# Assign #3: Oct Mock Exam暨选做题目满百

Updated 1537 GMT+8 Oct 14, 2024

2024 fall, Complied by 胡杨 元培学院

**说明：**

1）Oct月考： AC4 。考试题目都在"题库（包括计概、数算题目）"里面，按照数字题号能找到，可以重新提交。作业中提交自己最满意版本的代码和截图。

2）请把每个题目解题思路（可选），源码Python, 或者C++/C（已经在Codeforces/Openjudge上AC），截图（包含Accepted, 学号），填写到下面作业模版中（推荐使用 typora https://typoraio.cn ，或者用word）。AC 或者没有AC，都请标上每个题目大致花费时间。

3）提交时候先提交pdf文件，再把md或者doc文件上传到右侧"作业评论"。Canvas需要有同学清晰头像、提交文件有pdf、作业评论有md或者doc。

4）如果不能在截止前提交作业，请写明原因。

## 1. 题目

### E28674:《黑神话：悟空》之加密

http://cs101.openjudge.cn/practice/28674/

思路：字母不太好操作，但是转化成ASCII码就可以直接求余数来操作了，于是我想到了用chr()和ord()，只需要把偏移值k求26的余数，再用原字符的ASCII值减去它。输入中同时包含大小写字母，所以可以分开讨论A-Z和a-z的情况。题目要求把a当作z的下一个字母，连成一个圈，所以如果s[i]-k%26<65or97（两种情况），就再加26就可以了

#### 题解启示

题解的思路和语法都和我差不多，但是代码看起来却更简洁。可以学习这种**初始化列表再添加元素**的方式，**不要一边遍历一边修改。**

（感觉题解的代码好优美随复制粘贴）

```
def decrypt_caesar_cipher(k, s):
    decrypted_text = []

    for char in s:
        if 'a' <= char <= 'z':
            decrypted_char = chr((ord(char) - ord('a') - k) % 26 + ord('a'))
        elif 'A' <= char <= 'Z':
            decrypted_char = chr((ord(char) - ord('A') - k) % 26 + ord('A'))
        else:
            decrypted_char = char  # Non-alphabetic characters remain unchanged
        decrypted_text.append(decrypted_char)

    return ''.join(decrypted_text)
```

代码

```
#读取输入
k=int(input()) #偏移量
s=list(input())

#定义函数
def puoyi(s,k):
    n=len(s)
    for i in range(n):
        #大写字母的破译
        if 'A' <=s[i]<='Z':
            m=ord(s[i])-k%26 #修正后的ASCII值
            if m<65: #如果修正后的ASCII值小于65，则需要加上26
                m+=26
            s[i]=chr(m)

        #小写字母的破译
        elif 'a' <=s[i]<='z':
            m=ord(s[i])-k%26
            if m<97:
                m+=26
            s[i]=chr(m)
    return ''.join(s)

#调用函数
print(puoyi(s,k))
```

代码运行截图 （至少包含有"Accepted"）

状态: Accepted

源代码

```
#读取输入
k=int(input()) #偏移量
s=list(input())

#定义函数
def puoyi(s,k):
    n=len(s)
```

基本信息

#: 46479889
题目: 28674
提交人: 24n2300017728
内存: 3640kB
时间: 21ms
语言: Python3
提交时间: 2024-10-14 15:44:19

# E28691: 字符串中的整数求和

http://cs101.openjudge.cn/practice/28691/

思路：把第三个字母去掉，前面两个转化成int求和就好了

## 题解启示

读取**标准输入**：

```
import sys
input_data = sys.stdin.read()
```

`sys.stdin.read()` 函数会读取**所有输入**直到 EOF（文件结束符）。程序会在按下组合键 `Ctrl+D` (在 Unix 系统中) 或 `Ctrl+Z` (在 Windows 命令行中，之后按 Enter) 来表示输入结束时才会继续执行。

但是其实更常用的是`input()`函数，它每次读取**一行输入**

代码

```
a,b=input().split()

#把ab中最后一个字母删掉，把剩下的数字转化为int型
a=int(a[:-1])
b=int(b[:-1])

print(a+b)
```

代码运行截图 （至少包含有"Accepted"）

**#46480549提交状态**

状态: Accepted

源代码

```
a,b=input().split()

#把ab中最后一个字母删掉，把剩下的数字转化为int型
a=int(a[:-1])
b=int(b[:-1])

print(a+b)
```

## M28664: 验证身份证号

http://cs101.openjudge.cn/practice/28664/

思路：只需要检验第18位是否一样，按照题干一步一步操作即可，就是有点费手

### 题解启示

题解中增加了**检查数据长度**是否等于18这一步，我认为这或许是必要的，因为题干说了保证前17位合法，但是没说保证位数合法。只是测试数据里没有这种情况所以可以AC！

```
# 检查输入字符串的长度
    if len(s) != 18:
        print('NO')
        continue
```

