Лекция 5

Матричные вычисления. Регулярные выражения

1 Массивы (реализация через списки)

```
1   A = [[1, 4, 5, 12],
2        [-5, 8, 9, 0],
3        [-6, 7, 11, 19]]
4
5   print("A =", A)
6   print("A[1] =", A[1])
7   print("A[1][2] =", A[1][2])
8   print("A[0][-1] =", A[0][-1])
9
10  column = [];
11  for row in A:
12      column.append(row[2])
13
14  print("2ой столбец =", column)
```

```
A = [[1, 4, 5, 12], [-5, 8, 9, 0], [-6, 7, 11, 19]]
A[1] = [-5, 8, 9, 0]
A[1][2] = 9
A[0][-1] = 12
2ой столбец = [5, 9, 11]
```

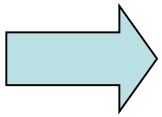
```
[[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8], [0, 3, 6, 9, 12], [0, 4, 8, 12, 16]]

[0, 0, 0, 0, 0]
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8]
[0, 3, 6, 9, 12]
[0, 4, 8, 12, 16]

0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12
0 4 8 12 16
```

```python
d1 = []

for j in range(5):
    d2 = []
    for i in range(5):
        d2.append(i*j)
    d1.append(d2)

print(d1)
print()

for j in range(5):
    print( d1[j])
print()

for j in range(5):
    for i in range(5):
        print( d1[i][j], end=" ")
    print()
```

2 Массивы (библиотека array)

Библиотека array:

```
1  from array import *
2
3  # вывод значений массива
4  def Print(d):
5      for i in d:
6          print(i, end=' ')
7      print()
8
9  # создание целочисленного массива
10 # 'i' - целое знаковое число
11 data = array('i', [0, 10, 20, 30, 40])
12 Print(data)
13
14 # добавление элемента
15 data.insert(3, 77)
16 Print(data)
17
18 # удаление элемента
19 data.pop(2)
20 Print(data)
```



```
0 10 20 30 40
0 10 20 77 30 40
0 10 77 30 40
```

3 Массивы (пакет NumPy)

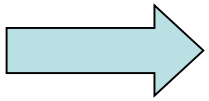
Создание массивов:

```python
import numpy as np

# одномерный массив
A = np.array([1, 2, 3])
print("A = ", A)

# двумерный целочисленный массив
B = np.array([[1, 2, 3], [3, 4, 5]])
print('B = \n', B)

# двумерный вещественный массив
C = np.array([[1.1, 2, 3], [3, 4, 5]])
print("C = \n", C)

# двумерный массив комплексных чисел
D = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex)
print("D = \n", D)

# массив нулей
Z = np.zeros( (2, 3) )
print("Z = \n", Z)
# массив единиц
O = np.ones( (1, 5), dtype=np.int32 )
print("O = \n", O)

# массив из чисел 0,1,2,...
X = np.arange(4)
print('X =', X)

Y = np.arange(12).reshape(2, 6)
print('Y = \n', Y)
D1 = D.reshape(3, 2)
print('D1 = \n', D1)
```

```
A =  [1 2 3]
B =
 [[1 2 3]
 [3 4 5]]
C =
 [[ 1.1  2.   3. ]
 [ 3.   4.   5. ]]
D =
 [[ 1.+0.j  2.+0.j  3.+0.j]
 [ 3.+0.j  4.+0.j  5.+0.j]]
Z =
 [[ 0.  0.  0.]
 [ 0.  0.  0.]]
O =
 [[1 1 1 1 1]]
X = [0 1 2 3]
Y =
 [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]]
D1 =
 [[ 1.+0.j  2.+0.j]
 [ 3.+0.j  3.+0.j]
 [ 4.+0.j  5.+0.j]]
```

Операции с матрицами:

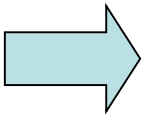
```python
import numpy as np

A = np.array([[2, 4], [5, -6]])
B = np.array([[9, -3], [3, 6]])

C = A + B
print('A + B = \n', C)

# умножение 1
C = A.dot(B)
print('1) A * B = \n', C)

# умножение 2
C = A * B
print('2) A * B = \n', C)

# транспонирование
print("AT = \n", A.transpose())
```



```
A + B =
 [[11  1]
 [ 8  0]]
1) A * B =
 [[ 30  18]
 [ 27 -51]]
2) A * B =
 [[ 18 -12]
 [ 15 -36]]
AT =
 [[ 2  5]
 [ 4 -6]]
```

Доступ к элементам массива:

```
1  import numpy as np
2
3  # доступ к элементам
4  A = np.array([2, 4, 6, 8, 10])
5  print("A[0] =", A[0])
6  print("A[-2] =", A[-2])
7
8  A = np.array([[1, 4, 5, 12],
9                [-5, 8, 9, 0],
10               [-6, 7, 11, 19]])
11 print("A[1][1] =", A[1][1])
12 print("A[-1][-1] =", A[-1][-1])
13
14 # доступ к строкам
15 print("A[2] =", A[2])
16 print("A[-2] =", A[-2])
17
18 # доступ к столбцам
19 print("A[:,1] =",A[:,1])
20 print("A[:,-1] =", A[:,-1])
```



```
A[0] = 2
A[-2] = 8
A[1][1] = 8
A[-1][-1] = 19
A[2] = [-6  7 11 19]
A[-2] = [-5  8  9  0]
A[:,1] = [4 8 7]
A[:,-1] = [12  0 19]
```

Срезы:

```python
import numpy as np
M = np.array([1, 3, 5, 7, 9, 7, 5])

# с 2-го по 4-ый элементы
print(M[2:5])

# с 0-го по 3-ый элементы
print(M[:-5])

# с 5-го до последнего элемента
print(M[5:])

# с 0-го до последнего элемента
print(M[:])

# в обратном порядке
print(M[::-1])

# срез с шагом
print(M[1:-2:2])
```

```
[5 7 9]
[1 3]
[7 5]
[1 3 5 7 9 7 5]
[5 7 9 7 5 3 1]
[3 7]
```

```
A[:2, :4] =
 [[ 1  4  5 12]
 [-5  8  9  0]]
A[:1,] =
 [[ 1  4  5 12 14]]
A[:,2] =
 [ 5  9 11]
A[:, 2:5] =
 [[ 5 12 14]
 [ 9  0 17]
 [11 19 21]]
```

```python
import numpy as np

A = np.array([[1, 4, 5,  12, 14],
              [-5, 8, 9,  0,  17],
              [-6, 7, 11, 19, 21]])

# две строки, четыре столбца
print("A[:2, :4] = \n", A[:2, :4])

# первая строка, все столбцы
print("A[:1,] = \n", A[:1,])

# все строки, второй столбец
print("A[:,2] = \n", A[:,2])

# все строки, со второго по четвертый столбец
print("A[:, 2:5] = \n", A[:, 2:5])
```

4 Регулярные выражения

Регулярное выражение — это строка, задающая шаблон поиска подстрок в тексте

Спецсимволы, являющиеся управляющими конструкциями:

```
.  ^  $  *  +  ?  {  }  [  ]  |  (  )
```

Шаблоны, соответствующие одному символу (1)

Шаблон	Описание	Пример	Применяем к тексту
.	Один любой символ, кроме новой строки \n.	м.л.ко	молоко, малако, Им0л0коИхлеб
\d	Любая цифра	СУ\d\d	СУ35, СУ111, АЛСУ14
\D	Любой символ, кроме цифры	926\D123	926)123, 1926-1234
\s	Любой пробельный символ (пробел, табуляция, конец строки и т.п.)	бор\soda	бор ода, бор ода, борода
\S	Любой непробельный символ	\S123	X123, я123, !123456, 1 + 123456
\w	Любая буква (то, что может быть частью слова), а также цифры и _	\w\w\w	Год, f_3, qwert
\W	Любая не буква, не цифра и не подчёркивание	сом\W	сом!, сом?

