳
38
```

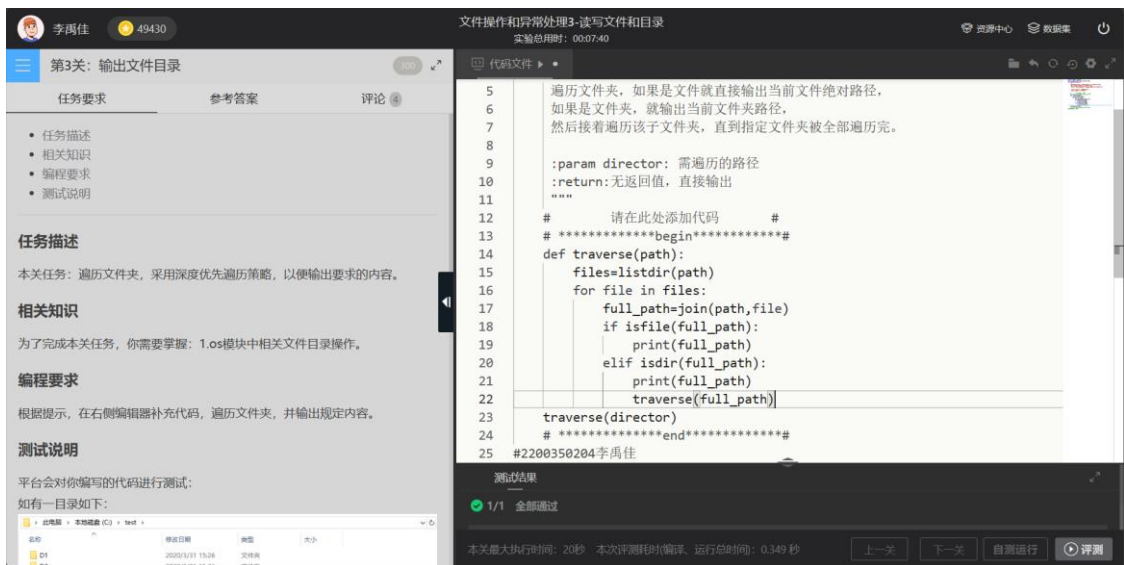
由题将不同范围的素数存入相对应文件，在代码中我首先是用循环来获取不同区间的数再依次判断是否为素数，后尝试使用简写放在了第一个数组中。后使用 `open` 打开文件并使用 `write` 用循环将存入数组的数依次写入相应文件

```
14 # 请在此处添加代码 #
15 # *****begin*****#
16 pria=[num for num in range(1,100) if is_prime(num)]
17 prib=[]
18 pric=[]
19
20 for num in range(100,1000):
21     if is_prime(num):
22         prib.append(num)
23 for num in range(1000,10000):
24     if is_prime(num):
25         pric.append(num)
26
27 with open("a.txt","w") as filea:
28     for pr in pria:
29         filea.write(str(pr)+"\n")
30 with open("b.txt","w") as fileb:
31     for pr in prib:
32         fileb.write(str(pr)+"\n")
33 with open("c.txt","w") as filec:
34     for pr in pric:
35         filec.write(str(pr)+"\n")
36 # *****end*****#
37 #2200350204李禹佳
38
```

通关：



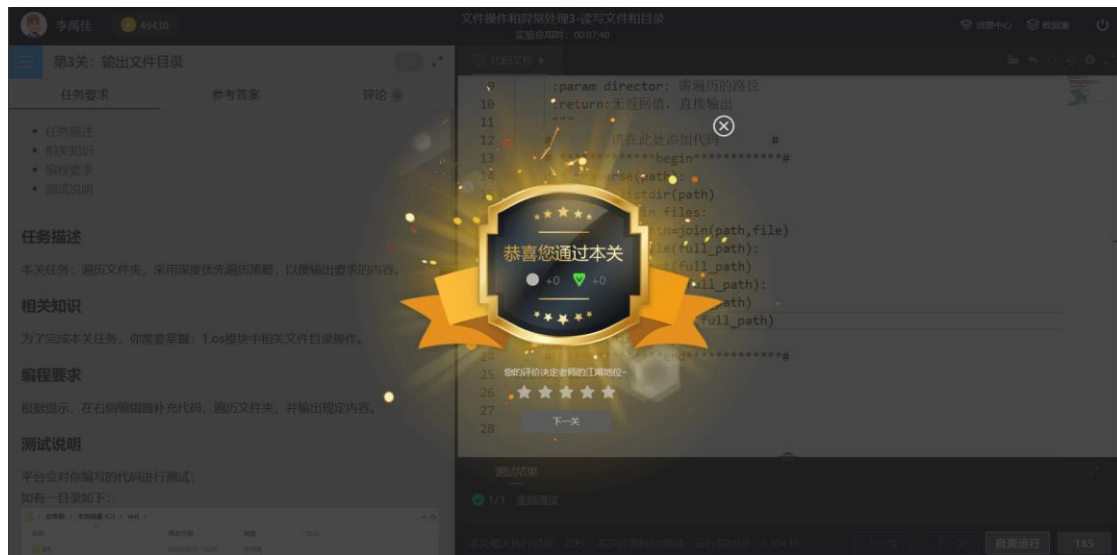
3. 遍历输出文件目录 代码截图：



Traverse 方法首先通过 `listdir(path)` 获取指定路径下的所有文件和文件夹的列表，将它们存储在变量 `files` 中，然后循环遍历 `files` 列表。使用 `join(path, file)` 获取当前文件或文件夹的完整路径，存储在变量 `full_path` 中。通过 `isfile(full_path)` 判断当前遍历的是不是一个文件，是文件则直接打印该文件的完整路径；是文件夹则打印该文件夹下的文件和文件夹。最后，代码调用 `traverse(director)` 函数，传入一个路径作为参数，从该路径开始递归遍历

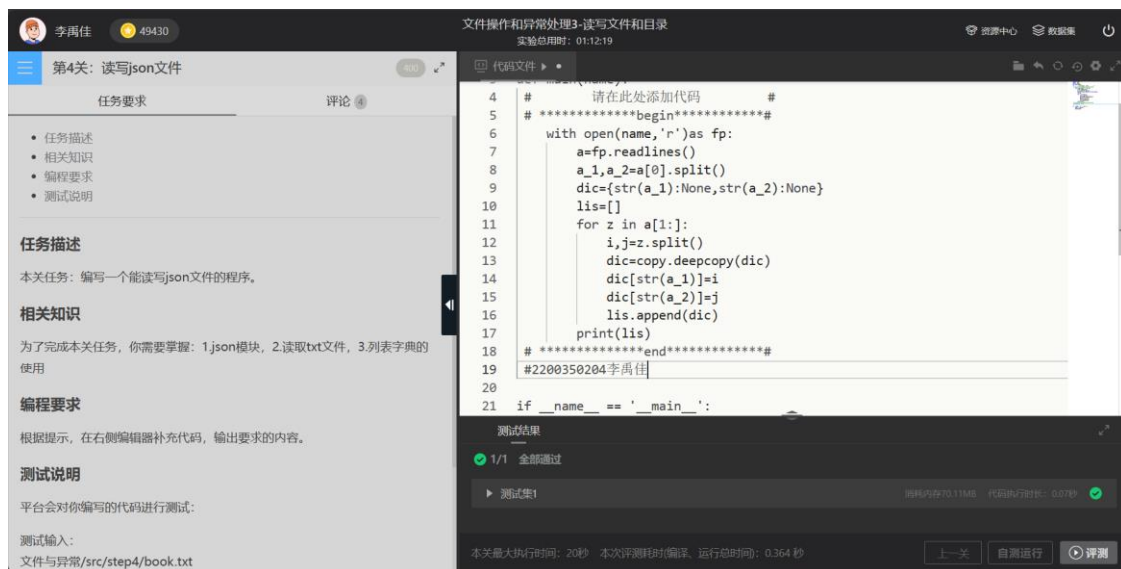


通关：



4. 读写 json 文件

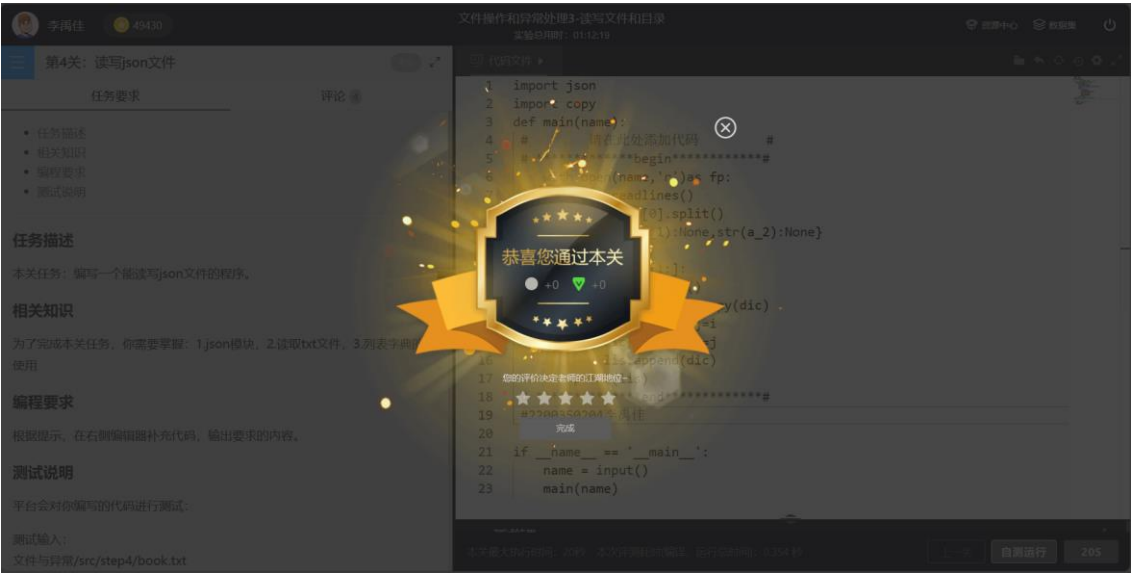
代码截图：



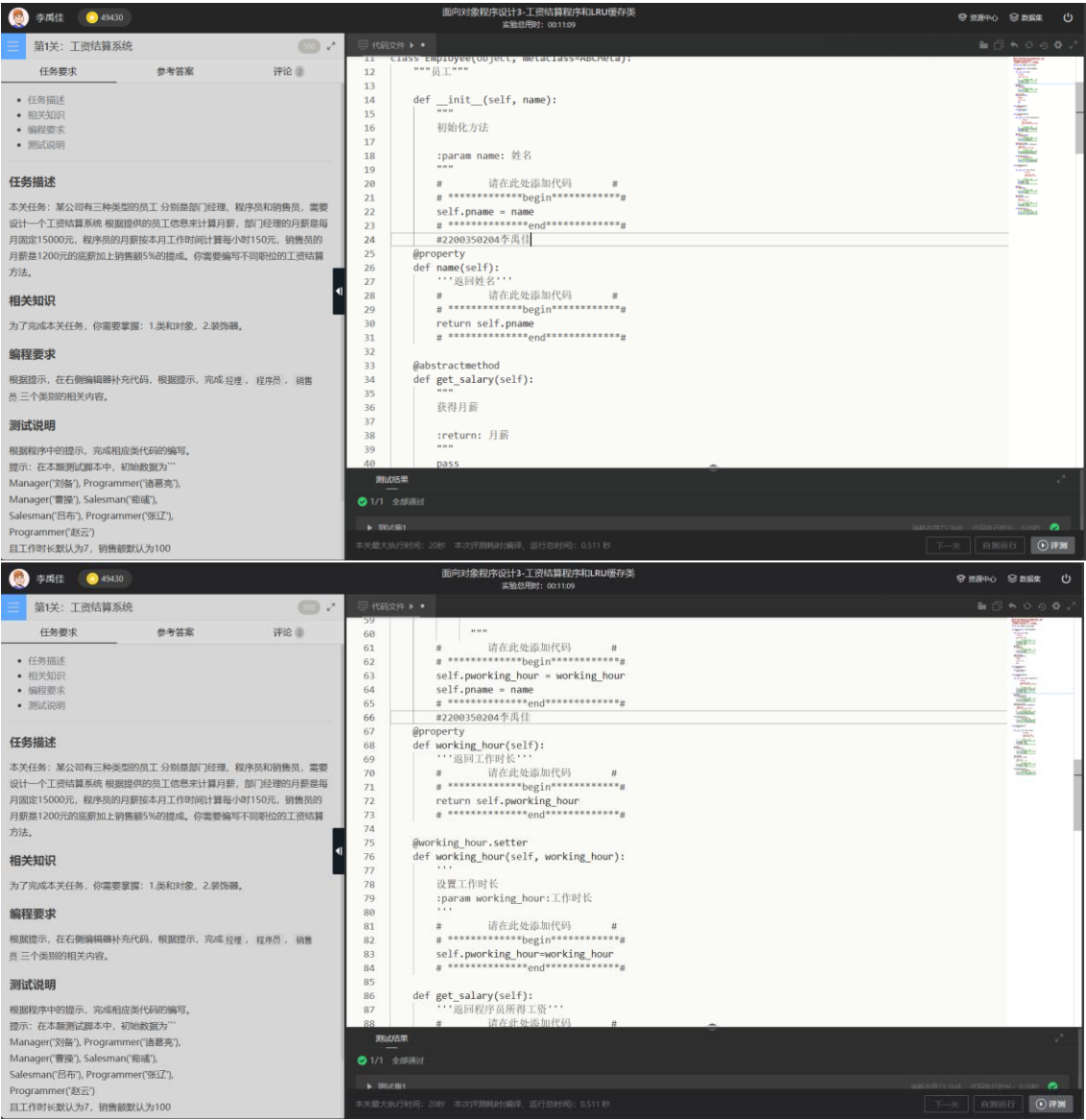
打开 `name` 值所在文件，逐行读取文件内容，并存入 `a`。通过 `split()` 方法按空格分割字符串，并将结果存入 `a_1` 和 `a_2`。循环遍历从第二行开始的所有行，将每行的内容按空格分割为两个值，存储在 `i` 和 `j` 中。通过 `copy.deepcopy(dic)` 创建字典 `dic` 的深拷贝，将深拷贝结果存回。将 `i` 赋值给 `dic[str(a_1)]`，将 `j` 赋值给 `dic[str(a_2)]`。将 `dic` 添加到列表 `lis` 中并打印 `lis`

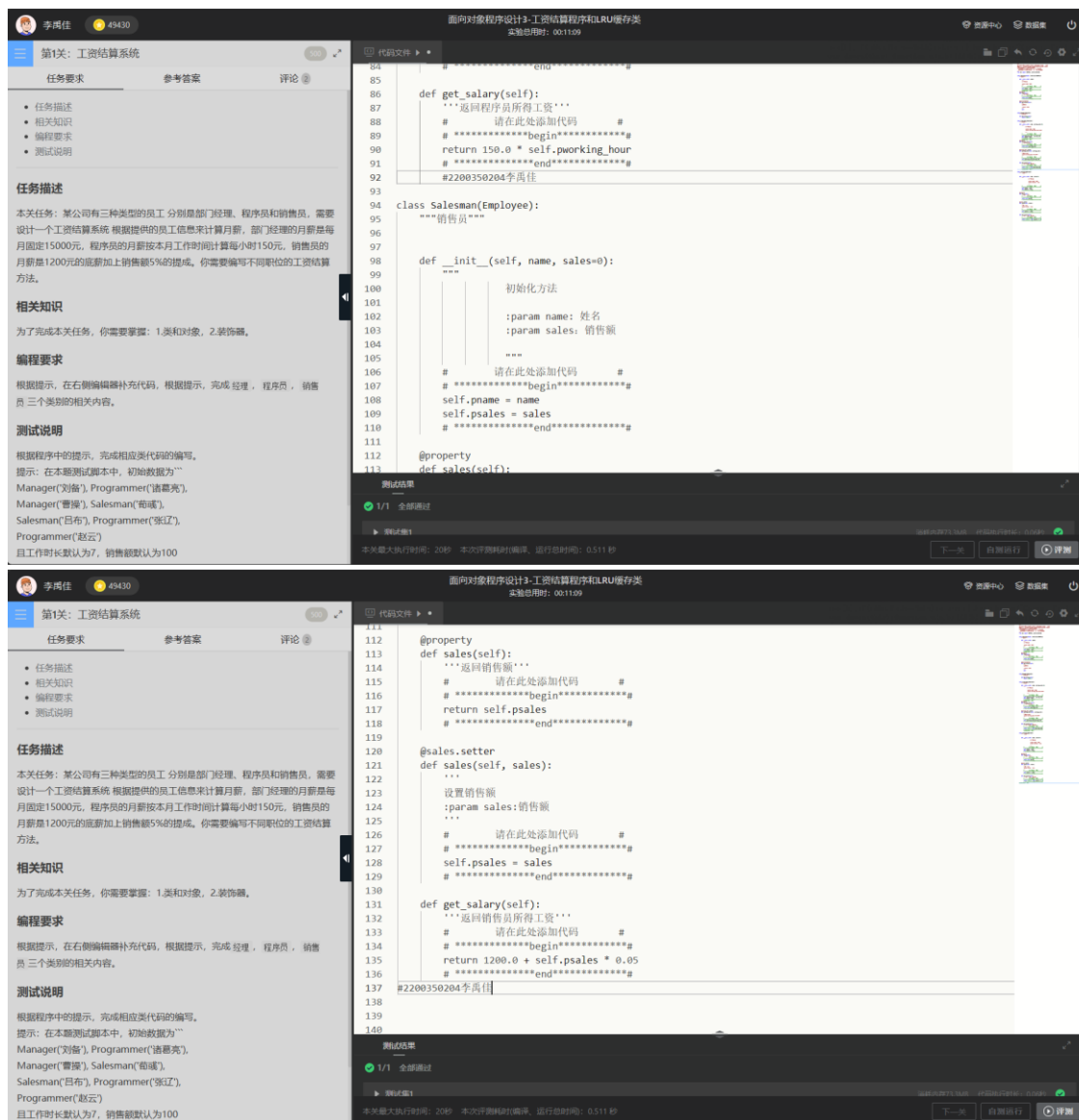


通关：



5. 工资结算计算
代码截图：





再__init__方法中，使用参数 name 初始化 pname,将参数值传入，在 name 方法中使用 getter 方法返回实例的姓名



在__init__方法中，初始化 pname 和 pworking_hour，在 working_hour 方法中使用 getter 方法中返回实例的工作时长，在 get_salary 中返回*150 后的工资数

```
63 self.pname = name
64 self.psales = sales
