

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Системный анализ процессов химической технологии

Лабораторная работа №3  
Структуры данных: списки

---

Вячеслав Алексеевич Чузлов,  
к.т.н., доцент ОХИ ИШПР

8 февраля 2023 г.

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Списки

- Списки в Python – наиболее гибкая разновидность объектов упорядоченных коллекций.
- Списки могут содержать объекты любого типа: строки, числа или другие списки.
- Списки **можно изменять** на месте присваиванием по индексам, с использованием срезов, вызвав специальные методы или выполнив оператор удаления.

Операция	Описание
<code>a = []</code>	Пустой список
<code>a = [123, 'abc', 1.354, []]</code>	Четыре элемента: индексы 0...3
<code>a = ['Joe', 30.0, ['dev', 'prof']]</code>	Вложенные списки
<code>a = list('hello')</code>	Список элементов итерируемого объекта
<code>a = list(range(-5, 6))</code>	Список последовательных целых чисел
<code>a[i]</code>	Индекс
<code>a[i][j]</code>	Индекс индекса
<code>a[i:j]</code>	Срез
<code>len(a)</code>	Длина
<code>a1 + a2</code>	Конкатенация
<code>a * 3</code>	Повторение
<code>x in a</code>	Вхождение
<code>a.append(5)</code>	Добавление элемента в конец списка
<code>a.extend([10, 20, 30])</code>	Добавление списка в конец исходного списка

## Базовые операции со списками

- Списки являются **последовательностями**, поэтому поддерживают многие операции, характерные для строк.
- Например, для списков определены операторы `+` и `*`. Данные операторы, также как и в случае со строками, означают конкатенацию и повторение, только возвращают в качестве результата новый список, а не строку.

```
1 >>> len([1, 2, 3, 4, 5])           # Длина
2 5
3
4 >>> [1, 2, 3, 4, 5] + [6, 7, 8, 9, 10] # Конкатенация
5 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
6
7 >>> ['Hi!'] * 5                     # Повторение
8 ['Hi!', 'Hi!', 'Hi!', 'Hi!', 'Hi!']
```

# Итерация по спискам

В общем смысле для списков определены все операции над последовательностями, в том числе и инструменты итерации:

```
1 >>> 'banana' in ['banana', 'orange', 'apple'] # Проверка вхождения
2 True
3
4 >>> for fruit in ['banana', 'orange', 'apple']: # Итерация
5 ...     print(fruit, end=' ')
6 ...
7 banana orange apple
```

Оператор цикла **for** проходит (итерируется) по всем элементам в любой последовательности (итерируемом объекте) слева направо, выполняя операторы для каждого из них.

## Генераторы списков (list comprehension)

**Генераторы списков** – это способ создания нового списка с применением выражения к каждому элементу последовательности (по факту в любом итерируемом объекте).

```
1 >>> res = [c * 4 for c in 'HELLO']
2
3 >>> res
4 ['HHHH', 'EEEE', 'LLLL', 'LLLL', '0000']
```

- Генераторы списков записываются более кратко и выполняются чуть быстрее.
- В сложных случаях лучше использовать цикл **for** из-за его более высокой читаемости.

```
5 >>> res = []
6
7 >>> for c in 'HELLO':
8 ...     res.append(c * 4)
9 ...
10
11 >>> res
12 ['HHHH', 'EEEE', 'LLLL', 'LLLL', '0000']
```

# Индексация и срезы

- Индексация и срезы для списков работают аналогично тому, как это было описано для объектов строк.
- Результатом индексации списка может быть объект любого типа, находящийся по указанному индексу, тогда как срезы всегда возвращают новый объект списка.

```
1 >>> fruits = ['banana', 'orange', 'apple']
2
3 >>> fruits[2]           # Индексы начинаются с нуля
4 'apple'
5
6 >>> fruits[-2]          # Отрицательные индексы отсчитываются справа
7 'orange'
8
9 >>> fruits[1:3]         # Срезы получают сегменты
10 ['orange', 'apple']
11
12 >>> fruits[-1]         # Результат среза всегда новый список
13 ['apple']
```

