High-Level Programming Languages 3

The Python Programming Language

Laszlo SZATHMARY

University of Debrecen

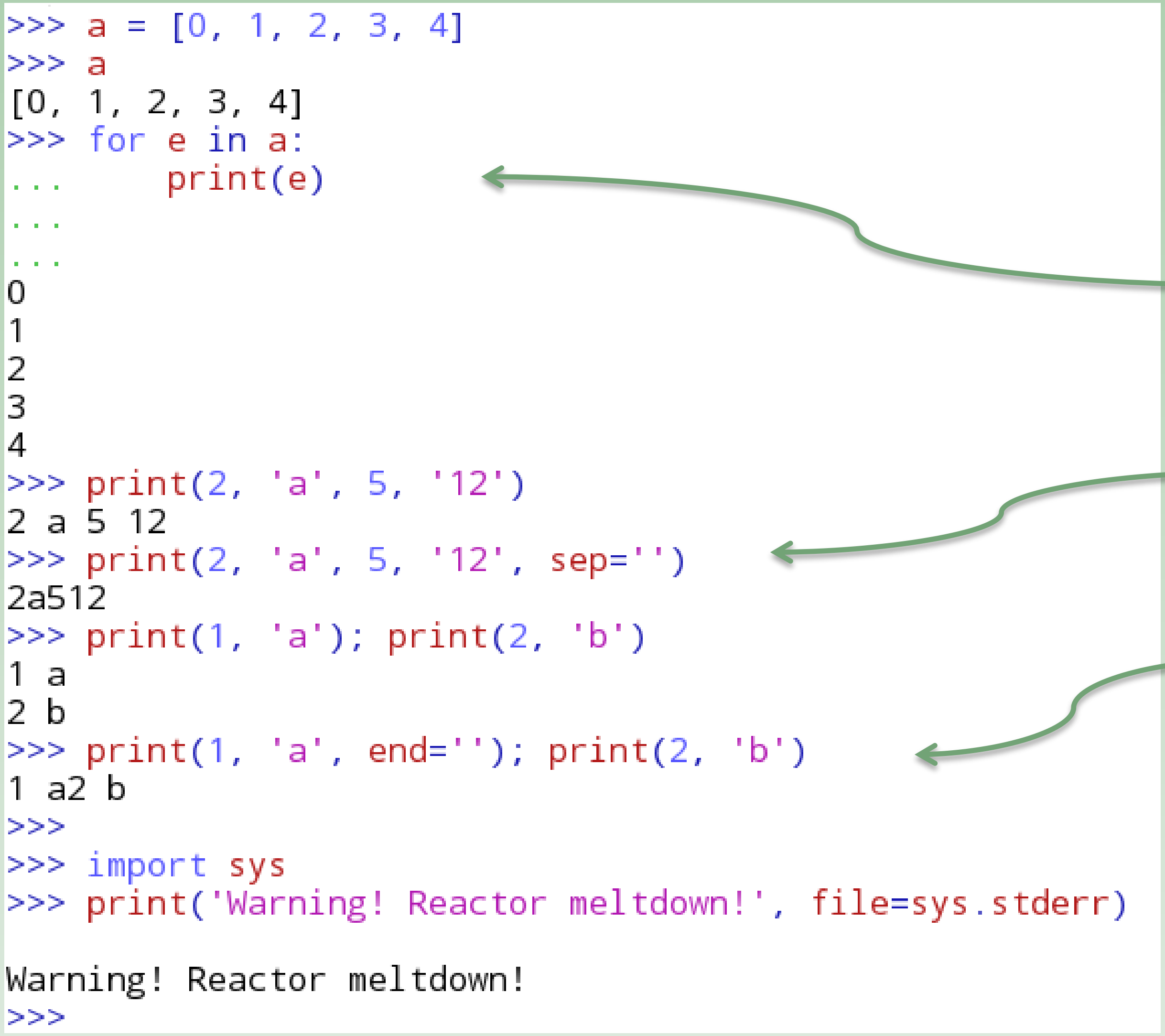
Faculty of Informatics

Lab #3

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

(last update: 2019-09-17 [yyyy-mm-dd])

