Google Colab

Colaboratory, или сокращенно Colab - это продукт компании Google Research. Colab позволяет любому человеку писать и выполнять произвольный код на языке python через браузер и особенно хорошо подходит для машинного обучения, анализа данных и образования. С технической точки зрения, Colab — это размещенный сервис Jupyter notebook, который не требует настройки для использования, но при этом предоставляет бесплатный доступ к вычислительным ресурсам, включая GPU. Colab можно использовать бесплатно. Ресурсы Colab не гарантированы и не безграничны, а лимиты использования иногда колеблются. Это необходимо для того, чтобы Colab мог предоставлять ресурсы бесплатно.

Блокноты Colab хранятся в Google Drive или могут быть загружены из GitHub. Блокнотами Colab можно делиться так же, как и документами Google Docs или Sheets. Просто нажмите кнопку "Поделиться" в правом верхнем углу любого блокнота Colab. Или можно скачать в формате .ipynb или .py

Виртуальная машина, которую вы используете, включая любые пользовательские файлы и библиотеки, которые вы установили, не будет использоваться совместно. Поэтому хорошей идеей будет включить код для установки и загрузки любых пользовательских библиотек или файлов, которые нужны вашему ноутбуку.

Код выполняется на виртуальной машине, являющейся частной для вашего аккаунта. Виртуальные машины удаляются после некоторого простоя и имеют максимальное время жизни, установленное службой Colab.

В бесплатной версии Colab ноутбуки могут работать не более 12 часов, в зависимости от доступности и режима использования.

https://colab.research.google.com/notebooks/markdown_guide.ipynb (https://colab.research.google.com/notebooks/markdown_guide.ipynb) Руководство по разметке текста

Генераторы списков

Генераторы списков - удобный способ создания списков в Python.

Вывести список всех кубов ряда от 0 до 6.

In [1]:

```
example = [0, 1, 2, 3, 4, 5, 6]
qubes = [i ** 3 for i in example]
print(*qubes, '- кубы')
```

```
0 1 8 27 64 125 216 - кубы
```

Вывести список из i элементов равных i в ряду четных цифр от 1 до 7

In [2]:

```
ex = [1, 2, 3, 4, 5, 6, 7]
arr = [[i]*i for i in ex if i % 2 == 0]
print(arr)
print(len(arr))
```

```
[[2, 2], [4, 4, 4, 4], [6, 6, 6, 6, 6, 6]]
3
```

In [3]:

```
ex = [2, 3, 4, 5, 6, 5, 4, 3, 2, 3, 4, 5, 6]
arr = [[i]*i for i in ex if i % 2 == 0]
arr
```

Out[3]:

```
[[2, 2],
 [4, 4, 4, 4],
 [6, 6, 6, 6, 6, 6],
 [4, 4, 4, 4],
 [2, 2],
 [4, 4, 4, 4],
 [6, 6, 6, 6, 6, 6]]
```

In [4]:

```
ex = [2, 3, 4, 5, 6, 5, 4, 3, 2, 3, 4, 5, 6]
arr = [i ** 2 if i % 2 == 0 else i ** 3 for i in ex]
print(*arr)
```

```
4 27 16 125 36 125 16 27 4 27 16 125 36
```

Функции

Функция - это блок организованного, многократно используемого кода, который используется для выполнения конкретного задания. Функции в Python - объект, который принимает аргументы, производит обработку данных, возвращает значение. Функция может принимать произвольное количество аргументов.

Они все обладают общим свойством: они могут принимать параметры (ноль, один или несколько), и они могут возвращать значение (хотя могут и не возвращать).

Встроенные функции языка Python

Язык Python включает много уже определенных, то есть встроенных в него, функций.

int() - преобразование к целому числу.

In [5]:

```
int('3') # преобразование из типа str в тип int
```

Out[5]:

```
3
```

float() - преобразование к числу с плавающей точкой. Если аргумент не указан, возвращается 0.0.

In [6]:

```
float('3.5') # преобразование из типа str в тип float
```

Out[6]:

```
3.5
```

str() - строковое представление объекта.

In [7]:

```
str(123) # преобразование из типа int в тип str
```

Out[7]:

```
'123'
```

input() - Возвращает введенную пользователем с консоли строку.

In [8]:

```
x = input() # функция попросит пользователя ввести с клавиатуры и вернет это значение в
x (тип возвращаемого значения str)
type(x)

x = int(input()) # преобразование введенного с клавиатуры числа в тип str
```

```
12
3
```

len() - Возвращает число элементов в указанном объекте.

In [9]:

```
x = [5, 7, 8, 2, 5]
len(x) # вернет значение 5 (5 элементов в списке)

s = 'Hello'
len(s) ## вернет значение 5 (5 букв в слове)
```

Out[9]:

```
5
```

max() - функция используется для нахождения «максимального» значения в последовательности, итерируемом объекте и так далее. В параметрах можно менять способ вычисления максимального значения.

min() - функция используется для нахождения «минимального» значения в последовательности, итерируемом объекте и так далее. В параметрах можно менять способ вычисления минимального значения.

In [10]:

```
x = [5, 7, 8, 2, 5]
max(x) # вернет значение 8
min(x) # вернет значение 2

x = ["Яблоко", "Апельсин", "Автомобиль"]
max(x, key = len) # считает самое длинное слово (по длине) - вернет "Автомобиль"
#max(x) # вернет "Яблоко" (сравнение слов будет по лексиграфическому признаку)
```

Out[10]:

```
'Автомобиль'
```

reversed() - функция предоставляет простой и быстрый способ развернуть порядок элементов в последовательности. В качестве параметра она принимает валидную последовательность, например список, а возвращает итерируемый объект.

In [11]:

```
x = [3,4,5]
b = reversed(x)
list(b)
```

Out[11]:

```
[5, 4, 3]
```

set() - создает множество. Обычно в качестве аргументов функция принимает последовательность, например строка или список, которая затем преобразуется в множество уникальных значений.

In [12]:

```
set("Hello")
# {'e', 'l', 'o', 'H'}

set((1,1,1,2,2,3,4,5))
# {1, 2, 3, 4, 5}
```

Out[12]:

```
{1, 2, 3, 4, 5}
```

range() - Используется для создания последовательности чисел с заданными значениями от и до, а также интервалом. Такая последовательность часто используется в циклах, особенно в цикле for.

In [13]:

```
# range(start, stop, step)
print(list(range(10,20,2)))
# вернет список [10, 12, 14, 16, 18]
```

```
[10, 12, 14, 16, 18]
```

enumerate() - В качестве параметра эта функция принимает последовательность. После этого она перебирает каждый элемент и возвращает его вместе со счетчиком в виде перечисляемого объекта. Основная особенность таких объектов — возможность размещать их в цикле для перебора.

Если range() позволяет получить только индексы элементов списка, то enumerate() – сразу индекс элемента и его значение.

In [14]:

```
x = "Строка"
print(list(enumerate(x)))
# [(0, 'С'), (1, 'т'), (2, 'р'), (3, 'о'), (4, 'к'), (5, 'а')]
```

```
[(0, 'С'), (1, 'т'), (2, 'р'), (3, 'о'), (4, 'к'), (5, 'а')]
```

In [15]:

```
import pandas as pd
pd.read_csv("/content/sample_data/california_housing_test.csv")
```

Out[15]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	hou
0	-122.05	37.37	27.0	3885.0	661.0	1537.0	
1	-118.30	34.26	43.0	1510.0	310.0	809.0	
2	-117.81	33.78	27.0	3589.0	507.0	1484.0	
3	-118.36	33.82	28.0	67.0	15.0	49.0	
4	-119.67	36.33	19.0	1241.0	244.0	850.0	
...	...	...	...	...	...	...	
2995	-119.86	34.42	23.0	1450.0	642.0	1258.0	
2996	-118.14	34.06	27.0	5257.0	1082.0	3496.0	
2997	-119.70	36.30	10.0	956.0	201.0	693.0	
2998	-117.12	34.10	40.0	96.0	14.0	46.0	
2999	-119.63	34.42	42.0	1765.0	263.0	753.0	

3000 rows × 9 columns

Определение собственных функций

Также в Python можно определить свою функцию. Функция определяется с помощью инструкции **def**.

Существуют некоторые правила для создания функций в Python.

- Блок функции начинается с ключевого слова def, после которого следуют название функции и круглые скобки ().
- Любые аргументы, которые принимает функция, должны находиться внутри этих скобок.
- После скобок идет двоеточие (:) и с новой строки с отступом начинается тело функции.

Выражение **return** прекращает выполнение функции и возвращает указанное после выражения значение. Выражение return без аргументов это то же самое, что и выражение return None.

```
def <Название функции>(параметр1, параметр2, ...):
    --код--
    return параметр1 + параметр2
```

Например, определим простейшую функцию, которая принимает в качестве аргументов два числа и возвращает их сумму.

- аргумент (фактический параметр): фактическая переменная передается в функцию;
- параметр (формальный параметр): принимающая переменная, которая используется в функции.

In [16]:

```
# x, y - это параметры функции
# функция возвращает значение переменной s
def summa(x, y):
    s = x + y
    return s
```

In [17]:

```
# x, y - это параметры функции
# данная функция ничего не возвращает, но выводит на экран сумму двух чисед
def summa(x, y):
    s = x + y
    print(s)
```

После создания функции, ее можно исполнять вызывая из другой функции или напрямую из оболочки Python. Для вызова функции следует ввести ее имя и добавить скобки.

In [18]:

```
# Для того, чтобы использовать функцию ее необходимо вызвать и передать ей необходимые
 аргументы
summa(4,5) # функция вернет сумму двух чисел 4 и 5 (4, 5 - это аргументы функции)

a = 1
b = 3
summa(a,b) # функция вернет сумму двух чисел a и b (a, b - это аргументы функции)
```

```
9
4
```

Также функции могут принимать любые типы аргументов как числовые, так и строковые, списки, словари и так далее.

In [19]:

```
# функция, которая в качестве параметра принимает список
# и возвращает сумму его элементов
def summa_spiska(lst):
    s = 0
    for i in lst:
        s += i
    return s

spisok = [1,2,3,4,5]
summa_spiska(spisok)

summa_spiska([1,9,10])
```

Out[19]:

```
20
```

Область видимости

Некоторые переменные программы могут быть недоступны некоторым областям данной программы. Все зависит от того, где вы объявили эти переменные.

В Python две базовых области видимости переменных:

- Глобальные переменные
- Локальные переменные

Переменные объявленные внутри тела функции имеют локальную область видимости, те что объявлены вне какой-либо функции имеют глобальную область видимости.

Это означает, что доступ к локальным переменным имеют только те функции, в которых они были объявлены, в то время как доступ к глобальным переменным можно получить по всей программе в любой функции.

In [20]:

```
z = 100 # z - глобальная переменная и может быть использована в любом месте программы
 (в том числе внутри функции)

def summa(x, y):
    s = x + y
    return None

def summa1(x, y):
    s = x + y + z
    return s

summa(1,2) #вызовем функцию и передадим ей два числа 1 и 2
print(s) # тут возникнет ошибка (так как вне функции переменная s не доступна)

summa1(1,2) # в данной случае функция вернет сумму трех числе 1 + 2 + 100, так как пере
менная z доступна в любой части программы
            # в том числе внутри функций
```

```
Hello
```

Out[20]:

```
103
```

Если перед нами стоит задача изменить глобальную переменную внутри функции - необходимо использовать ключевое слово **global**.

In [21]:

```
z = 0

def summa(x, y):
    z = x + y
    return None

def summa1(x, y):
    global z
    z = x + y
    return None

summa(1,2) # при вызове данной функции значение глобальной переменной z не измениться
summa1(1,2) # при вызове данной функции значение глобальной переменной z  измениться
```

Чтобы облегчить управление большими программами, программное обеспечение делится на более мелкие части. В Python эти части называются *модулями*. Модуль должен быть единицей, которая настолько независима от других модулей, насколько это возможно. Каждый файл в Python соответствует модулю. Модули могут содержать классы, объекты, функции, ... Например, функции для работы с регулярными выражениями находятся в модуле `re`

Стандартная библиотека Python состоит из сотен модулей. Некоторые из наиболее распространенных стандартных модулей включают.

- `re`
- `math`
- `random`
- `os`
- `sys`.