## Вложенность списков

- Внутри списков могут содержаться вложенные списки или объекты других типов.
- Матрицы в Python можно представить в виде вложенных списков. Пример матрицы  $3 \times 3$ :

```
1 >>> matrix = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]
```

- Если указать один индекс, то будет получена целая строка, а при указании двух индексов будет возвращен элемент строки:

```
2 >>> matrix[1]
3 [40, 50, 60]
4
5 >>> matrix[2][0]
6 70
7
8 >>> matrix = [[10, 20, 30],
9 ...           [40, 50, 60],
10 ...          [70, 80, 90]]
11
12 >>> matrix[1][1]
13 50
```



## Изменение списков

- Так как списки – **изменяемый** тип объектов, для них определены операции, которые могут модифицировать объект списка **на месте**.
- Операции модифицирования списков изменяют объект списка напрямую, перезаписывая его старое значение, без необходимости создания новой копии, как в случае работы со строками.

### Присваивание по индексам и срезам

Содержимое списка может быть изменено присваиванием значения либо отдельному элементу (по его индексу), либо целому сегменту (по срезу):

```
1 >>> food = ['burger', 'pizza', 'buritto']
2
3 >>> food[1] = 'toast' # Присваивание по индексу
4
5 >>> food
6 ['burger', 'toast', 'buritto']
7
8 >>> food[:2] = ['need', 'more'] # Присваивание срезу
9
10 >>> food
11 ['need', 'more', 'buritto']
```

## Вызовы методов списков

- Подобно строкам, списки имеют набор специфичных методов, многие из которых ведут к изменению исходного списка на месте:

```
1 >>> a = ['eat', 'more', 'SPAM']
2
3 >>> a.append('please')
4
5 >>> a
6 ['eat', 'more', 'SPAM', 'please']
7
8 >>> a.sort() # Сортировка элементов списка ('S' < 'e')
9
10 >>> a
11 ['SPAM', 'eat', 'more', 'please']
```

- Наиболее распространенный метод – `append()` добавляет объект в конец списка.
- Эффект выполнения выражения `a.append(x)` аналогичен `a + [x]` с одним принципиальным отличием: первый вариант изменяет `a` на месте, а второй вариант создает новый объект списка.
- Метод `sort()` упорядочивает элементы в списке.

## Дополнительные сведения о методе sort()

- В методе `sort()` аргумент `reverse` позволяет производить сортировку в порядке убывания вместо возрастания, а параметр `key` задает функцию с одним аргументом, которая возвращает значение для использования при сортировке.

```
1 >>> a = ["abc", "ABD", "aBe"]
2
3 >>> a.sort()                                # Сортировка со смешанным регистром
4
5 >>> a
6 ['ABD', 'aBe', 'abc']
7
8 >>> a = ["abc", "ABD", "aBe"]
9
10 >>> a.sort(key=str.lower)                  # Приведение к нижнему регистру
11
12 >>> a
13 ['abc', 'ABD', 'aBe']
14
15 >>> a.sort(key=str.lower, reverse=True)    # Изменение порядка сортировки
16
17 >>> a
18 ['aBe', 'ABD', 'abc']
```

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Пример

## Пример

По имеющимся исходным данным определите состав потока в объемных долях, используя следующую формулу:

$$\varphi_i = \frac{\frac{\omega_i}{\rho_i}}{\sum_{i=1}^n \frac{\omega_i}{\rho_i}}$$

где  $\varphi_i$  – объемная доля  $i$ -го компонента;  $\omega_i$  – массовая доля  $i$ -го компонента;  $\rho_i$  – плотность  $i$ -го компонента;  $n$  – число компонентов в системе;  $i$  – индекс компонента в системе.