65 # *****end*****#
66 #2200350204李禹佳
67
68 @property
69 def working_hour(self):
70     """返回工作时长"""
71     # 请在此处添加代码 #
72     # *****begin*****#
73     return self.pworking_hour
74     # *****end*****#
75
76 @working_hour.setter
77 def working_hour(self, working_hour):
78     """
79     设置工作时长
80     :param working_hour: 工作时长
81     """
82     # 请在此处添加代码 #
83     # *****begin*****#
84     self.pworking_hour=working_hour
85     # *****end*****#
86
87 def get_salary(self):
88     """返回程序员所得工资"""
89     # 请在此处添加代码 #
90     # *****begin*****#
91     return 150.0 * self.pworking_hour
92     # *****end*****#
93 #2200350204李禹佳
94
95 class Salesman(Employee):
96     """销售员"""
97     pass
```

在__init__方法中初始化 pname 和 psales 并传入数据，在@property sales 方法中使用 getter 返回实例的销售额，在 @sales.setter sales 方法中使用 setter 将销售额属性改为可修改，在 get_salary 方法中*1200 返回销售员工资

```
105 # 请在此处添加代码 #
106 # *****begin*****#
107 self.pname = name
108 self.psales = sales
109 # *****end*****#
110 #2200350204李禹佳
111
112 @property
113 def sales(self):
114     """返回销售额"""
115     # 请在此处添加代码 #
116     # *****begin*****#
117     return self.psales
118     # *****end*****#
119
120 @sales.setter
121 def sales(self, sales):
122     """
123     设置销售额
124     :param sales: 销售额
125     """
126     # 请在此处添加代码 #
127     # *****begin*****#
128     self.psales = sales
129     # *****end*****#
130
131 def get_salary(self):
132     """返回销售员所得工资"""
133     # 请在此处添加代码 #
134     # *****begin*****#
135     return 1200.0 + self.psales * 0.05
136     # *****end*****#
137 #2200350204李禹佳
138
```

通关：

第1关：工资结算系统

任务要求 参考答案 评论

任务描述

本关任务：某公司有三种类型的员工，分别是部门经理、程序员和销售员。设计一个工资结算系统，根据提供的员工信息计算月薪。部门经理的月薪是月固定15000元，程序员的月薪按本月工作时间计算每小时150元，销售员的月薪是1200元的底薪加上销售额5%的提成。你需要编写不同职位的工资计算方法。

相关知识

为了完成本关任务，你需要掌握：1.类和对象，2.类属性。

编程要求

根据提示，在右侧编辑器补充代码，根据提示，完成经理、程序员、销售员三个类的相关内容。

测试说明

```
