

Лекция 5

Язык программирования Python.

Хайрулин Сергей Сергеевич

email: s.khairulin@g.nsu.ru, s.khayrulin@gmail.com

Ссылка на [материалы](#)

План

- Лекции/практические занятия
 - Тест
- Дифференцированный зачет в конце семестра
 - Защита задания

Литература

Начальный уровень

- Mark Pilgrim. Dive into Python - <http://www.diveintopython.net/>
- Марк Лутц. Изучаем Python, 4-е издание // Символ-Плюс 2011.
- ...

Стандарт/Документация

- PEP-8 - <https://www.python.org/dev/peps/pep-0008/>
- <https://www.python.org/>
- <https://github.com/python/cpython>

Экспертный уровень

- Лучано Рамальо: Python. К вершинам мастерства
- Mitchell L. Model. Bioinformatics Programming Using Python // O'Reilly 2010.

Версии Python

- Python 2 вышел 2010 году последняя версия 2.7.16 - исправлялись только баги(ошибки) с января 2020 года поддержка прекращена.
- Python 3 появился в 2008, является актуальной версией языка. Текущая стабильная версия 3.8.5 -> в пред релиз 3.9, в разработке 3.10
 - Python 3 не гарантирует совместимости кода с Python 2

План занятия

- Коллекция
- Индексация
- tuple
- list
 - Срезы
 - Списковые включения
- Операции над списками
- Практика

Коллекции

- **Коллекция** — программный объект, содержащий в себе, тем или иным образом, набор значений одного или различных типов, и позволяющий обращаться к этим значениям.
- **Коллекция** позволяет записывать в себя значения и извлекать их.
- **Назначение коллекции** — служить хранилищем объектов и обеспечивать доступ к ним.

Индексация

Некоторые коллекции поддерживают операцию индексацию

В python операция обозначается [...] - аргументом операции является индекс, который указывает какой элемент коллекции нужно взять

