语法基础与算法分析

1.1 求两数之和

题目要求

1. 从数组 nums 中找到两个值，使其等于目标值 target
2. 数组中的值不能重复使用
3. 找到的值的下标和必须为最小值
4. 返回的下标值需以正序从小到大的顺序返回

解题思路

可以使用遍历循环解决，依次遍历每一个值，就像之前求两两组合有几种组合方法一样

1. 使用一次遍历，遍历每一个值 a
2. 再使用一次遍历，遍历 a 值后面的每一个值b
3. 判断 a 与 b 的和是否等于目标值 target
4. 因为索引下标是从小到大的，如果第一次找到满足要求的 a，b，那么此时 a、b 下标的和就是最小的，所以则直接返回其下标

因为该题与数组索引下标有关，所以为了方便解题，在遍历的时候可以直接遍历数组 nums 的索引

参考答案

```python
class Solution:
    def twoSum(self, nums: list, target: int) -> list:
        # 遍历数组nums
        for i in range(len(nums)):
            # 遍历 i 位置后面的数值，依次判断 i 位置的数值 与 其后面数值的和是否等于目标值 target
                for j in range(i+1,len(nums)):
                    # 直接返回下标 i、j，for循环不再执行
                    # 如果不在此返回，需要处理 for 循环
                    if nums[i] + nums[j] == target:
                        return [i, j]
```

1.2 判断数组重复元素

题目要求

1. 如果数组中有重复元素，且存在相等元素相邻的情况，返回 ‘01’ ，否则返回 ‘03’
2. 如果数组中没有重复元素，且偶数个数多于奇数个数，返回 ‘02’ ，否则返回 ‘03’

解题思路

1. 首先判断数组 arr 是否含有重复元素，可以用 set() 方法，将其转为集合，判断长度是否等于原 list 的长度
2. 然后分别对两种情况进行再次判断，有重复元素，判断相等元素是否相邻（在这里可以判断相邻元素是否相等），没有重复元素，判断偶数个数是否多于奇数个数
3. 在判断相邻元素是否相等的情况中，为了方便可以遍历索引下标，且需要将数组 arr 全部遍历查找判断一次，遍历时需要注意索引范围，只要存在相邻元素相等的情况，即返回 ‘01’ ，否则返回 ‘03'
4. 在判断偶数个数是否多于奇数个数时，正常进行判断即可，没有其他注意的地方。偶数个数多，返回 ‘02’ ，否

则返回 ‘03’

参考答案

```python
class Solution:
    def arrayRepeat(self, arr: list) -> str:

        # 判断数组arr中是否有重复元素，有重复元素时，去判断相邻元素是否重复
        if len(arr) != len(set(arr)):
            # 遍历数组arr中每一个元素，判断前后元素是否相等
            # 再进行元素比对的时候，最好遍历索引下标
            for i in range(len(arr)):
                # 排除索引 i 为 0 的时候，避免结果错误
                # ['a','b','c','d','e']
                #   0   1   2   3   4
                # 如果索引 i 为左端点 0 ，那么arr[i-1] == arr[-1]，结果不准
                # 如果索引 i 为右端点 4 ，那么arr[i-1] == arr[3]，结果正常
                if i != 0:
                    # 如果两相邻元素不相等，略过
                    if arr[i] != arr[i-1]:
                        pass
                    # 如果两相邻元素相等，返回‘03’
                    else:
                        return '03'
            # 当for遍历完即数组arr中相邻元素不相等即没有返回‘03’时，返回‘01’
            return '01'

        # 当数组arr中没有重复元素时，判断偶数个数与奇数个数的关系
        else:
            # 设定偶数个数 n ，奇数个数 m
            n = 0
            m = 0
            # 遍历每个元素，判断奇偶性，然后进行加 1 操作
            for i in arr:
                if i % 2 == 0:
                    n += 1
                else:
                    m += 1
            # 如果偶数个数多，返回‘02’，否则返回‘03’
            if n > m:
                return '02'
```

```
38            else:
39                return '03'
```

1.3 简单数学公式

题目要求

1. 根据提供的数学表达式进行代码编写
2. 返回的结果需向上取整

解题思路

1. 将数学表达式转为代码，注意 ^ 在 python 中为位运算符，应转为 ** ，√ 也应转为对应的方法
2. 利用 math 库中的方法进行向上取整或替换数学符号

参考答案

```python
# 导入数学函数库
import math
class Solution:
    def SpMtFml(self, num: int) -> int:
            # math.ceil(x)，向上取整，即大于或等于 x 的最小的整数，如 1.1，返回 2
            # math.sqrt(x)，开平方
            # pow(x,n)，求x的n次幂，返回 int / math.pow(x,n)，求x的n次幂，返回 float
            return math.ceil(math.sqrt((3 ** 4 + 5 * 6 ** 5) / num))
```

1.4 句子拼写检查

题目要求

1. 判断句子中的单词是否满足除特殊情况外，第一个单词首字母大写，其他单词全为小写的情况，是返回 True，否则返回 False
2. 特殊情况有三种：1. 如果单词全为大写，则忽略判断；2. 如果含有 “Python”、“Java”、“MachineLearning”、“DataMining”这 4 个单词且大小写一致，则忽略判断；3. 如果单词为数字+大写字母构成，则忽略判断

解题思路

对每个单词进行遍历，依次对单词进行判断（先判断特殊情况，特殊情况判断顺序不分先后，再进行正常情况判断）。即对每个单词先依次进行其中一种特殊情况的判断（排除），如果不满足该特殊情况，返回False，如果满足，则继续下一个特殊情况的判断，都三种特殊情况都满足时，最后再对正常情况进行判断。如果所有单词均满足了特殊的情况及正常的情况，那么该句子符合题意，返回 True

1. 题目提供的变量 st 为句子，由空格+单词组成，而且我们需要对每个单词进行判断，所以先对 st 利用 split 进行分隔，然后遍历
2. 先判断第2条特殊情况，即判断每个单词是否为那4个单词中的某一个。需要先进行粗略判断，即忽略大小写的情况进行一致性判断，如果一致，再进行大小写判断，如果大小写一致，则略过该单词，继续进行下一个单词的判断，否则，返回False
3. 再进行第3条特殊情况的判断，如果单词中含有数字，去掉数字，然后对剩余部分进行字母大写的判断，如果均为大写，则略过该单词，则进行下一个单词的判断，否则返回False
4. 再进行第1条特殊情况的判断，如果均为大写，则略过该单词，继续后续单词的判断，否则，不略过该单词，继续进行正常情况的判断（因为当isupper()为False时，表明单词不是均为大写，那么可能为都是小写或小写+大写组