109 self.psales = sales
110 # *****end*****#
111
112 @property
113 def sales(self):
114     """返回销售额"""
115     # 请在此处添加代码 #
116     # *****begin*****#
117     return self.psales
118     # *****end*****#
119
120 @sales.setter
121 def sales(self, sales):
122     """
123     设置销售额
124     :param sales: 销售额
125     """
126     # 请在此处添加代码 #
127     # *****begin*****#
```

恭喜您通过本关

+0 +0

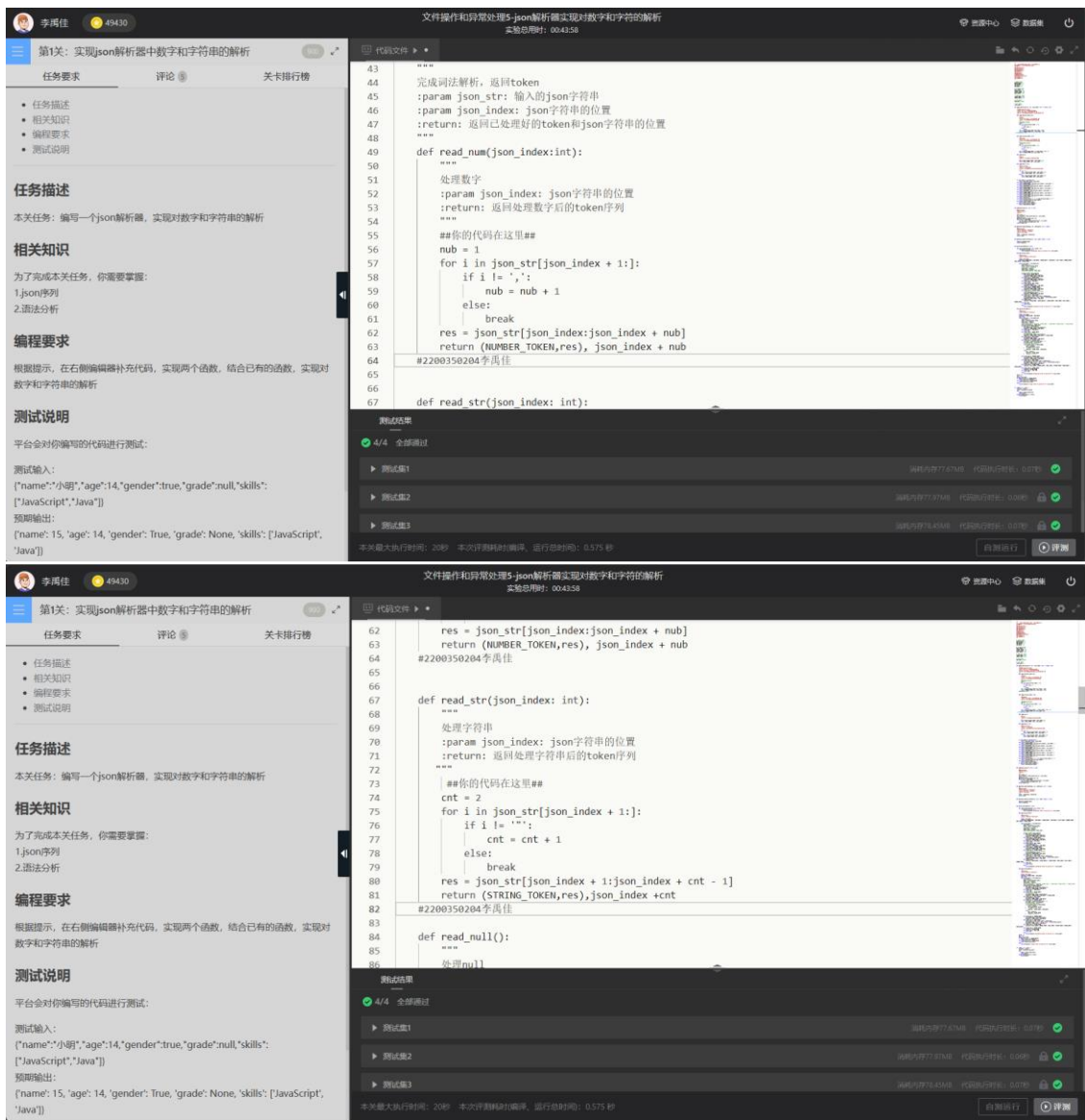
您评价决定老师的工资啦！

1/1 全部通过

下一关

6. Json 解析器

代码截图：



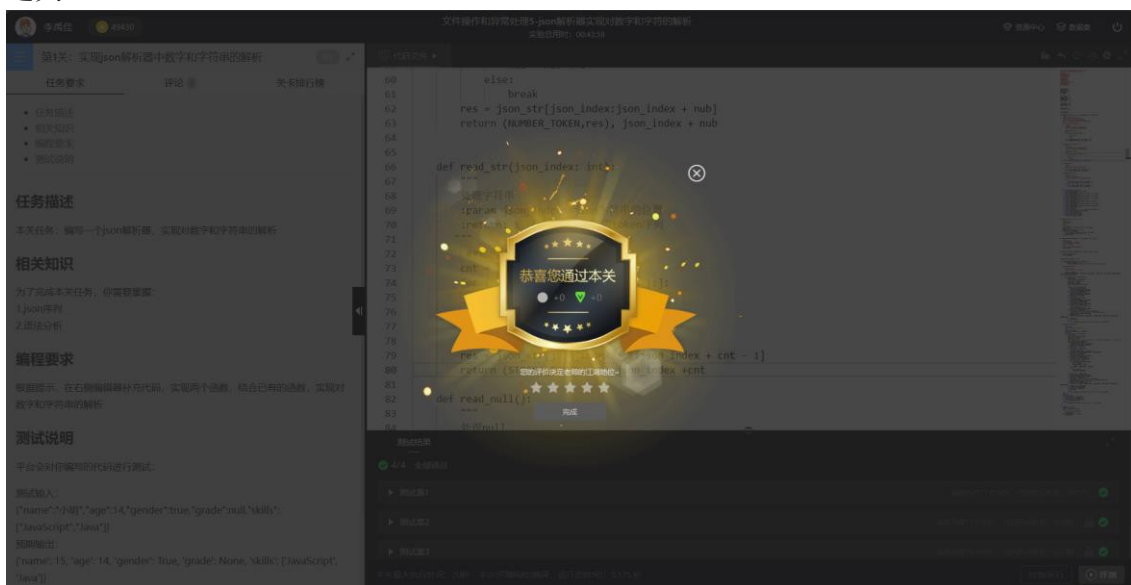
找到数字以,结束



找到以"开始再以"结尾的字符串

```
60         else:
61             break
62         res = json_str[json_index:json_index + nub]
63         return (NUMBER_TOKEN,res), json_index + nub
64     #2200350204李禹佳
65
66
67 def read_str(json_index: int):
68     """
69     处理字符串
70     :param json_index: json字符串的位置
71     :return: 返回处理字符串后的token序列
72     """
73     ##你的代码在这里##
74     cnt = 2
75     for i in json_str[json_index + 1:]:
76         if i != '"':
77             cnt = cnt + 1
78         else:
79             break
80     res = json_str[json_index + 1:json_index + cnt - 1]
81     return (STRING_TOKEN,res), json_index + cnt
82     #2200350204李禹佳
83
84 def read_null():
85     """
86     处理null
87     :return: 返回处理null后的token序列
88     """
89     res = json_str[json_index: json_index + 4]
90     return (NULL_TOKEN, res), json_index + 4
91
92 def read_bool(s: str):
```

通关:



7. 类的设计

7-1 重载小于等于运算符

代码截图:

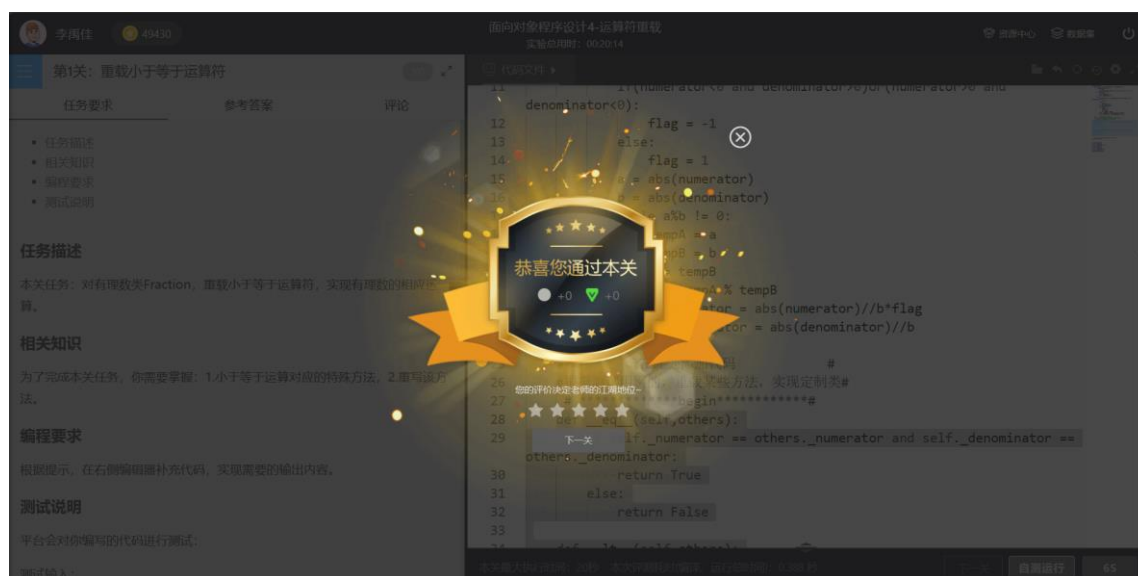
```
20     a = tempB
21     b = tempA % tempB
22     self._numerator = abs(numerator)//b*flag
23     self._denominator = abs(denominator)//b
24
25     # 请在此处添加代码
26     # 根据测试用例的, 重载某些方法, 实现定制类
27     # *****begin*****
28     def __eq__(self,others):
29         if self._numerator == others._numerator and self._denominator == others._denominator:
30             return True
