Знакомство с языком Python (семинары)

Задание 1: Видеокарты

В базе магазина электроники есть список видеокарт компании NVIDIA разных поколений. Вместо полных названий хранятся только числа, которые обозначают модель и поколение видеокарты. Недавно компания выпустила новую линейку видеокарт. Самые старшие поколения разобрали за пару дней.

Напишите программу, которая удаляет наибольшие элементы из списка видеокарт.

Пример:

`Количество видеокарт: 5`

`Видеокарта 1: 3070`

`Видеокарта 2: 2060`

`Видеокарта 3: 3090`

`Видеокарта 4: 3070`

`Видеокарта 5: 3090`

`Старый список видеокарт: [ 3070 2060 3090 3070 3090 ]`

`Новый список видеокарт: [ 3070 2060 3070 ]`

Подсказка № 1

Для начала получите количество видеокарт от пользователя и создайте пустые списки для хранения введённых видеокарт и нового списка без наибольших элементов.

Подсказка № 2

Используйте цикл для ввода номеров видеокарт. Во время ввода также определите наибольшую видеокарту, обновляя переменную `maxItem` при каждом вводе, если текущий элемент больше предыдущего максимума.

Подсказка № 3

После ввода всех видеокарт, пройдитесь по списку и создайте новый список, в который добавляйте только те видеокарты, которые не равны максимальному значению, найденному ранее.

Подсказка № 4

Для вывода старого списка видеокарт, используйте цикл, чтобы пройти по всем элементам списка и распечатать их в требуемом формате. Для вывода нового списка видеокарт, сделайте то же самое, что и для старого списка, но с новым списком, который не содержит наибольших значений. Убедитесь, что формат вывода совпадает с требуемым.

Эталонное решение:

```
# Ввод количества видеокарт
videoCardsNumber = int(input('Введите количество видеокарт: '))


# Инициализация списка для хранения видеокарт
videoCards = []
newVideoCardsList = []
maxItem = 0


# Заполнение списка видеокарт и определение наибольшего элемента
for item in range(videoCardsNumber):
    # Ввод номера видеокарты и добавление в список
    videoCards.append(int(input(f'Видеокарта {item + 1}: ')))

    # Обновление значения максимального элемента
    if videoCards[item] > maxItem:
        maxItem = videoCards[item]


# Формирование нового списка без наибольших элементов
for item in range(videoCardsNumber):
    if videoCards[item] != maxItem:
        newVideoCardsList.append(videoCards[item])
```

```
# Вывод старого списка видеокарт

print()

print('Старый список видеокарт: [', end=' ')

for item in range(videoCardsNumber):

    print(videoCards[item], end=' ')

print(']')



# Вывод нового списка видеокарт

print('Новый список видеокарт: [', end=' ')

for item in range(len(newVideoCardsList)):

    print(newVideoCardsList[item], end=' ')

print(']')
```

Задача 2. Кино

Илья зашёл на любительский киносайт, на котором пользователи оставляют рецензии на фильмы. Их список:

films = [‘Крепкий орешек’, ‘Назад в будущее’, ‘Таксист’, ‘Леон’, ‘Богемская рапсодия’, ‘Город грехов’, ‘Мементо’, ‘Отступники’, ‘Деревня’]

Илья на сайте в первый раз. Он хочет зарегистрироваться и сразу добавить часть фильмов в список любимых, чтобы позже прочитать рецензии на них.

Напишите программу, в которой пользователь вводит фильм. Если кинокартина есть в перечне, то добавляется в список любимых. Если её нет, то выводится ошибка. В конце выведите весь список любимых фильмов.

Пример:

Сколько фильмов хотите добавить? 3

Введите название фильма: Леон

Введите название фильма: Безумный Макс

Ошибка: фильма Безумный Макс у нас нет :(

Введите название фильма: Мементо

Ваш список любимых фильмов: Леон, Мементо

Подсказка № 1

Создайте список доступных фильмов (`films`), который содержит все фильмы, которые могут быть добавлены в список любимых. Также инициализируйте пустой список для хранения любимых фильмов (`my_list`).

Подсказка № 2

Используйте цикл, чтобы многократно запрашивать у пользователя название фильма. Внутри цикла вводите название фильма и сохраняйте его в переменной `movie`.

Подсказка № 3

Проверьте, содержится ли введённый фильм в списке доступных фильмов (`films`). Если фильм присутствует в списке, добавьте его в список любимых фильмов (`my_list`). Если фильма нет в списке, выведите сообщение об ошибке, указывая, что такого фильма нет.