成，所以要进行后续判断：首单词首字母大写判断、均为小写字母判断）

5. 最后对正常情况判断：首单词首字母大写。当单词为首字母时，对其进行 istitle() 的判断。因为可能存在单词重复的情况，所以可设立一个辅助值——索引，当辅助值为首单词的索引即为 0 时，进入首字母大写判断的程序，否则不进入程序。当首字母大写时，忽略该单词，继续进行下一个单词的判断，否则返回 False
6. 最后对正常情况判断：其余单词小写。如果单词均为小写，则忽略该单词，继续进行下一个单词的判断，否则返回 False
7. 排除了False情况，所以最后返回 True

参考答案

```python
class Solution:
    def detectCapital(self, st: str) -> bool:
        # 以空格为分隔符对st进行分割，转为列表
        li = st.split(' ')

        # 记录索引，方便后续判断句子第一个单词（防止单词重复）
        # 只有当单词为第一个单词即 n 为 0 时，才去判断
        n = -1
        # 设定一个辅助值，用于对数字+字母组成的单词的判断
        flag = False
        # 遍历句子单词
        for i in li:
            # 索引
            n += 1

            # 自上而下对每个单词依次进行特殊情况2、特殊情况3、特殊情况1、首单词首字母、其他字母的判断
            # 如果不满足要求，返回Fasle，如果满足要求，继续进行下一个单词的判断
            # 最后如果每个单词都满足要求，返回True

            # 首先判断单词是否是列表中的单词（第2条特殊情况）
            # 将单词小写化
            if i.lower() in ['python','java','machinelearning','datamining']:
                # 如果单词在这里面，说明满足特殊情况2，跳过本次判断（循环），所以进行下一个单词的判断
                # 如果单词不在这里面，说明大小写存在不同，不满足特殊情况2，所以返回False
                if i in ['Python','Java','MachineLearning','DataMining']:
                    continue
                else:
                    return False

```

```
            # 其次判断含有数字的单词是否全为大写（第3条特殊情况），是的话进行下一个单词的判断，否则返回False。
            # 设定一个辅助值 s，使其值与单词 i 相等
            s = i
            # 判断数字是否在单词 i 中
            for j in [str(i) for i in range(0,10)]:
                if j in i:
                    # 如果单词 i 中含有数字，则去掉 s 中的数字，使 flag 为 True
                    flag = True
                    s = s.replace(j,'')
            # 如果单词 i 中含有数字，才执行 if flag 这个代码块，才会判断单词中的字母是否都为大写
            # 如果都为大写，则进行下一个单词的判断，否则直接返回 False
            if flag:
                if s.isupper():
                    # 如果 s 均为大写字母，继续下一个单词的判断，并使 flag 变为初始值
                    flag = False
                    continue
                else:
                    return False

            # 再次判断单词是否都为大写（第1条特殊情况），如果都为大写，则进行下一个单词的判断
            if i.isupper():
                continue

            # 再次判断第一个单词的首字母，n为索引，当 i 为第一个单词时，进入下方 if n == 0 这个代码块
            # 首单词首字母判断必须要在三条特殊情况判断完毕之后，排除这些特殊情况，否则结果会不准
            if n == 0:
                # 如果第一个单词首字母大写，则进行下一个单词的判断
                if li[0].istitle():
                    continue
                else:
                    return False

            # 最后判断单词字母是否都为小写
            # 如果全为小写，则进行下一个单词的判断，否则返回False
            if i.islower():
                continue
            else:
```

```
                return False

        # 排除了False的情况，其他情况均为True

        return True
```

1.5 重复子字符串

题目要求

1. 判断字符串 s 是否由它的子串重复多次构成
2. 如果是返回其子串，否则返回字符串 s

解题思路

1. 可以利用子串与字符串s的长度关系、字符串包含字串的数量来解题，如 s 为 ‘abcabc’ 时，字串为 ‘abc’ ，那么字串在 s 中的数量 = s的长度与字串长度的比（其实这求的也是子串在 s 中的数量）
2. 如果 s 为空字符串或 s 中不存在子串时，返回原字符串 s

参考答案

```python
class Solution:
    def patternRepeatedSubstring(self, s: str) -> str:
        # 字串 ss 初始值
        ss = ''
        # 遍历 s 中的每个字符
        for i in s:
            # 每次对字串 ss 加上一个字符，然后判断 ss 是否为 s 的字串
            ss += i
            # 字串 ss 在 s 中的数量 == 字符串 s 的长度 / 字串 ss 的长度
            # 如 s 为‘abcabc’时，那么字串为‘abc’。字串在 s 中的数量：s.count(ss) = 2；两长度比：len(s) / len(ss) = 2。所以关系成立
            if s.count(ss) == len(s) / len(ss):
                return ss
        # 当 s 中不含有字串或字符串 s 为空字符串时，返回原字符串 s
        return s
```

网页处理与文本分析

2.1 选取网页表格中的数字

题目要求

1. 根据提供的网页，查找 row 行、col 列的数据
2. 返回的数据类型为 int

解题思路

1. 利用网页采集的套路模板：下载网页源代码—编码转换—网页解析，根据网页结构找到要查找的数据
2. 在查找数据时，注意 索引 与 row 以及 col 的关系

参考答案

```python
import requests
from bs4 import BeautifulSoup

class Solution:
    def table_num(self, row: int, col: int) -> int:
        # 下载网页源代码
        res = requests.get('http://72.itmc.org.cn:80/JS001/open/show/random-num/index.html')
        # 编码转换
        res.encoding = 'utf-8'
        # 网页解析
        soup = BeautifulSoup(res.text,'html.parser')
        # 先找到整体数据存在的节点：soup.find_all('tbody')[1]
        # 再找到目标数据所在的行：.find_all('tr')[row - 1]
        # 再找到目标数据所在的列（即该行第几个数据）：.find_all('td')[col]
        # 最后提取数据，并进行数据类型转换
        return int(soup.find_all('tbody')[1].find_all('tr')[row - 1].find_all('td')[col].string)
```