Синтаксис:

```
collection_name[index]
```

где collection_name - переменная содержащая ссылку на коллекцию

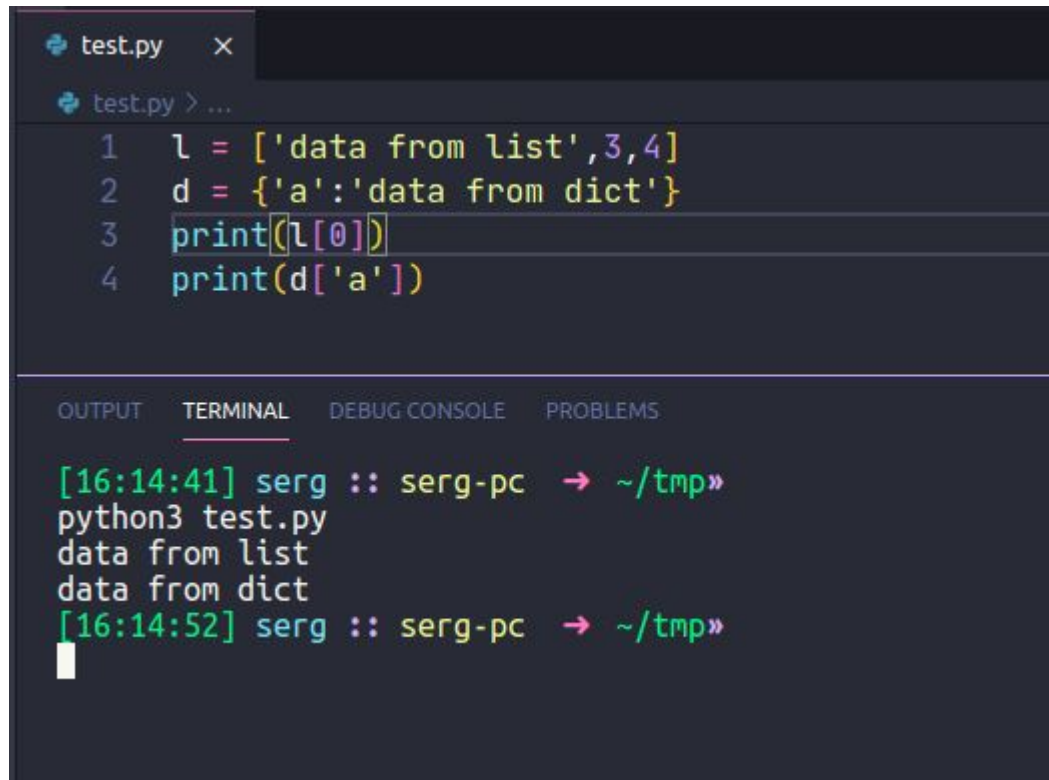
index - указатель на элемент коллекции

Индексация

Для коллекций, сохраняющих порядок элементов, таких как **list, tuple, range** - индексом является порядковый номер элемента в коллекции. **ИНДЕКСАЦИЯ В PYTHON ДЛЯ ТАКИХ КОЛЛЕКЦИЙ НАЧИНАЕТСЯ С 0!** Также в python поддерживаются отрицательные индексы (см. примеры).

Для **dict, set, ...** - индексом является значение ключа элемента (об этом на др лекции).

Индексация



```
test.py x
test.py > ...
1 l = ['data from list',3,4]
2 d = {'a':'data from dict'}
3 print(l[0])
4 print(d['a'])
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[16:14:41] serg :: serg-pc → ~/tmp»
python3 test.py
data from list
data from dict
[16:14:52] serg :: serg-pc → ~/tmp»
```

Индексация



The image shows a screenshot of a code editor with a dark theme. At the top, there is a tab labeled 'test.py' with a close button. Below the tab, the editor shows the following Python code:

```
1 l = ['zero', 'one', 'two', 'three', 'four']
2 print(l[0])
3 print(l[-1])
```

Below the code editor, there is a terminal window with tabs for 'OUTPUT', 'TERMINAL', 'DEBUG CONSOLE', and 'PROBLEMS'. The 'TERMINAL' tab is active, showing the output of running the script:

```
[16:30:31] serg :: serg-pc → ~/tmp»
python3 test.py
zero
four
[16:30:32] serg :: serg-pc → ~/tmp»
```

Python - collection package

`collections` — Container datatypes

Source code: [Lib/collections/__init__.py](#)

This module implements specialized container datatypes providing alternatives to Python's general purpose built-in containers, `dict`, `list`, `set`, and `tuple`.

<code>namedtuple()</code>	factory function for creating tuple subclasses with named fields
<code>deque</code>	list-like container with fast appends and pops on either end
<code>ChainMap</code>	dict-like class for creating a single view of multiple mappings
<code>Counter</code>	dict subclass for counting hashable objects
<code>OrderedDict</code>	dict subclass that remembers the order entries were added
<code>defaultdict</code>	dict subclass that calls a factory function to supply missing values
<code>UserDict</code>	wrapper around dictionary objects for easier dict subclassing
<code>UserList</code>	wrapper around list objects for easier list subclassing
<code>UserString</code>	wrapper around string objects for easier string subclassing

Deprecated since version 3.3, will be removed in version 3.10: Moved [Collections Abstract Base Classes](#) to the `collections.abc` module. For backwards compatibility, they continue to be visible in this module through Python 3.9.

<https://docs.python.org/3.8/library/collections.html>

Python built-in collection

Sequence Types — list, tuple, range

There are three basic sequence types: lists, tuples, and range objects. Additional sequence types tailored for processing of `binary data` and `text strings` are described in dedicated sections.