31         else:
32             return False
33
34     def __lt__(self,others):
35         if self._numerator / self._denominator < others._numerator / others._denominator:
36             return True
37         else:
38             return False
39
40     # *****end*****
41     #2200350204李禹佳
42
43
44 a = int(input())
45 b = int(input())
46 frac1 = Fraction(a,b)
47 c = int(input())
48 d = int(input())
49 frac2 = Fraction(c,d)
```

重载 `__eq__` 和 `__lt__` 方法, `__eq__` 方法判断两个分数是否相等。如果两个分数的分子和分母都相等, 则认为它们相等。方法中将 `self._numerator` 和 `self._denominator` 与另一个分数对象的对应属性进行比较。

`__lt__` 方法用于实现小于号 `<` 的比较操作。比较两个分数的大小实际上就是比较它们的大小关系，即比较两个分数的小数值的大小。实现中首先将两个分数转换为小数值进行比较，如果左边的分数小于右边的分数，则返回 `True`，否则返回 `False`。

```
18         tempA = a
19         tempB = b
20         a = tempB
21         b = tempA % tempB
22         self._numerator = abs(numerator)//b*flag
23         self._denominator = abs(denominator)//b
24
25     # 请在此处添加代码
26     # 根据测试用例，重载某些方法，实现定制类#
27     # *****begin*****#
28     def __eq__(self,others):
29         if self._numerator == others._numerator and self._denominator == others._denominator:
30             return True
31         else:
32             return False
33
34     def __lt__(self,others):
35         if self._numerator / self._denominator < others._numerator / others._denominator:
36             return True
37         else:
38             return False
39
40     # *****end*****#
41 #2200350204李禹佳
42
43 a = int(input())
44 b = int(input())
45 frac1 = Fraction(a,b)
46 c = int(input())
47 d = int(input())
48 frac2 = Fraction(c,d)
49 print(frac1<frac2)
50 print(frac1>frac2)
```

通关：



7-2 重载加减运算符

代码截图：

```
14         a = abs(numerator)
15         b = abs(denominator)
16         while a%b != 0:
17             tempA = a
18             tempB = b
19             a = tempB
20             b = tempA % tempB