2019-2020, 1st semester



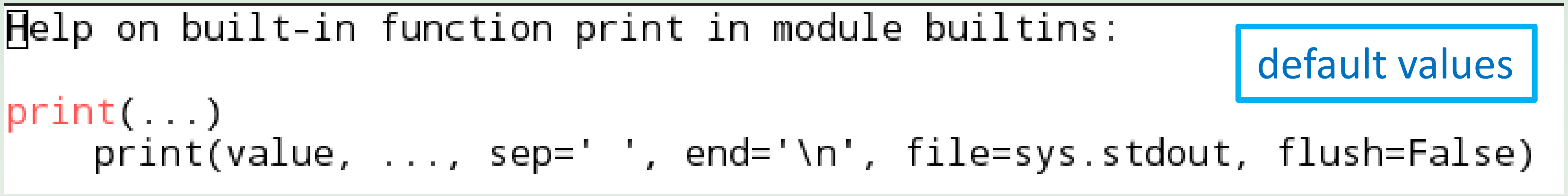
```
>>> a = [0, 1, 2, 3, 4]
>>> a
[0, 1, 2, 3, 4]
>>> for e in a:
...     print(e)
...
...
0
1
2
3
4
>>> print(2, 'a', 5, '12')
2 a 5 12
>>> print(2, 'a', 5, '12', sep='')
2a512
>>> print(1, 'a'); print(2, 'b')
1 a
2 b
>>> print(1, 'a', end=''); print(2, 'b')
1 a2 b
>>>
>>> import sys
>>> print('Warning!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

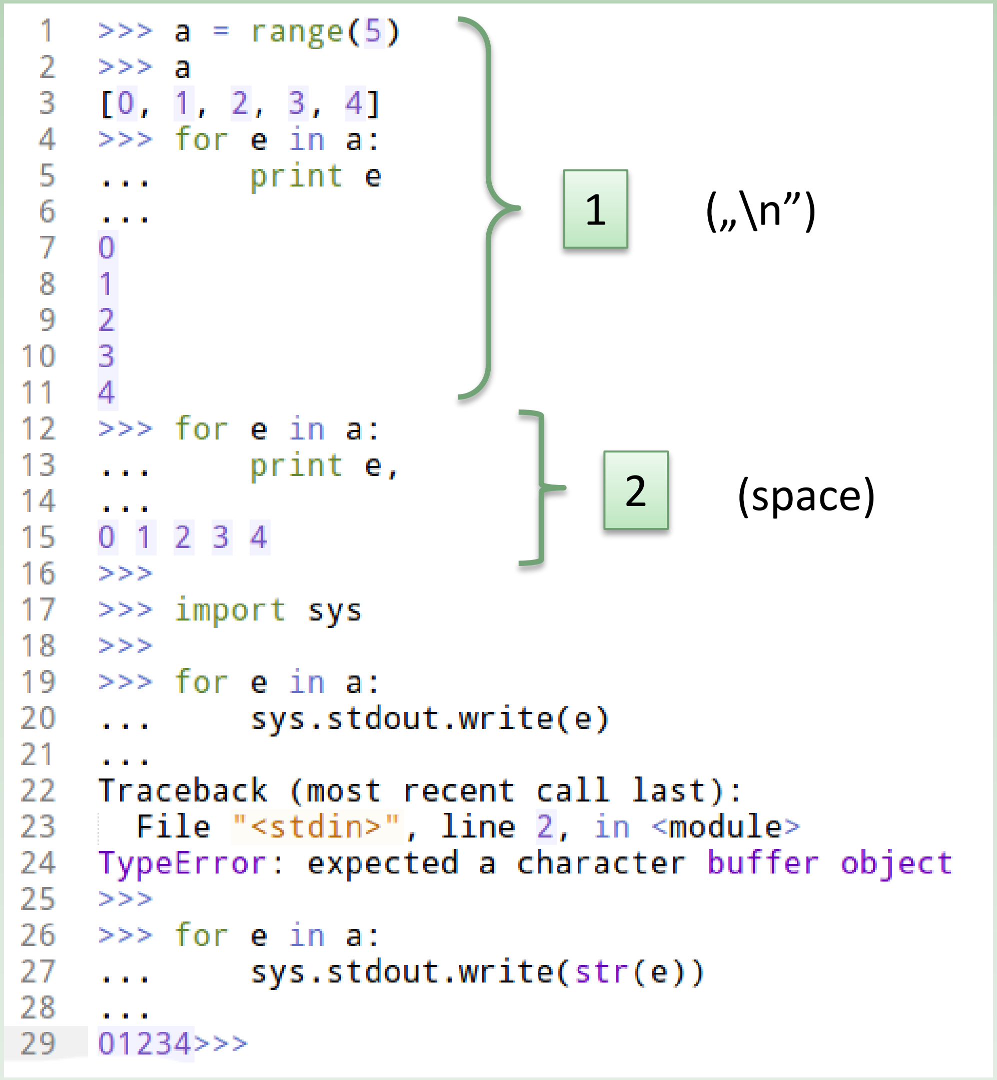
file: print the output to this file



```
Help on built-in function print in module builtins:

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

default values

Writing to the standard output



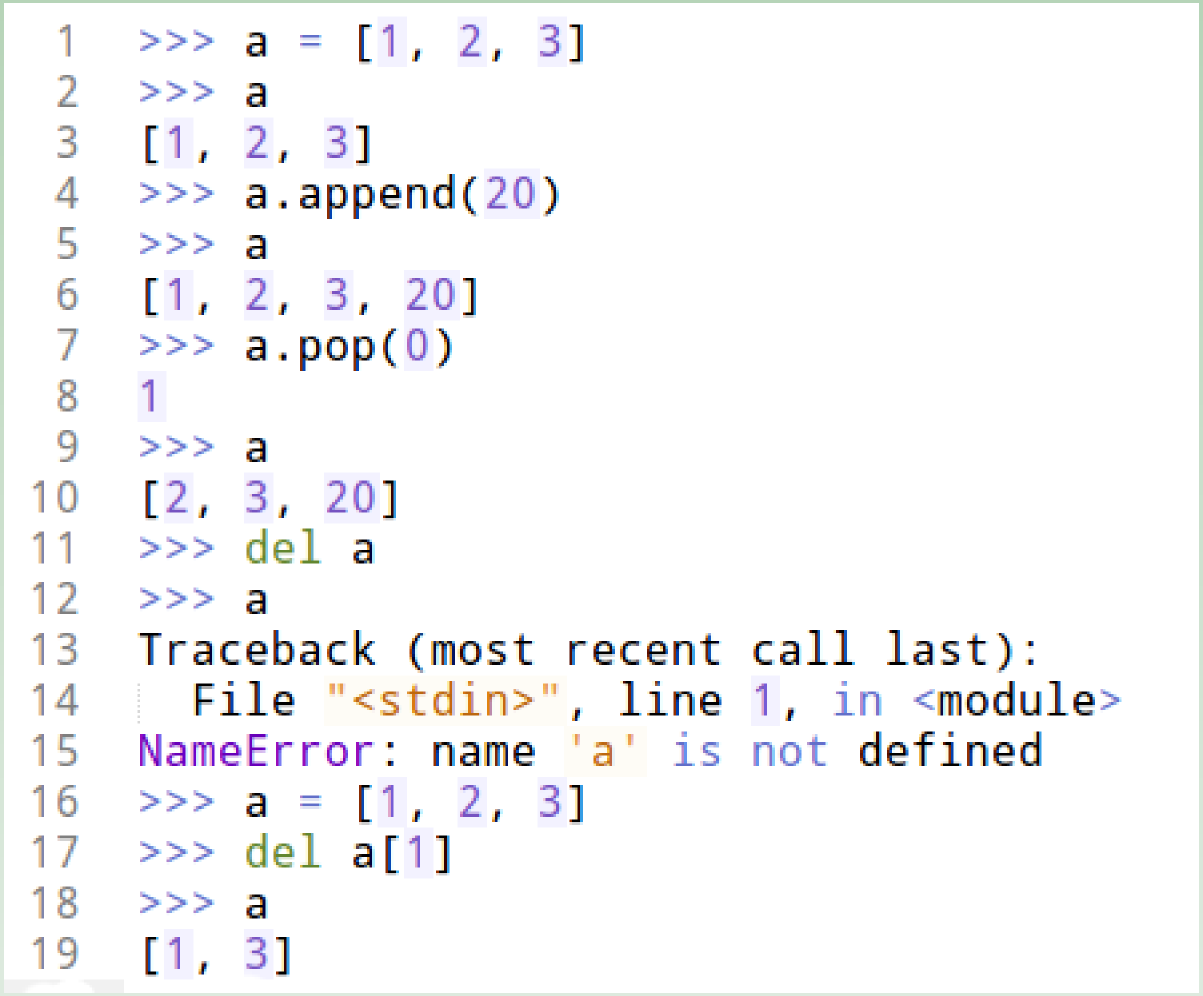
```
>>> a = range(5)
>>> a
[0, 1, 2, 3, 4]
>>> for e in a:
...     print e
...
0
1
2
3
4
>>> for e in a:
...     print e,
...
0 1 2 3 4
>>>
>>> import sys
>>>
>>> for e in a:
...     sys.stdout.write(e)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: expected a character buffer object
>>>
>>> for e in a:
...     sys.stdout.write(str(e))
...
01234>>>
```

1 („\n”)

2 (space)

3 („full control”)

