Знакомство с языком Python (семинары)

Задание 1. Сумма чисел

Во входном файле numbers.txt записано N целых чисел, которые могут быть разделены пробелами и концами строк. Напишите программу, которая выводит сумму чисел во выходной файл answer.txt.

Пример:

Содержимое файла numbers.txt

```
   2
2
 2
    2
```

Содержимое файла answer.txt

```
8
```

Подсказка № 1

Откройте входной файл в режиме чтения с помощью функции `open()`. Убедитесь, что вы используете правильное имя файла и кодировку. Функция `open()` с режимом `"r"` открывает файл для чтения. Убедитесь, что файл доступен и путь к файлу указан верно.

Подсказка № 2

Инициализируйте переменную для хранения суммы чисел перед началом чтения файла. Переменная `total_sum` используется для накопления суммы чисел из файла. Это должно быть сделано до чтения файла.

Подсказка № 3

Прочитайте файл построчно с помощью цикла `for`. Разделите каждую строку на части с помощью метода `split()` и преобразуйте их в целые числа.

Подсказка № 4

Закройте входной файл после завершения чтения, чтобы освободить ресурсы. Закрытие файла с помощью метода `close()` после завершения его использования

предотвращает утечку ресурсов и обеспечивает корректное завершение работы с файлом.

Эталонное решение:

```
import os


# Открываем входной файл для чтения
numbers_file = open("numbers.txt", "r", encoding="utf-8")


# Инициализируем переменную для хранения суммы чисел
total_sum = 0


# Читаем файл построчно
for line in numbers_file:
    # Разбиваем строку на части, используя пробелы и новые строки в качестве разделителей
    # Преобразуем каждую часть в целое число и накапливаем сумму
    numbers = [int(num) for num in line.split() if num]
    total_sum += sum(numbers)


# Закрываем файл после чтения
numbers_file.close()


# Открываем выходной файл для записи
sum_file = open("answer.txt", "w", encoding="utf-8")


# Записываем сумму в выходной файл
sum_file.write(str(total_sum))
```

```
# Закрываем выходной файл

sum_file.close()
```

Задание 2. Сумма чисел

В файле zen.txt хранится так называемый Дзен Пайтона — текст философии программирования на языке Python. Выглядит он так:

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than *right* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

Напишите программу, которая выводит на экран все строки этого файла в обратном порядке.

Кстати, попробуйте открыть консоль Python и ввести команду import this.

Результат работы программы:

Namespaces are one honking great idea -- let's do more of those!

If the implementation is easy to explain, it may be a good idea.

If the implementation is hard to explain, it's a bad idea.

Although never is often better than *right* now.

…

Подсказка № 1

Перед открытием файла убедитесь, что он существует в ожидаемом месте. Если файл не существует или имеет другое имя, программа не сможет его открыть. Проверьте путь к файлу и его наличие.

Подсказка № 2

Откройте файл для чтения с помощью функции `open()`. Убедитесь, что файл открыт в режиме чтения (`"r"`). Используйте метод `readlines()` для получения всех строк файла в виде списка.

Подсказка № 3

Закройте файл после чтения данных, чтобы освободить ресурсы и избежать утечек памяти. Используйте метод `close()` для закрытия файла.

Подсказка № 4

Используйте функцию `reversed()` для переворачивания списка строк. Обратите внимание, что `reversed()` возвращает итератор, который нужно итерировать для получения перевернутого списка.

Эталонное решение:

```
# Открываем файл для чтения

philosophical_text = open("zen.txt", "r")
```

```
# Считываем все строки в список

lines = philosophical_text.readlines()



# Закрываем файл после чтения

philosophical_text.close()



# Проходим по строкам в обратном порядке и выводим каждую строку без
символа новой строки в конце

for line in reversed(lines):

    print(line.strip())
```

Задача 3. Турнир

В файле first_tour.txt записано число K и данные об участниках турнира по настольной игре «Орлеан»: фамилии, имена и количество баллов, набранных в первом туре. Во второй тур проходят участники, которые набрали более K баллов в первом туре.

Напишите программу, которая выводит в файл second_tour.txt данные всех участников, прошедших во второй тур, с нумерацией.

В первой строке нужно вывести в файл second_tour.txt количество участников второго тура. Затем программа должна вывести фамилии, инициалы и количество баллов всех участников, прошедших во второй тур, с нумерацией. Имя нужно сократить до одной буквы. Список должен быть отсортирован по убыванию набранных баллов.

Пример:

Содержимое файла first_tour.txt:

80

Ivanov Serg 80

Sergeev Petr 92

Petrov Vasiliy 98

Vasiliev Maxim 78

Содержимое файла second_tour.txt:

2

1) V. Petrov 98

2) P. Sergeev 92

Подсказка № 1

Откройте файл `first_tour.txt` для чтения с помощью функции `open()` и метода `readlines()`, чтобы получить все строки файла. Метод `readlines()` позволяет считывать все строки файла и сохранить их в список. Первая строка файла будет содержать число `K`, а остальные строки будут содержать данные участников.