2.2 商品信息关键词权重计算

题目要求

1. 对网页中的内容进行权重分析，返回特定词性的、排名前 5 的词
2. 注意，进行权重分析的内容不能包括：标题、小标题（冒号前的内容）

解题思路

1. 先进行网页内容采集
2. 将不需要进行权重分析的内容排除，只选取进行权重分析的内容，注意：不同部分的内容之间要加上分隔符号，避免后续分析错误
3. 将最后整理出的内容进行权重分析，并返回特定词性的权重排名前5的词

参考答案

```python
import requests
import jieba
from jieba import analyse

class Solution:
    def wordTfidf(self) -> list:
        # 内容采集。因为采集的对象是txt文件，所以不需要解析，编码转换后直接可以使用
        res = requests.get('http://72.itmc.org.cn/JS001/data/user/4438/61/fj_chiffon_lady_dress.txt')
        # 编码转换
        res.encoding = 'utf-8'
```

```
11
12          # 因为要对内容进行筛选，所以先将整体内容进行按行分隔
13          s = res.text.splitlines()
14
15          # 将‘商品标题’的内容取出
16          ss = s[1]
17
18          # 将‘商品属性’的内容取出
19          # 注意冒号前面的内容不需要，只取冒号后的内容（冒号有两种形式）
20          # 还要注意下，在连接文本内容时，要加上分隔符号，不然会分析错误
21          for i in s[4:-3]:
22              if ':' in i:
23                  ss = ss + ',' + i.split(':')[-1]
24              if ': ' in i:
25                  ss = ss + ',' + i.split(': ')[-1]
26
27          # 将‘商品描述’的内容取出
28          ss = ss + ',' + s[-1]
29
30          # 权重分析，并返回
31          # analyse.tfidf(ana_text, topK=5, allowPOS=('nr', 'ns', 'n'))一样可以返回
32          # 返回的是 list
33          return analyse.extract_tags(ss,topK = 5,allowPOS = ('n','nr','ns'))
34
```

2.3 微信公众号排行榜分析

题目要求

1. 提供公众号信息网页，根据程序传入的公众号名称 name，判断公众号的三种信息是否满足要求，如果满足，返回 ‘YES’ ，否则返回 'NO'
2. 要求：“头条平均阅读”数 > 90000；“原创平均阅读”数 > 80000；“预估活跃粉丝”数 < 300万

解题思路

1. 先进行网页内容采集
2. 为了方便，可以先找到公众号名字所在的节点，然后再进一步找到该公众号三种信息都在的节点
3. 依次判断公众号是否满足这三种信息要求，如果有一条不满足，则直接返回 'NO'，如果满足，则忽略，继续进行下一条信息要求的判断，依次类推，如果都满足，则最后返回 ‘YES’ 。当然也可以对每一条信息要求的判断结果进行标记，最后对三条标记结果进行判断，如果为 True，则返回 'YES'，否则返回 ‘NO’
4. 注意判断信息要求时，因为三条要求都是对数字进行的比较，所以如果信息数据中存在 ‘万’ 、 ‘+’ 等符号，要先去掉再判断

参考答案

```
1  import re
```

```python
import requests
import bs4

class Solution:
    def weixinData(self, name: str) -> str:
        # 提供的代码是 import bs4，所以要 bs4.BeautifulSoup 这样书写
        res = requests.get('http://72.itmc.org.cn/JS001/open/show/weixindata.html')
        res.encoding = 'utf-8'
        soup = bs4.BeautifulSoup(res.text,'html.parser')

        # 先找到公众号名称所在的节点
        # 然后再找到三条信息所在的公共节点
        target_tag = soup.find('span',string = name).parent.parent.parent

        # 判断‘头条平均阅读’信息。
        # 如果信息中有‘万’，则数据均是 10万+ ，所以一定大于90000，所以略过，进行下一条要求的判断
        # 如果信息中不存在‘万’，则正常比较，如果大于90000，则略过，继续下一条要求的判断，否则直接返回 ‘NO’
        if '万' in target_tag.find_all('td')[4].string:
            pass
        else:
            if int(target_tag.find_all('td')[4].string) <= 90000:
                return 'NO'

        # 判断‘原创平均阅读’信息
        # 判断方法类似上一条
        if '万' in target_tag.find_all('td')[7].string:
            pass
        else:
            if int(target_tag.find_all('td')[7].string) <= 80000:
                return 'NO'

        # 判断‘预估活跃粉丝’信息
        # 这是最后一条要求，所以如果不满足的话直接返回‘NO’即可，不用再继续进行判断了
        if float(target_tag.find_all('td')[3].string.replace('万','')) >= 300:
            return 'NO'

        # 当都满足的时候，返回 ‘YES’
        return 'YES'
```

2.4 爱丽丝梦游仙境文本规范化

题目要求

1. 根据提供的网页文本，统计长度小于等于3的单词的词频，然后返回与 word 对应的词频
2. 不能使用jieba分词库直接分词
3. 单词不区分小写。所以需要进行大小写的转换
4. 如果找不到 word，则返回 0

解题思路

1. 先进行网页内容采集
2. 因为文本内容中存在特殊符号或标点符号，这些符号和单词组合在一起，会导致词频统计不准确。所以要先对文本内容如进行筛选处理，去掉这些特殊符号和标点符号，统一转换为可以进行分割的分割符号，如空格。
3. 对筛选后的文本以空格进行分割拆分
4. 再筛选出长度大于等于 3 的词语，在这里要主要对单词进行大小写的统一转换
5. 然后利用字典的 get 方法对单词的词频进行统计
6. 利用字典的查找方法对 word 单词的词频进行返回，因为如果找不到 word 需返回 0，所以最好还是用 get 进行查找。查找时需要将 word 的大小写与字典中单词的大小写保持一致