Some list operations

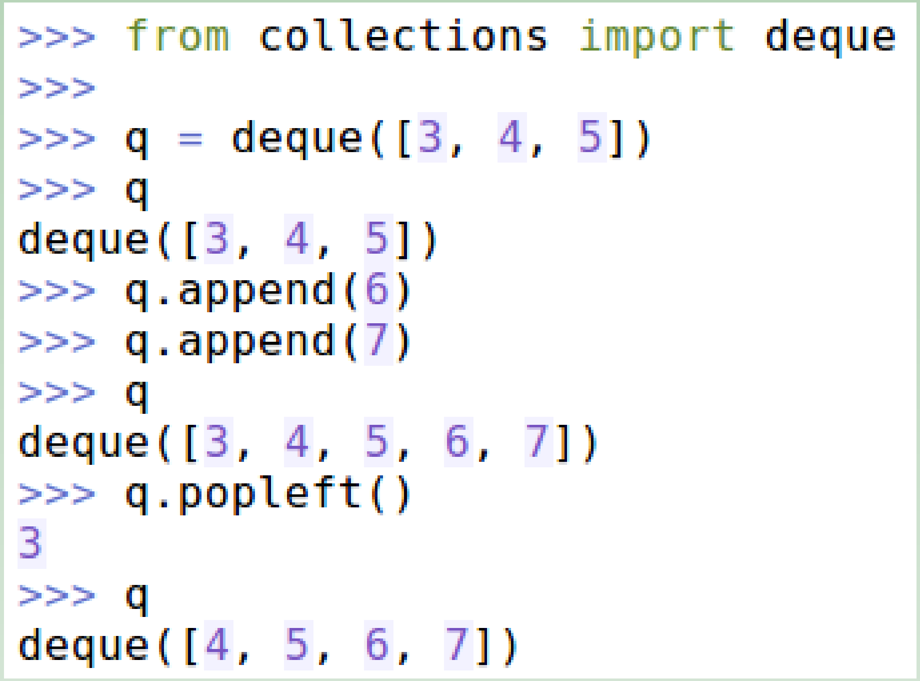


```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> a.append(20)
>>> a
[1, 2, 3, 20]
>>> a.pop(0)
1
>>> a
[2, 3, 20]
>>> del a
>>> a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>> a = [1, 2, 3]
>>> del a[1]
>>> a
[1, 3]
```

using a list as a **Stack**:

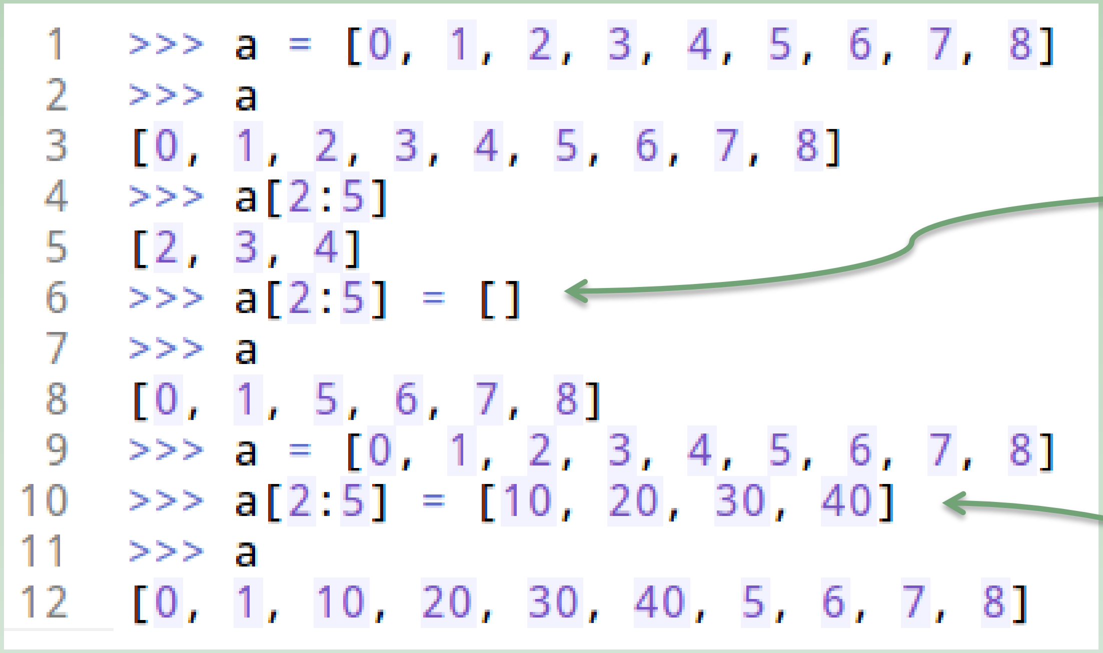
my_list.append(*elem*)
my_list.pop()

