

## 测试题型-阅读程序写输出

- 运行以下程序，输出结果是【 】。

```
lst = [2, 4, 1, 5, 1, 1, 5, 1, 4, 5, 2, 3]
```

```
n = max(lst)
```

```
blst = [lst.count(i) for i in range(n+1)]
```

```
print(lst.index(max(blst)))
```

===== (答案1) =====

1

# 测试题型-阅读程序写输出

```
data = '''Beijing,2185.8,43760.7
        Shanghai,2487.45,47218.66
        Guangzhou,1882.70,30355.73
        Shenzhen,1779.01,34606.40'''
cities = {}
for line in data.split():
    x = line.split(',')
    cities[x[0]] = [float(x[1]),float(x[2])]
print(len(cities))
sorted_ct = sorted(cities.items(), key=lambda item: item[1][1], reverse=True)
for x in sorted_ct:
    print(x[0], end=' ')
```

=====(答案)=====

4

Shanghai Beijing Shenzhen Guangzhou

# 测试题型-根据输入输出补充程序

- **【程序功能】**
- 编写程序，从键盘输入一个由数字和字母组成的字符串，数字表示紧跟着的字母串
- 连续出现的次数，请将字符串解码恢复原始的字符串输出到屏幕并保存到文件out.txt中。
- 
- **【测试数据及运行结果】**
- **【输入】**
- 3a2bc10d
- **【输出（文件out.txt中的内容）】**
- aaabcbcdcccccccccc

# 测试题型-根据输入输出补充程序

- 【待完善程序】

- '''
- s = input()
- s += '0'
- newstr = ''
- chars = ''
- nums = ''
- for ch in s:
- if ch.isdigit():
- if chars != '':
- #\*\*\*\*\*SPACE\*\*\*\*\*

#\*\*\*\*\*SPACE\*\*\*\*\*

newstr += 【?】

nums = ''

chars = ''

nums += ch

else:

chars += ch

print(newstr)

with open('out.txt', 'w') as fp:

#\*\*\*\*\*SPACE\*\*\*\*\*

【?】 (newstr)

===== (答案1) =====  
chars\*int(nums)

===== 或 =====  
int(nums)\*chars

===== (答案2) =====  
fp.write

# 测试题型-程序改错

- **【注意】**
- 不要新建文件，就在系统给出的程序中修改；
- 不要修改代码的其他部分；
- 不能删去整条语句；
- 不能增加其他语句；
- 可以调整语句的顺序。
- **【程序功能】**
- 输入一个整数列表，查找其中最长的山脉列表。
- 山脉列表lst的要求为：元素个数不小于3，并且存在索引 i，满足：  
 $lst[0] < lst[1] < \dots < lst[i-1] < lst[i]$  并且  $lst[i] > lst[i+1] > \dots > lst[-1]$
- 例如列表[2,-1,-1,8,4,-3,5,16,23,0]中存在两条山脉列表，[-1,8,4,-3]与[-3,5,16,23,0]，其中[-3,5,16,23,0]元素多，为最长的山脉列表。

# 测试题型-程序改错

- 【测试数据与运行结果】
- 运行过程1:
- 【输入】
- [2,3,3,8,4,4,-3,5,16,23,0]
- 【输出】
- [-3, 5, 16, 23, 0]
- 运行过程2:
- 【输入】
- [2,-1,-1,8,8,-3,5,16,23]
- 【输出】
- []

# 测试题型-程序改错

- 【含有错误的程序】
- ```
def longest_mountain(lst):  
    longest_lst = []  
    start = 0  
    i = 0  
    while i < len(lst)-1:  
        if lst[i] < lst[i+1]:  
            start = i  
            while i < len(lst)-1 and lst[i] < lst[i+1]:  
                i += 1  
            if i == len(lst)-1:  
                break
```
- ```
#*****FOUND*****  
    if lst[i] == lst[i+1]:  
        while i < len(lst)-1 and lst[i] > lst[i+1]:  
            i += 1  
#*****FOUND*****  
    templst = lst[start:i]  
#*****FOUND*****  
    if longest_lst < templst:  
        longest_lst = templst  
    else:  
        i += 1  
    return longest_lst
```

# 测试题型-程序改错

- `if __name__ == '__main__':`
- `#*****FOUND*****`
- `lst = input()`
- `print(longest_mountain(lst))`

=====(答案1)=====

```
lst[i] != lst[i+1]
```

=====~~或~~=====

```
lst[i+1] != lst[i]
```

=====(答案2)=====

```
lst[start:i+1]
```

=====(答案3)=====

```
len(longest_lst)<len(templst)
```

=====(答案4)=====

```
eval
```



# 测试题型-编程题

- **【程序功能】**
- 查找500以内满足如下要求的特殊数，并按运行结果的格式输出。
- 特殊数n的要求为：n为三位正整数，并且能找到一个三位正整数k，满足k加上k的反序数（反序数也是三位数，并且k不是回文数）之和等于n。
- 例如 $162+261=423$ ，423即为满足要求的特殊数，但202不满足要求，虽然 $101+101=202$ ，
- 但101为回文数。

# 测试题型-编程题

- 【运行结果】
- $102+201=303$
- $112+211=323$
- $122+221=343$
- $132+231=363$
- $142+241=383$
- $152+251=403$
- $103+301=404$
- $162+261=423$
- $113+311=424$
- $172+271=443$
- $123+321=444$
- $182+281=463$
- $133+331=464$
- $192+291=483$
- $143+341=484$

## 测试题型-编程题

- 答案： for n in range(200, 500): #遍历1分
- for i in range(100, n//2):   #遍历2分
- if i%10 != 0 and 2\*i != n and i+int(str(i)[::-1]) == n: #条件4分
- print(f'{i}+{n-i}={n}') #输出格式2分
- break   #每个数只输出一遍1分