



Introduction to the Python programming language

Laszlo SZATHMARY

University of Debrecen
Faculty of Informatics

Lab #3

- writing to the standard output
- list data type (cont.)
- loops (for, while)

(last update: 2017-08-03 [yyyy-mm-dd])

2017-2018, 1st semester



```
>>> a = [0, 1, 2, 3, 4]
>>> a
[0, 1, 2, 3, 4]
>>> for e in a:
...     print(e)
...
0
1
2
3
4
>>> print(2, 'a', 5, '12')
2 a 5 12
>>> print(2, 'a', 5, '12', sep='')
2a512
>>> print(1, 'a'); print(2, 'b')
1 a
2 b
>>> print(1, 'a', end=''); print(2, 'b')
1 a2 b
>>>
>>> import sys
>>> print('Warning! Reactor meltdown!', file=sys.stderr)

Warning! Reactor meltdown!
>>>
```

print has become
a function

sep: separator
between the elements

end: end sign
after the last element

file: print the output
to this file

Help on built-in function print in module builtins:

default values

```
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

Writing to the standard output



```

1  >>> a = range(5)
2  >>> a
3  [0, 1, 2, 3, 4]
4  >>> for e in a:
5  ...     print e
6  ...
7  0
8  1
9  2
10 3
11 4
12 >>> for e in a:
13 ...     print e,
14 ...
15 0 1 2 3 4
16 >>>
17 >>> import sys
18 >>>
19 >>> for e in a:
20 ...     sys.stdout.write(e)
21 ...
22 Traceback (most recent call last):
23 ...   File "<stdin>", line 2, in <module>
24 TypeError: expected a character buffer object
25 >>>
26 >>> for e in a:
27 ...     sys.stdout.write(str(e))
28 ...
29 01234>>>

```

1

(„\n”)

2

(space)

3

(„full control”)

Some list operations

```
1  >>> a = [1, 2, 3]
2  >>> a
3  [1, 2, 3]
4  >>> a.append(20)
5  >>> a
6  [1, 2, 3, 20]
7  >>> a.pop(0)
8  1
9  >>> a
10 [2, 3, 20]
11 >>> del a
12 >>> a
13 Traceback (most recent call last):
14   File "<stdin>", line 1, in <module>
15 NameError: name 'a' is not defined
16 >>> a = [1, 2, 3]
17 >>> del a[1]
18 >>> a
19 [1, 3]
```

using a list as a **Stack**:

my_list.append(elem)
my_list.pop()

Extra: Queue data structure

```
>>> from collections import deque
>>>
>>> q = deque([3, 4, 5])
>>> q
deque([3, 4, 5])
>>> q.append(6)
>>> q.append(7)
>>> q
deque([3, 4, 5, 6, 7])
>>> q.popleft()
3
>>> q
deque([4, 5, 6, 7])
```

More info about collections:

<http://docs.python.org/3/library/collections.html>

```
1 >>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8]
2 >>> a
3 [0, 1, 2, 3, 4, 5, 6, 7, 8]
4 >>> a[2:5]
5 [2, 3, 4]
6 >>> a[2:5] = []
7 >>> a
8 [0, 1, 5, 6, 7, 8]
9 >>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8]
10 >>> a[2:5] = [10, 20, 30, 40]
11 >>> a
12 [0, 1, 10, 20, 30, 40, 5, 6, 7, 8]
```

removing multiple
elements

changing multiple
elements

Some common list methods

- `list.append(elem)`
Insert an element to the end of the list. It doesn't return a new list; it modifies the list in place.
- `list.insert(index, elem)`
Insert an element to the given index position. Elements on the right are shifted one position to the right.
- `list.extend(list2)`
Elements in list2 are inserted to the end of the list. The operators `+` and `+=` work similarly.
- `list.index(elem)`
Searching for an element in the list. If it's in the list, then return its index position. If it's not in the list, then raise a `ValueError` exception. (If you want to avoid exceptions, use the „in” operator.)
- `list.remove(elem)`
Remove the first occurrence of the element from the list. If it's not in the list, then raise a `ValueError` exception.
- `list.sort()`
Sort the list in place. It has no return value!
- `list.reverse()`
Reverse the order of elements in place. It has no return value!
- `list.pop(index)`
Remove the element from the given index position. If no index position is specified, then remove the last (rightmost) element from the list.

Sorting a list

1

```
>>> a = [8, 5, 1, 3]
>>> a
[8, 5, 1, 3]
>>> sorted(a)
[1, 3, 5, 8]
>>> help(sorted)
```

← returns a new, sorted list

Help on built-in function sorted in module builtins:

```
sorted(iterable, key=None, reverse=False)
    Return a new list containing all items from the iterable in ascending order.