参考答案

```
import re
import requests

class Solution:
    def aliceText(self, word: str) -> int:
        # 下载文件内容
        res = requests.get('http://72.itmc.org.cn:80/JS001/data/user/4438/76/fj_alice_adventure.txt')
        # 编码转换
        res.encoding = 'utf-8'

        # 为了后续方便拆分，判断筛选文本内容res.text中是否有这些符号，如果有话，就将其替换为空格
        # 当然这样做可能会遗漏某些符号
        # 最好还是用 re 来进行替换，将不是下列字符的其他字符替换为空格，再替换换行符，如：
        # text = re.sub('[^A-Za-z1-9\']', ' ', rep.text).replace('\n', ' ')
        s = res.text
        for i in [',','-','.','!','?','\r','\n','\ufeff',';',':','"','_']:
            if i in res.text:
                s = s.replace(i,' ')

        # 将筛选后的文本内容用空格进行分割
        li = s.split(' ')
```

```
22
23          # 筛选出长度大于等于 3 的词汇
24          # 在这里要将所有词汇小写
25          li = [i.lower() for i in li if len(i) >= 3]
26
27          # 创建一个字典，为了对单词进行计数统计
28          # 字典的get方法就实现了这一操作，查找某个‘键’，如果存在，就返回对应的值，不存在返回 0，或其他自定义的内容
29          dict_ = {}
30          for i in li:
31              dict_[i] = dict_.get(i,0) + 1
32
33          # 查找 word 出现的次数，如果不存在返回 0
34          # 注意，这里也要将 word 小写化，与‘键’相对应
35          return dict_.get(word.lower(),0)
```

2.5 影片票房任务

题目要求

1. 根据提供的电影信息网页，查找目标电影 movie_name 的三条信息
2. ‘上映时间’：点映为 -1，首映为 0，其他情况正常采集，类型为 float
3. ‘综合票房’，类型为 float
4. ‘排片占比’，将百分数转为小数，类型为 float

解题思路

1. 先进行网页内容采集
2. 因为网页源代码部分标签不完整，所以不能用之前先查找电影名所在的节点来进行其他信息的查找的方法。所以找到包含所有电影信息的公共节点，然后遍历每一条电影信息
3. 依次对每一条电影信息进行判断，是否包含目标电影 movie_name
4. 找到目标电影后，再进行三条信息的查找
5. ‘排片占比’任务中，注意百分号 % 的转换、小于号 > 的去除，还有精度问题

参考答案

```
1  import bs4
2  import requests
3
4  class Solution:
5      def BoxOfficeSpider(self, movie_name: str) -> list:
6          # 下载网页源代码
7          res = requests.get('http://72.itmc.org.cn/JS001/open/show/box_office_on_a_certain_day.html')
8          # 编码转换
9          res.encoding = 'utf-8'
```

```
        # 网页解析，注意bs4
        soup = bs4.BeautifulSoup(res.text,'html.parser')

        # 因为网页源代码部分标签不完整，所以不能用之前先查找电影名所在的节点来进行其他信息的查找的方法
        # 只能进行遍历，判断其电影名是否是 movie_name
        # 如果是的话，再进行其他信息的查找

        for i in soup.find('tbody',class_ = 'table-body').find_all('tr'):
            # tr节点包含某个电影的所有信息
            # 判断tr节点中是否存在目标电影名 movie_name 所在的节点
            # 如果tr节点中存在目标电影，则进行信息查找
            if i.find('p',class_ = 'movie-name').string == movie_name:

                # 上映时间
                if '点映' in i.find('span',class_ = 'releaseInfo').string:
                    date = -1
                elif '首日' in i.find('span',class_ = 'releaseInfo').string:
                    date = 0
                else:
                    date = int(i.find('span',class_ = 'releaseInfo').string.replace('上映','').replace('天',''))

                # 综合票房
                sales = float(i.find('div',class_ = 'boxDesc-wrap red-color').string)

                # 排片占比
                # 百分号转换、小于号去除
                # 因为精度问题，如果不保留小数，则结果不准确，所以进行小数的保留，数据都是三位的，所以保留三位小数
                rat = round(float(i.find('div',class_= 'countRate-wrap').string.replace('%','').replace('<','')) / 100,3)

        return [date, sales, rat]
```

2.6 商品搜索结果分析

题目要求

1. 根据提供的商品信息，判断 shop_name 对应的商品是否参与满减、查找商品的价格（有会员价返回会员价）、评论数
2. 如果 shop_name 下的商品有多个，则返回价格最高的商品的信息，如果价格相同，则返回最靠前的商品的信息
3. ‘评论数’ 为 int 类型， ‘价格’ 为 float 类型， ‘是否参与满减’ 为 bool 类型

解题思路

1. 先进行网页内容采集
2. 再找到 shop_name 下所有的商品，然后找出价格最高的商品
3. 找到目标商品后，对商品进行信息判断查找
4. 除了下方这种查找价格最大的商品的方法，还可以设定一个最大价格的辅助值max_pr，对每个商品依次进行价格判断

参考答案

```
import re
# 题目中没有提供requests库的导入，所以自行导入
import requests
import bs4
from bs4 import BeautifulSoup

class Solution:
    def itemSearch(self, shop_name: str) -> list:
        # 下载网页源代码
        res = requests.get('http://72.itmc.org.cn/JS001/open/show/ecjd.html')
        # 编码转换
        res.encoding = 'utf-8'
        # 网页解析
        soup = BeautifulSoup(res.text,'html.parser')

        # 找到所有shop_name 店的商品信息
        pros_tag = soup.find_all('a',title = shop_name)

        # 为了判断商品价格，创建一个列表
        li = []
        # 遍历每一个商品信息
        for j in pros_tag:
            # 会员价
            vip_pr_tag = j.parent.parent.parent.find('span',class_ = 'price-plus-1')
            # 判断是否存在会员价，如果存在，将价格float后放入列表 li 中，默认保留一位小数
            if vip_pr_tag:
                li.append(float(vip_pr_tag.em.string[1:]))
            else:
                # 如果会员价不存在，将原价 float 后放入列表 li 中，默认保留一位小数
                li.append(float(j.parent.parent.parent.find('div',class_ = 'p-price').i.string))