\d≈[0-9], \D≈[^0-9], \w≈[0-9a-zA-Z, а-яА-ЯёЁ], \s≈[\f\n\r\t\v]

Шаблоны, соответствующие одному символу (2)

Шаблон	Описание	Пример	Применяем к тексту
[..]	Один из символов в скобках, а также любой символ из диапазона a-b	[0-9][0-9A-Fa-f]	12, 1F, 4B
[^..]	Любой символ, кроме перечисленных	<[^>]>	<1>, <a>, <>
[abc-], [-1]	если нужен минус, его нужно указать последним или первым		
[*[(+\\\]\t]	внутри скобок нужно экранировать только] и \		
\b	Начало или конец слова (слева пусто или не буква, справа буква и наоборот)	\bвал	вал, перевал, Перевалка
\B	Не граница слова: либо и слева, и справа буквы, либо и слева, и справа НЕ буквы	\Bвал	перевал, вал, Перевалка
		\Bвал\B	перевал, вал, Перевалка

Квантификаторы (указание количества повторений)

Шаблон	Описание	Пример	Применяем к тексту
{n}	Ровно n повторений	\d{4}	1, 12, 123, 1234, 12345
{m,n}	От m до n повторений включительно	\d{2,4}	1, 12, 123, 1234, 12345
{m,}	Не менее m повторений	\d{3,}	1, 12, 123, 1234, 12345
{,n}	Не более n повторений	\d{,2}	1, 12, 123
?	Ноль или одно вхождение, синоним {0,1}	валы?	вал, валы, валов
*	Ноль или более, синоним {0,}	СУ\d*	СУ, СУ1, СУ12, ...
+	Одно или более, синоним {1,}	a\)+	a), a)), a))), ba)])

5 Модуль re

Функция	Её смысл
re.search(pattern, string)	Найти в строке string первую строчку, подходящую под шаблон pattern;
re.fullmatch(pattern, string)	Проверить, подходит ли строка string под шаблон pattern;
re.split(pattern, string, maxsplit=0)	Аналог str.split(), только разделение происходит по подстрокам, подходящим под шаблон pattern;
re.findall(pattern, string)	Найти в строке string все шаблоны pattern (получаем список);
re.finditer(pattern, string)	Найти в строке string шаблоны pattern (выдаются match-объекты);
re.sub(pattern, repl, string, count=0)	Заменить в строке string все шаблоны pattern на repl; count>0 – количество замен

match-объекты

Метод	Описание
match.group()	Подстрока, соответствующая шаблону
match.start()	Индекс в исходной строке, начиная с которого идёт найденная подстрока
match.end()	Индекс в исходной строке, который следует сразу за найденной подстрокой

```
Строка 4: <_sre.SRE_Match object; span=(9, 14), match='23-12'>
Строка 5: 23-12
Строка 8: None
Строка 11: 12-12
Строка 14: None
['Где', 'скажите', 'мне', 'мои', 'очки', '']
['18.08.2018', '19.09.2019']
Дата 18.08.2018 начинается с позиции 5
Дата 19.09.2019 начинается с позиции 22
Дата DD.MM.YYYY, дата DD.MM.YYYY
```

```python
import re

m = re.search('\d\d\D\d\d', 'Телефон 123-12-12')
print("Строка 4:", m)
print("Строка 5:", m.group())

m = re.search('\d\d\D\d\d', 'Телефон 1231212')
print("Строка 8:", m)

m = re.fullmatch('\d\d\D\d\d', '12-12')
print("Строка 11:", m.group())

m = re.fullmatch('\d\d\D\d\d', 'Т. 12-12')
print("Строка 14:", m)

print("Строка 16:", re.split('\W+', 'Где, скажите мне, мои очки??!'))

print("Строка 18:", re.findall('\d\d\.\d\d\.\d{4}',
                               'Дата 18.08.2018, дата 19.09.2019'))

for m in re.finditer('\d\d\.\d\d\.\d{4}', 'Дата 18.08.2018, дата 19.09.2019'):
    print("Строка 22:", 'Дата', m.group() , 'начинается с позиции', m.start())

print("Строка 24:", re.sub('\d\d\.\d\d\.\d{4}',
                           'DD.MM.YYYY',
                           'Дата 18.08.2018, дата 19.09.2019'))
```

