TOMSK POLYTECHNIC UNIVERSITY
ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Системный анализ процессов химической технологии

Лабораторная работа №4
Функции

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

8 ноября 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.

```
>>> def say_hi():
...     print("Hi!")

>>> say_hi()
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.

```
>>> def maximum(x, y):
...     if x > y:
...         return x
...
...     elif x == y:
...         return 'Equals.'
...
...     else:
...         return y

>>> print(maximum(2, 3))
3
```

```
>>> def maximum(x, y):
...     if x > y:
...         return x
...
...     if x == y:
...         return 'Equals.'
...
...     return y

>>> print(maximum(2, 3))
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.

```
>>> x = 50

>>> def func(x):
...     print('x =', x)
...     x = 2
...     print('Replace x to', x)
...

>>> func(x)
x = 50
Replace x to 2

>>> print('x =', x)
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:

```
>>> def f(x, y, z):
...     return x, y, z
...

>>> f(0, 1, 2)
(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.

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

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

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

```
>>> f(0)
(0, 1, 2)
>>> f(x=0)
(0, 1, 2)
```

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

```
>>> f(10, 20)
(10, 20, 2)
>>> f(10, 20, 30)
(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 φ_i – объемная доля i-го компонента; ω_i – массовая доля i-го компонента; ρ_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
ω_i	0.1	0.1	0.1	0.4	0.2	0.05	0.03	0.02
ρ_i, г/см^3	0.416	0.546	0.585	0.5510	0.6	0.616	0.6262	0.6594
M_i, г/моль	16	30	44	58	58	72	72	86

Вычисления необходимо реализовать в виде функции.

Пример


```python
def mass_to_volume_fractions(
    mass_fractions: list[float],
    densities: list[float]
) -> list[float]:

    volume_fractions = [
        mf / d for mf, d in zip(mass_fractions, densities)
    ]
    s = sum(volume_fractions)
    volume_fractions = [v / s for v in volume_fractions]

    return volume_fractions


mass_fractions = [.1, .1, .1, .4, .2, .05, .03, .02]
densities = [0.416, 0.546, 0.585, 0.5510, 0.6, 0.616, 0.6262, 0.6594]
mol_mass_list = [16, 30, 44, 58, 58, 72, 72, 86]

volume_fractions = mass_to_volume_fractions(mass_fractions, densities)

for vf in volume_fractions: # 0.1326 0.1010 0.0943 0.4004 0.1838 0.0448 0.0264 0.0167
    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 χ_i – мольная доля i-го компонента; ω_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 ρ – плотность потока; ω_i – массовая доля i-го компонента; ρ_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; ω_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 смесевого потока можно найти следующим образом:

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

где ω_i – массовая доля i-го компонента в смесевом 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
+7-962-782-66-15

Благодарю за внимание!