Знакомство с языком Python (семинары)

Задание 1: Поставьте оценку!

Вася выложил своё новое приложение на платформу для начинающих разработчиков, на ней пользователи могут ставить оценку приложению: от −100 до 100. Ему захотелось сравнить количество положительных и отрицательных отзывов, для этого он заранее отфильтровал все отзывы так, чтобы в конце были те, которые равны нулю.

Напишите программу, которая находит количество положительных и количество отрицательных чисел в последовательности. Последовательность заканчивается на числе 0.

Пример

Введите число: −4

Введите число: −90

Введите число: 6

Введите число: 0

Кол-во положительных чисел: 1

Кол-во отрицательных чисел: 2

Подсказка № 1

Для решения задачи используйте цикл `while`. Он должен работать до тех пор, пока вводимое число не будет равно 0. Это число будет служить сигналом к завершению ввода и завершению программы.

Подсказка № 2

Инициализируйте две переменные для подсчета положительных и отрицательных чисел. Начальные значения этих переменных должны быть равны нулю.

Подсказка № 3

Внутри цикла `while` проверяйте введенное число. Если число больше нуля, увеличьте счетчик положительных чисел. Если меньше нуля — увеличьте счетчик отрицательных чисел. Число 0 не учитывается ни в одном из счетчиков.

Подсказка № 4

Не забудьте обновлять переменную `rating` внутри цикла. Это нужно для того, чтобы пользователь мог вводить новые значения чисел на каждой итерации цикла.

Эталонное решение:

```
# Ввод первого числа
rating = int(input('Введите число:'))
# Инициализация счетчиков положительных и отрицательных чисел
positive_count = 0
negative_count = 0

# Цикл продолжается до тех пор, пока не введено число 0
while rating != 0:
    # Увеличиваем счетчик положительных чисел, если число больше 0
    if rating > 0:
        positive_count += 1
    # Увеличиваем счетчик отрицательных чисел, если число меньше 0
    else:
        negative_count += 1
    # Ввод следующего числа
    rating = int(input('Введите число:'))

# Вывод количества положительных и отрицательных чисел
print('Кол-во положительных чисел:', positive_count)
print('Кол-во отрицательных чисел:', negative_count)
```

Задача 2. Обычный день на работе

Максим программирует целый день на работе и вечером идёт домой. Каждый час начальство кидает ему несколько задач, которые нужно решить до следующего

рабочего часа. Вдобавок каждый час Максиму звонит супруга. Он знает, что если он возьмёт трубку, то жена попросит зайти вечером в магазин.

Напишите программу, в которой считается, сколько задач выполнил Максим за день (восемь часов). Если он хотя бы раз взял трубку, то в конце дополнительно выводите сообщение: «Нужно зайти в магазин».

Пример

Начался восьмичасовой рабочий день.

1-й час

Сколько задач решит Максим? 1

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

2-й час

Сколько задач решит Максим? 2

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

3-й час

Сколько задач решит Максим? 3

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

4-й час

Сколько задач решит Максим? 4

Звонит жена. Взять трубку? (1 — да, 0 — нет): 1

5-й час

Сколько задач решит Максим? 5

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

6-й час

Сколько задач решит Максим? 1

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

7-й час

Сколько задач решит Максим? 2

Звонит жена. Взять трубку? (1 — да, 0 — нет): 1

8-й час

Сколько задач решит Максим? 3

Звонит жена. Взять трубку? (1 — да, 0 — нет): 0

Рабочий день закончился. Всего выполнено задач: 21

Нужно зайти в магазин.

Подсказка № 1

Используйте цикл `while`, чтобы организовать рабочий день, который длится 8 часов. Внутри цикла будет происходить учет выполненных задач и проверка звонков от жены.

Подсказка № 2

Создайте переменную для подсчета общего количества выполненных задач (`tasks_sum`) и увеличивайте её на значение, введенное пользователем каждый час. Это значение должно быть целым числом.

Подсказка № 3

Используйте переменную-флаг (`go_to_store`), чтобы отслеживать, взял ли Максим трубку хотя бы один раз за день. Если флаг установлен в `True`, то в конце рабочего дня программа должна вывести сообщение о необходимости зайти в магазин.

Подсказка № 4

Для ввода данных используйте функцию `input()` и не забудьте преобразовать входные данные в целочисленный тип с помощью `int()`. Это нужно как для количества выполненных задач, так и для решения о том, брать ли трубку.