Подсказка № 2

Извлеките значение `K` из первой строки и сохраните его как целое число с помощью функции `int()`.

Подсказка № 3

Используйте цикл `for` для обработки оставшихся строк, разделяя каждую строку на части и формируя имя участника в формате инициалов и фамилии, а также извлекайте количество баллов. Для формирования имени участника инициал должен быть первой буквой имени, а фамилия — полной. Количество баллов должно быть преобразовано в целое число.

Подсказка № 4

Создайте два словаря: один для хранения всех участников с их баллами и второй для фильтрации участников, которые набрали больше `K` баллов. Отфильтруйте участников и отсортируйте их по убыванию баллов. Сначала добавьте участников в словарь с баллами, затем используйте условие для фильтрации и сохранения только тех, кто набрал больше `K` баллов. Отсортируйте отфильтрованных участников по баллам в порядке убывания.

Подсказка № 5

Откройте файл `second_tour.txt` для записи и запишите количество участников второго тура. Затем запишите данные участников с нумерацией в нужном формате. При записи в файл укажите количество участников, а затем для каждого участника укажите номер, имя с инициалом и количество баллов. Нумерация должна начинаться с 1 и увеличиваться на 1 для каждого следующего участника.

Эталонное решение:

```
# Открываем файл first_tour.txt для чтения
with open("first_tour.txt", "r") as file:
    lines = file.readlines()
    # Первая строка содержит число K
    K = int(lines[0])
    # Словарь для хранения данных участников
    participants = {}
    # Словарь для хранения участников, прошедших во второй тур
    filter_player = {}
    count = 1

# Обрабатываем оставшиеся строки
for line in lines[1:]:
    # Разделяем строку на части
    data = line.strip().split()
    # Формируем имя участника с инициалом
    name = f"{data[1][0]}. {data[0]}"
    # Получаем количество баллов
    score = int(data[-1])
    # Сохраняем участника и его баллы
    participants[name] = score

# Фильтруем участников, набравших более K баллов
for player, score in participants.items():
    if score > K:
        filter_player[player] = score
```

```
# Сортируем участников по убыванию баллов

sorted_filter_player = dict(sorted(filter_player.items(), key=lambda
x: x[1], reverse=True))



# Открываем файл second_tour.txt для записи

with open("second_tour.txt", "w") as file:

    # Записываем количество участников второго тура

    file.write(f"{len(sorted_filter_player)}\n")

    # Записываем данные участников с нумерацией

    for player, score in sorted_filter_player.items():

        file.write(f"{count}) {player} {score}\n")

        count += 1
```

Задача 4. Частотный анализ

Есть файл text.txt, который содержит текст. Напишите программу, которая выполняет частотный анализ, определяя долю каждой буквы английского алфавита в общем количестве английских букв в тексте, и выводит результат в файл analysis.txt. Символы, не являющиеся буквами английского алфавита, учитывать не нужно.

В файл analysis.txt выводится доля каждой буквы, встречающейся в тексте, с тремя знаками в дробной части. Буквы должны быть отсортированы по убыванию их доли. Буквы с равной долей должны следовать в алфавитном порядке.

Пример:

Содержимое файла text.txt:

Mama myla ramu.

Содержимое файла analysis.txt:

a 0.333

m 0.333

l 0.083

r 0.083

u 0.083

y 0.083

Подсказка № 1

Откройте файл `text.txt` и считайте весь текст в переменную. Приведите текст к нижнему регистру для корректного подсчета частоты букв. Используйте функцию `open()` для открытия файла и метод `read()` для считывания всего содержимого. Метод `lower()` преобразует текст в нижний регистр, чтобы анализ был нечувствителен к регистру букв.

Подсказка № 2

Создайте словарь для хранения количества вхождений каждой буквы английского алфавита. Инициализируйте все буквы значением 0. Словарь должен содержать ключи для всех букв английского алфавита с начальным значением 0. Это поможет в дальнейшем подсчете частоты каждой буквы.

Подсказка № 3

Проходите по каждому символу в тексте. Если символ является буквой английского алфавита, увеличивайте его счётчик в словаре и общий счётчик английских букв. Используйте цикл для обхода каждого символа в тексте. Проверьте, является ли символ буквой (например, с помощью оператора `in` для проверки наличия в наборе букв), и обновите соответствующий счётчик.

Подсказка № 4

Создайте новый словарь, в котором будут храниться доли (или частоты) встречаемости каждой буквы, вычисленные как отношение количества вхождений к общему количеству английских букв. Чтобы вычислить частоту, разделите количество вхождений каждой буквы на общее количество английских букв. Используйте словарь для хранения этих значений, исключая буквы, которые не встречаются в тексте.