代码

```
n = int(input())
l=[7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2] #定义权重数组

for _ in range(n):
    s=input()
    x=sum(int(s[i])*l[i] for i in range(17)) #求和
    a=x%11 #求余数
    a_new=str((12-a)%11) #更新余数

    #处理10的特殊情况
    if a_new=='10':
        a_new='X'

    #比较是否一致
```

```
    if a_new==s[17]:
        print('YES')
    else:
        print('NO')
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
n = int(input())
l=[7,9,10,5,8,4,2,1,6,3,7,9,10,5,8,4,2] #定义权重数组

for _ in range(n):
    s=input()
    x=sum(int(s[i])*l[i] for i in range(17)) #求和
    a=x%11 #求余数
    a_new=str((12-a)%11) #更新余数

    #处理10的特殊情况
    if a_new=='10':
        a_new='X'

    #比较是否一致
    if a_new==s[17]:
        print('YES')
    else:
        print('NO')
```

基本信息

| | |
|---|---|
| #: | 46481620 |
| 题目: | 28664 |
| 提交人: | 24n2300017728 |
| 内存: | 3620kB |
| 时间: | 20ms |
| 语言: | Python3 |
| 提交时间: | 2024-10-14 16:49:10 |

# M28678: 角谷猜想

http://cs101.openjudge.cn/practice/28678/

思路：可以使用while循环，while x>1:

然后根据奇偶性分别写一个if，按题干要求操作，并把该步骤输出成文字。注意转成int()型！！

最后得到1了，循环自动结束，并且print('end')

## 题解启示：格式化字符串

**1. 使用 `str.format()`**

这是一种更现代的方法，提供了更多的灵活性和功能。

```
name = "Alice"
age = 30
message = "Hello, {}. You are {} years old.".format(name, age)
print(message)
```

也可以通过指定位置或关键字来提高可读性：

```
message = "Hello, {1}. You are {0} years old.".format(age, name)
print(message)

message = "Hello, {name}. You are {age} years old.".format(name=name, age=age)
print(message)
```

**2. 使用 f-string （更简单好用！）**

f-string 是一种非常直观和高效的字符串格式化方法，直接在字符串前加上 `f` 或 `F`，并在字符串中使用 `{}` 来包含表达式。

```
name = "Alice"
age = 30
message = f"Hello, {name}. You are {age} years old."
print(message)
```

f-string 还允许在 `{}` 内部执行表达式：

```
a = 5
b = 10
message = f"The sum of {a} and {b} is {a + b}."
print(message)
```

代码

```
n=int(input())

while n>1:
     if n%2==0:
         print(str(n)+'/2='+str(int(n/2)))
         n=int(n/2)
     elif n%2==1:
         print(str(n)+'*3+1='+str(int(n*3+1)))
         n=int(n*3+1)

print('End')
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

---

状态: Accepted

源代码

```
n=int(input())

while n>1:
     if n%2==0:
         print(str(n)+'/2='+str(int(n/2)))
         n=int(n/2)
     elif n%2==1:
         print(str(n)+'*3+1='+str(int(n*3+1)))
         n=int(n*3+1)

print('End')
```

基本信息

| | |
|---|---|
| #: | 46482286 |
| 题目: | 28678 |
| 提交人: | 24n2300017728 |
| 内存: | 3628kB |
| 时间: | 22ms |
| 语言: | Python3 |
| 提交时间: | 2024-10-14 17:04:40 |

# M28700: 罗马数字与整数的转换

http://cs101.openjudge.cn/practice/28700/

思路：先定义两个函数：罗马数字转整数的函数，整数转罗马数字的函数

以整数转罗马数字为例，先编写一个罗马数字字符的字典。罗马数字本质上是个加法，从最大的开始，能加几个1000就写几个M，900写一个CM，然后能有几个500就写几个D，以此类推。然后把我们input的整数从减1000开始，直到为0

罗马数字转整数类似，但是需要注意罗马数字中类似'CD'这种特殊的情况，可以把特殊的和普通的单字符写成两个字典，先遍历特殊的，把特殊的双字符删掉过后，再遍历普通的单字符

题解中采用的处理特殊情况的方法是先遍历，如果后面一个数比前面一个数大，则加上后面一个减去两倍的前面一个。但感觉有点小绕

提示告诉我们可以根据第一个字是不是1-9来判断input是整数还是罗马数字，但我还是不会

我会写的代码是

```
if n[1]=='1' or '2' or '3' or '4' or '5' or '6' or '7' or '8' or '9':
```

但这段代码并不是正确的方式来检查输入是否为数字。这实际上会总是返回 `True`，因为 `'2'`, `'3'`, ..., `'9'` 都是字符串，Python 会将非空字符串视为 `True`。

## 启示

（此题是边查ai边写的）

**1. `str.replace()` 方法**

在 Python 中，`str.replace()` 方法用于**返回一个新的字符串，其中原字符串中的某个子字符串被另一个子字符串替换**。这个方法的基本语法如下：