Флаги

Константа	Её смысл
re.ASCII	По умолчанию \w, \W, \b, \B, \d, \D, \s, \S соответствуют все юникодные символы. Флаг re.ASCII оставляет только символы ASCII
re.IGNORECASE	Не различать заглавные и маленькие буквы
re.MULTILINE	Возможность использования специальных символов ^ и $ (начала и конца каждой строки)
re.DOTALL	По умолчанию символ \n конца строки не подходит под точку. С этим флагом точка - любой символ

```python
import re
print(re.findall('\d+', '12 + ٦٧'))
print(re.findall('\w+', 'Hello, мир!'))
print(re.findall('\d+', '12 + ٦٧', flags=re.ASCII))
print(re.findall('\w+', 'Hello, мир!', flags=re.ASCII))
print(re.findall('[уеыаоэяию]+', 'ОООО аааaа ppppp ЫЫЫЫ яяяя'))
print(re.findall('[уеыаоэяию]+', 'ОООО аааaа ppppp ЫЫЫЫ яяяя', flags=re.IGNORECASE))

text ="""Торт
с вишней1
вишней2
"""

print(re.findall('Торт.с', text))
print(re.findall('Торт.с', text, flags=re.DOTALL))
print(re.findall('виш\w+', text, flags=re.MULTILINE))
print(re.findall('^виш\w+', text, flags=re.MULTILINE))

```

```
['12', '٦٧']
['Hello', 'мир']
['12']
['Hello']
['аааaа', 'яяяя']
['ОООО', 'аааaа', 'ЫЫЫЫ', 'яяяя']
[]
['Торт\nс']
['вишней1', 'вишней2']
['вишней2']
```

Перечисления (операция «ИЛИ») – выбор подходящего

```
1  import re
2
3  print(re.findall('морковк|св[её]кл|картошк|редиск',
4  'Вкусные свёкла и картошка'))
5
```

```
['свёкл', 'картошк']
```

Скобочные группы (группировка плюс квантификаторы) – повторение указанное количество раз ((?: …){N})

```
import re

print(re.findall('[0-9a-fA-F]{2}(?:[:-][0-9a-fA-F]{2}){5}',
'11-22-33-44-55-66 11-111-11-11-11-11 11:22-22:33-44-11 '))
```

```
['11-22-33-44-55-66', '11:22-22:33-44-11']
```

Скобки плюс перечисления – перебор вариантов ((?:...))

```
1  import re
2
3  print(re.findall('(?:он|тот) (?:шёл|плыл)',
4  ' он плыл плыл шел тот шёл тот плыл'))
```

```
['он плыл', 'тот шёл', 'тот плыл']
```

Шаблон	Применяем к тексту
(?:\w\w\d\d)+	Есть миг29а, ту154б. Некоторые делают даже миг29ту154ил86.
(?:\w+\d+)+	Есть миг29а, ту154б. Некоторые делают даже миг29ту154ил86.
(?:\+7|8)(?:-\d{2,3}){4}	+7-926-123-12-12, 8-926-123-12-12
(?:[Xx][аоеи]+)+	Муха — хахахехо, ну хаааховоохе, да хахахехохииии! Хам трамвайный.
\b(?:[Xx][аоеи]+)+\b	Муха — хахахехо, ну хаааховоохе, да хахахехохииии! Хам трамвайный.

Группирующие скобки (...)

```
 1  import re
 2
 3  r = '\s*([А-Яа-яЁё]+)(\d+)\s*'
 4  s = '---   Опять45   ---'
 5
 6  m = re.search(r, s)
 7
 8  print (m.group())
 9
10
11  r = '\s*([А-Яа-яЁё]\d)+\s*'
12  s = '---   Опять4ф5   ---'
13
14  m = re.search(r, s)
15  print (m.group())
```