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



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

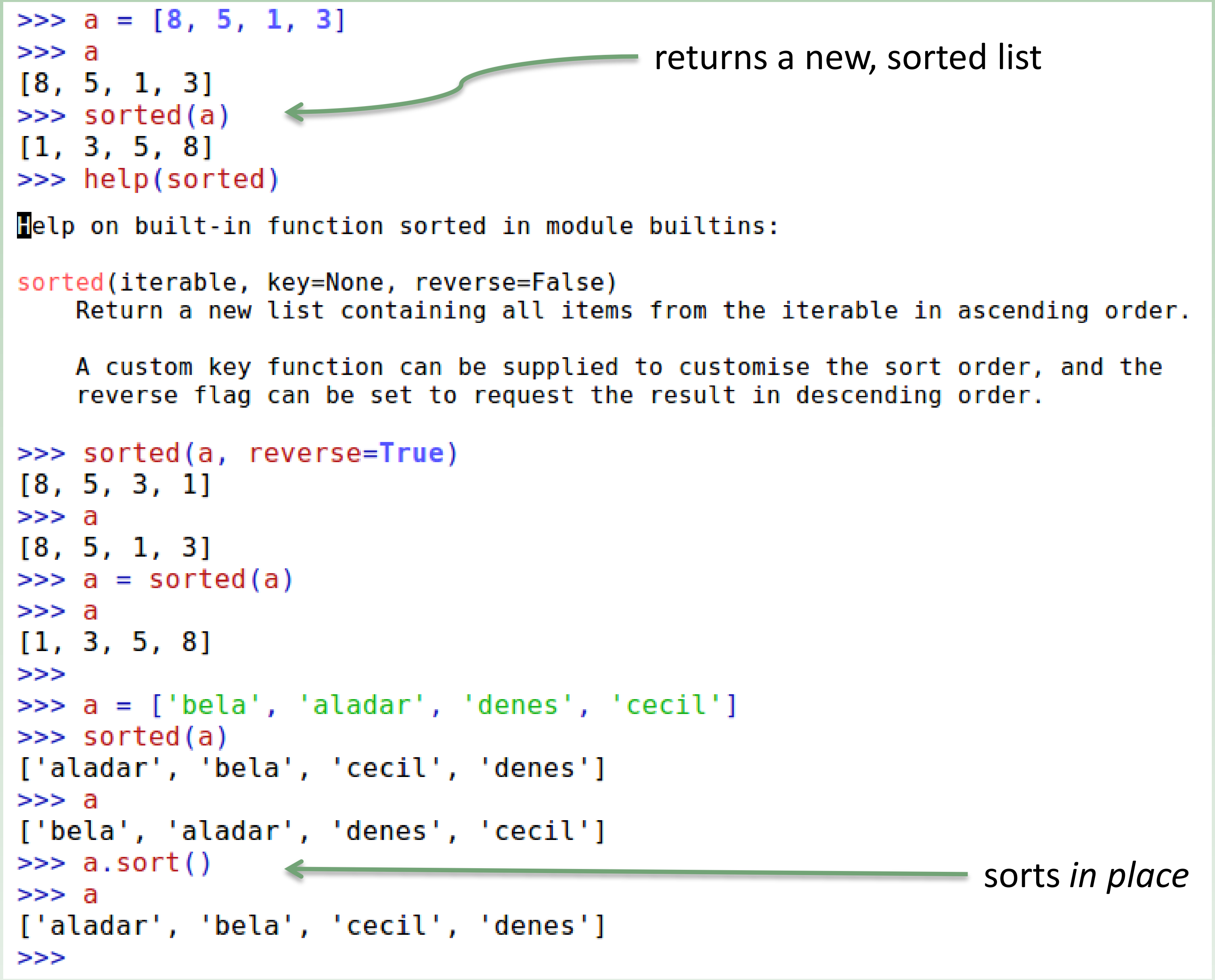
  Sort the list in place. It has no return value!
- `list.reverse()`

  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)

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.