21         self._numerator = abs(numerator)//b*flag
22         self._denominator = abs(denominator)//b
23
24     # 请在此处添加代码
25     # 根据测试用例，重载某些方法，实现定制类#
26     # *****begin*****#
27     def __add__(self,oth):
28         tmp1 = self._numerator * oth._denominator + self._denominator * oth._numerator
29         tmp2 = self._denominator * oth._denominator
30         return Fraction(tmp1,tmp2)
31
32     def __sub__(self,oth):
33         tmp1=self._numerator * oth._denominator - self._denominator * oth._numerator
34         tmp2=self._denominator * oth._denominator
35         return Fraction(tmp1,tmp2)
36
37     def __repr__(self):
38         return str(self._numerator) + '/' + str(self._denominator)
39
40     # *****end*****#
41 #2200350204李禹佳
42
43 a = int(input())
44 b = int(input())
45 frac1 = Fraction(a,b)
46 c = int(input())
47 d = int(input())
48 frac2 = Fraction(c,d)
49 print(frac1+frac2)
50 print(frac1-frac2)
```

重载 `__add__`、`__sub__` 和 `__repr__` 方法：`__add__` 方法将两个分数的分子分别乘以对方的分母再相加得到分子。分母则是两个分数的分母相乘。最后，将得到的分子和分母作为参数创建一个新的分数对象，并返回；`__sub__` 方法和加法类似，将两个分数的分子分别乘以对方的分母再相减得到分子。分母则是两个分数的分母相乘。最后，将得到的分子和分母作为参数创建一个新的分数对象，并返回；`__repr__` 方法用于定义分数类的字符串表示形式。将分数的分子和分母以 `/` 分隔并以字符串的形式返回。

```
16         tempA = a
17         tempB = b
18         a = tempB
19         b = tempA % tempB
20         self._numerator = abs(numerator)//b*flag
21         self._denominator = abs(denominator)//b
22
23         # 请在此处添加代码
24         # 根据测试用例的，重载某些方法，实现定制类
25         # *****begin*****
26         def __add__(self,oth):
27             tmp1 = self._numerator * oth._denominator + self._denominator * oth._numerator
28             tmp2 = self._denominator * oth._denominator
29             return Fraction(tmp1,tmp2)
30
31         def __sub__(self,oth):
32             tmp1=self._numerator * oth._denominator - self._denominator * oth._numerator