Любой файл с расширением `.py`, содержащий исходный код Python является модулем. Таким образом, для создания модуля не требуется специальной нотации.

Использование модулей

Допустим, нам необходимо использовать функцию косинуса. Эта функция и многие другие математические функции находятся в модуле `math`. Чтобы сообщить Python, что мы хотим получить доступ к функциям, предлагаемым этим модулем, мы можем передать оператор `import math`. Теперь модуль загружен в память. Теперь мы можем вызвать функцию следующим образом:

```
 python
math.cos(0)
1.0
```

Обратите внимание, что мы должны указать имя модуля, в котором находится функция `cos`. Это связано с тем, что другие модули могут иметь функцию (или другой атрибут модуля) с таким же именем. Использование различных пространств имен для каждого модуля предотвращает столкновения имен. Например, функции `gzip.open`, `os.open` не следует путать со встроенной функцией `open`.

Нарушение пространства имен

Если косинус нужен часто, то может оказаться утомительным всегда указывать пространство имен, особенно если имя пространства имен/модуля длинное. Для таких случаев существует другой способ импорта модулей. Принесите имя в текущую область видимости с помощью `from math import cos`. Теперь мы можем использовать его без спецификатора пространства имен: `cos(1)`.

Несколько имен можно импортировать в текущую область видимости с помощью оператора `from math import name1, name2, ...`. Или даже все имена модуля с помощью `from math import *`. Последняя форма имеет смысл только в некоторых случаях, обычно она просто запутывает ситуацию, поскольку пользователь может не иметь представления о том, какие имена будут импортированы.

Иерархия модулей

Стандартная библиотека содержит сотни модулей. Следовательно, трудно понять, что включает в себя библиотека. Поэтому модули необходимо как-то упорядочить. В Python модули могут быть организованы в иерархии с помощью *пакетов*. Пакет - это модуль, который может содержать другие пакеты и модули. Например, пакет `numpy` содержит подпакеты `core`, `distutils`, `f2py`, `fft`, `lib`, `linalg`, `ma`, `numarray`, `oldnumeric`, `random`, и `testing`. А пакет `numpy.linalg` в свою очередь содержит модули `linalg`, `lapack_lite` и `info`.

Импортирование из пакетов

Утверждение `import numpy` импортирует пакет верхнего уровня `numpy` и его подпакеты.

- `import numpy.linalg` импортирует только подпакет, и
- `import numpy.linalg.linalg` импортирует только модуль.

Если мы хотим пропустить длинную спецификацию пространства имен, мы можем использовать форму

```
 python
from numpy.linalg import linalg
```

или

 python
```
from numpy.linalg import linalg as lin
```

если мы хотим использовать другое имя для модуля. Следующая команда импортирует функцию `det` (вычисляет определитель матрицы) из модуля linalg, который содержится в подпакете linalg, принадлежащем пакету numpy: `python`
```
from numpy.linalg.linalg import det
```

Если бы мы импортировали только пакет верхнего уровня `numpy` , нам пришлось бы обращаться к функции `det` с полным именем `numpy.linalg.linalg.det` .

Вот краткое описание иерархии модулей:

```
пакет numpy
  .
подпакет linalg
  .
модуль linalg
  .
функция det
```

Соответствие между иерархиями папок и модулей

Пакеты представлены папками в файловой системе. Папка должна содержать файл с именем `__init__.py` , который составляющий тело пакета. Он обрабатывает инициализацию пакета. Папка может также содержать другие папки (подпакеты) или файлы Python (обычные модули).

``` a/ **init**.py b.py c/ **init**.py d.py e

Пример использования функций sqrt (извлечение квадратного корня) и pow (возведение в степень) из модуля math

In [22]:

```
from math import sqrt, pow
print(sqrt(4), sqrt(5))
print(pow(2, 8), pow(5, 2))
```

```
2.0 2.23606797749979
256.0 25.0
```

Задания
```

Задача 1.

Даны четыре действительных числа: x1, y1, x2, y2. Напишите функцию distance(x1, y1, x2, y2), вычисляющая евклидово расстояние (https://ru.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%BE%D0%B2%D0 (https://ru.wikipedia.org/wiki/%D0%95%D0%B2%D0%BA%D0%BB%D0%B8%D0%B4%D0%BE%D0%B2%D( ) между точкой (x1,y1) и (x2,y2).

In [22]:

Задача 2.

Написать функцию season(month), принимающую 1 аргумент — номер месяца (от 1 до 12), которая присваивает глобальной переменной s время года, которому этот месяц принадлежит (зима, весна, лето или осень).

In [22]:

Задача 3.

Написать функцию is_prime, принимающую 1 аргумент — число от 0 до 1000, и возвращающую True, если оно простое, и False - иначе.

In [22]:

Задача 4.

Написать функцию reverse_list(lst), которая принимает в качестве аргумента список и возвращаем его в перевернутом виде.

Например,

- исходный список: 8, 1, 0, 4
- полученный список: 4, 0, 1, 8

Использовать встроенные функции Python нельзя.

In [22]:

Задача 5.

Распечатайте с 4 по 8 символ фразы "Привет мир!" приведенные к верхнему регистру.

In [22]:

```
```

Задача 6.

Напишите код, который все элементы массива x с нечетными индексами переставит в обратном порядке.

Т.е. если x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], то код должен получать [0, 9, 2, 7, 4, 5, 6, 3, 8, 1]

In [22]:

```
```

Задача 7.

Напишите цикл, который выводит все числа от 0 до 500, делящиеся на 7, если в них есть цифра 8.

In [22]:

```
```

Задача 8.

На вход функция more_than_five(lst) получает список из целых чисел. Результатом работы функции должен стать новый список, в котором содержатся только те числа, которые больше 10 по модулю.

In [22]:

```
```

Numpy

NumPy (https://numpy.org/) это open-source модуль для python, который предоставляет общие математические и числовые операции.

NumPy - один из ключевых модулей в экосистеме Python, в том числе при решении задач машинного обучения и искусственного интеллекта.

NumPy является наследником Numeric и NumArray. Основан NumPy на библиотеке LAPAC, которая написана на Fortran. Когда-то numpy была частью SciPy. Да, это напоминает мыльную оперу.

Мы подробно разбираем особенности библиотеки, так как на работе с ней основаны все остальные библиотеки, работающие с искусственным интеллектом.

Текст урока опирается на небольшой, но полезный мануал (https://sites.engineering.ucsb.edu/~shell/che210d/numpy.pdf).

Установка

Если вы используете Google Colab, то numpy уже установлен на виртуальном сервере и вы можете им пользоваться.

Если вы открыли собственный ноутбук, то можете воспользоваться командой установки, записанной через восклицательный знак

```
!command
```

Так запущенная команда в среде ipy вызывает системную команду pip, которая сама установит данный модуль в вашу виртуальную среду.

Обратите внимание, что каждый ноутбк имеет свой набор используемых библиотек и системных установок.

In [23]:

```
!pip install numpy
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/col
ab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-pack
ages (1.21.6)
```

In [24]:

```
!pip list
```

Package	Version
absl-py	1.3.0
aeppl	0.0.33
aesara	2.7.9
aiohttp	3.8.3
aiosignal	1.2.0
alabaster	0.7.12
albumentations	1.2.1
altair	4.2.0
appdirs	1.4.4
arviz	0.12.1
astor	0.8.1
astropy	4.3.1
astunparse	1.6.3
async-timeout	4.0.2
asynctest	0.13.0
atari-py	0.2.9
atomicwrites	1.4.1
attrs	22.1.0
audioread	3.0.0
autograd	1.5
Babel	2.10.3
backcall	0.2.0
beautifulsoup4	4.6.3
bleach	5.0.1
blis	0.7.9
bokeh	2.3.3
branca	0.5.0
bs4	0.0.1
CacheControl	0.12.11
cached-property	1.5.2
cachetools	4.2.4
catalogue	2.0.8
certifi	2022.9.24
cffi	1.15.1
cftime	1.6.2
chardet	3.0.4
charset-normalizer	2.1.1
click	7.1.2
clikit	0.6.2
cloudpickle	1.5.0
cmake	3.22.6
cmdstanpy	1.0.8
colorcet	3.0.1
colorlover	0.3.0
community	1.0.0b1
confection	0.0.3
cons	0.4.5
contextlib2	0.5.5
convertdate	2.4.0
crashtest	0.3.1
crcmod	1.7
cufflinks	0.17.3
cvxopt	1.3.0
cvxpy	1.2.1
cycler	0.11.0
cymem	2.0.7
Cython	0.29.32
daft	0.0.4
dask	2022.2.0

```
datascience                   0.17.5
debugpy                       1.0.0
decorator                     4.4.2
defusedxml                    0.7.1
descartes                     1.1.0
dill                          0.3.6
distributed                   2022.2.0
dlib                          19.24.0
dm-tree                       0.1.7
dnspython                     2.2.1
docutils                      0.17.1
dopamine-rl                   1.0.5
earthengine-api               0.1.329
easydict                      1.10
ecos                          2.0.10
editdistance                  0.5.3
en-core-web-sm                3.4.1
entrypoints                   0.4
ephem                         4.1.3
et-xmlfile                    1.1.0
etils                         0.8.0
etuples                       0.3.8
fa2                           0.3.5
fastai                        2.7.9
fastcore                      1.5.27
fastdownload                  0.0.7
fastdtw                       0.3.4
fastjsonschema                2.16.2
fastprogress                  1.0.3
fastrlock                     0.8
feather-format                0.4.1
filelock                      3.8.0
firebase-admin                4.4.0
fix-yahoo-finance             0.0.22
Flask                         1.1.4
flatbuffers                   1.12
folium                        0.12.1.post1
frozenlist                    1.3.1
fsspec                        2022.10.0
future                        0.16.0
gast                          0.4.0
GDAL                          2.2.2
gdown                         4.4.0
gensim                        3.6.0
geographiclib                 1.52
geopy                         1.17.0
gin-config                    0.5.0
glob2                         0.7
google                        2.0.3
google-api-core               1.31.6
google-api-python-client      1.12.11
google-auth                   1.35.0
google-auth-httplib2          0.0.4
google-auth-oauthlib          0.4.6
google-cloud-bigquery         1.21.0
google-cloud-bigquery-storage 1.1.2
google-cloud-core             1.0.3
google-cloud-datastore        1.8.0
google-cloud-firestore        1.7.0
google-cloud-language         1.2.0
google-cloud-storage          1.18.1
```

google-cloud-translate	1.5.0
google-colab	1.0.0
google-pasta	0.2.0
google-resumable-media	0.4.1
googleapis-common-protos	1.56.4
googledrivedownloader	0.4
graphviz	0.10.1
greenlet	1.1.3.post0
grpcio	1.50.0
gspread	3.4.2
gspread-dataframe	3.0.8
gym	0.25.2
gym-notices	0.0.8
h5py	3.1.0
HeapDict	1.0.1
hijri-converter	2.2.4
holidays	0.16
holoviews	1.14.9
html5lib	1.0.1
httpimport	0.5.18
httplib2	0.17.4
httplib2shim	0.0.3
httpstan	4.6.1
humanize	0.5.1
hyperopt	0.1.2
idna	2.10
imageio	2.9.0
imagesize	1.4.1
imbalanced-learn	0.8.1
imblearn	0.0
imgaug	0.4.0
importlib-metadata	4.13.0
importlib-resources	5.10.0
imutils	0.5.4
inflect	2.1.0
intel-openmp	2022.2.0
intervaltree	2.1.0
ipykernel	5.3.4
ipython	7.9.0
ipython-genutils	0.2.0
ipython-sql	0.3.9
ipywidgets	7.7.1
itsdangerous	1.1.0
jax	0.3.23
jaxlib	0.3.22+cuda11.cudnn805
jieba	0.42.1
Jinja2	2.11.3
joblib	1.2.0
jpeg4py	0.1.4
jsonschema	4.3.3
jupyter-client	6.1.12
jupyter-console	6.1.0
jupyter-core	4.11.2
jupyterlab-widgets	3.0.3
kaggle	1.5.12
kapre	0.3.7
keras	2.9.0
Keras-Preprocessing	1.1.2
keras-vis	0.4.1
kiwisolver	1.4.4
korean-lunar-calendar	0.3.1