Эталонное решение:

```
hour = 0  # Счетчик часов рабочего дня

tasks_sum = 0  # Суммарное количество выполненных задач

go_to_store = False  # Флаг, указывающий на необходимость зайти в
магазин



print('Начался 8-часовой рабочий день')
```

```
while hour != 8:  # Цикл на 8 часов

    hour += 1  # Увеличение счетчика часов

    print(hour, 'час')



    # Ввод количества задач, решенных в текущий час

    solved_tasks = int(input('Сколько задач решит Максим? '))

    tasks_sum += solved_tasks  # Увеличение общего числа задач



    # Ввод решения о том, брать ли трубку, если звонит жена

    call = int(input('Звонит жена. Взять трубку? (1 — да, 0 — нет): 
'))

    if call == 1:

        go_to_store = True  # Если ответ "да", устанавливаем флаг 
для захода в магазин



# Вывод результатов после окончания рабочего дня

print('Рабочий день закончился. Всего выполнено задач:', tasks_sum)

if go_to_store:

    print('Нужно зайти в магазин')  # Вывод сообщения о 
необходимости зайти в магазин
```

Задача 3. Игра «Угадай число»

Папа-программист написал для сына программу, которая просит его угадать число. Недостаток программы был в том, что бедному сыну приходилось её каждый раз перезапускать, чтобы угадывать число. Теперь, когда мы знаем гораздо больше, пришло время исправить этот недостаток и заодно немного улучшить саму игру.

Напишите программу-игру, которая запрашивает у пользователя число до тех пор, пока он его не отгадает. Выводите сообщения в соответствии с примером.

Пример (загадали число 7)

Введите число: 3

Число меньше, чем нужно. Попробуйте ещё раз!

Введите число: 10

Число больше, чем нужно. Попробуйте ещё раз!

Введите число: 8

Число больше, чем нужно. Попробуйте ещё раз!

Введите число: 7

Вы угадали! Число попыток: 4

Подсказка № 1

Используйте бесконечный цикл `while True`, чтобы повторно запрашивать ввод числа у пользователя до тех пор, пока он не угадает загаданное число. Цикл должен завершаться только при правильном ответе.

Подсказка № 2

Создайте переменную для хранения загаданного числа. Это число нужно угадать. Также создайте переменную для подсчета количества попыток угадывания (`guess_count`) и увеличивайте её при каждой новой попытке.

Подсказка № 3

Для ввода числа от пользователя используйте функцию `input()`, а затем преобразуйте введенное значение в целое число с помощью `int()`. Это число нужно сравнивать с загаданным числом.

Подсказка № 4

Используйте условные конструкции (`if`, `elif`, `else`) внутри цикла для проверки введенного числа. Если оно больше загаданного, выведите сообщение, что число слишком велико. Если оно меньше, выведите сообщение, что число слишком мало. Если угадали, выведите сообщение об успехе и количество попыток, после чего завершите цикл с помощью `break`.

Эталонное решение:

```
number = 7  # Загаданное число, которое нужно угадать

guess_count = 0  # Счетчик количества попыток
```

```
while True:  # Бесконечный цикл для повторного запроса числа у
пользователя

    guess_num = int(input('Введите число: '))  # Ввод числа
пользователем

    guess_count += 1  # Увеличиваем счетчик попыток



    # Проверка, больше ли введенное число, чем загаданное

    if guess_num > number:

        print('Число больше, чем нужно. Попробуйте ещё раз!')

    # Проверка, меньше ли введенное число, чем загаданное

    elif guess_num < number:

        print('Число меньше, чем нужно. Попробуйте ещё раз!')

    # Если число угадано правильно

    else:

        print('Вы угадали! Число попыток:', guess_count)

        break  # Прерываем цикл, так как число угадано
```

Задача 4. Посчитай чужую зарплату...

Бухгалтер устала постоянно считать вручную среднегодовую зарплату сотрудников компании и, чтобы облегчить себе жизнь, обратилась к программисту.

Напишите программу, которая принимает от пользователя зарплату сотрудника за каждый из 12 месяцев и выводит на экран среднюю зарплату за год.

Подсказка № 1

Для решения задачи используйте цикл `for`, который будет проходить по диапазону от 1 до 12. Этот цикл необходим для сбора данных о зарплате за каждый месяц года.

Подсказка № 2

Создайте переменную `salary_year` для хранения суммарной зарплаты за год. Изначально её значение должно быть равно нулю.

Подсказка № 3

Внутри цикла запрашивайте у пользователя ввод зарплаты за текущий месяц. Используйте функцию `input()` и преобразуйте введенное значение в целое число с помощью `int()`. После ввода зарплаты за месяц добавляйте её к общей сумме зарплат в переменной `salary_year`.

Подсказка № 4

После завершения цикла вычислите среднюю зарплату за год, разделив общую сумму на 12 (количество месяцев).

Эталонное решение:

```
salary_year = 0  # Инициализация переменной для хранения суммарной зарплаты за год



# Цикл для ввода зарплаты за каждый из 12 месяцев

for month in range(1, 13):

    # Ввод зарплаты за текущий месяц

    salary_month = int(input('Введите зарплату сотрудника за месяц {}: '.format(month)))

    # Добавление введенной зарплаты к общей сумме

    salary_year += salary_month



# Вычисление средней зарплаты за год

average_salary = salary_year / 12

# Вывод средней зарплаты

print('Средняя зарплата за год:', average_salary)
```

Задача 5. Пропавшая карточка

Для настольной игры используются карточки с номерами от 1 до N. Одна карточка потерялась. Напишите программу, которая поможет найти потерянную карточку, если номера оставшихся известны. Найдите её, зная номера оставшихся карточек.

