

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Системный анализ процессов химической технологии

Лабораторная работа №3  
Функции

---

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

13 февраля 2023 г.

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Описание функций

# Написание кода функций

- **Функции** – это многократно используемые фрагменты программы. Они позволяют дать имя определенному блоку команд с тем, чтобы в последствии запускать блок по указанному имени в любом месте программы и сколь угодно много раз. Это называется **вызовом функции**.
- Функции определяются при помощи зарезервированного слова **def**. После этого слова указывается имя функции, за которым следует пара скобок, в которых можно указать имена некоторых переменных, и заключительное двоеточие в конце строки. Далее следует блок команд (инструкций), составляющих тело функции.
- **Сигнатура функции** – часть общего объявления функции, позволяющая средствами трансляции идентифицировать функцию среди других. Составляющие сигнатуры:
  1. имя функции;
  2. аргументы функции;
  3. возвращаемые значения.

```
1 >>> def say_hi():  
2 ...     print("Hi!")  
3  
4 >>> say_hi()  
5 Hi!
```

# Оператор return

- Тело функции почти всегда содержит оператор `return` :  

```
def имя_функции(аргумент1, аргумент2, ..., аргумент3):
    операторы
    ...
    return ...
```
- Оператор `return` в Python может появляться где угодно в теле функции; по достижении он заканчивает выполнение функции и возвращает результат обратно вызывающему коду.
- Оператор `return` состоит из необязательного выражения с объектным значением, которое дает результат функции.
- Если значение опущено, тогда `return` возвращает `None` .
- Оператор `return` сам по себе также необязателен; если он отсутствует, то выход из функции происходит, когда интерпретатор достигает конца тела функции. Формально функция без оператора `return` автоматически возвращает объект `None` .
- Хорошим тоном является явное использование пустого оператора `return` для дополнительного пояснения того, что функция ничего не возвращает в качестве результата.

# Оператор return

- Оператор `return` используется для возврата из функции, т.е. для прекращения её работы и выхода из неё. При этом можно также вернуть некоторое значение из функции.
- Оператор `return` в Python может появляться где угодно в теле функции; по достижении он **заканчивает выполнение функции** и возвращает результат обратно вызывающему коду.

```
1 >>> def maximum(x, y):
2 ...     if x > y:
3 ...         return x
4 ...
5 ...     elif x == y:
6 ...         return 'Equals.'
7 ...
8 ...     else:
9 ...         return y
10
11 >>> print(maximum(2, 3))
12 3
```

```
1 >>> def maximum(x, y):
2 ...     if x > y:
3 ...         return x
4 ...
5 ...     if x == y:
6 ...         return 'Equals.'
7 ...
8 ...     return y
9
10 >>> print(maximum(2, 3))
11 3
```

## Локальные переменные

- При объявлении переменных внутри определения функции, они никоим образом не связаны с другими переменными с таким же именем за пределами функции – т.е. имена переменных являются локальными в функции.
- Это называется **областью видимости переменной**. Область видимости всех переменных ограничена блоком, в котором они объявлены, начиная с точки объявления имени.

```
1 >>> x = 50
2
3 >>> def func(x):
4 ...     print('x =', x)
5 ...     x = 2
6 ...     print('Replace x to', x)
7 ...
8
9 >>> func(x)
10 x = 50
11 Replace x to 2
12
13 >>> print('x =', x)
14 x = 50
```

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Синтаксис передачи аргументов

# Позиционные аргументы

- Если не использовать какой-то специальный синтаксис сопоставления, то Python будет сопоставлять имена по позиции слева направо подобно большинству других языков. Например, если Вы определили функцию, которая требует трех аргументов, тогда должны вызывать ее с тремя аргументами:

```
1 >>> def f(x, y, z):  
2 ...     return x, y, z  
3 ...  
4  
5 >>> f(0, 1, 2)  
6 (0, 1, 2)
```

- Здесь аргументы передаются по позиции – x соответствует 0, y – 1 и z – 2.



# Именованные параметры

- **Именованные** аргументы делают возможным сопоставление по имени, а не по позиции.

```
7 >>> f(z=2, x=0, y=1)
8 (0, 1, 2)
```

- Здесь z=2 означает передачу значения 2 аргументу по имени z. Когда применяются ключевые слова, порядок следования аргументов несущественен, т.к. они сопоставляются по имени.
- Разрешено комбинировать позиционные и ключевые аргументы. Сначала сопоставляются позиционные аргументы слева направо в заголовке, а затем ключевые аргументы по имени:

```
9 >>> f(0, z=2, y=1)
10 (0, 1, 2)
```

- Ключевые аргументы делают вызовы функций самодокументированными. Вызов функции:

```
func(name='James', age=20, job='student')
```

выглядит более значащим по сравнению с вызовом, содержащим три разделенных запятыми значения, особенно в крупных программах.

## Значения по умолчанию

- Стандартные значения позволяют делать некоторые аргументы функции необязательными.
- Если значение для аргумента не было передано, то используется стандартное значение.

```
1 >>> def f(x, y=1, z=2): # Аргумент x обязательный
2 ...     return x, y, z # y и z необязательные
```

- При вызове такой функции обязательно нужно передать значение для x по позиции, либо по имени; передача значений для y и z необязательна:

```
3 >>> f(0)
4 (0, 1, 2)
5 >>> f(x=0)
6 (0, 1, 2)
```

- В случае передачи двух значений стандартное значение получит только аргумент z, а при передаче трех значений стандартные значения не применяются:

```
7 >>> f(10, 20)
8 (10, 20, 2)
9 >>> f(10, 20, 30)
10 (10, 20, 30)
```

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. 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

Вычисления необходимо реализовать в виде функции.

## Пример



```
1 def mass_to_volume_fractions(  
2     mass_fractions: list[float],  
3     densities: list[float]  
4 ) -> list[float]:  
5  
6     volume_fractions = [  
7         mf / d for mf, d in zip(mass_fractions, densities)  
8     ]  
9     s = sum(volume_fractions)  
10    volume_fractions = [v / s for v in volume_fractions]  
11  
12    return volume_fractions  
13  
14  
15 mass_fractions = [.1, .1, .1, .4, .2, .05, .03, .02]  
16 densities = [0.416, 0.546, 0.585, 0.5510, 0.6, 0.616, 0.6262, 0.6594]  
17 mol_mass_list = [16, 30, 44, 58, 58, 72, 72, 86]  
18  
19 volume_fractions = mass_to_volume_fractions(mass_fractions, densities)  
20  
21 for vf in volume_fractions: # 0.1326 0.1010 0.0943 0.4004 0.1838 0.0448 0.0264 0.0167  
22     print(f'{vf:.4f}', end=' ')
```

TOMSK  
POLYTECHNIC  
UNIVERSITY



ТОМСКИЙ  
ПОЛИТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ

# Задания

## Задание 1

Используя исходные данные из примера, рассчитайте, реализовав соответствующие функции:

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 200, 250 и 120 кг/ч, соответственно. Необходимо рассчитать состав итогового потока в массовых долях, реализовав соответствующую функцию.

Состав смесового потока 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!**