## Пример

Исходные данные:

Параметр	$C_1$	$C_2$	$C_3$	$iC_4$	$nC_4$	$iC_5$	$nC_5$	$C_6$
$\omega_i$	0.1	0.1	0.1	0.4	0.2	0.05	0.03	0.02
$\rho_i, \text{г/см}^3$	0.416	0.546	0.585	0.5510	0.6	0.616	0.6262	0.6594
$M_i, \text{г/моль}$	16	30	44	58	58	72	72	86

## Пример

```
1 mass_fractions = [.1, .1, .1, .4, .2, .05, .03, .02]
2 densities = [0.416, 0.546, 0.585, 0.5510, 0.6, 0.616, 0.6262, 0.6594]
3 mol_mass_list = [16, 30, 44, 58, 58, 72, 72, 86]
4
5 volume_fractions = [mf / d for mf, d in zip(mass_fractions, densities)]
6 s = sum(volume_fractions)
7 volume_fractions = [vof / s for vof in volume_fractions]
8
9 volume_fractions = [] # Alternative
10 for mf, d in zip(mass_fractions, densities):
11     volume_fractions.append(mf / d)
12
13 s = sum(volume_fractions)
14 for i in range(len(volume_fractions)):
15     volume_fractions[i] /= s
16
17 for vof in volume_fractions: # вывод результата
18     print(f'{vof:.4f}', end=' ')
```

0.1326 0.1010 0.0943 0.4004 0.1838 0.0448 0.0264 0.0167

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Задания



## Задание 1

Используя исходные данные из примера, рассчитайте:

1. Состав потока в мольных долях:

$$\chi_i = \frac{\frac{\omega_i}{M_i}}{\sum_{i=1}^n \frac{\omega_i}{M_i}}$$

где  $\chi_i$  – мольная доля  $i$ -го компонента;  $\omega_i$  – массовая доля  $i$ -го компонента;  $M_i$  – молярная масса  $i$ -го компонента;  $n$  – число компонентов в системе;  $i$  – индекс компонента в системе.

2. Плотность потока:

$$\rho = \frac{1}{\sum_{i=1}^n \frac{\omega_i}{\rho_i}}$$

где  $\rho$  – плотность потока;  $\omega_i$  – массовая доля  $i$ -го компонента;  $\rho_i$  – плотность  $i$ -го компонента;  $n$  – число компонентов в системе;  $i$  – индекс компонента в системе.

3. Среднюю молекулярную массу потока:

$$m = \frac{1}{\sum_{i=1}^n \frac{\omega_i}{M_i}}$$

где  $m$  – средняя молекулярная масса потока;  $\omega_i$  – массовая доля  $i$ -го компонента;  $M_i$  – молярная масса  $i$ -го компонента;  $n$  – число компонентов в системе;  $i$  – индекс компонента в системе.

## Задание 2

Пусть на смешение поступают материальные потоки следующего состава (массовые доли):

Поток	$C_1$	$C_2$	$C_3$	$iC_4$	$nC_4$	$iC_5$	$nC_5$	$C_6$
1	0.1	0.1	0.1	0.4	0.2	0.05	0.03	0.02
2	0.1	0.2	0.1	0.3	0.1	0.15	0.03	0.02
3	0.1	0.1	0.15	0.35	0.1	0.05	0.08	0.07

Расходы потоков 100, 150 и 120 моль/ч, соответственно. Необходимо рассчитать состав итогового потока в массовых и мольных долях.

Состав смесового потока (в массовых долях) можно найти следующим образом:

$$\frac{\sum_{j=1}^n G_j \cdot \omega_{i,j}}{\sum_{j=1}^n G_j}$$

где  $\omega_i$  – массовая доля  $i$ -го компонента в смесовом потоке;  $\omega_{i,j}$  – массовая доля  $i$ -го компонента в  $j$ -ом потоке;  $G_j$  – массовый расход  $j$ -го потока;  $j$  – индекс потока;  $i$  – индекс компонента в системе;  $n$  – число потоков, подаваемых на смешение.

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Контакты

Вячеслав Алексеевич Чузлов,  
к.т.н., доцент ОХИ ИШПР



Учебный корпус №2, ауд. 136



[chuva@tpu.ru](mailto:chuva@tpu.ru)



+7-962-782-66-15

**Благодарю за внимание!**