>>> 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']
>>>
```

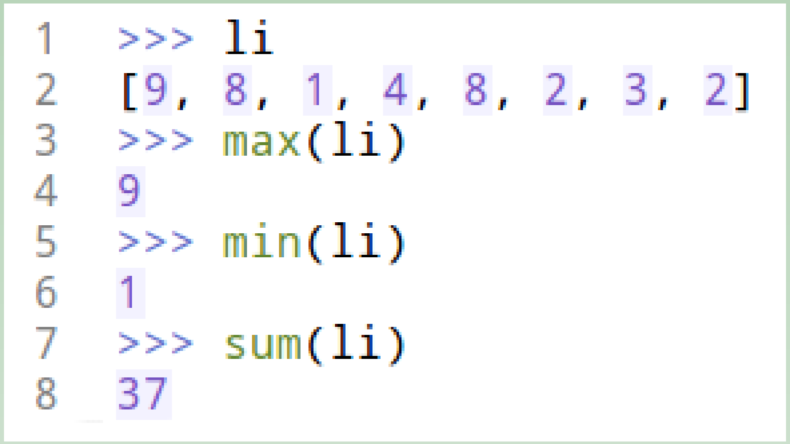
returns a new, sorted list

key, *reverse*: optional parameters

2

sorts *in place*

Some common operations with lists

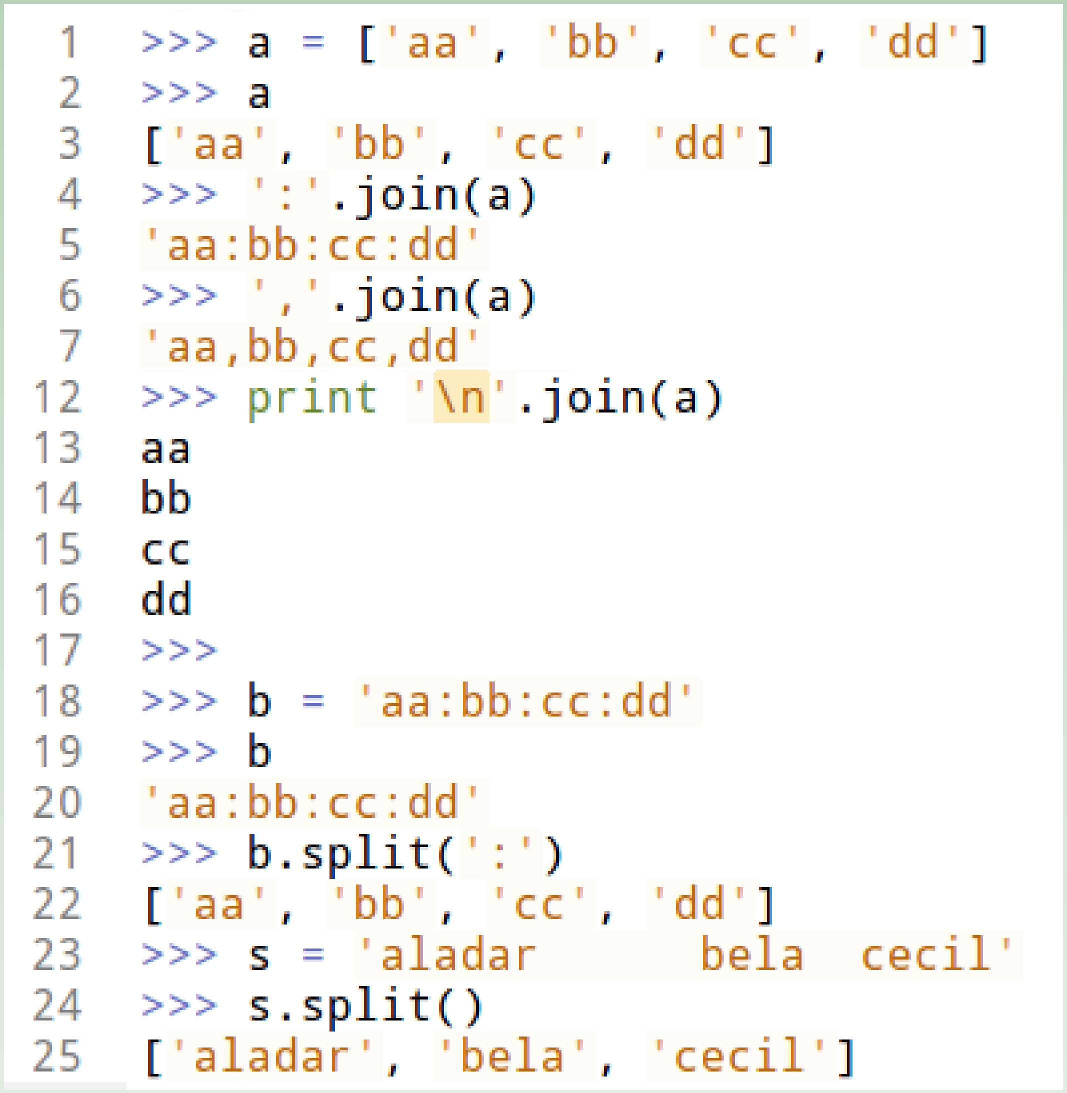