<https://docs.python.org/3.8/library/stdtypes.html#sequence-types-list-tuple-range>

tuple

```
test.py x
test.py > ...
1 l1 = 1,2,3
2 l2 = (1,2,3)
3 print(l1)
4 print(l2)
5
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[15:10:10] serg :: serg-pc → ~/tmp»
cd /home/serg/tmp ; /usr/bin/env /usr/bin/python3
g/tmp/test.py
(1, 2, 3)
(1, 2, 3)
[15:10:14] serg :: serg-pc → ~/tmp»
```

```
test.py x
test.py > ...
1 l1 = 1,2,3
2 l1[0] = 123
```

Exception has occurred: TypeError
'tuple' object does not support item assignment
File "/home/serg/tmp/test.py", line 2, in <module>
l1[0] = 123

<https://docs.python.org/3.8/library/stdtypes.html#tuple>

list

```
test.py x
test.py > ...
1 l = [1,2,3,4]
2 print(l)
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[15:12:37] serg :: serg-pc → ~/tmp»
cd /home/serg/tmp ; /usr/bin/env /usr/bin/python3
g/tmp/test.py
[1, 2, 3, 4]
[15:12:43] serg :: serg-pc → ~/tmp»
```

```
test.py x
test.py > ...
1 l = [1,2,3,4]
2 print(l)
3 l[2] = 3.14
4 print(l)
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[15:15:11] serg :: serg-pc → ~/tmp»
cd /home/serg/tmp ; /usr/bin/env /usr/bin/python3
[1, 2, 3, 4]
[1, 2, 3.14, 4]
[15:15:16] serg :: serg-pc → ~/tmp»
```

list

Operation	Result
<code>s[i] = x</code>	item i of s is replaced by x
<code>s[i:j] = t</code>	slice of s from i to j is replaced by the contents of the iterable t
<code>del s[i:j]</code>	same as <code>s[i:j] = []</code>
<code>s[i:j:k] = t</code>	the elements of <code>s[i:j:k]</code> are replaced by those of t (1)
<code>del s[i:j:k]</code>	removes the elements of <code>s[i:j:k]</code> from the list
<code>s.append(x)</code>	appends x to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code>)
<code>s.clear()</code>	removes all items from s (same as <code>del s[:]</code>) (5)
<code>s.copy()</code>	creates a shallow copy of s (same as <code>s[:]</code>) (5)
<code>s.extend(t)</code> or <code>s += t</code>	extends s with the contents of t (for the most part the same as <code>s[len(s):len(s)] = t</code>)
<code>s *= n</code>	updates s with its contents repeated n times (6)
<code>s.insert(i, x)</code>	inserts x into s at the index given by i (same as <code>s[i:i] = [x]</code>)
<code>s.pop([i])</code>	retrieves the item at i and also removes it from s (2)
<code>s.remove(x)</code>	remove the first item from s where <code>s[i] == x</code> (3)
<code>s.reverse()</code>	reverses the items of s in place (4)