langcodes	3.3.0
libclang	14.0.6
librosa	0.8.1
lightgbm	2.2.3
llvmlite	0.39.1
lmdb	0.99
locket	1.0.0
logical-unification	0.4.5
LunarCalendar	0.0.9
lxml	4.9.1
Markdown	3.4.1
MarkupSafe	2.0.1
marshmallow	3.18.0
matplotlib	3.2.2
matplotlib-venn	0.11.7
miniKanren	1.0.3
missingno	0.5.1
mistune	0.8.4
mizani	0.7.3
mkl	2019.0
mlxtend	0.14.0
more-itertools	9.0.0
moviepy	0.2.3.5
mpmath	1.2.1
msgpack	1.0.4
multidict	6.0.2
multipledispatch	0.6.0
multitasking	0.0.11
murmurhash	1.0.9
music21	5.5.0
natsort	5.5.0
nbconvert	5.6.1
nbformat	5.7.0
netCDF4	1.6.1
networkx	2.6.3
nibabel	3.0.2
nltk	3.7
notebook	5.5.0
numba	0.56.3
numexpr	2.8.4
numpy	1.21.6
oauth2client	4.1.3
oauthlib	3.2.2
okgrade	0.4.3
opencv-contrib-python	4.6.0.66
opencv-python	4.6.0.66
opencv-python-headless	4.6.0.66
openpyxl	3.0.10
opt-einsum	3.3.0
osqp	0.6.2.post0
packaging	21.3
palettable	3.3.0
pandas	1.3.5
pandas-datareader	0.9.0
pandas-gbq	0.13.3
pandas-profiling	1.4.1
pandocfilters	1.5.0
panel	0.12.1
param	1.12.2
parso	0.8.3
partd	1.3.0

pastel	0.2.1
pathlib	1.0.1
pathy	0.6.2
patsy	0.5.3
pep517	0.13.0
pexpect	4.8.0
pickleshare	0.7.5
Pillow	7.1.2
pip	21.1.3
pip-tools	6.2.0
plotly	5.5.0
plotnine	0.8.0
pluggy	0.7.1
pooch	1.6.0
portpicker	1.3.9
prefetch-generator	1.0.1
preshed	3.0.8
prettytable	3.4.1
progressbar2	3.38.0
promise	2.3
prompt-toolkit	2.0.10
prophet	1.1.1
protobuf	3.17.3
psutil	5.4.8
psycopg2	2.9.5
ptyprocess	0.7.0
py	1.11.0
pyarrow	6.0.1
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycocotools	2.0.5
pycparser	2.21
pyct	0.4.8
pydantic	1.10.2
pydata-google-auth	1.4.0
pydot	1.3.0
pydot-ng	2.0.0
pydotplus	2.0.2
PyDrive	1.3.1
pyemd	0.5.1
pyerfa	2.0.0.1
Pygments	2.6.1
pygobject	3.26.1
pylev	1.4.0
pymc	4.1.4
PyMeeus	0.5.11
pymongo	4.3.2
pymystem3	0.2.0
PyOpenGL	3.1.6
pyparsing	3.0.9
pyrsistent	0.18.1
pysimdjson	3.2.0
pysndfile	1.3.8
PySocks	1.7.1
pystan	3.3.0
pytest	3.6.4
python-apt	0.0.0
python-dateutil	2.8.2
python-louvain	0.16
python-slugify	6.1.2
python-utils	3.3.3

```
pytz                            2022.5
pyviz-comms                     2.2.1
PyWavelets                      1.3.0
PyYAML                          6.0
pyzmq                           23.2.1
qdldl                           0.1.5.post2
qudida                          0.0.4
regex                           2022.6.2
requests                        2.23.0
requests-oauthlib               1.3.1
resampy                         0.4.2
rpy2                            3.5.5
rsa                             4.9
scikit-image                    0.18.3
scikit-learn                    1.0.2
scipy                           1.7.3
screen-resolution-extra         0.0.0
scs                             3.2.0
seaborn                         0.11.2
Send2Trash                      1.8.0
setuptools                      57.4.0
setuptools-git                  1.2
Shapely                         1.8.5.post1
six                             1.15.0
sklearn-pandas                  1.8.0
smart-open                      5.2.1
snowballstemmer                 2.2.0
sortedcontainers                2.4.0
soundfile                       0.11.0
spacy                           3.4.2
spacy-legacy                    3.0.10
spacy-loggers                   1.0.3
Sphinx                          1.8.6
sphinxcontrib-serializinghtml   1.1.5
sphinxcontrib-websupport        1.2.4
SQLAlchemy                      1.4.42
sqlparse                        0.4.3
srsly                           2.4.5
statsmodels                     0.12.2
sympy                           1.7.1
tables                          3.7.0
tabulate                        0.8.10
tblib                           1.7.0
tenacity                        8.1.0
tensorboard                     2.9.1
tensorboard-data-server         0.6.1
tensorboard-plugin-wit          1.8.1
tensorflow                      2.9.2
tensorflow-datasets             4.6.0
tensorflow-estimator            2.9.0
tensorflow-gcs-config           2.9.1
tensorflow-hub                  0.12.0
tensorflow-io-gcs-filesystem    0.27.0
tensorflow-metadata             1.10.0
tensorflow-probability          0.16.0
termcolor                       2.0.1
terminado                       0.13.3
testpath                        0.6.0
text-unidecode                  1.3
textblob                        0.15.3
thinc                           8.1.5
```

```
threadpoolctl                3.1.0
tifffile                     2021.11.2
toml                         0.10.2
tomli                        2.0.1
toolz                        0.12.0
torch                        1.12.1+cu113
torchaudio                   0.12.1+cu113
torchsummary                 1.5.1
torchtext                    0.13.1
torchvision                  0.13.1+cu113
tornado                      5.1.1
tqdm                         4.64.1
traitlets                    5.1.1
tweepy                       3.10.0
typeguard                    2.7.1
typer                        0.4.2
typing-extensions            4.1.1
tzlocal                      1.5.1
uritemplate                  3.0.1
urllib3                      1.24.3
vega-datasets                0.9.0
wasabi                       0.10.1
wcwidth                      0.2.5
webargs                      8.2.0
webencodings                 0.5.1
Werkzeug                     1.0.1
wheel                        0.37.1
widgetsnbextension           3.6.1
wordcloud                    1.8.2.2
wrapt                        1.14.1
xarray                       0.20.2
xarray-einstats              0.2.2
xgboost                      0.90
xkit                         0.0.0
xlrd                         1.1.0
xlwt                         1.3.0
yarl                         1.8.1
yellowbrick                  1.5
zict                         2.2.0
zipp                         3.10.0
```

Для обновления модуля можно использовать следующую команду:

In [25]:

```
!pip install numpy --upgrade
# или можно так !pip install numpy --U
# а удалить пакет можно так !pip uninstall numpy --U
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/col
ab-wheels/public/simple/
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-pack
ages (1.21.6)
```

Также можно использовать другие команды (ls, cd и т.д.):

In [ ]:

```
!pwd # текущая директория
!ls # список файлов в текущей директории
# !cd - смена директории
# !mkdir, !mv
# конвертация ноутбука в html формат
!jupyter nbconvert --to html /content/example.ipynb
!ls
# а можно даже такое творить
contents = !ls
directory = !pwd
type(directory)

from google.colab import files, drive
# подсоединение к гугд-диску
drive.mount('/content/gdrive')
# скачивание файла локально
files.download('/content/example.ipynb')
# сохранение на гугл-диск
#!cp "/content/example.ipynb" "/content/drive/My Drive/"
```

Скачивать файлы из веба можно с помощью команды !wget

Установка пакетов через !pip install or !apt-get install

Запуск .py скрипта !python script.py

Клонирование git-репозитория !git clone

!curl

Подключение

Для подключения модулей в Python используется команда `import` или её вариации. В случае с numpy есть традиционная и привычная всем команда импорта с использованием алиаса (as) np

Алиас - это встроенная команда интерпретатора для сокращения команд и их последовательностей.

In [29]:

```
# классический вариант
import numpy
```

In [30]:

```
# традиционный вариант
import numpy as np
```

In [33]:

```
np?
```

При таком импорте к любым командам из модуля numpy придётся дописывать название модуля

```
numpy.command
np.command
```

Есть другой вариант, в котором можно использовать только команду без указания модуля. Но так как в разных модулях могут быть одинаковые функции, использовать такой вариант не рекомендуется.

In [34]:

```
from numpy import *
```

Особые константы

Numpy реализует несколько особых значнений через константы. Например:

In [ ]:

```
np.NaN
# not a number - Не число
```

In [36]:

```
np.Inf
# infinity - бесконечно
```

Out[36]:

```
inf
```

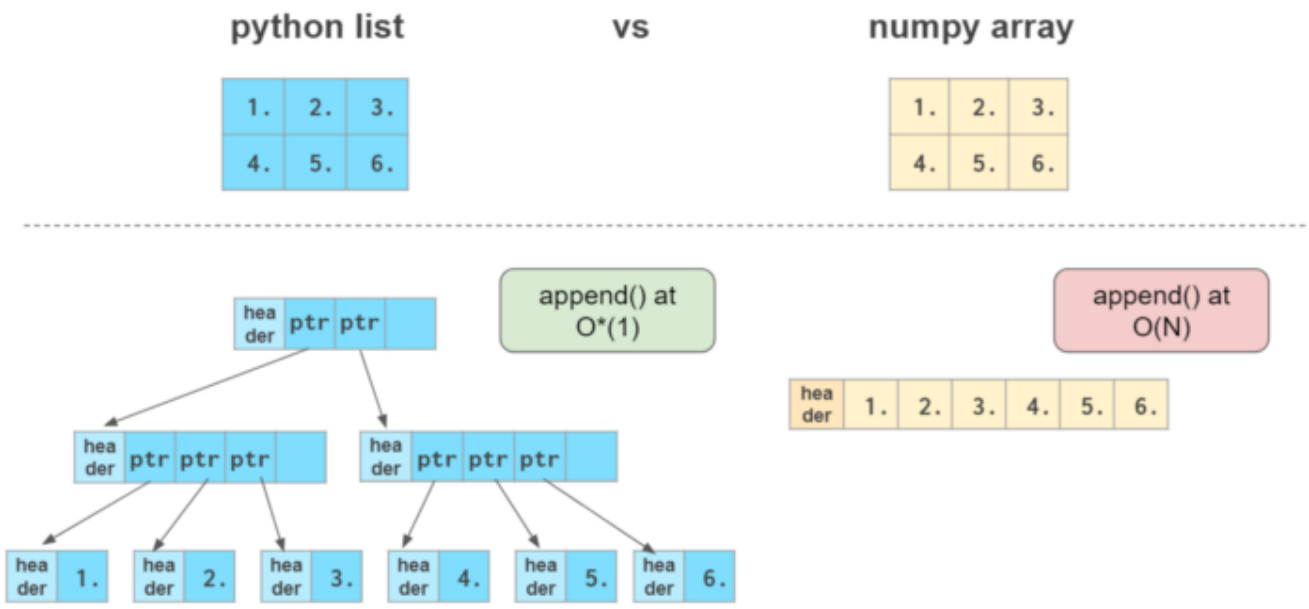
Массивы

Главная особенность и элемент, с которым необходимо работать, в numpy - это массивы.

Массивы NumPy:

```
1) более компактны, особенно когда есть более одного измерения

2) быстрее, чем списки, когда операция может быть векторизована

3) медленнее, чем списки, когда вы добавляете элементы в конец

4) обычно однородны: могут быстро работать только с элементами одного типа, что
 отличает их от классического списка (list) Python.
```

Создаются массивы разными способами, которые мы сейчас разберём.