```
>>> li
[9, 8, 1, 4, 8, 2, 3, 2]
>>> max(li)
9
>>> min(li)
1
>>> sum(li)
37
```

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



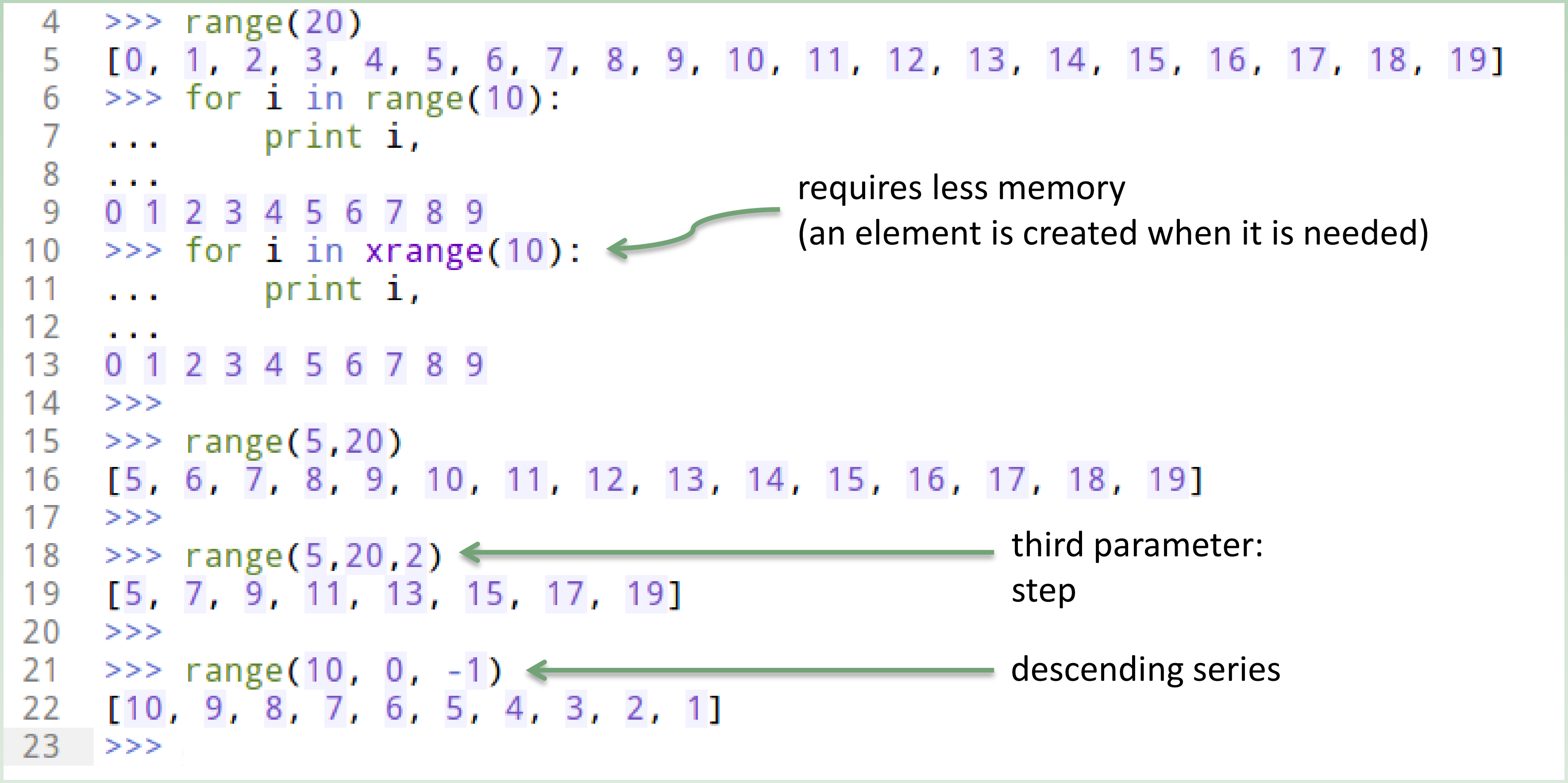
```
>>> a = ['aa', 'bb', 'cc', 'dd']
>>> a
['aa', 'bb', 'cc', 'dd']
>>> ':'.join(a)
'aa:bb:cc:dd'
>>> ','.join(a)
'aa,bb,cc,dd'
>>> print '\n'.join(a)
aa
bb
cc
dd
>>>
>>> b = 'aa:bb:cc:dd'
>>> b
'aa:bb:cc:dd'
>>> b.split(':')
['aa', 'bb', 'cc', 'dd']
>>> s = 'aladar      bela  cecil'
>>> s.split()
['aladar', 'bela', 'cecil']
```

list → string

by some delimiter

string → list

range / xrange



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

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

third parameter:
step

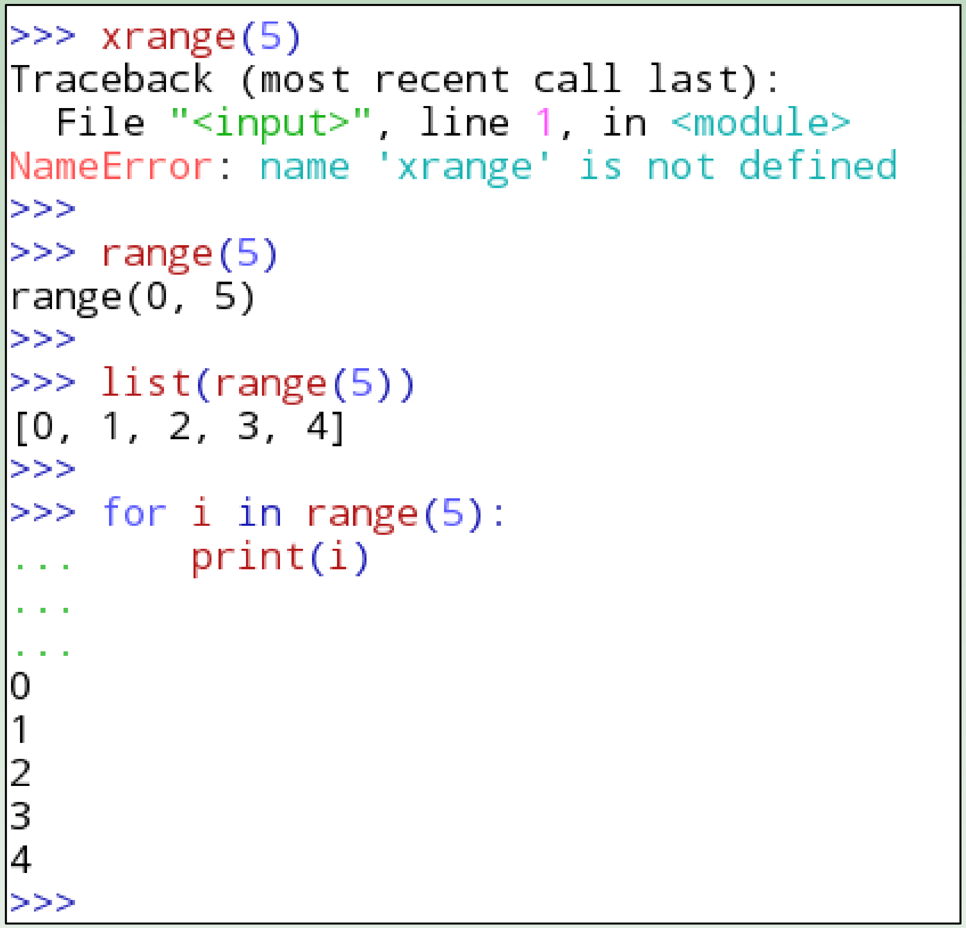
descending series

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

range

3

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
>>>
```