33             tmp2=self._denominator * oth._denominator
34             return Fraction(tmp1,tmp2)
35
36         def __repr__(self):
37             return str(self._numerator) + '/' + str(self._denominator)
38         # *****end*****
39
40
41 a = int(input())
42 b = int(input())
43 frac1 = Fraction(a,b)
44 c = int(input())
45 d = int(input())
46 frac2 = Fraction(c,d)
47 resultSUB = frac1-fac2
```

通关：

第2关：重载加减运算符

任务要求

参考答案

评论

任务描述

相关知识

编程要求

测试说明

恭喜您通过本关

您的代码已通过老师的正确性检测。

完成

```
16         tempA = a
17         tempB = b
18         a = tempB
19         b = tempA % tempB
20         self._numerator = abs(numerator)//b*flag
21         self._denominator = abs(denominator)//b
22
23         # 请在此处添加代码
24         # 根据测试用例的，重载某些方法，实现定制类
25         # *****begin*****
26         def __add__(self,oth):
27             tmp1 = self._numerator * oth._denominator + self._denominator * oth._numerator
28             tmp2 = self._denominator * oth._denominator
29             return Fraction(tmp1,tmp2)
30
31         def __sub__(self,oth):
32             tmp1=self._numerator * oth._denominator - self._denominator * oth._numerator
33             tmp2=self._denominator * oth._denominator
34             return Fraction(tmp1,tmp2)
35
36         def __repr__(self):
37             return str(self._numerator) + '/' + str(self._denominator)
38         # *****end*****
39
40
41 a = int(input())
42 b = int(input())
43 frac1 = Fraction(a,b)
44 c = int(input())
45 d = int(input())
46 frac2 = Fraction(c,d)
47 resultSUB = frac1-fac2
```

8. 类的定制
代码截图：

```
109         如果没有找到则返回-1，否则返回对应的value
110         ---
111         #你的代码在这里#
112         if key not in self.__keyMap:
113             return -1
114         node = self.__keyMap[key]
115         self.__increment(node)
116         return node.value
117
118     #2200350204李真佳
119     def put(self, key, value):