In [37]:

```
a = np.array([1, 2, 3, 4], dtype=float)
print('Array:', a)
print('Тип: ',type(a))
# обратное преобразование
print(a.tolist())
```

```
Array: [1. 2. 3. 4.]
Тип:  <class 'numpy.ndarray'>
[1.0, 2.0, 3.0, 4.0]
```

Можно указать dtype при инициализации массива не float, a object, что позволит хранить неоднородные данные - но это сведет массив к list и замедлит работу

In [38]:

```
# показываем, что можно также работать с np.array, как и с обычным list
a = np.array([1, 2, 4, 4])

print('1: ', a[0])
print('2: ', a[1:3])
print('3: ', a[-1])
a[0] = 5
print('4: ', a[0])
```

```
1:  1
2:  [2 4]
3:  4
4:  5
```

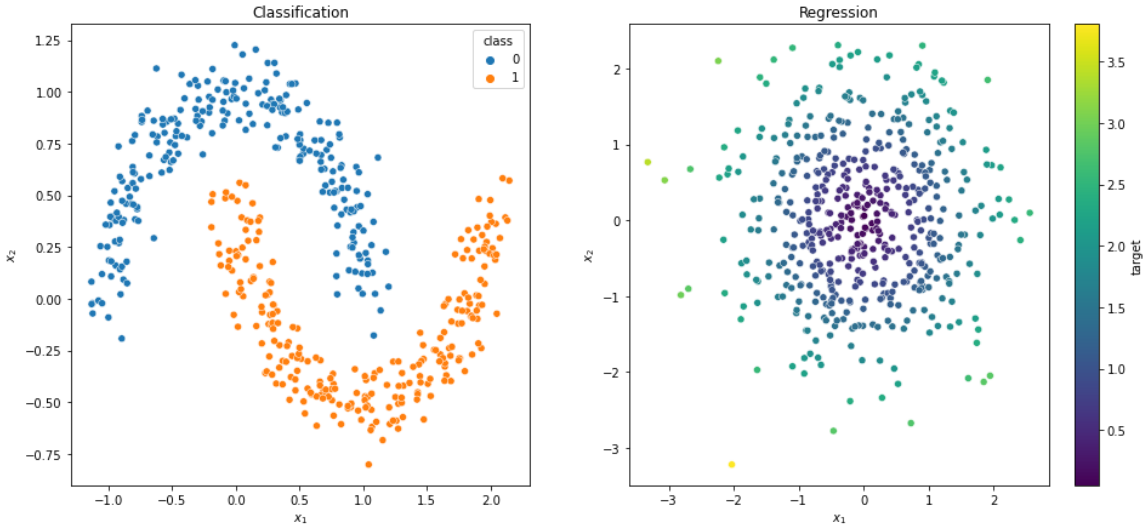
Многомерные массивы

Большая ценность numpy в том, что можно работать и многомерными массивами. Например, любое изображение - как минимум двумерный массив. А при обучении нейронных сетей для работы с компьютерным зрением используются, по сути, четырёхмерные массивы.

Например, табличные данные - классический вид данных машинного обучения

In [143]:

```
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

X, y = make_moons(n_samples=500, noise=0.1, random_state=0)
fig, axs = plt.subplots(figsize=(16, 7), nrows=1, ncols=2, gridspec_kw={'width_ratios':
[1.75, 2]})
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y, ax=axs[0])

X = np.random.randn(500, 2)
y = np.sqrt((X ** 2).sum(axis=1))
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=y, ax=axs[1], palette='viridis')

axs[0].legend(title='class')
axs[0].set_title('Classification')

norm = plt.Normalize(y.min(), y.max())
sm = plt.cm.ScalarMappable(cmap='viridis', norm=norm)
sm.set_array([])

axs[1].get_legend().remove()
axs[1].figure.colorbar(sm, label='target')
axs[1].set_title('Regression')

for ax in axs:
    ax.set_xlabel('$x_1$')
    ax.set_ylabel('$x_2$')

plt.show()
pd.DataFrame(np.hstack((X, y.reshape((-1, 1)))), columns=['X1', 'X2', 'y'])
```



Out[143]:

	X1	X2	y
0	-1.929544	0.638139	2.032329
1	0.461932	0.525027	0.699310
2	-0.296551	0.050682	0.300851
3	-0.740195	-1.072085	1.302787
4	-0.499353	-0.239295	0.553729
...	...	...	...
495	0.784738	0.735758	1.075711
496	0.071602	1.219523	1.221623
497	1.530606	1.026740	1.843082
498	0.705667	0.231087	0.742541
499	-0.943264	0.009772	0.943315

500 rows × 3 columns

Изображения

Направление машинного обучения, которое имеет дело с изображениями, называется компьютерным зрением (Computer Vision, CV). Как правило, изображения представляют в виде тензоров размерности (H, W, C), где обычно $C = 3$. Содержание тензора зависит от цветового пространства, чаще всего используется RGB.

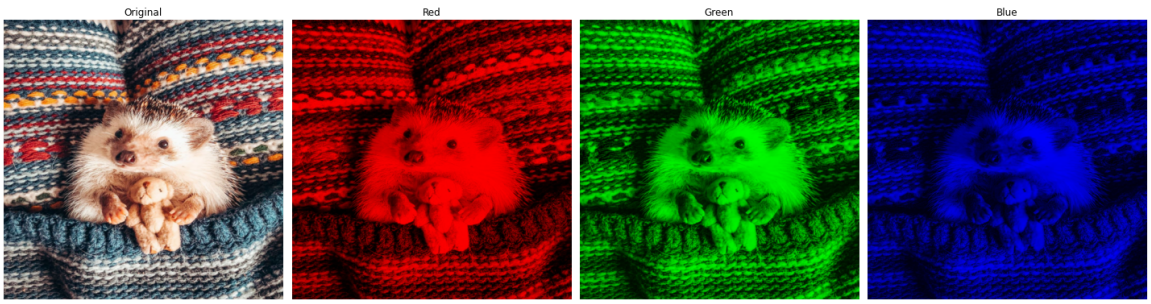
In [144]:

```
!wget -O image.jpg https://krasivosti.pro/uploads/posts/2021-06/1623727983_11-krasivosti_pro-p-milie-yezhiki-zhivotnie-krasivo-foto-11.jpg
```

```
--2022-11-03 06:43:12--  https://krasivosti.pro/uploads/posts/2021-06/1623
727983_11-krasivosti_pro-p-milie-yezhiki-zhivotnie-krasivo-foto-11.jpg
Resolving krasivosti.pro (krasivosti.pro)... 151.80.240.247
Connecting to krasivosti.pro (krasivosti.pro)|151.80.240.247|:443... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 465999 (455K) [image/jpeg]
Saving to: ‘image.jpg’

image.jpg           100%[===================>] 455.08K   973KB/s    in 0.5
s

2022-11-03 06:43:13 (973 KB/s) - ‘image.jpg’ saved [465999/465999]
```

In [145]:

```
from skimage import io, color
from skimage.transform import resize

rgb_image = resize(io.imread('image.jpg'), (512, 512))
fix, axs = plt.subplots(figsize=(20, 6), nrows=1, ncols=4)
axs[0].imshow(rgb_image)

red_image = np.copy(rgb_image)
red_image[..., 1:] = 0.0
axs[1].imshow(red_image)

green_image = np.copy(rgb_image)
green_image[..., [0, 2]] = 0.0
axs[2].imshow(green_image)

blue_image = np.copy(rgb_image)
blue_image[..., :2] = 0.0
axs[3].imshow(blue_image)

for ax in axs:
    ax.grid(False)
    ax.axis('off')

for ax, title in zip(axs, ['Original', 'Red', 'Green', 'Blue']):
    ax.set_title(title)

plt.tight_layout()
plt.show()
```



In [146]:

```
rgb_image.shape
```

Out[146]:

```
(512, 512, 3)
```

Текстовые данные

Обработка естественного языка (Natural Language Processing, NLP) - одно из наиболее востребованных направлений в машинном обучении. Зачастую методы из NLP перекочевывают в другие домены данных и находят там успешное применение. Основными особенностями текстовых данных являются, во-первых, их дискретность, а во-вторых, последовательная структура. Как правило, при работе с текстом его токенизируют (по символам, по словам или по сочетаниям символов -- так называемый byte-pair-encoding, BPE), а затем каждому токену присваивается некоторый вектор-эмбеддинг. Такое преобразование позволяет обрабатывать дискретные данные непрерывными моделями.

В последнее время также развиваются инструменты работы с данными, которые похожи на тексты, но таковыми не являются. Например, появляются новые задачи и данные, связанные с обработкой программного кода