Эталонное решение:

```
# Список доступных фильмов
films = ['Крепкий орешек', 'Назад в будущее', 'Таксист',
         'Леон', 'Богемская рапсодия', 'Город грехов',
         'Мементо', 'Отступники', 'Деревня']

# Инициализация пустого списка для любимых фильмов
my_list = []

# Ввод количества фильмов, которые пользователь хочет добавить
movies_count = int(input('Сколько фильмов хотите добавить? '))

# Цикл для ввода названий фильмов
```

```
for _ in range(movies_count):

    # Ввод названия фильма

    movie = input('Введите название фильма: ')



    # Проверка наличия фильма в списке доступных фильмов

    if movie in films:

        # Добавление фильма в список любимых, если он есть в списке

        my_list.append(movie)

    else:

        # Вывод сообщения об ошибке, если фильма нет в списке

        print(f'Ошибка: фильма {movie} у нас нет :(')



# Вывод списка любимых фильмов

print(f'\nВаш список любимых фильмов: {my_list}')
```

Задача 3. Сортировка

Дан список из N чисел. Напишите программу, которая сортирует элементы списка по возрастанию и выводит их на экран. Дополнительный список **использовать нельзя**.

Также нельзя использовать готовые функции sorted/min/max и метод sort

Постарайтесь придумать и написать как можно более эффективный алгоритм сортировки.

Пример:

Изначальный список: [1, 4, –3, 0, 10]

Отсортированный список: [–3, 0, 1, 4, 10]

Подсказка № 1

Используйте алгоритм сортировки пузырьком (Bubble Sort). В этом алгоритме два вложенных цикла сравнивают элементы списка и меняют их местами, если они находятся в неправильном порядке.

Подсказка № 2

Во внешнем цикле проходите по всем элементам списка, за исключением последнего элемента на каждом шаге, так как он уже будет на своем месте. Внутренний цикл сравнивает и меняет местами соседние элементы списка.

Подсказка № 3

Во внутреннем цикле сравнивайте текущий элемент с следующим. Если текущий элемент больше следующего, меняйте их местами. Это приведет к тому, что на каждом проходе элементы с наибольшим значением будут перемещаться в конец списка.

Эталонное решение:

```
# Исходный список чисел

original_list = [1, 4, -3, 0, 10]



# Вывод исходного списка

print('Изначальный список: ', original_list)



# Реализация сортировки пузырьком (Bubble Sort)

for i in range(len(original_list) - 1):

    # Проход по всем элементам списка, за исключением уже
отсортированных

    for j in range(len(original_list) - 1 - i):

        # Сравнение текущего элемента с следующим

        if original_list[j] > original_list[j + 1]:

            # Если текущий элемент больше следующего, меняем их
местами

            original_list[j], original_list[j + 1] = original_list[j
+ 1], original_list[j]



# Вывод отсортированного списка
```

```
print('Отсортированный список: ', original_list)
```

Задача 4. Товары

В базе данных магазина вся необходимая информация по товарам делится на два словаря: первый отвечает за коды товаров, второй — за списки количества разнообразных товаров на складе:

```
goods = {
'Лампа': '12345',
'Стол': '23456',
'Диван': '34567',
'Стул': '45678',
}
store = {
'12345': [
{'quantity': 27, 'price': 42},
],
'23456': [
{'quantity': 22, 'price': 510},
{'quantity': 32, 'price': 520},
],
'34567': [
{'quantity': 2, 'price': 1200},
{'quantity': 1, 'price': 1150},
],
'45678': [
{'quantity': 50, 'price': 100},
```

```
{'quantity': 12, 'price': 95},
{'quantity': 43, 'price': 97},
],
}
```

Каждая запись второго словаря отображает, сколько и по какой цене закупалось товаров. Цена указана за одну штуку.

Напишите программу, которая рассчитывает общую стоимость позиций для каждого товара на складе и выводит эту информацию на экран.

Результат работы программы:

```
Лампа — 27 штук, стоимость 1134 рубля.
Стол — 54 штуки, стоимость 27 860 рублей.
Диван — 3 штуки, стоимость 3550 рублей.
Стул — 105 штук, стоимость 10 311 рублей.
```

Подсказка № 1

В цикле по `goods.keys()` получайте код товара. Используйте этот код для поиска информации о товаре в словаре `store`. Это поможет вам получить все необходимые данные о количестве и цене.

Подсказка № 2

Инициализируйте переменные `total_quantity` и `total_cost` перед началом цикла для подсчета общей информации по каждому товару. Это необходимо для накопления данных по каждому товару отдельно.

