第三次小测题解

一、练习网站

第三次小测题目已添加到之前小测发布的题单，链接如下：

2025 小测练习 - 题单 - 洛谷 | 计算机科学教育新生态

（提示：按住 ctrl 点击链接即可直接跳转到网站）

二、题目一：加密通话

1. 思路：

本题目的要求为将句子中的每个单词根据规则进行修改，所以问题的解决思路按照以下流程进行即可：

（1）将句子按空格分割成单词列表

（2）遍历每个单词，记录单词位次

（3）对每个单词判断首字母是否为元音（大小写都要注意）

（4）根据规则转换单词

（5）加上对应数量的'x'

（6）用空格连接所有加密后的单词

注意题目要求，为了和判题程序进行交互，需要添加以下代码：

```
while True:
    print(eval(input()))
```

以下给出两个实现代码，解决思路没有区别，仅实现方式上有一些不同。

代码 1：

```
def Encryption(sentence):
    # 用字符串创建一个集合（set），包含所有元音字母的大小写形式
    vowels = set('aeiouAEIOU')
    # 也可直接创建集合
    # vowels = {'a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'}
    # 将句子按空格分割成单词列表
    words = sentence.split()
    # 创建用于存储处理后的单词的空列表
    result = []

    # enumerate() 返回 (索引, 元素) 的元组序列
    # for i, word in enumerate(words, 1)指定从 1 开始计数
    for i, word in enumerate(words, 1):
        # 判断首字母是否为元音，即 word[0]是否在集合 vowels 中
        # 并按题目要求处理单词
        if word[0] in vowels:
            encrypted_word = word + "mi"
        else:
            encrypted_word = word[1:] + word[0] + "mi"

        # 根据索引添加对应数量的 'x'
```

```
        encrypted_word += 'x' * i
        # 将处理后的单词添加进结果列表
        result.append(encrypted_word)

    # 将列表 result 中的所有字符串元素用一个空格连接起来
    # 形成一个新的字符串，并作为返回值
    return ' '.join(result)

# 按题目要求添加与判题程序交互的代码
while True:
    print(eval(input()))
```

代码 2：

```
def Encryption(sentence):
    # 创建一个列表，包含所有元音字母的大小写形式
    vowels = ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']
    # 将句子按空格分割成单词列表
    words = sentence.split()
    # 创建用于存储处理后的单词的空字符串
    result = ''

    # 用变量 i 作为单词位次的索引
    i=1
    for word in words:
        # 判断首字母是否为元音，即 word[0]是否在列表 vowels 中
        # 并按题目要求处理单词
        if word[0] in vowels:
            encrypted_word = word + "mi"
        else:
            encrypted_word = word[1:] + word[0] + "mi"

        # 根据索引添加对应数量的 'x'，并将 i 的值+1
        encrypted_word += 'x' * i
        i+=1
        # 将处理后的单词添加进结果字符串，并添加' '
        result = result + encrypted_word + ' '

    # 将结果字符串的最后一位' '去除，并作为返回值
    result = result[:-1]
    return result

# 按题目要求添加与判题程序交互的代码
while True:
    print(eval(input()))
```

对比：

元音字母 vowels：代码 1 使用集合，代码 2 使用列表，在执行 if word[0] in vowels:语句进行元音查找时，集合查找时间复杂度为 O(1)，列表查找时间复杂度为 O(n)。

单词处理循环：代码 1 使用 enumerate(words, 1) 同时获取索引和单词，代码 2 需手动维护索引变量 i。

字符串拼接方式：代码 1 使用使用 result.append(encrypted_word) 收集单词，然后使用高效的 ' '.join(result) 一次性连接，时间复杂度为 O(n)，代码 2 使用 result = result + encrypted_word + ' ' 逐步拼接，最后用 result = result[:-1] 去掉末尾空格，因为每次拼接都创建新字符串，时间复杂度为 $O(n^2)$。

