|  |
| --- |
|  |
| Задачник по TensorFlow |
|  |

|  |
| --- |
|  |

Оглавление

[Тензоры 3](#_Toc170581205)

[Задание 1 3](#_Toc170581206)

[Задание 2 3](#_Toc170581207)

[Задание 3 3](#_Toc170581208)

[Задание 4 4](#_Toc170581209)

[Задание 5 4](#_Toc170581210)

[Задание 6 4](#_Toc170581211)

[Переменные тензора 5](#_Toc170581212)

[Задание 1 5](#_Toc170581213)

[Операции над тензорами 6](#_Toc170581214)

[Задание 1 6](#_Toc170581215)

[Задание 2 6](#_Toc170581216)

[Задание 3 6](#_Toc170581217)

[Задание 4 6](#_Toc170581218)

[Задание 5 7](#_Toc170581219)

[Индексирование с помощью фрагмента : 8](#_Toc170581220)

[Задание 1 8](#_Toc170581221)

[Многоосевое индексирование 9](#_Toc170581222)

[Задание 1 9](#_Toc170581223)

[Управление фигурами 10](#_Toc170581224)

[Задание 1 10](#_Toc170581225)

[Задание 2 10](#_Toc170581226)

[Задание 3 10](#_Toc170581227)

[DTypes 11](#_Toc170581228)

[Задание 1 11](#_Toc170581229)

[Вещание 12](#_Toc170581230)

[Задание 1 12](#_Toc170581231)

[Задание 2 12](#_Toc170581232)

[Задание 3 12](#_Toc170581233)

[Рваные тензоры 13](#_Toc170581234)

[Задание 1 13](#_Toc170581235)

[Тензоры струн (строк?) 14](#_Toc170581236)

[Задание 1 14](#_Toc170581237)

[Задание 2 14](#_Toc170581238)

[Задание 3 14](#_Toc170581239)

[Задание 4 14](#_Toc170581240)

[Задание 5 15](#_Toc170581241)

[Разреженные тензоры 16](#_Toc170581242)

[Задание 1 16](#_Toc170581243)

# Тензоры

**Normal tf.Tensor objects are immutable**

### Задание 1

Создать 2 тензора размерностью 2 на 2.

Назвать первый тензор именем first.

Второй тензор именем second.

Заполнить их значениями 1 2 3 4

Вывести в консоль тип данных и форму тензоров

### Задание 2

Создать 2 тензора размерностью 2 на 2.

Назвать первый тензор именем first\_int. Содержимое массива должно иметь тип int32

Второй тензор именем second\_float. Содержимое массива должно иметь тип float32

Заполнить их значениями 1 2 3 4

Вывести в консоль тип данных и форму тензоров

### Задание 3

Создать 3 одномерных тензора.

Заполнить их числами от 1 до 30, по 10 значений в каждом.

Назвать тензоры first second third .

first должен содержать значения от 1 до 10, second – 11-20 и так далее.

Вывести в консоль тип данных и форму тензоров

### Задание 4

Создать 3 одномерных тензора.

Заполнить их любым одним значением.

Вывести в консоль тип данных и форму тензора

### Задание 5

Создать тензор с целыми значениями, но при создании указать тип данных «число с плавающей точкой»

Вывести в консоль тип данных и форму тензора

### Задание 6

Создать трехмерный тензор, заполнить любыми значениями.

Вывести в консоль тип данных и форму тензора, сам тензор.

## Переменные тензора

### Задание 1

Создать любой тензор, заполнить любыми значениями.

Вывести в консоль:

* тип данных
* число осей
* форму
* количество элементов на каждой оси
* количество элементов на последней оси (предположим, что вы не знаете размерность тензора)
* общее число элементов в тензоре

## Операции над тензорами

### Задание 1

Создать любой тензор, заполнить любыми значениями.

Создать массив numpy на основе этого тензора.

Результат вывести в консоль.

Выполнить задание 2-мя способами.

### Задание 2

Создать 2 тензора.

Выполнить их сложение, поэлементное умножение и матричное умножение.

Результат вывести в консоль.

Выполнить задание 2-мя способами.

### Задание 3

Создать любой тензор, заполнить любыми значениями.

Найти наибольшее значение в тензоре.

Результат вывести в консоль.

### Задание 4

Создать любой тензор, заполнить любыми значениями.

Найти индекс наибольшего значения в тензоре.

Результат вывести в консоль.

### Задание 5

Создать любой тензор, заполнить любыми значениями.

Найти softmax.