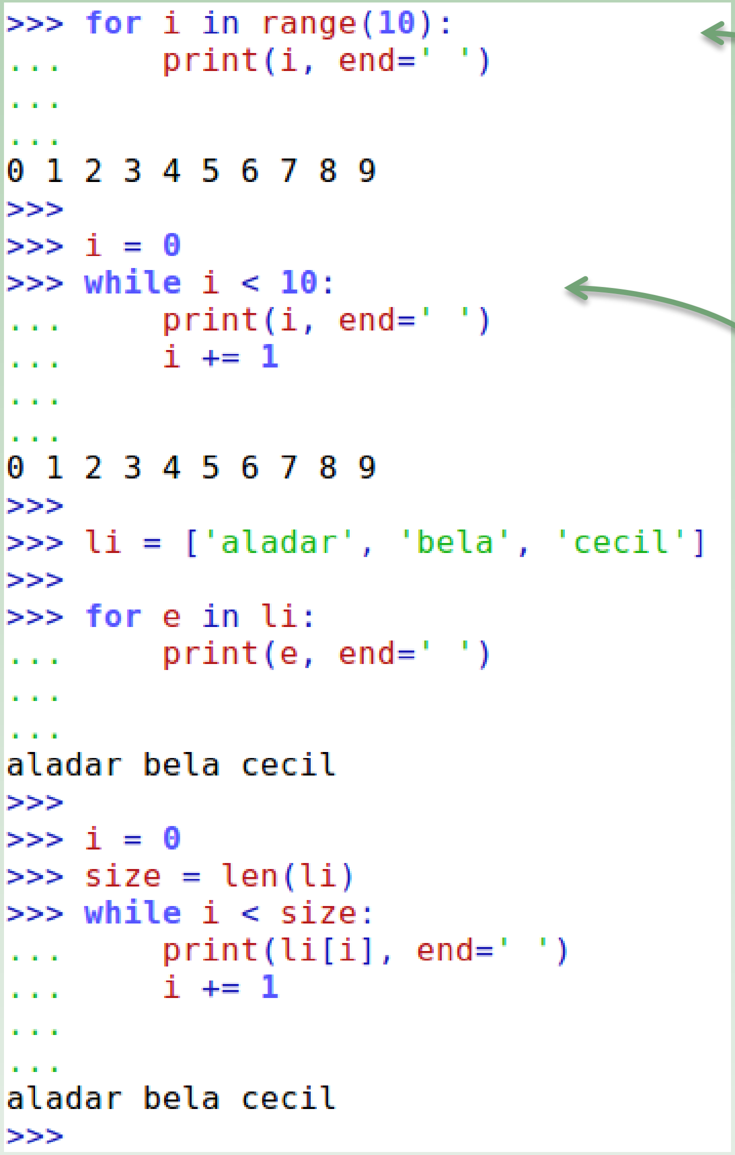
Exercise

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`

▢ homework

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)