In [148]:

```
import tensorflow_hub as hub

embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4")

Sentences = [
    "How old are you",
    "What is your age",
    "I love to watch Television",
    "I am wearing a wrist watch"
]
embeddings = embed(Sentences)

print(embeddings)

for i in range(len(Sentences)):
    print(Sentences[i])
    print(embeddings[i])
    break
```

```
tf.Tensor(
[[-0.06045125 -0.00204541  0.02656925 ...  0.00764413 -0.02669661
   0.05110302]
 [-0.08415681 -0.08687922  0.03446117 ... -0.01439389 -0.04546221
   0.03639964]
 [ 0.0816019  -0.01570276 -0.05659244 ... -0.07133698  0.11040761
  -0.0071095 ]
 [-0.00369539  0.03064634 -0.05556112 ...  0.01751423  0.0316496
  -0.05139377]], shape=(4, 512), dtype=float32)
How old are you
tf.Tensor(
[-0.06045125 -0.00204541  0.02656925  0.02483987 -0.09206301  0.01760055
 -0.00284112 -0.00158421 -0.00648677  0.05091851 -0.01242187  0.0196162
  0.01128109 -0.04664037 -0.00683796 -0.04489101 -0.08644009  0.00650328
 -0.05588394 -0.09097648 -0.01062818  0.00379061  0.00729432 -0.03010405
 -0.02222248  0.01104948  0.02357136 -0.01104616  0.0022157   0.01935781
  0.02867085 -0.07079582  0.05549346  0.00179186 -0.02146762 -0.00257316
 -0.00539005 -0.03071919  0.00420406  0.0023769  -0.03511341  0.01090724
  0.01152228  0.10254128  0.04378743 -0.06699706 -0.10248536  0.00487916
  0.06548391 -0.0132634  -0.02285226  0.11008765 -0.09174919 -0.04022533
  0.00245567  0.03807915 -0.09587954  0.03216569 -0.05647296 -0.0092262
 -0.02832855 -0.01865764  0.06763721 -0.03797805  0.00380764  0.02795104
  0.00482427 -0.05925604 -0.00132146  0.00646655  0.02378649 -0.03549853
 -0.03697146 -0.01133168 -0.02775426 -0.00576924  0.02420777  0.0156711
  0.03766861 -0.10901458 -0.03557547  0.03258683 -0.08582282 -0.04159031
 -0.04307481 -0.07554604 -0.00135031  0.05407195 -0.04491962  0.06613255
 -0.05416534 -0.0135482   0.07910454  0.02208845 -0.03984658 -0.04217036
  0.05219576  0.09406526 -0.0299576   0.01244447  0.01027207  0.00108326
 -0.00387798  0.01307962 -0.02755091  0.04380766 -0.03764504 -0.01876564
 -0.0025468   0.02528893  0.04400486  0.02859843 -0.05940965 -0.07587431
  0.0061207   0.01270822  0.01687385  0.01942195 -0.00170139  0.05851777
  0.04113562  0.02398655 -0.01965653 -0.02660001 -0.07510682 -0.0156631
 -0.01484451 -0.01067223  0.04613434  0.04155705 -0.0056817  -0.05051666
  0.04287304  0.027521    0.01502921 -0.03377119 -0.0444997  -0.00791443
  0.02430577 -0.00203638 -0.04406145 -0.08691692  0.08827817  0.03371754
 -0.02259916  0.02028665  0.01936296  0.00947016  0.05391425 -0.01432816
 -0.00628367 -0.02428202 -0.02830358 -0.03781512 -0.03811835 -0.10129569
 -0.04673885  0.09986857  0.02402731  0.00493672 -0.00102915 -0.01137885
 -0.05344289  0.00654126 -0.02941149  0.03012461 -0.05634551 -0.03261634
 -0.02545495  0.02436141  0.03135832  0.08505724  0.02935501 -0.00714586
  0.00691799 -0.07481728  0.00633727 -0.02819034 -0.0003587   0.10050821
  0.01975427 -0.02046341  0.11006147 -0.01865015  0.00073067 -0.04685991
  0.02256736  0.01032875 -0.03461596  0.00686473  0.05090384 -0.01351599
  0.06034109  0.03594441 -0.02509799  0.0472048  -0.00864083 -0.04504239
  0.00353577  0.0135932   0.00995236  0.01369962 -0.0542173  -0.08059129
  0.06753725 -0.03574559  0.05404207 -0.0132688   0.00668525  0.0222158
  0.00115469  0.00772014 -0.0009705   0.04064095 -0.0173293  -0.00985907
 -0.07807792 -0.02983265 -0.01361165  0.01361167 -0.007105   -0.01972496
  0.000122    0.05403951 -0.0392362   0.02924993  0.03447536  0.09660447
  0.04014966  0.01944557 -0.02900655 -0.07659646 -0.02683486  0.10600524
 -0.02046891  0.01470214 -0.04470241 -0.06767991  0.0679819   0.01647552
 -0.08040208 -0.00105591  0.07032613  0.01053682  0.05592678  0.01833372
 -0.05859426  0.01366992  0.10758807 -0.09209943  0.07488038 -0.03402849
  0.00478643  0.07082231  0.10186784  0.05105784  0.06321965  0.05317153
 -0.01237961  0.01402194 -0.01251294  0.07837544 -0.07235587  0.03748485
 -0.02670813 -0.03542441 -0.01462621  0.04126989  0.04184546  0.03538936
  0.00179791  0.101612   -0.00854288  0.06636288 -0.05760877 -0.02262902
  0.03202924  0.06278779 -0.04574149 -0.07212305  0.03385026 -0.09136895
 -0.0202541   0.01260154  0.03872314  0.03312309 -0.04048205 -0.02116848
 -0.03972117  0.06208019 -0.05567107 -0.05032091 -0.08866157  0.02720379
  0.00613067 -0.00155733  0.02704135  0.03788527  0.07137241  0.05030008
```

```
-0.0306305   0.02619931 -0.05845411  0.00865892 -0.01383821  0.06349964
-0.03056945  0.02235948 -0.03368198  0.04317337 -0.02703194  0.0721937
 0.03109808  0.01800174  0.01340949  0.03420349  0.03084787 -0.04560824
-0.03791029  0.01409329  0.00333317 -0.07557759 -0.04386066 -0.00075197
 0.05958378  0.04582742  0.04103404  0.02889142 -0.03972253  0.0006737
 0.03637769  0.0304617   0.00470297 -0.04965226 -0.03135933 -0.01560437
 0.06140454 -0.00878903  0.05179127  0.00384064 -0.02629672  0.00832837
-0.04031477  0.08868903  0.01783241  0.02076994  0.0260689  -0.00626407
-0.01969612  0.08694895  0.02706335  0.01660867 -0.00216502  0.01133792
 0.02191536  0.04492361 -0.10157027  0.03234165  0.04988328  0.03451608
-0.10478352  0.01526531  0.08659794  0.00793927 -0.05365328  0.03835471
-0.05944474 -0.01340998  0.05740954  0.01230434 -0.01850316  0.00560911
 0.01906535 -0.02667349 -0.01127984  0.10955353 -0.01335264  0.02368696
 0.00245678 -0.03541937 -0.01954655  0.04979192 -0.03137618 -0.01882681
-0.04443992  0.05433447  0.04253041  0.01737742  0.08055392  0.01613655
 0.04015921  0.04319952 -0.02312539 -0.01146666  0.01036766 -0.02896691
-0.01297201 -0.03913989 -0.00755901  0.06484879 -0.04514799  0.05298594
 0.03448734 -0.03786335  0.00201303  0.01048455 -0.01040853  0.07110377
-0.02348094  0.01771266 -0.02033903  0.02665153  0.02616197 -0.01000245
-0.04644519  0.01130423  0.01483327  0.06707779 -0.01461387 -0.00452696
-0.04304872  0.00689362 -0.05182236 -0.01519322 -0.04088411  0.03092237
 0.00254572  0.08904892 -0.00792992  0.01870976  0.04452706  0.03333293
 0.05689116  0.07747299 -0.01120676  0.03132071 -0.0055206  -0.02491163
 0.00083916  0.06260858  0.04215233  0.0432216   0.07809894  0.04484669
-0.02300675 -0.04632314  0.04653197 -0.00201554 -0.02685321 -0.05020576
 0.04530709  0.04188759  0.0328692  -0.03730487 -0.0781797  -0.06690054
 0.03342807 -0.04680061 -0.01493964  0.02251878 -0.10957455 -0.01221834
-0.0659089  -0.1001366   0.06179824 -0.06596868 -0.01495706  0.04030221
 0.01698426  0.01884837 -0.01466365  0.00414858  0.02326314 -0.05161554
 0.04524233  0.0716437  -0.03854608  0.07572064 -0.04058167  0.07632438
-0.0174949  -0.02530942 -0.06067707 -0.04749297 -0.01579468 -0.01677551
-0.10685218 -0.03555699 -0.04327874 -0.0121555  -0.0105649  -0.05180348
 0.0268241  -0.04316089 -0.03254637 -0.02068835  0.02562678 -0.02623284
 0.0781391  -0.02213159  0.05033635 -0.01542311 -0.05477818  0.08987653
 0.04600991 -0.01985893  0.02022584 -0.02211026  0.00164093  0.00764413
-0.02669661  0.05110302], shape=(512,), dtype=float32)
```

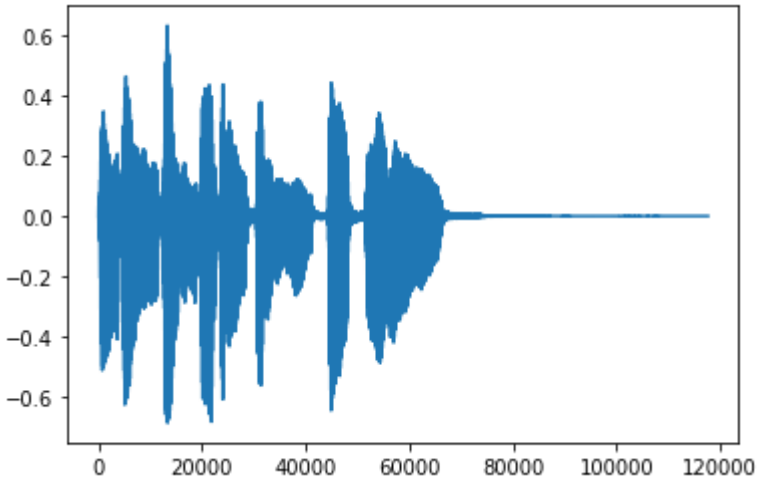
Аудио

Аудиозаписи часто представляют как последовательность вещественных чисел из промежутка $[-1, 1]$ (wav-формат). Такие аудио могут быть как одноканальными, так и многоканальными. При работе с музыкой популярен MIDI-формат. Но наиболее часто встречаются спектрограммы -- результат применения к звуковой волне преобразования Фурье или его вариаций.

In [152]:

```
import librosa
import torch
import torchaudio
from IPython.display import Audio

waveform, sample_rate = librosa.load(librosa.example('trumpet'))
print(waveform.shape)
plt.plot(waveform)
print(waveform[:10])
```

```
(117601,)
[-1.4068224e-03 -4.4607223e-04 -4.1098078e-04  9.9920901e-05
  4.3150427e-05 -1.7485349e-04 -3.6783377e-04  1.4553138e-04
  5.0557830e-04  1.4929948e-03]
```



Работа с видео-файлами похожа на работу с изображениями: к 2 пространственным размерностям и размерности цветовых каналов добавляется время. Иногда в контексте видео рассматривают и звуковую дорожку.

Графовые данные

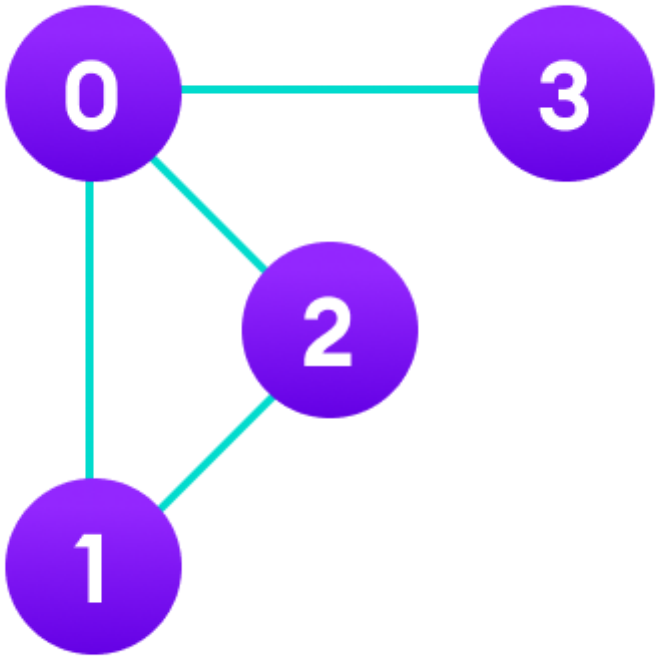
Иногда данные имеют явную или неявную графовую структуру (например, молекулы как графы атомов или профили в социальных сетях)

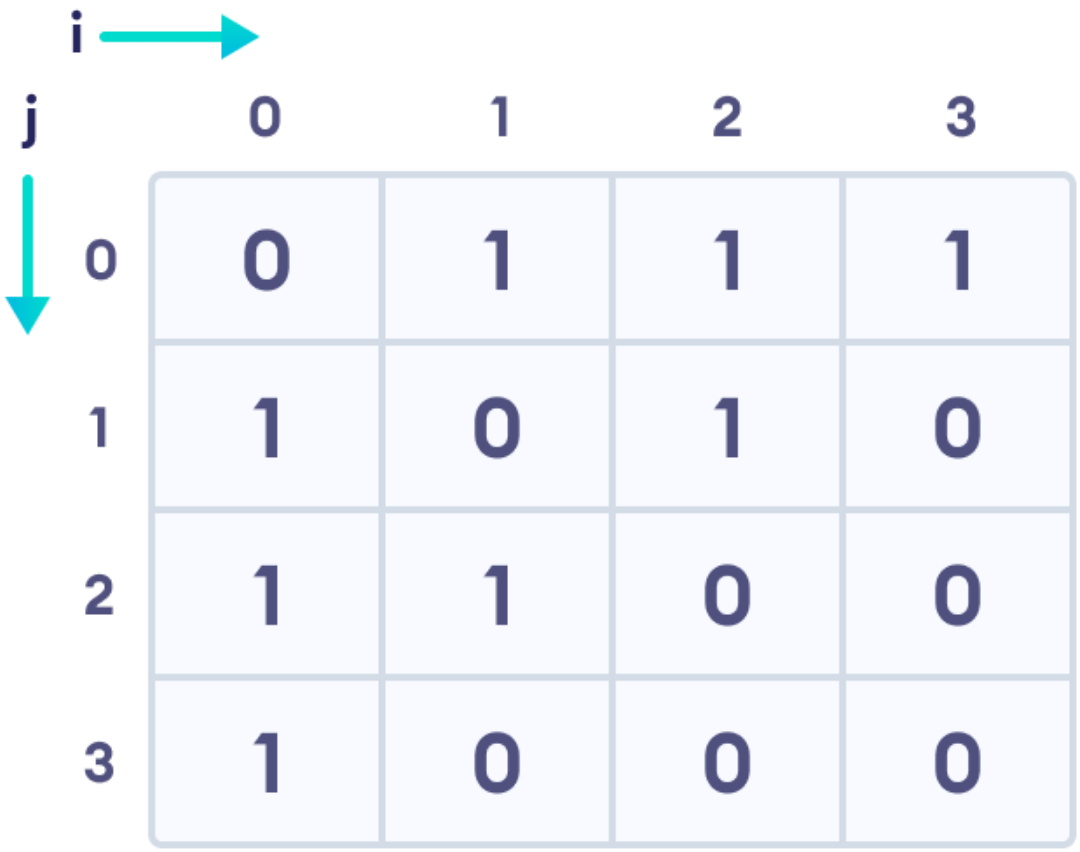
In [154]:

```
class Graph(object):
    def __init__(self, size):
        self.adjMatrix = []
        for i in range(size):
            self.adjMatrix.append([0 for i in range(size)])
        self.size = size

    def add_edge(self, v1, v2):
        if v1 == v2:
            print("Same vertex %d and %d" % (v1, v2))
        self.adjMatrix[v1][v2] = 1
        self.adjMatrix[v2][v1] = 1

    def remove_edge(self, v1, v2):
        if self.adjMatrix[v1][v2] == 0:
            print("No edge between %d and %d" % (v1, v2))
            return
        self.adjMatrix[v1][v2] = 0
        self.adjMatrix[v2][v1] = 0

    def __len__(self):
        return self.size

    def print_matrix(self):
        for row in self.adjMatrix:
            for val in row:
                print('{:4}'.format(val)),
            print


g = Graph(5)
g.add_edge(0, 1)
g.add_edge(0, 2)
g.add_edge(1, 2)
g.add_edge(2, 0)
g.add_edge(2, 3)

g.adjMatrix
```