```

```
        # 找到价格最高的商品信息
        # max(li)返回价格最高的商品的索引
        # pros_tag[li.index(max(li))]返回价格最高的商品的价格节点
        # .parent.parent.parent返回价格最高的商品的信息节点
        pro_tag = pros_tag[li.index(max(li))].parent.parent.parent

        # 判断商品是否满减
        # 因为属性'data-tips'中有‘-’，书写不规则，所以利用 attrs 来进行查找
        if pro_tag.find('i',attrs = {'data-tips':'本商品参与满减促销'}):
            pref = True
        else:
            pref = False

        # 价格，之前已经找出来了，直接写上
        pr = max(li)

        # 评论数
        # 评论数有两种书写形式，一种如‘1.3万+’，一种如‘9876+’，要分别进行判断采集
        commit_str = pro_tag.find('div',class_ = 'p-commit').a.string
        # 如果有‘万’字，去掉‘万+’后，乘 10000
        if '万' in commit_str:
            commit = int(float(commit_str[:-2]) * 10000)
        # 在这里要注意，因为有的评价节点的文本中可能包含换行符，所以，这里不能用索引获取，可以用split方法
        else:
            commit = int(commit_str.split('+')[0])

        return [pref, pr, commit]
```

2.7 商品关键词权重分析

题目要求

1. 根据提供text文件内容，对其进行权重分析，返回名词（n）、动词（v）权重排名最靠前的 5 个关键词
2. 使用 jieba 分词库中的 TextRank 算法进行分析，这个与前面用的 TF-IDF 算法不同

解题思路

1. 先进行text内容采集
2. 然后按照要求进行权重分析

参考答案

```
import requests
import jieba.analyse
```

```
3
4  class Solution:
5      def itemAnalyse(self) -> list:
6          # 内容采集
7          res =
   requests.get('http://72.itmc.org.cn:80/JS001/data/user/4438/77/fj_5392_hangzhou_top_woma
   n_wear.txt')
8          # 编码转换
9          res.encoding = 'utf-8'
10
11         # 对名词、动词进行权重分析
12         # 取权重排名前 5 的词语
13         return jieba.analyse.textrank(res.text, topK = 5, allowPOS=('n','v'))
```

数据处理与分析

3.1 电商订单数据清洗

题目要求

1. 将指定列‘商品描述(choice_description)’中的缺失值‘NaN’填充为‘banana’
2. 返回该列中缺失值的数量
3. 无需再去读取文件，传入的 order_data 的类型就是 'DataFrame'

解题思路

1. 先确定‘商品描述(choice_description)’列中缺失值的数量
2. 再将缺失值填充为‘banana’
3. 返回缺失值数量

参考答案

```
1  class Solution:
2      def nanCount(self, order_data: 'pd.DataFrame') -> int:
3          # 从此处开始编写代码
4          # 后台读取csv示例代码如下（参考）
5          # order_data = pandas.read_csv(url, sep=',')
6
7          # 无需读取文件，直接操作即可
8          # 确定缺失值的数量，转为 int 类型
9          num = int(order_data['choice_description'].isna().sum())
10         # 将缺失值填充为‘banana’，原地填充
11         order_data['choice_description'].fillna('banana', inplace=True)
12
13         return num
```

3.2 电商订单数据计算

题目要求

1. 根据提供的 csv 文件，要求返回订单总金额最大或最小的商品的名称
2. 传入的 condition 不区分大小写，可能为 ‘Max’ 、 ‘max’ 、 ‘Min’ 、 ‘min’ 等大小写不确定的形式
3. 返回的商品名称不能进行修改，需与文件中的名称保持一致

解题思路

1. 首先将文件中单价列 ‘item_price’ 中的美元符号去掉，然后再转为 float 类型
2. 因为数量列 ‘quantity’ 中的数据不都为 1 ，即一笔订单可能有多个商品，所以要先计算每笔订单的总金额：数量 * 单价
3. 再去计算在该文件所有订单中，每种商品的订单总金额
4. 最后根据要求，对 condition 进行大小写的转化，返回对应的商品的名称

参考答案

```python
import pandas as pd

class Solution:
    def salesStr(self, condition: str) -> str:

        # 读取文件
        df = pd.read_csv('http://72.itmc.org.cn:80/JS001/data/user/4438/80/fj_order_data.csv')
        # 将列‘item_price’数据中的美元符号 $ 去掉，然后再将数据类型转为 flaot
        df['item_price'] = df['item_price'].map(lambda x:x[1:]).astype('float')
        # 创建列 ‘sales’，其值为每笔订单（每行）的总金额，一行数据为一笔订单
        df['sales'] = df['quantity'] * df['item_price']

        # 如果求的是最大值
        if condition.lower() == 'max':
            # 将商品进行分组，求每种商品（所有订单中每种商品）的总金额，然后排序（默认正序排序），返回最大值的商品的名称
            return df.groupby('item_name').sum().sort_values('sales').index[-1]

        # 如果求的是最小值
        else:
            # 返回最小值的商品的名称
            return df.groupby('item_name').sum().sort_values('sales').index[0]
```

3.3 DataFrame计算平均值

题目要求

1. 根据程序传入的 DataFrame 对象 df，添加新列 ‘avg’ ，使其值为每行的平均值，且保留两位小数

2. 返回列 ‘avg’ ，数据类型需为 list

解题思路

1. 先添加新列 ‘avg’
2. 然后返回 list 数据类型的列 ‘avg’ 数据

参考答案

```
class Solution:

    def CalAvg(self, df: 'pandas.DataFrame') -> list:

        # 创建新列‘avg’，调用 mean 方法，传入参数axis = 1，使新列的值为每行的平均值，且保留两位小数
        df['avg'] = df.mean(axis = 1).round(2)

        # 返回列‘avg’,并转变数据类型为 list
        return list(df['avg'])
```

3.4 乐高商店收入计算

题目要求

1. 根据提供的 excel 文件，将价格 ‘price’ 列中为空的行删掉，将销量 ‘sales_num’ 列中的缺失值填充为 0
2. 计算文件中所有商品的总收入
3. 计算指定商品 title 的总收入
4. 计算指定商品 title 价格的平均值