Подсказка № 5

Отсортируйте буквы по убыванию их частоты, и в случае равенства частоты — по алфавиту. Запишите результат в файл `analysis.txt` с форматированием до трёх знаков после запятой. Для сортировки используйте функцию `sorted()` с параметром `key`, указывающим на частоту в убывающем порядке и букву в алфавитном порядке при равенстве частот. Запишите результат в файл с помощью метода `write()` и форматируйте частоту до трёх знаков после запятой.

Эталонное решение:

```
# Определяем английский алфавит

english_alphabet = 'abcdefghijklmnopqrstuvwxyz'

total_letters = 0



# Открываем файл text.txt и считываем текст

with open("text.txt", "r") as file:

    text = file.read().lower()  # Приводим текст к нижнему регистру



# Создаем словарь для подсчета количества каждой буквы

letter_count = {letter: 0 for letter in english_alphabet}



# Подсчитываем количество вхождений каждой буквы

for char in text:

    if char in english_alphabet:

        letter_count[char] += 1

        total_letters += 1



# Вычисляем частоту встречаемости каждой буквы

letter_freq = {letter: (count / total_letters) for letter, count in
letter_count.items() if count > 0}



# Сортируем буквы по убыванию частоты и по алфавиту при равенстве
частоты

sorted_letters = sorted(letter_freq.items(), key=lambda x: (-x[1],
x[0]))



# Записываем результат в файл analysis.txt

with open("analysis.txt", "w") as file:
```

```
    for letter, freq in sorted_letters:

        file.write(f"{letter} {freq:.3f}\n")
```

Задача 5*. «Война и мир»

Мало кто не знает знаменитый роман Л. Н. Толстого «Война и мир». Это довольно объёмное произведение лежит в архиве voina-i-mir.zip. Напишите программу, которая подсчитывает статистику по буквам (не только русского алфавита) в этом романе и выводит результат на экран (или в файл). Результат должен быть отсортирован по частоте встречаемости букв (по возрастанию или убыванию). Регистр символов имеет значение.

Архив можно распаковать вручную, но, если хотите, можете изучить документацию по модулю zipfile (можно использовать и другой модуль) и попробовать написать код, который будет распаковывать архив за вас.

Подсказка № 1

Для работы с архивом `.zip` используйте модуль `zipfile`. Начните с открытия архива и выяснения, какие файлы он содержит. Найдите текстовый файл, который содержит нужный вам текст. Используйте метод `zipfile.ZipFile()` для открытия архива и метод `namelist()` для получения списка файлов. Фильтруйте файлы по расширению `.txt` или другим критериям, чтобы найти нужный файл.

Подсказка № 2

Откройте найденный текстовый файл из архива и считайте его содержимое в строку. Используйте метод `read()` и декодируйте байты в текстовый формат, например, UTF-8. Используйте метод `open()` для чтения файла из архива, а затем метод `read().decode('utf-8')` для преобразования содержимого в строку. Это позволит работать с текстом в читаемом виде.

Подсказка № 3

Создайте словарь для хранения количества вхождений каждого символа. Проходите по каждому символу в тексте и обновляйте счётчики в словаре только для букв (или других символов, если это требуется). Используйте цикл для перебора символов в тексте. Проверьте, является ли символ буквой с помощью метода `isalpha()`, и обновите счётчик в словаре. Увеличивайте счётчик для каждого символа по мере его встречи.

Подсказка № 4

Преобразуйте словарь подсчета в словарь частот, если требуется. Отсортируйте символы по убыванию частоты их появления. При равенстве частот отсортируйте их по алфавиту. Используйте функцию `sorted()` для сортировки словаря по значениям. Для этого можно использовать параметр `key` для сортировки по частоте в убывающем порядке и по символам в алфавитном порядке при равенстве частот.

Эталонное решение:

```
import zipfile



# Открываем архив и ищем текстовый файл

with zipfile.ZipFile("voina-i-mir.zip", "r") as zip_ref:

    # Получаем список файлов в архиве и выбираем первый текстовый
файл

    file_name = [name for name in zip_ref.namelist() if
name.endswith('.txt')][0]



    # Открываем выбранный файл и читаем его содержимое

    with zip_ref.open(file_name) as file:

        text = file.read().decode('utf-8')



# Инициализируем словарь для подсчета количества символов

char_count = {}



# Подсчитываем количество вхождений каждого символа

for char in text:

    if char.isalpha():  # Учитываем только буквы

        if char in char_count:

            char_count[char] += 1

        else:
```

```
            char_count[char] = 1



# Сортируем символы по частоте (в убывании) и по алфавиту при
равенстве частоты

sorted_chars = sorted(char_count.items(), key=lambda x: (-x[1],
x[0]))



# Записываем результаты в файл

with open("statistik.txt", "w", encoding="utf-8") as file:

    for char, freq in sorted_chars:

        file.write(f"{char}: {freq}\n")
```