120         插入指定的key&value，如果key存在则更新value，同时更新频率
121         如果key不存在并且缓存满了，则删除频率最低的元素，并插入新元素
122         否则，直接插入新元素
123         Args:
124             key:要插入的关键字
125             value:要插入的值
126         ---
127         #你的代码在这里#
128         if key in self.__keyMap:
129             node = self.__keyMap[key]
130             node.value = value
131             self.__increment(node)
132         else:
133             if self.__capacity==0:
134                 return
135             if len(self.__keyMap)==self.__capacity:
136                 self._removeMinFreqElement()
137             node = Node(key,value,1)
138             self.__increment(node,True)
139             self.__keyMap[key] = node
140
141     def _increment(self, node, is_new_node=False):
142         ---
```

Get 中获取到了 key 对应的元素值，如果 key 不存在返回-1，如果存在则返回对应的 value 值

```
94         self.__freqMap[freq->linkedlist 这种结构的字典
95         self.__minFreq:记录缓存中最低频率
96         """
97         self.__capacity = capacity
98         self.__keyMap = dict()
99         self.__freqMap = dict()
100         self.__minFreq = 0
101
102     def get(self, key):
103         """
104         获取一个元素, 如果key不存在则返回-1, 否则返回对应的value
105         同时更新被访问元素的频率
106         Args:
107             key:要查找的關鍵字
108         Returns:
109             如果没找到则返回-1, 否则返回对应的value
110         """
111         #你的代码在这里#
112         if key not in self.__keyMap:
113             return -1
114         node = self.__keyMap[key]
115         self.__increment(node)
116         return node.value
117
118     #2200350204李尚恒
119     def put(self, key, value):
120         """
121         插入指定的key和value, 如果key存在则更新value, 同时更新频率
122         如果key不存在并且缓存满了, 则删除频率最低的元素, 并插入新元素
123         否则, 直接插入新元素
124         Args:
125             key:要插入的關鍵字
126             value:要插入的值
127         """
128         #把key放进linkedlist里