Введите число карточек — N.

Затем введите N − 1 номера оставшихся карточек. Они могут быть введены в любом порядке.

Подсказка № 1

Используйте цикл `for` для расчета суммы всех карточек от 1 до `N`. Сумма этих чисел равна `(N * (N + 1)) / 2`. В этом случае цикл `for` используется для накопления суммы, что эквивалентно формуле.

Подсказка № 2

Создайте второй цикл для ввода номеров оставшихся карточек. Этот цикл должен выполняться `N - 1` раз, так как одна карточка потерялась.

Подсказка № 3

Внутри второго цикла вычитайте каждый введенный номер оставшейся карточки из общей суммы всех карточек. Это поможет найти разницу между полной суммой и суммой оставшихся карточек.

Подсказка № 4

После завершения цикла с оставшимися карточками значение переменной `total_sum` будет равно номеру потерянной карточки. Выведите это значение на экран.

Эталонное решение:

```
# Ввод общего количества карточек
total_cards = int(input('Введите кол-во карточек: '))

# Инициализация переменной для хранения суммы всех номеров карточек
total_sum = 0

# Вычисление суммы всех номеров карточек от 1 до N
for card in range(1, total_cards + 1):
    total_sum += card

# Вычисление суммы оставшихся карточек
```

```
for card in range(total_cards - 1):

    remaining_card = int(input('Номер оставшейся карты: '))

    total_sum -= remaining_card  # Вычитание номера оставшейся карты
из общей суммы



# Вывод номера потерянной карточки, который равен оставшейся сумме

print('Номер потерявшейся карты:', total_sum)
```

Задача 6 *. Кинотеатр

X мальчиков и Y девочек пошли в кинотеатр и купили билеты на идущие подряд места в одном ряду. Напишите программу, которая выдаст, как нужно сесть мальчикам и девочкам, чтобы рядом с каждым мальчиком сидела хотя бы одна девочка, а рядом с каждой девочкой — хотя бы один мальчик.

На вход подаются два числа: количество мальчиков X и количество девочек Y. В ответе выведите какую-нибудь строку, в которой будет ровно X символов B, обозначающих мальчиков, и Y символов G, обозначающих девочек, удовлетворяющую условию задачи. Пробелы между символами выводить не нужно. Если рассадить мальчиков и девочек согласно условию задачи невозможно, выведите строку «Нет решения».

Подсказка № 1

Проверьте, можно ли разместить мальчиков и девочек так, чтобы удовлетворять условиям задачи. Если количество мальчиков или девочек больше чем в два раза больше другого типа, то корректное размещение невозможно. Выведите "Нет решения" в этом случае.

Подсказка № 2

Если мальчиков больше или их количество равно количеству девочек, определите, сколько мальчиков нужно поместить между девочками так, чтобы обеспечить правильное чередование. Используйте переменную k для подсчета лишних мальчиков.

Подсказка № 3

Для ситуации, когда мальчиков больше или их количество равно количеству девочек, создайте строку, чередующую мальчиков и девочек с учетом лишних мальчиков. Используйте цикл для вставки лишних мальчиков между девочками.

Подсказка № 4

Используйте два цикла для создания строки с чередованием мальчиков и девочек. Первый цикл используется для вставки лишних символов, а второй для добавления оставшихся символов.

Эталонное решение:

```
# Ввод количества мальчиков и девочек

boys = int(input('Введите кол-во мальчиков: '))

girls = int(input('Введите кол-во девочек: '))



# Инициализация пустой строки для результата

answer = ''



# Проверка возможности корректного распределения мальчиков и девочек

if (boys > 2 * girls) or (girls > 2 * boys):

    # Если мальчиков или девочек слишком много для корректного
чередования

    print('Нет решения')

elif boys >= girls:

    # Если мальчиков больше или их количество равно количеству
девочек

    k = boys - girls  # Количество лишних мальчиков

    # Вставка лишних мальчиков между девочками

    for bgb in range(k):

        answer += 'BGB'

    # Добавление оставшихся мальчиков и девочек

    for bg in range(girls - k):

        answer += 'BG'

else:
```

```
    # Если девочек больше, чем мальчиков
    k = girls - boys  # Количество лишних девочек
    # Вставка лишних девочек между мальчиками
    for gbg in range(k):
        answer += 'GBG'
    # Добавление оставшихся мальчиков и девочек
    for gb in range(boys - k):
        answer += 'GB'


# Вывод ответа
print(answer)
```