Результат вывести в консоль.

## Индексирование с помощью фрагмента :

### Задание 1

Создать любой тензор, заполнить любыми значениями.

Используя индексирование с помощью фрагмента : вывести в консоль указанные ниже данные.

print("Everything:", ???)  
print("Before 4:", ???)  
print("From 4 to the end:", ???)  
print("From 2, before 7:", ???)

print("Every other item:", ???)  
print("Reversed:", ???)

## Многоосевое индексирование

### Задание 1

Создать любой тензор, заполнить любыми значениями.

Используя многоосевое индексирование вывести в консоль указанные ниже данные.

# Get row and column tensors  
print("Second row:", ???)  
print("Second column:", ???)  
print("Last row:", ???)

print("First item in last column:", ???)  
print("Skip the first row:", ???)

## Управление фигурами

### Задание 1

Создать одномерный тензор, заполнить любыми значениями.

Изменить форму тензора так, чтобы строка стала столбцом и наоборот.

Результат вывести в консоль.

### Задание 2

Создать 3-хмерный тензор с любыми значениями.

Сделать его одномерным.

Изменить формулу по шаблону a\*(b\*c).

Изменить формулу по шаблону (a\*b)\*c.

Результат вывести в консоль.

### Задание 3

Создать тензор формы 3\*2\*4\*5\*1 с любыми значениями.

Сделать его одномерным.

Изменить формулу по шаблону 3\*2\*4\*(5\*1 ).

Изменить формулу по шаблону 3\*2\*(4\*5\*1 ).

Результат вывести в консоль.

## DTypes

### Задание 1

Пока не решил делать из этого задание или нет

the\_f64\_tensor = tf.constant([2.2, 3.3, 4.4], dtype=tf.float64)  
the\_f16\_tensor = tf.cast(the\_f64\_tensor, dtype=tf.float16)  
# Now, cast to an uint8 and lose the decimal precision  
the\_u8\_tensor = tf.cast(the\_f16\_tensor, dtype=tf.uint8)  
print(the\_u8\_tensor)

## Вещание

### Задание 1

Дан тензор:

x = tf.constant([1, 2, 3])  
  
Увеличить все его значения на 5 операцией умножения.

Выполнить задание тремя способами вывести результат в консоль.

### Задание 2

Создать два одномерных тензора.

Размерность должна быть такой, чтобы при их умножении получился тензор размерностью 6\*4

Вывести результат в консоль.

### Задание 3

Создать любые два одномерных тензора таких, чтобы один тензор имел только одну строку, а другой - только один столбец.

Выполнить вещание таким образом, чтобы тензоры стали квадратными.

Вывести результат в консоль.

## Рваные тензоры

### Задание 1

ragged\_list = [

[0, 1, 2, 3],

[4, 5],

[6, 7, 8],

[9]]

ragged\_tensor = tf.ragged.constant(ragged\_list)

print(ragged\_tensor)

## Тензоры струн (строк?)

### Задание 1

# Tensors can be strings, too here is a scalar string.

scalar\_string\_tensor = tf.constant("Gray wolf")

print(scalar\_string\_tensor)

### Задание 2

# If you have three string tensors of different lengths, this is OK.

tensor\_of\_strings = tf.constant(["Gray wolf",

"Quick brown fox",

"Lazy dog"])

# Note that the shape is (3,). The string length is not included.

print(tensor\_of\_strings)

### Задание 3

# You can use split to split a string into a set of tensors  
print(tf.strings.split(scalar\_string\_tensor, sep=" "))

# ...but it turns into a `RaggedTensor` if you split up a tensor of strings,  
# as each string might be split into a different number of parts.  
print(tf.strings.split(tensor\_of\_strings))

### Задание 4

text = tf.constant("1 10 100")  
print(tf.strings.to\_number(tf.strings.split(text, " ")))

### Задание 5

byte\_strings = tf.strings.bytes\_split(tf.constant("Duck"))

byte\_ints = tf.io.decode\_raw(tf.constant("Duck"), tf.uint8)

print("Byte strings:", byte\_strings)

print("Bytes:", byte\_ints)

## Разреженные тензоры

### Задание 1

# Sparse tensors store values by index in a memory-efficient manner

sparse\_tensor = tf.sparse.SparseTensor(indices=[[0, 0], [1, 2]],

values=[1, 2],

dense\_shape=[3, 4])

print(sparse\_tensor, "\n")

# You can convert sparse tensors to dense

print(tf.sparse.to\_dense(sparse\_tensor))