```
str.replace(old, new[, count])
```

- `old`：要被替换的子字符串。
- `new`：新的子字符串，用来替换 `old`。
- `count`（可选）：指定替换的最大次数。如果不提供此参数，则默认替换所有匹配的子字符串。

下面是一些使用 `str.replace()` 的示例：

（1）替换所有匹配的子字符串

```
text = "hello world, hello Python, hello everyone"
new_text = text.replace("hello", "hi")
print(new_text)
```

输出：

```
hi world, hi Python, hi everyone
```

（2）只替换前几次匹配的子字符串

```
text = "hello world, hello Python, hello everyone"
new_text = text.replace("hello", "hi", 2)
print(new_text)
```

输出：

```
hi world, hi Python, hello everyone
```

（3）替换空格

```
text = "hello world"
new_text = text.replace(" ", "_")
print(new_text)
```

输出：

```
hello_world
```

（4）替换多个字符

```
text = "hello123world"
new_text = text.replace("123", "")
print(new_text)
```

输出：

```
helloworld
```

**（5）多步替换**

有时你可能需要进行多次替换，可以链式调用 `replace` 方法：

```
text = "hello world, hello Python, hello everyone"
new_text = text.replace("hello", "hi").replace("world", "Earth")
print(new_text)
```

输出：

```
hi Earth, hi Python, hi everyone
```

**注意事项**

- `str.replace()` 方法不会修改原始字符串，而是返回一个新的字符串。
- 如果 `old` 子字符串在原字符串中不存在，`str.replace()` 方法会直接返回原字符串。

### 2. `str.isdigit()` 方法

`isdigit()` 是 Python 字符串的一个方法，用于**检查字符串中的所有字符是否都是数字**。如果字符串中的所有字符都是数字，并且字符串非空，则 `isdigit()` 返回 `True`；否则返回 `False`。

示例

```
n = "12345"
if n.isdigit():
    print("n 是一个纯数字字符串")
else:
    print("n 不是纯数字字符串")

n = "123abc"
if n.isdigit():
    print("n 是一个纯数字字符串")
else:
    print("n 不是纯数字字符串")

n = "123.45"
if n.isdigit():
    print("n 是一个纯数字字符串")
else:
    print("n 不是纯数字字符串")

n = ""
if n.isdigit():
    print("n 是一个纯数字字符串")
else:
    print("n 不是纯数字字符串")
```

输出

```
n 是一个纯数字字符串
n 不是纯数字字符串
n 不是纯数字字符串
n 不是纯数字字符串
```

#### 注意事项

1. **空字符串**：空字符串 `""` 被认为不是纯数字字符串，因此 `isdigit()` 会返回 `False`。
2. **浮点数**：包含小数点的字符串（如 `"123.45"`）不是纯数字字符串，因此 `isdigit()` 会返回 `False`。
3. **负号**：包含负号的字符串（如 `"-123"`）也不是纯数字字符串，因此 `isdigit()` 会返回 `False`。
4. **其他字符**：任何非数字字符（包括空格、字母等）都会导致 `isdigit()` 返回 `False`。

#### 其他相关方法

- `isnumeric()`：检查字符串中的所有字符是否都是数字字符，包括 Unicode 数字。
- `isdecimal()`：检查字符串中的所有字符是否都是十进制数字。

示例：使用 `isnumeric()` 和 `isdecimal()`

```python
n = "12345"
print(n.isnumeric())  # True
print(n.isdecimal())  # True

n = "Ⅷ"  # 罗马数字
print(n.isnumeric())  # True
print(n.isdigit())    # False
print(n.isdecimal())  # False

n = "123.45"
print(n.isnumeric())  # False
print(n.isdigit())    # False
print(n.isdecimal())  # False
```

### 3. `str.count()` 方法

`str.count()` 是 Python 中字符串对象的一个方法，用于**计算某个子字符串在原字符串中出现的次数。**这个方法**会区分大小写，且不会重叠计数**（即一个子串的一部分不能作为另一个子串的一部分来计数）。

**基本语法**

```python
string.count(sub, start=..., end=...)
```

- `sub`：必需，要搜索的子字符串。
- `start`：可选，开始搜索的位置，默认为字符串的开始位置。
- `end`：可选，结束搜索的位置，默认为字符串的结束位置。

**示例**

假设我们有一个字符串 `s = "hello world, hello universe"`，我们可以使用 `count()` 方法来查找特定子字符串出现的次数。

示例 1: 没有指定范围

