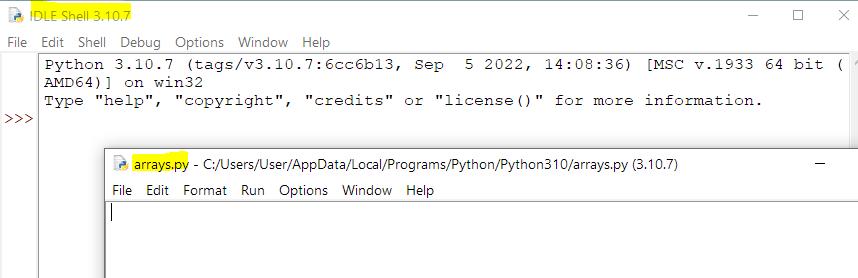
**Работа с массивами в Python**

**Маленький практикум**

**1. Начало работы**

Запускаем командную оболочку Python IDLE Shell. Далее в меню оболочки набираем File -> New File и создаем файл arrays.py. В результате получим:



**2. Списки и кортежи**

В Python есть свои структуры для хранения массивов данных — это **списки** (list) и **кортежи** (tuple [ˈtjuːp(ə)l]).

2.1. **Список** – это упорядоченный изменяемый (mutable) массив объектов. Объекты в списке могут иметь любые типы. Объекты разделяются запятыми и размещаются внутри квадратных скобок [ ].

Введем в файл arrays.py и запустим (Run->Run Module) следующий код:

Const=2

#списки

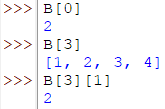
A = [1, 2, 3, 4]

B = [2, Const, -0.1, A, True]

Теперь все эти данные находятся в памяти оболочки. Посмотрим, как выглядит список B в командной строке оболочки:

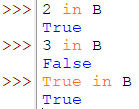


Индексация объектов в списке начинается с нуля:



В последней строчке извлекаем элемент с индексом 1 из объекта с индексом 3, который также является списком.

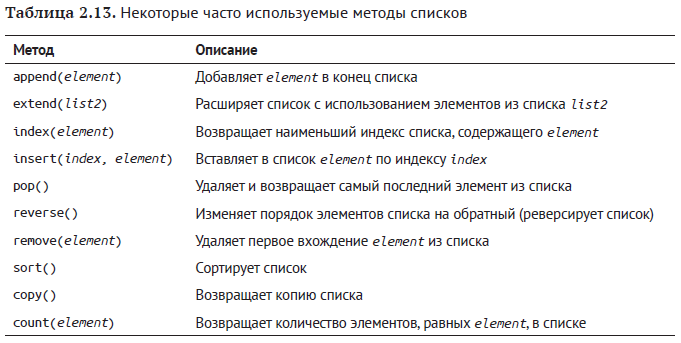
Проверка наличия элемента в списке:



Из списка можно вырезать группу элементов:



Некоторые часто используемые методы списков:



Список – изменяемая структура данных, так как есть возможность изменять размерность списка, можно добавить элемент в конец списка или удалить из него последний элемент (методами append() и pop()).

Пустой список вводится так: А=[ ]

2.2. **Кортеж** – это упорядоченный неизменяемый массив объектов, то есть его размерность определена при инициализации и далее не меняется. Объекты в кортеже могут иметь любые типы. Объекты разделяются запятыми и размещаются внутри круглых скобок ( ).

Добавим в файл arrays.py следующий код:

#кортежи

T1 = (1,2,3,4,5,6,7)

T2 = 1,2,3,4,5,6,7 #кортеж можно инициализировать и без скобок

G=(123,) # кортеж из одного элемента - ставим после него запятую!

В оболочке посмотрим, что представляет собой Т2:



**3. Векторы, матрицы и их инициализация**

Задавать векторы и матрицы удобнее не в виде списков, а с помощью методов-конструкторов библиотеки Numpy. Подключим Numpy, записав в файле arrays.py:

import numpy as np

**Примеры конструкторов**

**Конструктор array**

#Применение конструктора для создания вектора а и матрицы b

#из списка или кортежа

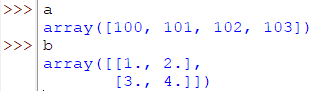
import numpy as np

a = np.array( (100, 101, 102, 103) ) #вектор из кортежа

b = np.array( [[1.,2.], [3.,4.]] ) #матрица из двух списков

Допишем эти три строки в array.py и запустим.

Выведем в оболочке вектор а и матрицу b:



Видно, что матрица — это список из двух списков.

Выведем элемент матрицы b(0,1) (индексация начинается с нуля)



и второй элемент вектора а(1)



**Конструктор range**

a = range(5) # 0, 1, 2, 3, 4

b = range(1, 6) # 1, 2, 3, 4, 5

c = range(0, 6, 2) # 0, 2, 4

d = range(10, 0, -2) # 10, 8, 6, 4, 2

**Конструкторы matrix, zeros, ones, eye**

#матрица

M1 = np.matrix([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

#матрица в стиле MATLAB - это должна быть строка!

M2 = np.matrix('1 2 3; 4 5 6; 7 8 9')

z0 = np.zeros((3,2)) #нулевая матрица 3х2

z1 = np.ones((3,3))  #матрица из единиц 3х3

E = np.eye(3, 3) #единичная матрица 3х3

**Конструктор linspace**

#генерация последовательности x равномерно распределенных значений

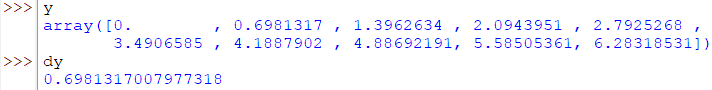
x = np.linspace(1, 20, 5)

# генерация последовательности

# и вывод шага этой последовательности

y, dy = np.linspace(0., 2\*np.pi, 10, retstep=True)

Последовательность y и ее шаг



**Конструктор fromfunction**

#генерация последовательности с помощью функции f

def f(i, j):

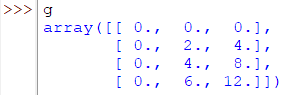
return 2 \* i \* j #функция должна быть размещена в тексте

#программы до ее вызова.

#Важно: отступ перед return 4 пробела

g = np.fromfunction(f,(4,3))

Результат:



**Использование цикла for**

#заполнение массивов

ax, ay = [], []

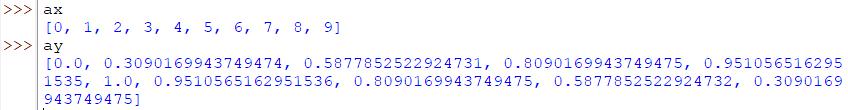
for i in range(10):

#добавляем в конец элемент i

ax.append(i)

#добавляем в конец элемент sin(i\*pi/10)

ay.append(np.sin(i\*np.pi/10.))



**4. Прочие операции с матрицами**

А = B \* 5 #умножение на скаляр

C = A.dot(B) #С = А \* B

C = np.dot(A, B) #С = А \* B

C = A \* B #Внимание! Это - поэлементное умножение

AT = A.transpose() #транспонирование

**5. Решение систем алгебраических уравнений**

# Ах=b

А = np.array([[3, -2, 0], [-2, 1, -3], [4, 6, 1]])

b = np.array([8, -20, 7])

x = np.linalg.solve(A, b)

**6. Собственные числа и собственные векторы матрицы**

# vals – собственные числа, vecs – матрица собственных векторов

vals, vecs = np.linalg.eig(A)

**7.Собственные частоты и формы колебаний**

import scipy.linalg as la

# Matlab: [V, D] = eig(C,A)

D, V = la.eig(C,A)