本次测试因没有严格要求时间复杂度的优化，所以两种写法皆可得到 100 分满分，但是建议学习代码 1 的写法。

二、题目二：仓库管理

1. 思路：

本题目需要按照题目要求实现 Warehouse 类，仓库中固定有 3 种货物。

仓库需要存储三种信息：货物名称：字符串列表；货物价格：整数列表；货物数量：整数列表。在初始化时，需要将这些信息存储为类的属性。

get_warehouse()方法用于显示仓库中所有货物的信息，输出格式为每行显示一种货物的名称、价格和数量，可以使用循环遍历三种货物，按指定格式输出。

update_stock(rank, amount)方法用于更新库存数量，根据 rank 索引物品位置，用 amount 确定出入库数量，如仓库物品数量不支持出库，则直接输出"Outbound failed."并返回，不执行后续操作，如可执行出入库操作，因直接用 amount 的正负判断出入库，绝对值判断大小，所以直接库存+amount 即可，出入库后输出物品信息。

total_value()方法用于计算仓库总价值，用每种货物的价格×数量，然后求和，最后按要求格式输出总价值。

注意题目要求，为了和判题程序进行交互，需要添加以下代码：

```
while True:
    exec(input())
```

代码：

```
class Warehouse:
    def __init__(self, names, prices, quantities):
        """
        初始化仓库
        names: 货物名称列表
        prices: 货物价格列表
        quantities: 货物数量列表
        """
        self.names = names
        self.prices = prices
        self.quantities = quantities

```

```
    def get_warehouse(self):
        """仓库查询方法"""
        for i in range(3):
            print(f"name:{self.names[i]} price:{self.prices[i]}
quantity:{self.quantities[i]}.")

    def update_stock(self, rank, amount):
        """
        物品出入库操作
        rank: 操作物品在列表中的位置
        amount: 出入库数量（正数表示入库，负数表示出库）
        """
        # 判断是否能够成功出库，不能则输出 Outbound failed.
        if self.quantities[rank] + amount < 0:
            print("Outbound failed.")
            return
        # 如果无法出库则直接返回，也就不会执行下面的语句
        # 因直接用 amount 的正负判断出入库，绝对值判断大小，所以直接库存
+amount 即可
        self.quantities[rank] += amount
        # 出入库后输出物品信息
        print(f"name:{self.names[rank]} price:{self.prices[rank]}
quantity:{self.quantities[rank]}.")

    def total_value(self):
        """仓库价值统计方法"""
        total = 0
        for i in range(3):
            total += self.prices[i] * self.quantities[i]
        print(f"The total value of the warehouse is {total}.")

while True:
    exec(input())
```

未通过第一个测试数据表示代码无法正确处理商品全出库情况；
未通过第二个测试数据说明代码无法正确处理出库数大于库存数的情况；
未通过第三个测试数据说明代码无法正确处理计算总价值的情况；
未通过第四个测试数据说明代码无法正确处理正确出入库，计算总价值的情况；
未通过第五个测试数据说明代码无法正确处理遇到错误出库后计算总价值的情况；
针对第 6 个测试测试数据，初始化时会遇到负数和 0 价格，对于这种错误价格，需将价格改为 1。初始化时数量存在负数，对于错误数量，应将数量改为 0。在代码运行过程中，会遇到出入库时商品的 rank 不正确，对于不正确的 rank，

不进行任何操作。

代码：

```python
class Warehouse:
    def __init__(self, names, prices, quantities):
        """
        初始化仓库
        names: 货物名称列表
        prices: 货物价格列表
        quantities: 货物数量列表
        """
        self.names = names
        # 处理价格：负数或 0 改为 1
        self.prices = []
        for price in prices:
            if price <= 0:
                self.prices.append(1)
            else:
                self.prices.append(price)
        # 处理数量：负数改为 0
        self.quantities = []
        for quantity in quantities:
            if quantity < 0:
                self.quantities.append(0)
            else:
                self.quantities.append(quantity)

    def get_warehouse(self):
        """仓库查询方法"""
        for i in range(3):
            print(f"name:{self.names[i]} price:{self.prices[i]}
quantity:{self.quantities[i]}.")

    def update_stock(self, rank, amount):
        """
        物品出入库操作
        rank: 操作物品在列表中的位置
        amount: 出入库数量（正数表示入库，负数表示出库）
        """
        # 检查 rank 是否有效
        if rank < 0 or rank >= 3:
            return

        # 判断是否能够成功出库，不能则输出 Outbound failed.
        if self.quantities[rank] + amount < 0:
```

```
            print("Outbound failed.")
            return
        # 如果无法出库则直接返回，也就不会执行下面的语句
        # 因直接用 amount 的正负判断出入库，绝对值判断大小，所以直接库存+amount 即可
        self.quantities[rank] += amount
        # 出入库后输出物品信息
        print(f"name:{self.names[rank]} price:{self.prices[rank]} quantity:{self.quantities[rank]}.")

    def total_value(self):
        """仓库价值统计方法"""
        total = 0
        for i in range(3):
            total += self.prices[i] * self.quantities[i]
        print(f"The total value of the warehouse is {total}.")

while True:
    exec(input())
```