```

Put 中插入指定的 key 和 value,如果 key 已经存在, 则更新 value,如果 key 不存在, 则删除频率最低的元并插入新元素

The screenshot shows a code editor with a dark theme. The top bar includes a user profile (李瑞佳), a file explorer (49430), and a title bar (面向对象程序设计3-工资计算程序和LRU缓存类). The main code area displays Python code for a class. A green box highlights the `_increment` method. The code includes comments in Chinese explaining the logic of the frequency-based node removal and linked list management.

```
136         self.__removeMinFrequencyElement()
137         node = Node(key,value,1)
138         self.__increment(node,True)
139         self.__keyMap[key] = node
140
141     def __increment(self, node, is_new_node=False):
142         """
143         更新节点的访问频率
144         Args:
145             node:要更新的节点
146             is_new_node:是否是新节点。新插入的节点和非新插入节点更新逻辑不同
147         """
148         if is_new_node:
149             self.__minFreq = 1
150             self.__setDefaultLinkedList(node)
151         else:
152             self.__deleteNode(node)
153             node.freq += 1
154             self.__setDefaultLinkedList(node)
155             if self.__minFreq not in self.__freqMap:
156                 self.__minFreq += 1
157
158     def __setDefaultLinkedList(self, node):
159         """
160         根据节点的频率，插入到对应的LinkedList中，如果LinkedList不存在则创建
161         Args:
162             node:将要插入到LinkedList的节点
163         """
164         if node.freq not in self.__freqMap:
165             self.__freqMap[node.freq] = LinkedList()
166         linkedList = self.__freqMap[node.freq]
167         linkedList.InsertFirst(node)
168
```

通关:

[illegible]

六、问题记录和实验总结（必写）

1. 对实现的方法，函数还不是很熟悉，需查找资料
2. 对代码的理解能力欠缺，长代码需耗时读完

ReadMe:

本实验报告中课内实验 2-1 的 1, 2, 3, 4 对应实验内容的前四个，课内实验 2-3 进阶部分 1 中两关对应着实验内容的 7: 类的设计(将两关分为 7-1 和 7-2)，课内实验 2-3 进阶部分 2 中两关对应着实验内容的 6: Json 解析器和 8: 类的定制