Out[154]:

```
[[0, 1, 1, 0, 0],
 [1, 0, 1, 0, 0],
 [1, 1, 0, 1, 0],
 [0, 0, 1, 0, 0],
 [0, 0, 0, 0, 0]]
```

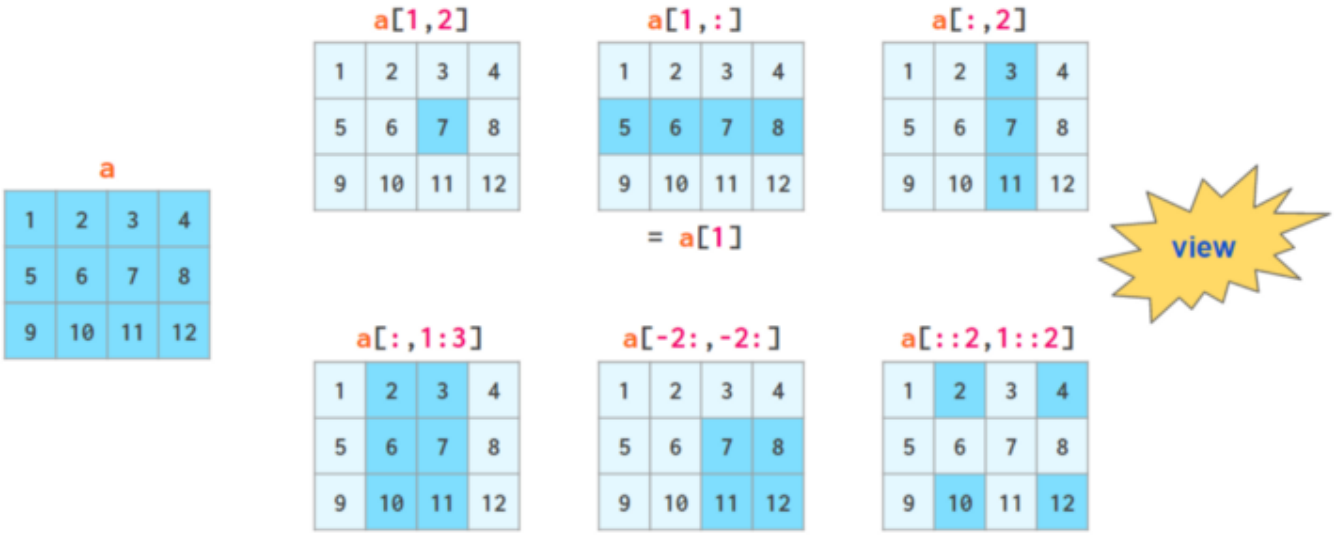

0
3
2
1


i
j
0
1
2
3
0
1
2
3
0 1 1 1
1 0 1 0
1 1 0 0
1 0 0 0

In [39]:

```
a = np.array([[1, 2, 3], [4, 5, 6]], int)
print('0: ', a)
print('1: ', a[0,0])
print('2: ', a[1,0])
print('3: ', a[0,1])
```

```
0:  [[1 2 3]
 [4 5 6]]
1:  1
2:  4
3:  2
```



In [40]:

```
# срезы (сленг - слайсы) с двумерным массивом
print('4: ', a[1,:])
print('5: ', a[:,2])
print('6: ', a[-1:, -2:])
```

```
4:  [4 5 6]
5:  [3 6]
6:  [[5 6]]
```

In [41]:

```
x1 = np.random.randint(10, size=6)  # одномерный массив
x2 = np.random.randint(10, size=(3, 4))  # двумерный
x3 = np.random.randint(10, size=(3, 4, 5))  # трехмерный
```

Характеристики объектов numpy

In [42]:

```
a = np.array([[1, 2, 3], [4, 5, 6]], int)
a
```

Out[42]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [43]:

```
print("a ndim: ", a.ndim) # число измерений
print("a shape:", a.shape) # размерность
print("a size: ", a.size) # итоговый размер массива (сколько всего элементов)
print("itemsize:", a.itemsize, "bytes") # байтовый размер элементов
print("nbytes:", a.nbytes, "bytes") # общий байтовый размер
```

```
a ndim:  2
a shape: (2, 3)
a size:  6
itemsize: 8 bytes
nbytes: 48 bytes
```

In [44]:

```
# тип данных внутри
# напоминаем, массив питру может хранить только один тип данных
a.dtype
```

Out[44]:

```
dtype('int64')
```

In [45]:

```
x1[0] = 3.14159  # усечение до int
x1
```

Out[45]:

```
array([3, 3, 9, 0, 1, 6])
```

Изменение размеров массива

In [46]:

```
a
```

Out[46]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [51]:

```
a = a.reshape(3,2)
```

In [48]:

```
# Обратите внимание, что для того, чтобы это сработало, размер исходного массива должен соответствовать размеру переформируемого массива.
# По возможности метод reshape будет использовать представление исходного массива без копирования, но это не всегда так.
a
```

Out[48]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

Другим распространенным шаблоном переформирования является преобразование одномерного массива в двумерную матрицу строк или столбцов.

Это можно сделать с помощью метода reshape или более просто, используя ключевое слово newaxis в операции среза:

In [49]:

```
a.flatten().shape
```

Out[49]:

```
(6,)
```

In [52]:

```
a.flatten()[np.newaxis, :].shape
```

Out[52]:

```
(1, 6)
```

In [53]:

```
# с помощью этой команды можно вытянуть массив в одномерную "строку"
a.flatten()
```

Out[53]:

```
array([1, 2, 3, 4, 5, 6])
```

In [54]:

```
# обратите внимание, что в процессе изменения размера создан новый массив, а не изменён старый
a
```

Out[54]:

```
array([[1, 2],
       [3, 4],
       [5, 6]])
```

Одна из важных и чрезвычайно полезных вещей, которые необходимо знать о срезах массивов, заключается в том, что они возвращают представления, а не копии данных массива.

Это одна из областей, в которой срезы массивов NumPy отличается от срезов списков Python: в list срезы будут копиями.

Это поведение по-умолчанию на самом деле довольно полезно: оно означает, что при работе с большими наборами данных мы можем получать доступ и обрабатывать их части без необходимости копировать основной буфер данных.

Рассмотрим наш двумерный массив:

In [55]:

```
print(x2)
```

```
[[1 3 4 9]
 [8 6 0 1]
 [6 0 2 0]]
```

In [56]:

```
x2_sub = x2[:2, :2]
print(x2_sub)
```

```
[[1 3]
 [8 6]]
```

In [57]:

```
x2_sub[0, 0] = 99
print(x2_sub)
```

```
[[99  3]
 [ 8  6]]
```

In [58]:

```
print(x2)
```

```
[[99  3  4  9]
 [ 8  6  0  1]
 [ 6  0  2  0]]
```

А можно явно уточнить с помощью метода copy(), что нам нужна копия массива:

In [59]:

```
x2_sub_copy = x2[:2, :2].copy()
print(x2_sub_copy)
```

```
[[99  3]
 [ 8  6]]
```

In [60]:

```
x2_sub_copy[0, 0] = 42
print(x2_sub_copy)
```

```
[[42  3]
 [ 8  6]]
```

In [61]:

```
print(x2)
```

```
[[99  3  4  9]
 [ 8  6  0  1]
 [ 6  0  2  0]]
```

Конкатенация массивов

Конкатенация, или объединение двух массивов в NumPy, в основном выполняется с помощью процедур np.concatenate, np.vstack и np.hstack. np.concatenate принимает кортеж или список массивов в качестве первого аргумента, как мы видим здесь:

In [62]:

```
x = np.array([1, 2, 3])
y = np.array([3, 2, 1])
np.concatenate([x, y])
```

Out[62]:

```
array([1, 2, 3, 3, 2, 1])
```

Также можно конкатенировать более двух массивов одновременно:

In [63]:

```
z = [99, 99, 99]
print(np.concatenate([x, y, z]))
```

```
[ 1  2  3  3  2  1 99 99 99]
```

In [64]:

```
grid = np.array([[1, 2, 3],
                 [4, 5, 6]])
```

In [65]:

```
# конкатенация по первому измерению
print(np.concatenate([grid, grid]))
print(np.concatenate([grid, grid], axis=0).shape)
```

```
[[1 2 3]
 [4 5 6]
 [1 2 3]
 [4 5 6]]
(4, 3)
```

In [66]:

```
# конкатенация по второму измерению
print(np.concatenate([grid, grid], axis=1))
print(np.concatenate([grid, grid], axis=1).shape)
```

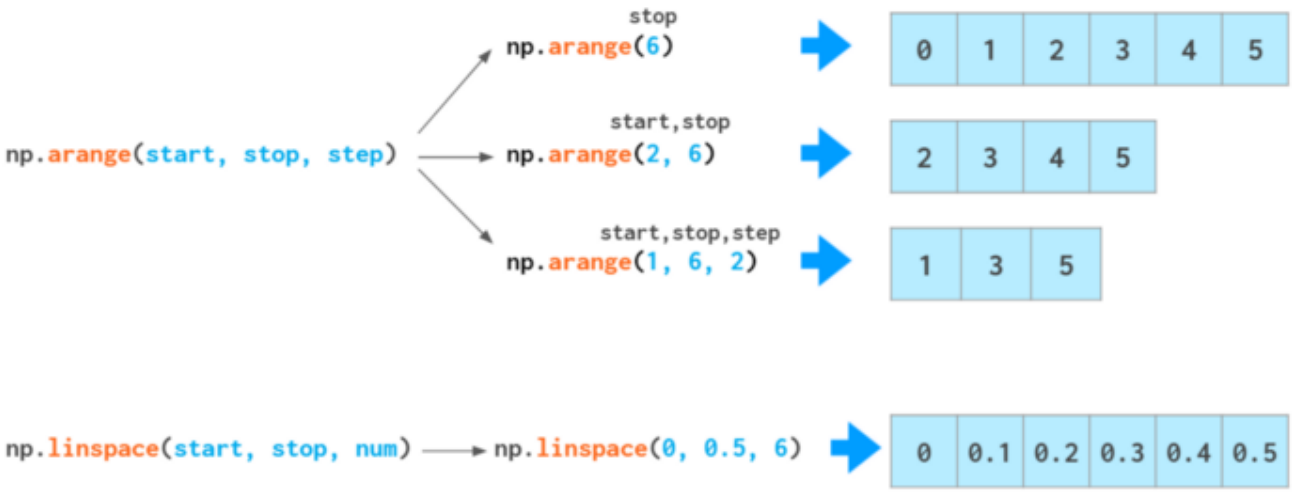
```
[[1 2 3 1 2 3]
 [4 5 6 4 5 6]]
(2, 6)
```

Создание по-разному заполненных массивов

In [67]:

```
# аналог range для массивов
print(np.arange(5))
print(np.arange(1, 6, 2))
```

```
[0 1 2 3 4]
[1 3 5]
```



In [68]:

```
np.ones(shape=(2,3))
```

Out[68]:

```
array([[1., 1., 1.],
       [1., 1., 1.]])
```

In [ ]:

```
np.zeros((5,4))
```

In [ ]:

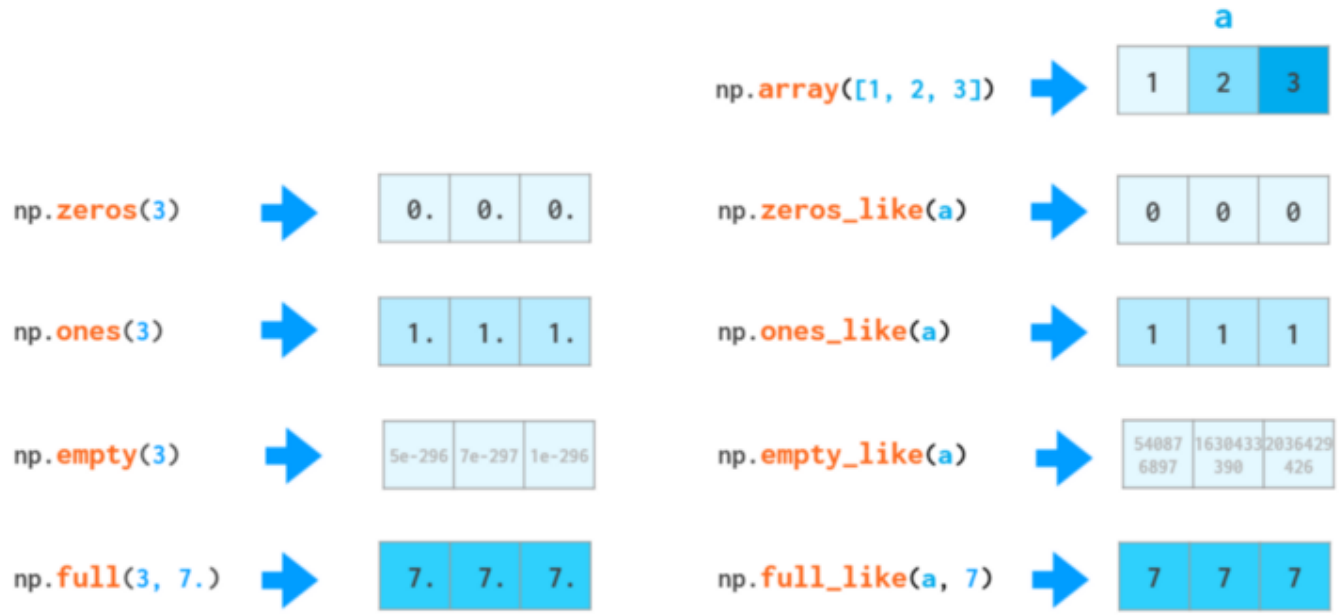
```
np.identity(4)
```

In [71]:

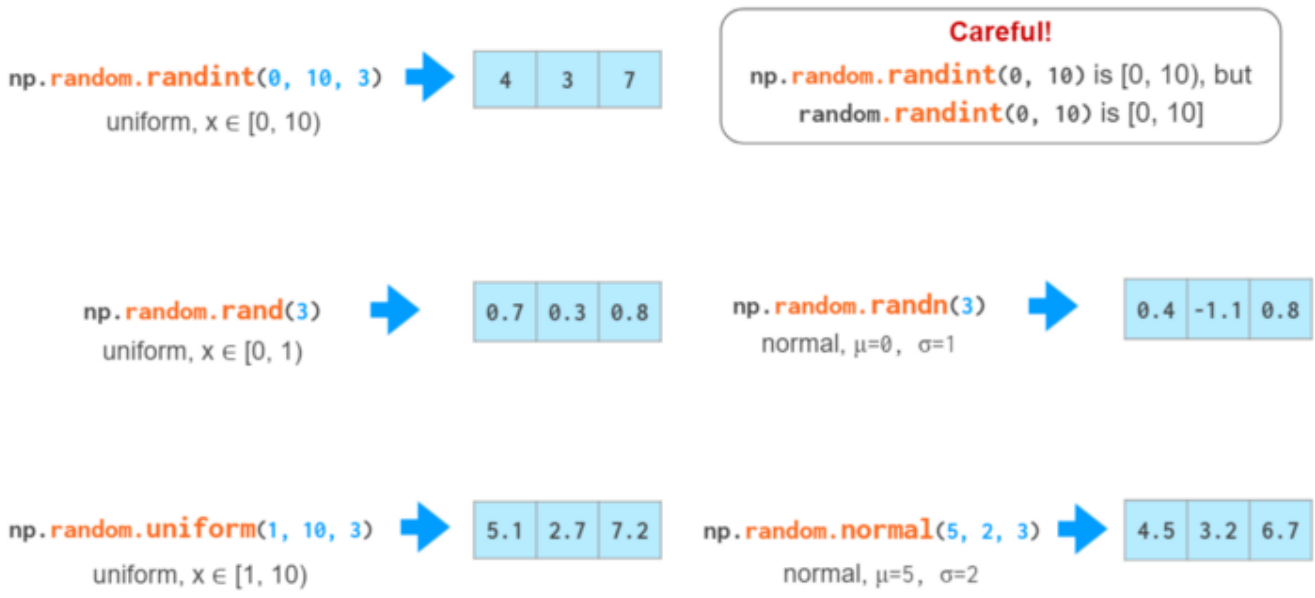
```
# k - номер диагонали, заполненный единицами
np.eye(5,4, k=2)
```

Out[71]:

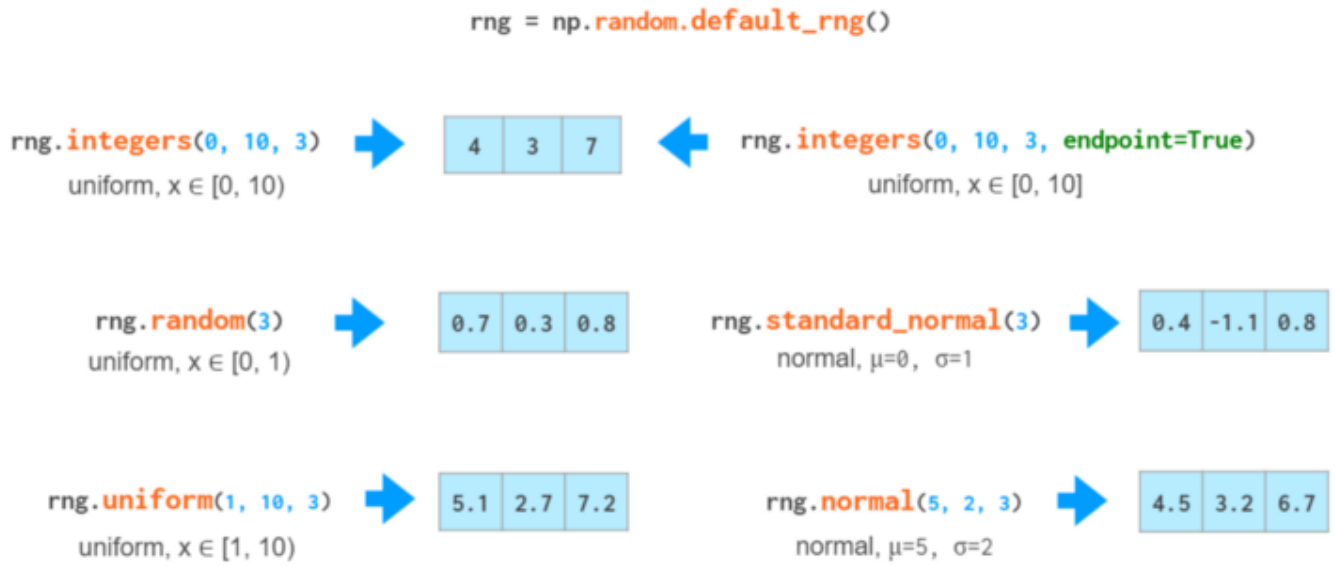
```
array([[0., 0., 1., 0.],
       [0., 0., 0., 1.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```



Генерация случайных чисел



В новых версиях numpy такая генерация скоро исчезнет и останется только новый способ



Перебор элементов массива

In [72]:

```
a = np.array([1, 4, 5], int)
```

In [73]:

```
# простой перебор для одномерного случая
for x in a:
    print(x)
```

```
1
4
5
```

In [74]:

```
# простой перебор для многомерного случая сработает плохо, он перебирает по первой разм
ерности
a = np.array([[1, 2], [3, 4], [5, 6]], float)
for x in a:
    print(x)
```

```
[1. 2.]
[3. 4.]
[5. 6.]
```

In [75]:

```
# перебор правильным способом
for x in range(a.shape[0]):
  for y in range(a.shape[1]):
    print(a[x, y])
```

```
1.0
2.0
3.0
4.0
5.0
6.0
```

Операции над массивами

Математические операции над массивами

С массивами можно применять стандартные математические операции. Они будут работать так, как будто происходит поэлементная работа одной и той же операции. Для матричных операций есть специальные команды.

Стандартные математические операции применимы только к массивам одинаковых размеров

In [76]:

```
a = np.arange(1, 4, 1, dtype=int)
b = np.arange(6, 9, 1, dtype=int)
print('a: ', a)
print('b: ', b)
```

```
a:  [1 2 3]
b:  [6 7 8]
```

In [77]:

```
a + b
```

Out[77]:

```
array([ 7,  9, 11])
```

In [78]:

```
a - b
```

Out[78]:

```
array([-5, -5, -5])
```

In [79]:

```
a * b
```

Out[79]:

```
array([ 6, 14, 24])
```

In [80]:

```
b / a
```

Out[80]:

```
array([6.        , 3.5       , 2.66666667])
```

In [81]:

```
a % b
```

Out[81]:

```
array([1, 2, 3])
```

In [82]:

```
b**a
```

Out[82]:

```
array([  6,  49, 512])
```

In [84]:

```
a // b
```

Out[84]:

```
array([0, 0, 0])
```

Кроме того, поэлементно могут быть применены другие математические операции

In [83]:

```
# корень
np.sqrt(a)
```

Out[83]:

```
array([1.        , 1.41421356, 1.73205081])
```

In [85]:

```
a = np.array([1.1, 1.5, 1.9], float)
```

In [86]:

```
# округление вниз
np.floor(a)
```

Out[86]:

```
array([1., 1., 1.])
```

In [87]:

```
# округление вверх
np.ceil(a)
```

Out[87]:

```
array([2., 2., 2.])
```

In [88]:

```
# округление по правилам математики
np.rint(a)
```

Out[88]:

```
array([1., 2., 2.])
```

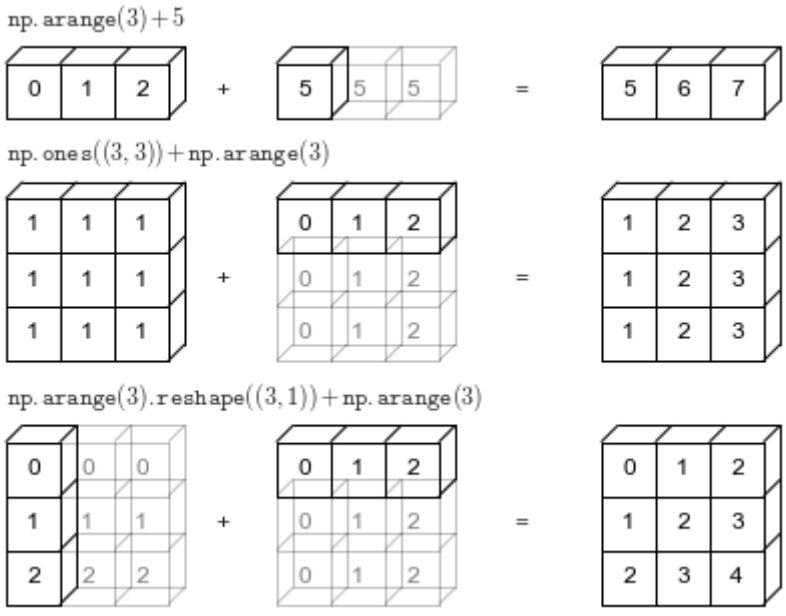
Также можно умножать, делить и т.д. на одно число (array broadcasting).

In [89]:

```
a * 5
```

Out[89]:

```
array([5.5, 7.5, 9.5])
```



Простые операции над массивами

Одномерные массивы

In [90]:

```
a = np.arange(1, 6, 1)
print(a)
print('Сумма: ', a.sum())
print('Перемножение: ', a.prod())
```

```
[1 2 3 4 5]
Сумма:  15
Перемножение:  120
```

In [91]:

```
# среднее (математическое ожидание)
a.mean()
```

Out[91]:

```
3.0
```

In [92]:

```
# дисперсия (смещенная - это будет важно в дальнейшем)
a.var()
```

Out[92]:

```
2.0
```

In [93]:

```
# стандартное отклонение (несмещенное)
a.std()
```

Out[93]:

```
1.4142135623730951
```

In [94]:

```
a.min()
```

Out[94]:

```
1
```

In [95]:

```
a.argmin()
```

Out[95]:

```
0
```

In [96]:

```
# clip позволяет "отрезать" значения сверху и снизу
a = np.array([6, 2, 5, -1, 0, 6, 2, 5, 4], float)
a.clip(0, 5)
```

Out[96]:

```
array([5., 2., 5., 0., 0., 5., 2., 5., 4.])
```

In [97]:

```
np.unique(a)
```

Out[97]:

```
array([-1.,  0.,  2.,  4.,  5.,  6.])
```

Многомерные массивы

Для работы с многомерными массивами можно использовать параметр `axis` .

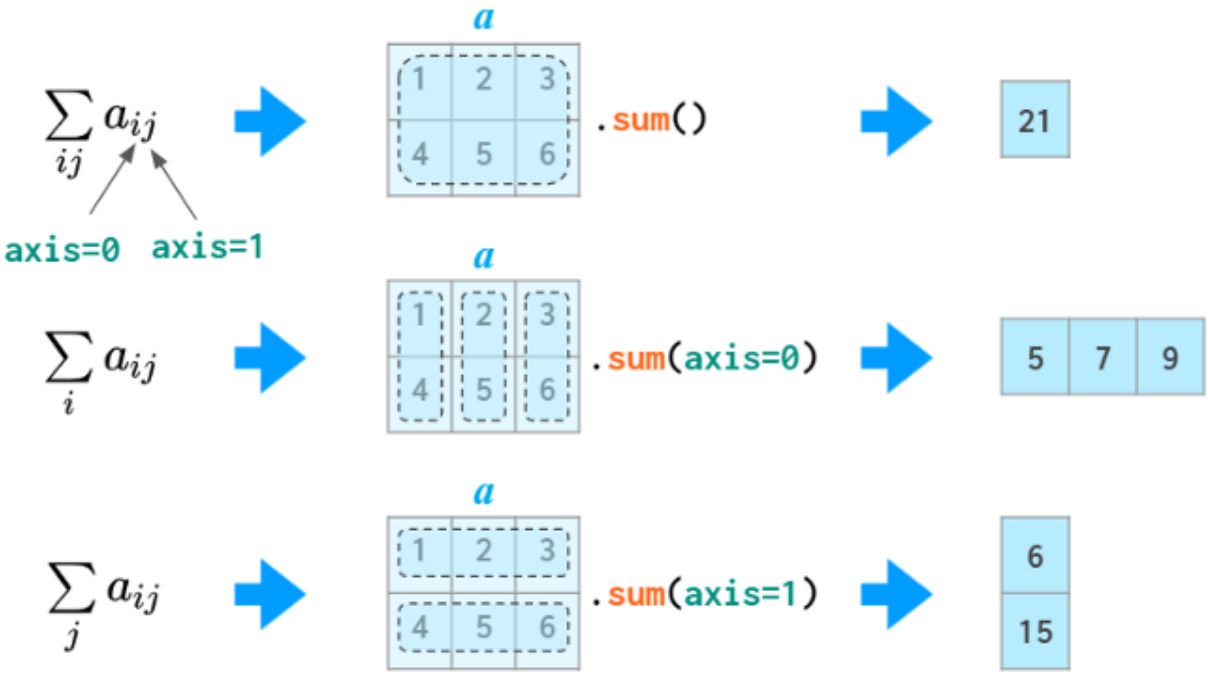
In [98]:

```
a = np.array([[5, 2], [4, 1], [3, -1]])
print(a)
print(a.mean(axis=0))
print(a.mean(axis=1))
a.mean()
```

```
[[ 5  2]
 [ 4  1]
 [ 3 -1]]
[4.         0.66666667]
[3.5 2.5 1. ]
```

Out[98]:

```
2.3333333333333335
```



Логические операции над массивами

In [99]:

```
a = np.array([1, 3, 0])
b = np.array([0, 3, 2])

print(a > b, type(a>b))
```

```
[ True False False] <class 'numpy.ndarray'>
```

In [100]:

```
c = a > 2
c
```

Out[100]:

```
array([False,  True, False])
```

In [101]:

```
# проверяем, что хотя бы один элемент истинен
print(any(c))
# проверяем, что все элементы истинны
print(all(c))
```

```
True
False
```

Если вы хотите провести сравнение логическим И или лолгическим ИЛИ, то необходимо воспользоваться специальнымыми методами:

```
np.logical_and(_, _)
np.logical_or(_, _)
np.logical_not(_)
```

In [102]:

```
(a < 3) * (a > 0)
```

Out[102]:

```
array([ True, False, False])
```

In [103]:

```
np.logical_and(a > 0, a < 3)
```

Out[103]:

```
array([ True, False, False])
```

С помощью `np.where` можно создать массив на основании условий. Синтаксис:

```
where(boolarray, truearray, falsearray)
```

In [104]:

```
a = np.array([1, 3, 0])
a
```

Out[104]:

```
array([1, 3, 0])
```

In [105]:

```
np.where(a != 0, 1 / a, a)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWar
ning: divide by zero encountered in true_divide
  """Entry point for launching an IPython kernel.
```

Out[105]:

```
array([1.        , 0.33333333, 0.        ])
```

Можно проверять элементы массива на наличие NaN и бесконечностей.

In [106]:

```
a = np.array([1, np.NaN, np.Inf], float)
a
```

Out[106]:

```
array([ 1., nan, inf])
```

In [107]:

```
np.isnan(a)
```

Out[107]:

```
array([False,  True, False])
```

In [108]:

```
np.isfinite(a)
```

Out[108]:

```
array([ True, False, False])
```

Выбор элементов массива по условию

Очень важной особенностью массивов является то, что элементы в них можно брать на основании других массивов. Это особенно полезно при реализации свёрточных нейронных сетей.

In [109]:

```
# это результат применения логической операции к многомерному массиву
a = np.array([[6, 4], [5, 9]], float)
a >= 6
```

Out[109]:

```
array([[ True, False],
       [False,  True]])
```

In [110]:

```
# а это результат фильтрации элементов
# обратите внимание, получился одномерный массив, содержащий только элементы, удовлетво
ряющие условию
a[a >= 6]
```

Out[110]:

```
array([6., 9.])
```

In [111]:

```
a[np.logical_and(a > 5, a < 9)]
```

Out[111]:

```
array([6.])
```

Обратите внимание, что если передать целочисленные значения, в качестве условий, то результат будет другой. Будут выбраные соответствующие элементы.

In [112]:

```
a = np.array([2, 4, 6, 8], float)
b = np.array([0, 0, 1, 3, 2], int)
a[b]
```

Out[112]:

```
array([2., 2., 4., 8., 6.])
```

In [113]:

```
# Для выбора значений из многомерных массивов необходимо передать массивы, которые опре
деляют индексы по каждому из направлений. Они должны быть, естественно, целочисленными.
a = np.array([[1, 4], [9, 16]], float)
b = np.array([0, 0, 1, 1, 0], int)
c = np.array([0, 1, 1, 1, 1], int)
a[b,c]
```

Out[113]:

```
array([ 1.,  4., 16., 16.,  4.])
```

Векторная и матричная математика с использованием numpy

Векторная математика в numpy - это главная причина того, что numpy стал ключевым модулем Python среди всех представленных модулей. Векторные вычисления позволяют значительно ускорить обработку численной информации.

Часто сравнивая Python с C++/C говорят том, что первый гораздо менее производителен. Но с учётом современных модулей верно следующее утверждение: хорошо написанная программа на Python будет производительнее, чем средняя программа на C/C++, хорошую программу на C/C++ написать крайне сложно.

В дополнении к арифметически операциям мы рассмотрим некоторые векторные операции. Глубокое погружение в их суть требует значительных знаний математики, которые мы будем получать по мере необходимости.

Скалярное произведение

Для двух векторов a и b одинаковой длины скалярное произведение считается по следующей формуле:

$$a * b = \sum_{i=0}^{len(a)} a_i * b_i$$

In [114]:

```
# скалярное произвдение векторов, также операция свёртки в свёрточных нейронных сетях

a = np.array([1, 2, 3], float)
b = np.array([0, 1, 1], float)
np.dot(a, b)
```

Out[114]:

```
5.0
```

In [115]:

```
a @ b
```

Out[115]:

```
5.0
```

Произведение матриц

Произведение матриц - это особая математическая операция, которая не эквивалентна произведени соответствующих элементов матриц. О матричном произведении целесообразно говорить в рамках соответствующих разделов математики. Тем не менее, используя numpy легко получить матричное произведение.

In [116]:

```
a = np.array([[0, 1], [2, 3]], float)
b = np.array([2, 3], float)
d = np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]], float)
```

In [117]:

```
np.dot(b, a)
```

Out[117]:

```
array([ 6., 11.])
```

In [118]:

```
np.dot(a, b)
```

Out[118]:

```
array([ 3., 13.])
```

In [119]:

```
# следите за размерностью, иначе ничего не получится
np.dot(b, d)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
<ipython-input-119-024e0aa2161c> in <module>
      1 # следите за размерностью, иначе ничего не получится
----> 2 np.dot(b, d)

<__array_function__ internals> in dot(*args, **kwargs)

ValueError: shapes (2,) and (3,3) not aligned: 2 (dim 0) != 3 (dim 0)
```

Определитель матриц

Многие математические операции, связанные с линейной алгеброй реализованы в модуле linalg внутри numpy. Мы не будем углублбляться в различные функциия модуля, рассмотрим для примера определитель.

In [120]:

```
np.linalg.det(a)
```

Out[120]:

```
-2.0
```

Дополнительный материал для желающих

Нескучный туториал по numpy (https://habr.com/ru/post/469355/)

Задания

Задача 1.

Создать матрицу размером `10x10` с 0 внутри, и 1 на границах. Например для `3x3`.

```
1 1 1
1 0 1
1 1 1
```

In [ ]:

```

```

Задача 2.

Создать 5x5 матрицу с 1,2,3,4 над диагональю. Все остальные элементы - 0.

In [ ]:

Задача 3.

Создайте случайную матрицу и вычтите из каждой строки среднее.

In [ ]:

Задача 4.

Написать функцию, принимающую на вход массив и меняющую знак у элементов, значения которых между 3 и 8. Протестировать на нескольких заданных вами массивах.

In [ ]:

Задача 4.

Написать функцию, принимающую на вход массив и вычитающую среднее из каждой строки в матрице. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 5.

Дан вектор [1, 2, 3, 4, 5], построить новый вектор с тремя нулями между каждым значением.

In [ ]:

Задача 6.

Написать функцию, принимающую на вход матрицу MxN и меняющую 2 любые строки в матрице. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 7.

Написать функцию, принимающую на вход одномерный массив и возвращающую наиболее частое значение в массиве и частоту его встречи. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 8.

Написать функцию, принимающую на вход массив 16x16 и считающую сумму по блокам 4x4. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 9.

Написать функцию, принимающую на вход матрицу и возвращающую n наибольших значений в массиве. n вводится с клавиатуры. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 10.

Написать функцию, принимающую на вход 10x3 матрица и находящую строки из неравных значений (например [2,2,3]). Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 11.

Написать функцию, принимающую на вход двумерный массив и находящую все различные строки. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 12.

Написать функцию, принимающую на вход два вектора одинакого размера и считающую расстояние между векторами. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 13.

Написать функцию, принимающую на вход два вектора одинакого размера и находящую косинус угла между векторами. Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 14.

Написать функцию, принимающую на вход вектор A содержит float числа как больше, так и меньше нуля.

Функция должна округлить их до целых и результат записать в глобальную переменную Z. Округление должно быть "от нуля", т.е.:

- положительные числа округляем всегда вверх до целого
- отрицательные числа округляем всегда вниз до целого
- 0 остаётся 0

Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 15.

Написать функцию, принимающую на вход 2 вектора целых чисел A и B.

Функция должна находить числа, встречающиеся в обоих векторах и добавлять их по возрастанию в глобальную переменную - вектор Z.

Если пересечений нет, то вектор Z будет пустым.

Протестировать на нескольких заданных вами примерах.

In [ ]:

Задача 16.

Написать функцию, принимающую на вход вектор и возвращающую максимальный элемент в векторе среди элементов, перед которыми стоит 0.

Например для:

```
x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0])
```

Ответ: 5

In [ ]:

Задача 17.

Написать функцию, принимающую на вход матрицу 5x3 и считающую длину каждого вектора в матрице (строка) и ищущую самый длинный вектор, вернуть его координаты и длину.

Как выглядит матрица:

```
| x | y | z |
| 1 | 2 | 3 |
| 3 | 4 | 1 |
| ...       |
```