解题思路

1. 因为文件中价格 ‘price’ 列中并没有缺失值，所以不用理会，只填充销量 ‘sales_num’ 列中的缺失值即可，填充为 0
2. 根据题目要求计算每一个问题：所有商品的总收入、指定商品 title 的总收入、指定商品 title 价格的平均值（保留两位小数）
3. 需要注意，每一个要求的数据类型为 float，在返回前需要将其转为 float 类型

参考答案

```
import pandas as pd

class Solution:
    def task(self, title):

        # 读取文件
        df = pd.read_excel('http://72.itmc.org.cn:80/JS001/data/user/4438/241/fj_lego_tmallshop_sales_data.xlsx')
```

```
        # 将‘销量’列中的缺失值填充为 0 ，原地修改
        df['sales_num'].fillna(0,inplace = True)
        # 计算每行商品的收入（销售金额）
        df['sale'] = df['price'] * df['sales_num']

        # 计算所有产品的总收入，转为 float 类型
        pros_sales = float(df['sale'].sum())

        # 计算指定商品 title 的总收入，转为 float 类型
        pro_sales = float(df[df['title'] == title]['sale'].sum())

        # 计算指定商品 title 价格的平均值，且保留两位小数，转为 float 类型
        mean_pr = float(df['price'].mean().round(2))

        return [pros_sales, mean_pr, pro_sales]
```

3.5 部门平均薪资汇总计算

题目要求

1. 根据提供的 excel 文件，计算各部门的 ‘平均薪资’ ，保留两位小数、降序返回、返回Series、Series名字为‘平均薪资’
2. 根据题目要求进行个人薪资的计算：基本工资（基本薪资 + 岗位工资 + 绩效工资 - 五险一金）+ 通勤薪资（加班工资 - 请假扣除）

解题思路

1. 文件中存在两个 sheet 表，分别读取
2. 根据两个文件中的列信息，将两个文件以 ‘姓名’ 列为连接键进行合并
3. 因为表中含有缺失值，先进行清洗，将其填充为 0 ，方便后续计算
4. 然后根据题目要求：薪资计算方法进行各种薪资计算、金额扣除，计算出个人薪资总额
5. 计算部门薪资总额，然后进行分组、求平均薪资，保留两位小数
6. 注意：题目要求返回Series、降序排序、name属性设置

参考答案

```
import pandas as pd

class Solution:

    def department_salary_summary(self) -> pd.Series:
        # 读取基本薪资工作表
        df1 =
pd.read_excel('http://72.itmc.org.cn:80/JS001/data/user/4438/242/fj_employee_salary_work
_books.xlsx',sheet_name = 0)
```

```
        # 读取职工薪酬工作表
        df2 = pd.read_excel('http://72.itmc.org.cn:80/JS001/data/user/4438/242/fj_employee_salary_work_books.xlsx',sheet_name = 1)
        # 将两个表以‘姓名’列为连接键进行连接合并
        df = pd.merge(df1,df2)
        # 因为表中存在缺失值，所以先将其填充为 0 ，方便后续计算，原地修改
        df.fillna(0,inplace = True)

        # 计算‘五险一金’扣除金额
        df['五险一金'] = df['社会保险缴费基数'] * (0.08 + 0.02 + 0.01 + 0.1)

        # 计算员工‘基本工资’
        df['基本工资'] = df['基本薪资'] + df['岗位工资'] + df['绩效工资'] - df['五险一金']

        # 计算员工的‘时薪’
        df['时薪'] = (df['基本薪资'] + df['岗位工资'] + df['绩效工资']) / df['应出勤天数（天）'] / 8

        # 计算员工的‘通勤工资’
        df['通勤工资'] = df['工作日加班（小时）'] * df['时薪'] + df['周末加班（小时）'] * df['时薪'] * 1.5 + df['法定假日加班（小时）'] * df['时薪'] * 2 - df['时薪'] * df['请假（小时）']

        # 计算员工的总工资
        df['总工资'] = df['基本工资'] + df['通勤工资']

        # 计算各部门员工的‘平均薪资’，保留两位小数，并转为 Series (取‘总工资’列)，进行降序排序（默认正序排序）
        series = df[['部门','总工资']].groupby('部门').mean().round(2)['总工资'].sort_values(ascending = False)

        # 将 Series 的名字改为‘平均薪资’
        series.name = '平均薪资'

        return series
```

3.6 编制比较资产负债表

题目要求

1. 根据提供的 excel 文件，先将缺失值填充为 0，再计算 2021 年各项指标的 ‘变动额’ （保留两位小数）、 ‘变动率’ （保留四位小数）
2. 以列表的形式返回指定项目 name 的 ‘变动额’ 、 ‘变动率’ ，数据类型为 float

解题思路

1. 因为在文件中，第三个 sheet 表中的数据正好可以满足题目要求：计算 2021 年各项指标的变动，所以只读取第三个 sheet 表即可
2. 填充缺失值为 0
3. 添加新列 ‘变动额’ 、 ‘变动率’ ，分别保留对应的小数位数
4. 找到项目 name 下的 ‘变动额’ 、 ‘变动率’
5. 以列表的形式返回 float 类型的 ‘变动额’ 、 ‘变动率’

参考答案

```
import pandas as pd

class Solution:
    def balance_sheet(self, name):

        # 读取第三个 sheet 表
        df = pd.read_excel('http://72.itmc.org.cn:80/JS001/data/user/4438/243/fj_Interprice_balance_data.xlsx',sheet_name = 2)
        # 将缺失值填充为 0 ，原地修改
        df.fillna(0, inplace = True)

        # 添加新列 ‘变动额’，保留两位小数
        df['变动额'] = (df['2021.12.31'] - df['2020.12.31']).round(2)
        # 添加新列 ‘变动率’，保留四位小数
        df['变动率'] = ((df['2021.12.31'] - df['2020.12.31']) / df['2020.12.31']).round(4)

        # 找到项目 name
        df = df[df['项目'] == name]

        # 返回该项目的 ‘变动额’、‘变动率’，记得转为 float 类型
        return [float(df['变动额']), float(df['变动率'])]
```