Подсказка № 3

Внутри цикла по `store[item_code]`, где `item_code` — код товара, проходите по каждому элементу списка и обновляйте переменные `total_quantity` и `total_cost`. Убедитесь, что вы правильно добавляете количество и рассчитываете стоимость.

Эталонное решение:

```
# Словарь с названиями товаров и их кодами
```

```
goods = {
    'Лампа': '12345',
    'Стол': '23456',
    'Диван': '34567',
    'Стул': '45678',
}

# Словарь с информацией о товарах на складе
store = {
    '12345': [
        {'quantity': 27, 'price': 42},
    ],
    '23456': [
        {'quantity': 22, 'price': 510},
        {'quantity': 32, 'price': 520},
    ],
    '34567': [
        {'quantity': 2, 'price': 1200},
        {'quantity': 1, 'price': 1150},
    ],
    '45678': [
        {'quantity': 50, 'price': 100},
        {'quantity': 12, 'price': 95},
        {'quantity': 43, 'price': 97},
    ],
}
```

```
# Проходим по всем товарам в словаре goods

for item_name in goods.keys():

    # Получаем код товара из словаря goods

    item_code = goods[item_name]



    # Инициализируем переменные для подсчета общего количества и
стоимости

    total_quantity = 0

    total_cost = 0



    # Проходим по всем записям для данного товара в словаре store

    for entry in store[item_code]:

        # Увеличиваем общее количество товара

        total_quantity += entry['quantity']

        # Увеличиваем общую стоимость товара

        total_cost += entry['price'] * entry['quantity']



    # Выводим информацию о товаре

    print('{} — {} штук, стоимость {} рубля(ей).'.format(item_name,
total_quantity, total_cost))
```

Задача 5. Пицца

В базе данных интернет-магазина PizzaTime хранятся сведения о том, кто, что и сколько заказывал у них в определённый период. Вам нужно структурировать эту информацию и определить, сколько всего пицц купил каждый заказчик.

На вход в программу подаётся N заказов. Каждый заказ представляет собой строку вида «Покупатель — название пиццы — количество заказанных пицц». Реализуйте код, который выводит список покупателей и их заказов по

алфавиту. Учитывайте, что один человек может заказать одну и ту же пиццу несколько раз.

Пример

Введите количество заказов: 6

Первый заказ: Иванов Пепперони 1

Второй заказ: Петров Де-Люкс 2

Третий заказ: Иванов Мясная 3

Четвёртый заказ: Иванов Мексиканская 2

Пятый заказ: Иванов Пепперони 2

Шестой заказ: Петров Интересная 5

Иванов:

Мексиканская: 2

Мясная: 3

Пепперони: 3

Петров:

Де-Люкс: 2

Интересная: 5

Подсказка № 1

Для обработки каждого заказа используйте метод `split()` для разделения строки на компоненты: покупателя, название пиццы и количество. Преобразуйте количество из строки в целое число для последующих операций.

Подсказка № 2

Используйте вложенные словари для хранения данных. Внешний словарь будет содержать покупателя как ключ и внутренний словарь как значение. Внутренний словарь будет хранить названия пицц и их количества.

Подсказка № 3

Если покупатель уже существует в словаре, проверьте, есть ли в его списке пицца, которую он заказал. Если пицца уже есть, увеличьте количество. Если нет — добавьте новую запись.

Эталонное решение:

```
# Ввод количества заказов

orders_count = int(input('Введите количество заказов: '))



# Создаем пустой словарь для хранения данных о заказах

database = dict()



# Обрабатываем каждый заказ

for i_order in range(orders_count):

    # Вводим заказ и разбиваем его на части

    customer, pizza_name, count = input('{} заказ: '.format(i_order
+ 1)).split()



    # Преобразуем количество в целое число

    count = int(count)



    # Если покупатель еще не добавлен в словарь

    if customer not in database:

        # Добавляем покупателя и начальную запись о пицце

        database[customer] = {pizza_name: count}

    else:

        # Если покупатель уже есть

        if pizza_name in database[customer]:

            # Если пицца уже была заказана ранее, увеличиваем
количество
```

```
            database[customer][pizza_name] += count

        else:

            # Если пицца новая для этого покупателя, добавляем
запись

            database[customer][pizza_name] = count



# Сортируем список покупателей в алфавитном порядке и выводим
информацию

for customer in sorted(database.keys()):

    print('{}:'.format(customer))

    # Сортируем пиццы по алфавиту и выводим информацию

    for pizza_name in sorted(database[customer].keys()):

        print('     {}: {}'.format(pizza_name,
database[customer][pizza_name]))
```