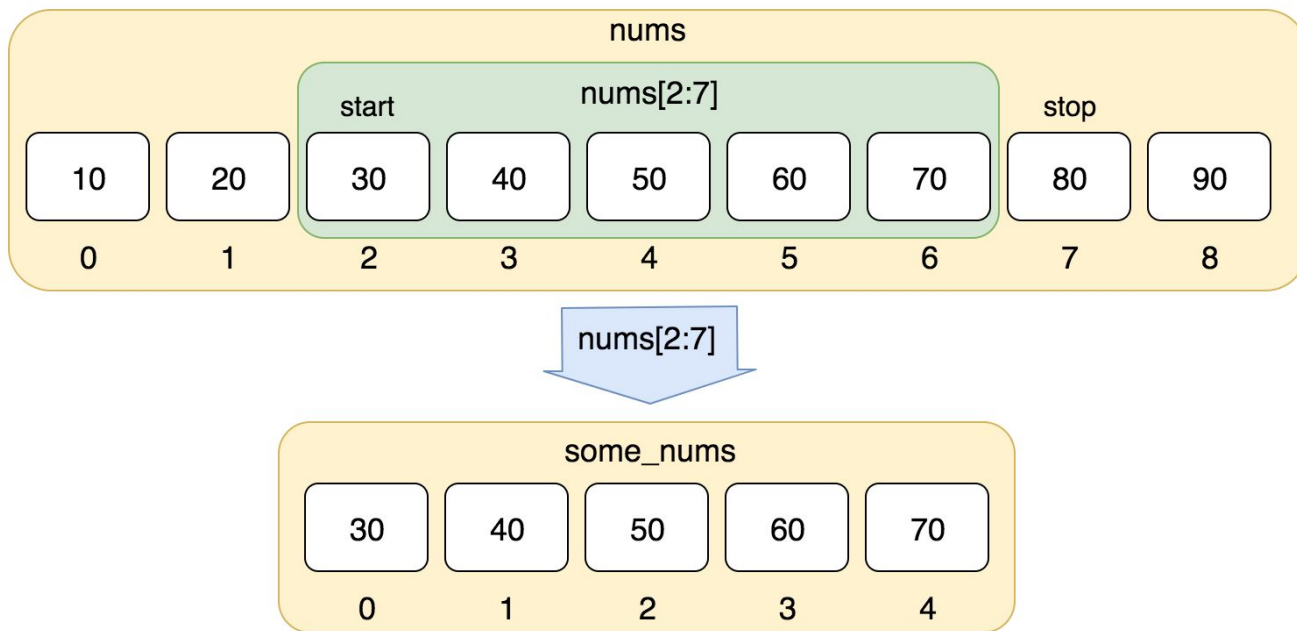
Срезы (slice)

Расширенная операция индексации, над сохраняющими порядок коллекциями, которая позволяет делать выборки из них. Параметры задаются значениями индексов правым (**который включается в выборку**) и левым (**не включается в выборку**). Синтаксис

`collection_name[start:stop:step]` -> вернет список в который является

Значение индексов могут быть < 0 . Значение `step` - шаг может отсутствовать по умолчанию он равен 1.

Срезы (slice)



Срезы (slice)

```
1 l = [i for i in range(10)]
2
3 print(l)
4 print(l[1:7])
5 print(l[4:])
6 print(l[:5])
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[16:43:49] serg :: serg-pc → ~/tmp»
python3 test.py
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4, 5, 6]
[4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4]
[16:43:50] serg :: serg-pc → ~/tmp»
```

Списковые включения

```
test.py x
test.py > ...

1  # How to create list
2  #1. loops
3  l = [] # empty list
4  for i in range(10):
5      l.append(i)
6
7  print(l)
8
9  # list comprehension
10 l = [i for i in range(10)]
11 print(l)
```

Списковые включения

```
test.py x
test.py > ...

1  # How to create list
2  #1. loops
3  l = [] # empty list
4  for i in range(10):
5      l.append(i)
6
7  print(l)
8
9  # list comprehension
10 l = [i for i in range(10)]
11 print(l)
```

Списковые включения

```
1 # inner list comprehension
2 l = [[j for j in range(i)] for i in range(10)]
3 for row in l:
4     print(row)
```

OUTPUT

TERMINAL

DEBUG CONSOLE

PROBLEMS

```
[17:07:56] serg :: serg-pc → ~/tmp»
python3 test.py
[]
[0]
[0, 1]
[0, 1, 2]
[0, 1, 2, 3]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5]
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 2, 3, 4, 5, 6, 7]
[0, 1, 2, 3, 4, 5, 6, 7, 8]
```

Списковые включения

```
test.py > ...  
1  # inner list comprehension  
2  l = [i for i in range(10) if i % 2 == 0]  
3  print(l)
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
[17:09:16] serg :: serg-pc → ~/tmp»  
python3 test.py  
[0, 2, 4, 6, 8]  
[17:09:18] serg :: serg-pc → ~/tmp»  
█
```

Практика

1. Списки