Информатика. Введение в Python

Множества, кортежи. Numpy.

Шевская Наталья Владимировна,
СПбГЭТУ “ЛЭТИ”, ФКТИ, МОЭВМ

Множества

- Класс set
- Изменяемый объект
- Множество – это неупорядоченная последовательность, состоящая из неповторяющихся элементов
- Удобно использовать для удаления повторяющихся элементов из списка

Инициализация множества

на основе строки:

```
>>> a = set('Hello!')
>>> a
{'e', 'l', 'o', '!', 'H'}
```

на основе списка:

```
>>> a = set(['hello', 'daddy', 'hello', 'mum'])
>>> a
{'mum', 'daddy', 'hello'}
```

на основе range:

```
>>> set(range(10))
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

на основе ключей словаря:

```
>>> set({'a': 2, 'b': 4, 'c': 6})
{'b', 'c', 'a'}
```

Работа со множествами

Проверка вхождения элемента:

```
>>> a = set([1, 2, 3])
>>> a
{1, 2, 3}
>>> 1 in a
True
```

Сравнение множеств:

```
>>> b = {1, 2, 3}
>>> a == b
True
```

```
>>> c
{1, 2, 4}
>>> a > c
False
```

```
>>> d = {1, 2}
>>> d < c
True
>>> d < a
True
```

Объединение множеств:

```
>>> a.union(c)
{1, 2, 3, 4}
```

Пересечение множеств:

```
>>> a.intersection(d)
{1, 2}
```

Разность множеств:

```
>>> c.difference(d)
{4}
```

Не поддерживаются дублирование и конкатенация!

Кортежи

Класс tuple

Неизменяемый объект

Кортеж – это упорядоченная последовательность, состоящая из элементов разной природы

Как список, только неизменяемый

Часто встречается в функциях

Инициализация кортежей

Инициализация на основе строки:

```
>>> a = tuple('Hello!')
>>> a
('H', 'e', 'l', 'l', 'o', '!')
```

на основе списка:

```
>>> a = tuple(['H', 'e', 'l', 'l', 'o', '!'])
>>> a
('H', 'e', 'l', 'l', 'o', '!')
```

на основе range:

```
>>> a = tuple(range(10))
>>> a
(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

на основе ключей словаря:

```
>>> tuple({'a': 2, 'b': 4, 'c': 6})
('a', 'b', 'c')
```

Работа с кортежами

Проверка вхождения элемента:

```
>>> a
(0, 1, 2)
>>> 0 in a
True
```

Сравнение кортежи:

```
>>> b = (0, 1, 2)
>>> a == b
True
```

```
>>> c = (0, 1, 3)
>>> c > a
True
```

Конкатенация и дублирование:

```
>>> a + b
(0, 1, 2, 0, 1, 2)
>>> a * 3
(0, 1, 2, 0, 1, 2, 0, 1, 2)
```

Кортежное присваивание (unpacking, распаковка)

```
x, y, z = 1, 2, 3
```

- скобки кортежа подставляются неявно
- поочередно: x = 1, y = 2, z = 3

- распаковка значений, возвращаемых из функции
- оператор распаковки *

Полезные ссылки

Про множества:
https://pythonworld.ru/tipy-dannyx-v-python/mnozhestva-set-i-frozenset.html

Про кортежи:
https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html

numpy

Основные определения

Специальная библиотека для работы с массивами.

Позволяет быстро выполнять математические и статистические операции над массивами.

Часто используется в математике и физике.

Основной объект – однородный и неизменяемый многомерный массив (numpy.ndarray)

Описание библиотеки: https://numpy.org/

Матрицы на основе списков list

```
L = [[i for i in range(5)]
     for j in range(5)]
```

построили матрицу 5 на 5:

выводим матрицу на экран:

```
for item in L:
    print(item)
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4]
```

Создание объекта ndarray

```
import numpy as np

L = [[i for i in range(5)] for j in range(5)]
arr = np.array(L)
print(type(arr))
print(arr)
```

```
<class 'numpy.ndarray'>
[[0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]
 [0 1 2 3 4]]
```

Массивы

```
import numpy as np


M = np.random.random((3, 3))
M_min, M_max, M_mean = M.min(), M.max(), M.mean()
sum_x = M.sum(axis=1)
sum_y = M.sum(axis=0)
print(M_min, M_max, M_mean, sum_y, sum_x)

# поиск минимума, максимума, среднего значения
# суммы по строкам axis=1, суммы по столбцам axis=0
```

Атрибуты объекта ndarray

`ndarray.ndim` - число измерений (осей)

`ndarray.shape` - размеры массива (сколько элементов по каждой оси)

`ndarray.size` - количество элементов массива (по всем осям)

`ndarray.dtype` - объект, описывающий тип элементов массива

`ndarray.itemsize` - размер каждого элемента массива в байтах

Атрибуты объекта ndarray. Пример

```
print(arr.ndim)                    2

print(arr.shape)                   (5, 5)

print(arr.size)                    25

print(arr.dtype)                   int64

print(arr.itemsize)                8
```

Тип элементов во время создания объекта ndarray

Параметр по умолчанию dtype может принимать значения:

```
bool_, character, int8, int16, int32, int64, float8,
float16, float32, float64, complex64, object_
```

```
import numpy as np

L = [[i for i in range(5)] for j in range(5)]
arr = np.array(L,dtype=np.int8)
print(arr.itemsize)
print(arr)
```

```
float16
[[0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]
 [0. 1. 2. 3. 4.]]
```

Другие способы создания массивов

```
print(np.zeros((3, 5)))
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

```
print(np.ones((3, 5)))
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

```
print(np.empty((3, 5)))
print(np.eye(3))
```

Тригонометрия над ndarrays

```
a = np.array([20, 30, 40, 50])

np.cos(a)

np.sin(a)

np.tan(a)
```

Другие тригонометрические функции:

Mathematical functions — NumPy v1.21 Manual

Срезы в ndarrays

- Для одномерных массивов
- Взятие среза:

a[3:7]

- Использование среза:

a[3:7] = 8

Особенности работы с массивами

1. Математические операции выполняются над массивами одного размера
2. Математические операции выполняются поэлементно
3. Допустимы математические операции с массивом и числом
4. Всегда создается новый массив (т.к. неизменяемый тип)
5. Можно получить inf как результат операции

Операции с массивами

1. Доступны математические операции **+, -, *, /, **, %**
2. Фильтрация элементов массива с использованием операций сравнения: **>, <, >=, <=, ==, !=**
3. Доступны тригонометрические функции: **numpy.cos(<ndarray>)**
4. Доступны основные операции для работы с задачами линейной алгебры **numpy.linalg**
5. Можно применять срезы **[i:j:k]**

Элементы линейной алгебры

Система линейных алгебраических уравнений (СЛАУ)

– это набор двух или более линейных уравнений, включающих один и тот же набор переменных.

Решение СЛАУ:

$$\begin{cases} 2x + 5y = 1 \\ x - 10y = 3 \end{cases} \rightarrow \begin{pmatrix} 2 & 5 \\ 1 & -10 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \rightarrow$$

```
import numpy

M1 = numpy.array([[2., 5.], [1., -10.]])
v1 = numpy.array([1., 3.])

print(numpy.linalg.solve(M1, v1))
```

Вопросы по курсу задавайте по почте:

Шевская Наталья Владимировна
natalya.shevskaya@moevm.info

Правила коммуникации по электронной почте:
http://se.moevm.info/doku.php/inf:communication_rules