    A custom key function can be supplied to customise the sort order, and the
    reverse flag can be set to request the result in descending order.
```

key, reverse:
optional
parameters

```
>>> sorted(a, reverse=True)
[8, 5, 3, 1]
>>> a
[8, 5, 1, 3]
>>> a = sorted(a)
>>> a
[1, 3, 5, 8]
>>>
>>> a = ['bela', 'aladar', 'denes', 'cecil']
>>> sorted(a)
['aladar', 'bela', 'cecil', 'denes']
>>> a
['bela', 'aladar', 'denes', 'cecil']
>>> a.sort()
>>> a
['aladar', 'bela', 'cecil', 'denes']
>>>
```

← sorts in place

2

Some common operations with lists

```
1 >>> li
2 [9, 8, 1, 4, 8, 2, 3, 2]
3 >>> max(li)
4 9
5 >>> min(li)
6 1
7 >>> sum(li)
8 37
```

these are built-in functions

Exercise: write a function, which receives a list of integers and returns the *product* of the elements in the list.

split / join

```
1  >>> a = ['aa', 'bb', 'cc', 'dd']
2  >>> a
3  ['aa', 'bb', 'cc', 'dd']
4  >>> ':'.join(a)
5  'aa:bb:cc:dd'
6  >>> ','.join(a)
7  'aa,bb,cc,dd'
12 >>> print '\n'.join(a)
13 aa
14 bb
15 cc
16 dd
17 >>>
18 >>> b = 'aa:bb:cc:dd'
19 >>> b
20 'aa:bb:cc:dd'
21 >>> b.split(':')
22 ['aa', 'bb', 'cc', 'dd']
23 >>> s = 'aladar    bela  cecil'
24 >>> s.split()
25 ['aladar', 'bela', 'cecil']
```

list → string

by some
delimiter

string → list



range / xrange

```
4 >>> range(20)
5 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
6 >>> for i in range(10):
7 ...     print i,
8 ...
9 0 1 2 3 4 5 6 7 8 9
10 >>> for i in xrange(10):
11 ...     print i,
12 ...
13 0 1 2 3 4 5 6 7 8 9
14 >>>
15 >>> range(5,20)
16 [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
17 >>>
18 >>> range(5,20,2)
19 [5, 7, 9, 11, 13, 15, 17, 19]
20 >>>
21 >>> range(10, 0, -1)
22 [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
23 >>>
```

requires less memory
(an element is created when it is needed)

third parameter:
step

descending series

Python 3 only has „range”, but it works like Python 2’s „xrange”.



range

Python 3 only has „range”, but it works like Python 2’s „xrange”.

```
>>> xrange(5)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'xrange' is not defined
>>>
>>> range(5)
range(0, 5)
>>>
>>> list(range(5))
[0, 1, 2, 3, 4]
>>>
>>> for i in range(5):
...     print(i)
...
...
0
1
2
3
4
>>>
```

Exercise

Calculate the sum of natural numbers from 1 to 100 (included).

Time available: 30 seconds.

Link: <https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=EnPy3.20121001b>

for loop and while loop

```
>>> for i in range(10):
...     print(i, end=' ')
...
0 1 2 3 4 5 6 7 8 9
>>>
>>> i = 0
>>> while i < 10:
...     print(i, end=' ')
...     i += 1
...
0 1 2 3 4 5 6 7 8 9
>>>
>>> li = ['aladar', 'bela', 'cecil']
>>>
>>> for e in li:
...     print(e, end=' ')
...
aladar bela cecil
>>>
>>> i = 0
>>> size = len(li)
>>> while i < size:
...     print(li[i], end=' ')
...     i += 1
...
aladar bela cecil
>>>
```

for loop

the same with a *while* loop

HW: list1.py and list2.py



Exercises

1. [[20120905b](#)] product of the elements in a list
 2. [[20121001b](#)] sum of natural numbers from 1 to 100 (2nd version too)
 3. [[20120818bc](#)] lists #1
 4. [[20120922a](#)] lists #2
 5. [[20120815h](#)] a-z
 6. [[20130225a](#)] string cleaning
 7. [[20120815d](#)] ASCII table
 8. [[20120820b](#)] decimal → binary conversion
 9. [[20120818e](#)] multiples of 3 or 5 (PE #1)
 10. [[20120815l](#)] secret message
 11. [[20120815e](#)] palindrome (iterative method)
- 