Знакомство с языком Python (семинары)

Задание 1. Гласные буквы

Команде лингвистов понравилось качество ваших программ, поэтому они решили заказать функцию для анализатора текста, которая создавала бы список гласных букв в нём и считала бы их количество.

Напишите программу, которая запрашивает у пользователя текст и генерирует список гласных букв этого материала (сама строка вводится на русском языке). Выведите в консоль сам список и его длину.

Пример:

Введите текст: Нужно отнести кольцо в Мордор!

Список гласных букв: ['у', 'о', 'о', 'е', 'и', 'о', 'о', 'о', 'о']

Длина списка: 9

Подсказка № 1

Для начала определите, какие буквы являются гласными. Это можно сделать, создав строку с гласными буквами русского алфавита: `'ауоыиэяюёе'`.

Подсказка № 2

Создайте список `vowels`, который будет содержать только гласные буквы из введённого текста. Для этого используйте генератор списка (list comprehension), проверяя, входит ли каждая буква текста в строку с гласными.

Подсказка № 3

Для проверки на принадлежность буквы к гласным используйте оператор `in`. Например, `i in 'ауоыиэяюёе'` вернёт `True`, если `i` — гласная буква, и `False` в противном случае.

Эталонное решение:

```
# Запрашиваем у пользователя ввод текста

text = input('Введите текст: ')



# Создаём список гласных букв, которые есть в тексте
```

```
# Используем генератор списка для фильтрации гласных

vowels = [i for i in text if i in 'ауоыиэяюёе']



# Выводим список найденных гласных

print('Список гласных букв:', vowels)



# Выводим количество гласных в списке

print('Длина списка:', len(vowels))
```

Задача 2. Случайные соревнования

Мы хотим протестировать работу электронной таблицы для участников некоторых соревнований. Есть два списка, то есть две команды, по 20 участников в каждом. В них хранятся очки каждого участника — вещественные числа с двумя знаками после точки, например 4.03.

Член одной команды соревнуется с участником другой команды под таким же номером. То есть первый соревнуется с первым, второй — со вторым и так далее.

Напишите программу, которая генерирует два списка участников (по 20 элементов) из случайных вещественных чисел (от 5 до 10). Для этого найдите подходящую функцию из модуля random. Затем сгенерируйте третий список, в котором окажутся только победители из каждой пары.

Пример:

Первая команда: [7.86, 6.76, 9.97, 9.08, 5.45, 6.9, 8.65, 5.17, 8.17, 5.06, 7.56, 7.1, 7.18, 8.25, 5.53, 7.95, 8.91, 7.11, 8.29, 9.52]

Вторая команда: [7.13, 5.7, 8.89, 5.36, 5.62, 9.46, 5.82, 8.67, 8.41, 7.0, 5.31, 7.8, 9.93, 7.76, 7.4, 8.26, 7.94, 5.71, 7.89, 7.77]

Победители тура: [7.86, 6.76, 9.97, 9.08, 5.62, 9.46, 8.65, 8.67, 8.41, 7.0, 7.56, 7.8, 9.93, 8.25, 7.4, 8.26, 8.91, 7.11, 8.29, 9.52]

Подсказка № 1

Для генерации случайных чисел с плавающей запятой в определённом диапазоне используйте функцию `random.uniform(a, b)`, где `a` и `b` — границы диапазона. В данном случае `a = 5`, `b = 10`.

Подсказка № 2

Для округления числа до двух знаков после запятой используйте функцию `round()`. Например, `round(random.uniform(5, 10), 2)`.

Подсказка № 3

Для создания списка используйте генератор списков. Например, `[round(random.uniform(5, 10), 2) for _ in range(20)]` создаст список из 20 случайных вещественных чисел.

Подсказка № 4

Для определения победителя в каждой паре сравните соответствующие элементы двух списков. Используйте условный оператор внутри генератора списков: `first_team[i] if first_team[i] > second_team[i] else second_team[i]`.

Эталонное решение:

```
import random



# Генерация первой команды из 20 случайных вещественных чисел в
диапазоне от 5 до 10

first_team = [round(random.uniform(5, 10), 2) for _ in range(20)]



# Генерация второй команды аналогично первой

second_team = [round(random.uniform(5, 10), 2) for _ in range(20)]



# Определение победителей тура путем сравнения значений в первой и
второй командах

winners = [

    first_team[i_player] if first_team[i_player] >
second_team[i_player]

    else second_team[i_player]
```

```
    for i_player in range(20)

]



# Вывод данных

print('Первая команда:', first_team)

print('Вторая команда:', second_team)

print('Победители тура:', winners)
```

Задача 3. Двумерный список

Часто в программировании приходится писать код исходя из результата, который требует заказчик. В этот раз ему нужно получить двумерный список:

[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]

Напишите программу, которая генерирует такой список и выводит его на экран. Используйте только list comprehensions.