3.7 某店铺不同地区销售情况分析

题目要求

1. 根据提供的 csv 文件，要求计算电商中的各个指标情况：’ 订单的付款转化率’ 、’ 买家全额支付的转化率’ 、’ 买家实际支付总金额’ 、’ 客单价’ 、’ 销量（订单量）最多的产品的价格’ 、’ 买家实际支付总金额在所有地区实际支付总金额中的占比’
2. 转化率、占比任务返回百分号形式，所有任务均需保留小数点后两位小数
3. 如果某地区无销量（总实际支付金额为 0），则 ’ 客单价’ 、’ 销量（订单量）最多的产品的价格’ 需返回字

符串 ‘本地区无销量’

4. 以列表形式按任务顺序返回

解题思路

1. 读取 csv 文件

2. 根据题目要求及参考公式进行计算、返回

付款转化率 = 付款订单数 / 总订单数

买家全额支付的转化率 = 买家全额支付的订单数 / 总订单数

买家实际支付总金额 = ‘买家实际支付金额’ 列的总和

客单价 = 实际支付金额 / 支付买家数

销量（订单量）最多的产品的价格 = 买家实际支付金额大于0的订单中，对总金额（价格）列进行计数操作，求数量最多的产品的价格。当然也可以对价格进行分组求其出现的数量

买家实际支付总金额在所有地区实际支付总金额中的占比 = 本地买家实际支付总金额 / 所有地区买家实际支付总金额

参考答案

```python
import pandas as pd

class Solution:
    def task(self, area: str) -> list:
        # 读取文件
        df = pd.read_csv("http://72.itmc.org.cn:80/JS001/data/user/4438/240/fj_7568_tmall_order_report.csv")
        # 筛选出 area 地区的订单数据
        df_area = df[df["收货地址"] == area]
        # 总订单数
        ord_sum = len(df_area)

        # 订单的付款转化率:付款订单数 / 总订单数
        # 付款订单数：订单付款时间列中非缺失值的数量，即存在付款行为的数量，没有付款时间说明没有付过款
        pay_rate = "{:.2%}".format(df_area["订单付款时间"].notna().sum() / ord_sum)

        # 买家全额支付的转化率:买家全额支付的订单数 / 总订单数
        # 买家全额支付的订单数：买家实际支付金额 == 总金额 的数量
        fulpay_rate = "{:.2%}".format(len(df_area[df_area["买家实际支付金额"] == df_area["总金额"]]) / ord_sum)

        # 买家实际支付总金额:‘买家实际支付金额’ 列的总和
        # 保证后续计算数据准确，准备一个没有保留小数的数据
```

```
        pay_sum = df_area["买家实际支付金额"].sum()
        pay_sum_result = "{:.2f}".format(df_area["买家实际支付金额"].sum())

        # 客单价&销量最多的产品的价格
        # 如果本地区无销量，即买家实际支付总金额为 0
        if pay_sum == 0:
            # 客单价
            per_cus = "本地区无销量"
            # 销量最多的产品的价格
            pro_pir = "本地区无销量"
        else:
            # 客单价:实际支付金额 / 支付买家数
            # 支付买家数：买家实际支付金额大于 0 的数量
            per_cus = "{:.2f}".format(pay_sum / len(df_area[df_area["买家实际支付金额"] > 0]))
            # 销量最多的产品的价格:买家实际支付金额大于0的订单中，对总金额（价格）列进行计数操作（默认降序排序），求数量最多的产品的价格
            # 计数操作后，返回一个 Series，总金额为索引，对应出现的数量为数据，且数据（总金额）数量最多的在最前面
            pro_pir = "{:.2f}".format(df_area[df_area["买家实际支付金额"] > 0]["总金额"].value_counts().index[0])

        # 买家实际支付总金额在所有地区实际支付总金额中的占比:本地买家实际支付总金额 / 所有地区买家实际支付总金额
        pay_ratio = "{:.2%}".format(pay_sum / df["买家实际支付金额"].sum())

        return [pay_rate, fulpay_rate, pay_sum_result, per_cus, pro_pir, pay_ratio]
```

3.8 用户职位信息统计

题目要求

1. 根据提供的 excel 文件，计算指定职位 ‘occupation’ 下，女性用户的百分占比，写为百分数形式，且保留两位小数
2. 统计指定职位 ‘occupation’ 女性用户百分占比在所有职位女性用户百分占比中的排名（降序），输出数据类型为 int
3. 然后将两个任务以列表的形式返回

解题思路

1. 读取 excel 文件
2. 将指定职位为 ‘occupation’ 的数据筛选出来
3. 然后去计算第一个任务：女性用户数量 / 该职位用户总数量
4. 要求排名，可以利用分组，按照职位 ‘occupation’ 、性别 ‘gender’ 进行分组、计数

5. 因为第 4 条操作后，返回的行列索引不满足我们的需要，所以可以进行行列转换，unstack 操作，转换后出现了缺失值，将其填充
6. 添加 ‘占比’ 列，并求该列中各种职位的 ‘占比’ 值
7. 因为题中并没有提及排名相同的情况，所以直接对该列进行 rank 降序默认方法排名（数据为 float 类型）
8. 找到指定职位的排名，进行数据类型转换
9. 按要求返回