```python
s = "hello world, hello universe"
print(s.count("hello"))  # 输出: 2
```

这里 `s.count("hello")` 计算了整个字符串中 "hello" 出现的次数，结果是 2。

示例 2: 指定范围

如果你想只在字符串的某一部分查找，可以指定 `start` 和 `end` 参数。

```python
s = "hello world, hello universe"
print(s.count("hello", 0, 12))  # 输出: 1
```

这里 `s.count("hello", 0, 12)` 只会计算从索引 0 到 11（包括索引 0，但不包括索引 12）之间的 "hello" 的数量，因此输出为 1。

**注意事项**

- 如果 `start` 或 `end` **超过了字符串的实际长度**，Python **不会抛出错误**，而是将其视为字符串的最大或最小可能值。
- `count()` 方法**对大小写敏感**，所以 "Hello" 和 "hello" 会被认为是两个不同的子字符串。

代码

```
#定义整数转罗马数字的函数
def int_to_roman(n):
    #定义罗马数字的符号和值
    roman_dict = {1000: 'M', 900: 'CM', 500: 'D', 400: 'CD', 100: 'C', 90: 'XC',
50: 'L', 40: 'XL', 10: 'X', 9: 'IX', 5: 'V', 4: 'IV', 1: 'I'}
    #定义结果字符串
    result = ''

    #遍历罗马数字的符号和值
    for value,symbol in roman_dict.items():
        #从1000开始，计算当前值能整除的最大次数
        count = n//value
        #将当前符号重复count次添加到结果字符串中
        result += symbol*count

        #更新待转换的数字
        n -= value*count

    #返回
    return result

#定义罗马数字转整数的函数
def roman_to_int(n):
    #定义罗马数字的符号和值
    roman_dict_special = {'CM': 900, 'CD': 400, 'XC': 90, 'XL': 40, 'IX': 9,
'IV': 4}
    roman_dict_usual = {'M': 1000, 'D': 500, 'C': 100, 'L': 50, 'X': 10, 'V': 5,
'I': 1}
    #定义结果整数
    result = 0

    #遍历罗马数字的符号和值,先特殊后普通
    for symbol,value in roman_dict_special.items():
        #如果当前符号在待转换的数字中出现，则将其转换为整数并加到结果中
        if symbol in n:
            count=n.count(symbol)
            result += value*count
            n=n.replace(symbol,'') #将当前符号从待转换的罗马数字中移除

    for symbol,value in roman_dict_usual.items():
        if symbol in n:
            count=n.count(symbol)
            result+=value*count
            n=n.replace(symbol,'')

    #返回结果
```

```
    return result

#读取输入
n = input()

#测试输入是否为整数
if n.isdigit():
    n=int(n)
    print(int_to_roman(n))
else:
    print(roman_to_int(n))
```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

状态: Accepted

源代码

```
#定义整数转罗马数字的函数
def int_to_roman(n):
    #定义罗马数字的符号和值
    roman_dict = {1000: 'M', 900: 'CM', 500: 'D', 400: 'CD', 100: 'C', 90:
    #定义结果字符串
    result = ''
```

基本信息

#: 46501767
题目: 28700
提交人: 24n2300017728
内存: 3708kB
时间: 22ms
语言: Python3
提交时间: 2024-10-15 17:29:52

(终于AC了命都快做没了X(.....)

## *T25353: 排队 （选做）

http://cs101.openjudge.cn/practice/25353/

思路:

代码

```

```

代码运行截图 （AC代码截图，至少包含有"Accepted"）

# 2. 学习总结和收获

如果作业题目简单，有否额外练习题目，比如：OJ“计概2024fall每日选做”、CF、LeetCode、洛谷等网站题目。

只有AC4，看到群里AC5AC6还嫌不够的大佬，感觉前途一片灰暗

月考的一些心得：

1.一定要提前到教室调试好设备。月考时我来的有点晚，pc里有前人留下的乱七八糟代码，他调了运行让他一直跑的都是他的程序而不是当下的程序……我没发现，然后在那里对着我没有问题但就是“跑不动”的代码心态炸裂加迷惑了半个小时……一定要**检查是不是current file**！

2.**携带一张草稿纸和一支笔**，有助于理清思路

3.不要因为紧张忘记一些很基础的操作。比如数据可以复制粘贴不用对着一个一个打。

4.可以根据AC率尝试人数等大体判断哪道题最简单。先易后难，太难放弃

罗马数字转换已经写得快死了，排队果断放弃。发现自己对于字符串、列表的操作仍然不熟，对于字典、元组、集合更是知之甚少。并且**过于依赖列表**，很少想到用字典等其他形式会不会更简单。发现这一周都没怎么学计概，作业也是赶ddl，回顾了一下发现原来这周我有点时间都在医院（喜），事已至此，先活着吧，至于学习，我再想想办法X(