Подсказка № 1

Используйте генераторы списков (list comprehensions) для создания двумерных списков.

Подсказка № 2

Внутри генератора списка используйте функцию `range()` с нужными параметрами. В данном случае `range(1, 5)` задает первый элемент каждой строки, а шаг в 4 единицы позволяет получить последующие элементы строки.

Подсказка № 3

Для формирования каждого вложенного списка используйте выражение вида `[начальное_значение, начальное_значение + 4, начальное_значение + 8]`. Это позволяет получать элементы, которые находятся в фиксированном шаге друг от друга.

Эталонное решение:

```
# Создание двумерного списка с помощью list comprehension

# Внешний цикл по range(1, 5) определяет первый элемент каждой
вложенной строки

# Внутренний цикл формирует элементы каждой строки с шагом 4
```

```
my_list = [[j_num for j_num in range(i_list, 13, 4)] for i_list in
range(1, 5)]

print(my_list)



# Альтернативный вариант решения с использованием фиксированных
смещений

second_answer = [[value, value + 4, value + 8] for value in range(1,
5)]

print(second_answer)
```

Задача 4. Список списков

Дан многомерный список:

`nice_list = [[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 11, 12], [13, 14, 15], [16, 17, 18]]]`

Напишите код, который раскрывает все вложенные списки, то есть оставляет лишь внешний список. Для решения используйте только list comprehensions.

Ответ: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

Подсказка № 1

В каждом уровне вложенности используйте генератор для получения элементов следующего уровня. Например, `for each_list in nice_list` и `for each_list_2 in each_list` позволяют сначала пройти по верхнему списку, затем по среднему, и наконец, получить элементы из вложенных списков.

Подсказка № 2

Для доступа к элементам из самого внутреннего уровня используйте последний цикл, который итерируется по числам в списках: `for digit in each_list_2`. Это позволяет получить конечные значения из вложенных структур.

Подсказка № 3

Обратите внимание на синтаксис: все вложенные генераторы должны быть записаны в одну строку, разделенные пробелами, а не запятыми.

Эталонное решение:

```
# Исходный многомерный список
```

```
nice_list = [[[1, 2, 3], [4, 5, 6], [7, 8, 9]],

             [[10, 11, 12], [13, 14, 15], [16, 17, 18]]]



# Раскрытие всех вложенных списков с использованием list
comprehension

output = [digit

          for each_list in nice_list

          for each_list_2 in each_list

          for digit in each_list_2]



# Вывод результата

print(output)
```

Задача 5. Шифр Цезаря

Юлий Цезарь использовал свой способ шифрования текста. Каждая буква заменялась на следующую по алфавиту через K позиций по кругу. Если взять русский алфавит и K, равное 3, то в слове, которое мы хотим зашифровать, буква А станет буквой Г, Б станет Д и так далее.

Пользователь вводит сообщение и значение сдвига. Напишите программу, которая изменит фразу при помощи шифра Цезаря.

Пример:

Введите сообщение: это питон.

Введите сдвиг: 3

Зашифрованное сообщение: ахс тлхср.

Подсказка № 1

Перед началом работы убедитесь, что алфавит включает все буквы, включая букву 'ё', чтобы избежать ошибок в шифровании.

Подсказка № 2

При определении индекса нового символа используйте операцию взятия остатка (`%`), чтобы избежать выхода за пределы длины алфавита.

Подсказка № 3

Учтите, что пробелы и другие символы, не входящие в алфавит, должны оставаться без изменений. Это можно сделать, проверяя наличие символа в строке `alphabet`.

Подсказка № 4

Используйте функцию join() для преобразования списка зашифрованных символов в единую строку, что позволяет избежать использования дополнительного цикла для создания строки.

Эталонное решение:

```
# Определение русского алфавита, включая букву 'ё'
alphabet = 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'


# Функция для шифрования текста по методу Цезаря
def caesar_cipher(string, user_shift):
    # Создание списка зашифрованных символов
    char_list = [(alphabet[(alphabet.index(sym) + user_shift) %
len(alphabet)]
                  if sym in alphabet else sym)
                 for sym in string]

    # Преобразование списка символов в строку
    new_str = ''.join(char_list)
    return new_str


# Ввод пользователем исходного сообщения и сдвига
input_str = input('Введите сообщение: ')
shift = int(input('Введите сдвиг: '))
```

```
# Шифрование сообщения

output_str = caesar_cipher(input_str, shift)



# Вывод зашифрованного сообщения

print('Зашифрованное сообщение:', output_str)
```