参考答案

```
import pandas as pd

class Solution:
    def jobStatistics(self, occupation: str) -> list:
        # 表格链接
        url ='http://72.itmc.org.cn:80/JS001/data/user/4438/67/fj_jobstatics.xlsx'
        # 读取表格
        chipo = pd.read_excel('http://72.itmc.org.cn:80/JS001/data/user/4438/67/fj_jobstatics.xlsx')

        # 将指定的职位信息‘occupation’数据挑选出来
        chipo_1 = chipo[chipo['occupation'] == occupation]

        # 计算指定职位信息数据中‘女’性的数量：len(chipo_1[chipo_1['gender'] == 'F'])
        # 计算指定职位信息数据总数量：len(chipo_1)
        # 计算‘女’性占比，以百分号的形式返回，且保留两位小数
        proportion = '{:.2%}'.format(len(chipo_1[chipo_1['gender'] == 'F']) / len(chipo_1))

        # 将数据表按照‘职位’（occupation）、‘性别’（gender）分组：chipo.groupby(['occupation','gender'])
        # 计算每组数据‘user_id’列中数据的个数即每组数据的个数：.count()，返回的结果为层次化索引的 DataFrame
        #                         user_id  age  zip_code
        # occupation    gender
        # administrator   F         36      36     36
        #                 M         43      43     43
        # artist          F         13      13     13
                          M         15      15     15
          doctor          M          7       7      7
          educator        F         26      26     26
        #                    .....
```

```
        # 因为没有缺失值，数据数量相同，所以然后随便选取一列：['user_id']，返回的是一个Series，依旧是层次化索引
        # occupation     gender
        # administrator  F          36
        #                M          43
        # artist         F          13
        #                M          15
        # doctor         M           7
        # educator       F          26
        #           ......
        # 将返回结果进行行列转换，将数据的行‘旋转’为列：.unstack()，将‘gender’转为列索引，行索引为‘gender’，列索引为‘F’、‘M’
        # gender         F      M
        # occupation
        # administrator  36.0  43.0
        # artist         13.0  15.0
        # doctor         NaN    7.0
        # educator       26.0   69.0
        #          ......
        # 旋转后数据中存在缺失值，将缺失值填充为 0，方便后续比重计算
        df = chipo.groupby(['occupation','gender']).count()['user_id'].unstack().fillna(0)

        # 创建‘占比’列，并计算其数据：‘女’性数量 / 总数量
        df['占比'] = df['F'] / (df['F'] + df['M'])

        # 对列‘占比’数据进行排名，并降序排序：df['占比'].rank(ascending = False)，索引不变，数据为具体的排序
        # occupation
        # administrator     5.0
        # artist            4.0
        # doctor           21.0
        # educator         11.0
        # engineer         20.0
        #        ......
        # 因为题目中并没有说存在排名相同的情况，所以各个职位女性用户的占比必定不会相同，所以直接调用 rank 方法默认排名即可
        # 找到指定岗位‘occupation’的占比排名情况：[occupation]
        # 将数据类型转为 int
        # 以列表形式返回
        return [proportion, int(df['占比'].rank(ascending = False)[occupation])]
```

3.8 某知名网站高质量视频主挖掘

题目要求

1. 根据提供的 csv 文件及分类指标，判断返回指定的视频主 author 属于哪个类别
2. I = (总弹幕数 + 总评论数) / (总播放量 * 统计范围内视频数量) * 100
3. F = (统计范围内最晚发布视频时间 - 最早发布视频时间) / 发布视频的数量
4. L = (点赞数 * 1 + 投币数 * 2 + 收藏数 * 3 + 分享数 * 4) / (播放量 * 发布视频数) * 100

解题思路

1. 读取 csv 文件
2. 文件数据去重、发布时间 ‘pubdate’ 列数据类型转换
3. 将各个视频主的数据进行合并
4. 将 ‘author’ 列设为索引，创建新列 ‘count’ 用于标记视频主发布的视频数量
5. 根据要求分别计算每个视频主的 I、F、L 值
6. 计算 I、F、L 均值，判断视频主是否满足题目中与均值判断的要求
7. 将三者的判断结果综合成 IFL 模型
8. 创建判断视频主类型的函数
9. 对每个视频主进行判断
10. 返回指定的视频主的类型

参考答案

```
import pandas as pd

class Solution:
    def videoMining(self, author: str) -> str:

        # 读取数据表
        df = pd.read_csv('http://72.itmc.org.cn:80/JS001/data/user/4438/94/fj_B_video_web_data.csv')

        # 删除重复项
        df.drop_duplicates(inplace=True)
        # 将‘pubdate’列转换成时间序列
        df['pubdate'] = pd.to_datetime(df['pubdate'])

        # 计算各个视频主的总值，数据汇总，不能合并的数据会被抛弃
        df_total = df.groupby('author', as_index=False).sum()

        # 将作者‘author’列设置为索引，丢弃‘author’列
        df_total.set_index('author', inplace=True, drop=True)
```

```
19
20          # 添加新列‘count’，值为各个视频主的视频总数量
21          df_total = df_total.assign(count=df['author'].value_counts())
22
23          # 构建 I 值
24          df_total['I'] = (df_total['danmu'] + df_total['reply']) / (df_total['views'] *
    df_total['count']) * 100
25
26          # 计算F值
27          time_diff = df.groupby('author')['pubdate'].max() - df.groupby('author')
    ['pubdate'].min()
28          df_diff = time_diff.dt.total_seconds()
29          df_total = df_total.assign(time_diff=df_diff)
30          df_total['F'] = df_total['time_diff'] / df_total['count']
31          df_total.loc[df_total['F'] == 0, 'F'] = df_total['F'].max() + 1
32
33          # 计算L值
34          df_total['L'] = (df_total['likes'] + df_total['coins'] * 2 +
    df_total['favorites']* 3 + df_total['share'] * 4) / (df_total['views'] *
    df_total['count']) * 100
35
36          # 计算均值，并判断每个视频主的I、L是否大于均值，F是否小于均值，如果满足，则 * 1
37          df_total['df_I'] = (df_total['I'] > df_total['I'].mean()) * 1
38          df_total['df_F'] = (df_total['F'] < df_total['F'].mean()) * 1
39          df_total['df_L'] = (df_total['L'] > df_total['L'].mean()) * 1
40
41          # 将IFL综合分析
42          df_total['IFL'] = (df_total['df_I'] * 100) + (df_total['df_F'] * 10) +
    (df_total['df_L'] * 1)
43
44          # 创建判断函数
45          def transform_label(x):
46              if x == 111:
47                  label = '高质量视频主'
48              elif x == 101:
49                  label = '高质量拖更视频主'
50              elif x == 11:
51                  label = '高质量内容高深视频主'
52              elif x == 1:
53                  label = '高质量内容高深拖更视频主'
54              elif x == 110:
```

```
            label = '接地气活跃视频主'
        elif x == 10:
            label = '活跃视频主'
        elif x == 100:
            label = '接地气视频主'
        elif x == 0:
            label = '还在成长的视频主'
        return label

    # 根据IFL应用上述判断函数，创建‘人群类型’（视频主分类）列
    df_total['人群类型'] = df_total['IFL'].apply(transform_label)

    # 找到指定 author 的‘人群类型’(视频主分类)，然后将其提取出来，进行返回
    return df_total[df_total.index == author]['人群类型'].to_list()[0]
```