Python Crash Course

General

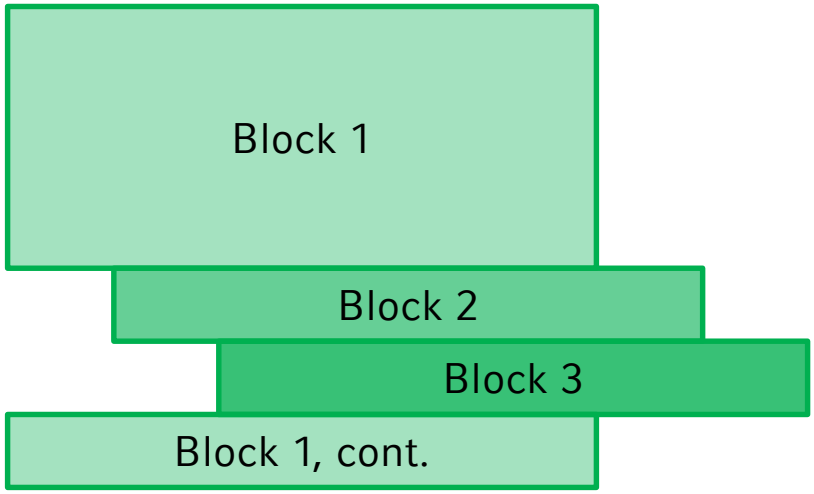
- Conceived in the late 1980s by Guido van Rossum at CWI in the Netherlands
- Successor to the ABC language
  - Improvement: small core language with a large standard library and easily extensible
- Multi-paradigm programming language
  - Object-oriented
  - Structured
  - Functional
  - …

The structure of a Python program

- Code blocks are defined by their indentations
  → Indentation is a requirement in Python!

- Structures that introduce blocks end with a colon ":"

```
from math import sqrt

my_list = [1,2,3,4]
result = 0
for i in my_list:
    if i%2 == 0:
        result += sqrt(i)
print(result)
```



Built-in Data Types: Numbers and Booleans

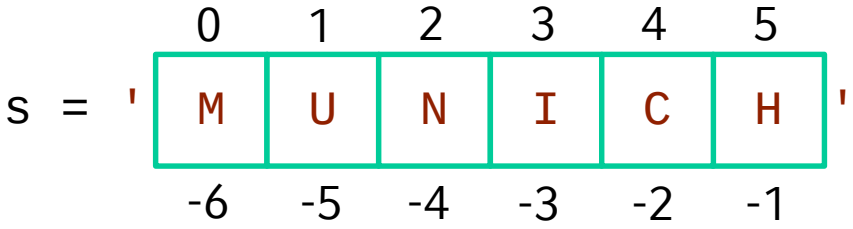
- Integer:
  - Normal Integer, e.g. `i = 345`
  - Octal Literals, e.g. `i = 0o10`
  - Hexadecimal Literals, e.g. `i = 0x1F`
  - Binary Literals, e.g. `i = 0b10110`
- Floating-point numbers
  - E.g. `i = 1.234e-2`
- Complex numbers
  - Composed of <real part> + <imaginary part>j, e.g. `i = 3+4j`
- Boolean Values
  - **`True`** and **`False`**

Built-in Data Types: String

- Strings are sequences of Unicode characters

- Marked by quotes:
  - Single-quote, e.g. ‘Hello, World!’
  - Double-quote, e.g. “Hello, World!”
  - Triple-quote, e.g. ‘’‘Hello, “World”!’’’

- Access:




```
>>> s[2]    >>> s[-1]    >>> s[2:]    >>> s[:-2]    >>> s[2:-2]
'N'         'H'          'NICH'       'MUNI'       'NI'
```

Common Operators

Operator	Description	Example
`+, -`	Addition, Subtraction	`12 + 3, 12 - 3`
`*, %`	Multiplication, Modulo	`12 * 3, 12 % 3`
`/`	Division	`10 / 3`
`//`	Floor Division	`10 // 3`
`**`	Exponentiation	`12**3`
`or, and, not`	Boolean operators	`not(True or False) and True`
`in`	Element of	`1 in [1,2,3]`
`<, <=, ==, !=, >=, >`	Comparison operators	`10 >= 3`
`|, &, ^`	Bitwise or, bitwise and, bitwise XOR	`6 ^ 3`

Data Structures: Python Lists


```
[0b100, ['times',True], 'is', 4.]
```

- Related to Java or C arrays, BUT more powerful
- List items do not need to have the same type
- Lists can grow dynamically
- Lists are ordered
- Lists are mutable and elements can be accessed by their index

Data Structures: Python Lists (cont.)

- Lists (resp. Iterables) are supported by many built-in functions
  - sum()
  - len()
  - max(), min()
  - …

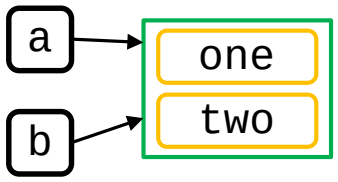
- List comprehension as an elegant way to create lists

```
>>> a = [x**2 for x in range(7)]
>>> a
[0, 1, 4, 9, 16, 25, 36]
>>> sum(a)
91
>>> a + [x**2 for x in range(7,9)]
[0, 1, 4, 9, 16, 25, 36, 49, 64]
>>> del a[:3]
[9, 16, 25, 36, 49, 64]
```

Excursus: Copying in Python

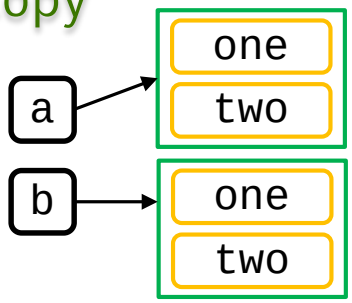
Assignment

```
>>> a = ['one','two']
>>> b = a
>>> print(id(a),id(b))
85992520 85992520
```



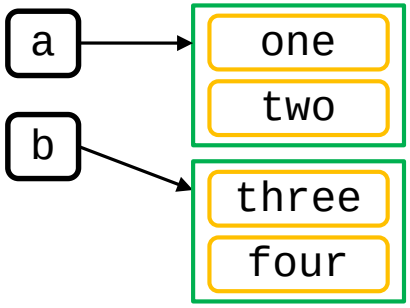
Shallow Copy

```
>>> a = ['one','two']
>>> b = a[:]
>>> print(id(a),id(b))
85992520 85995336
```



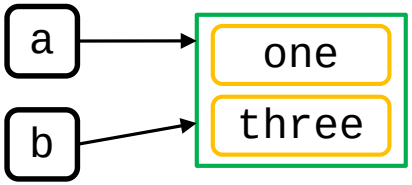
New Assignment

```
>>> b = ['three','four']
>>> print(id(a),id(b))
85992520 85995336
```



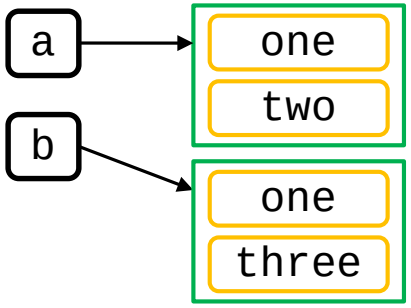
Side Effect

```
>>> b[1] = 'three'
>>> print(id(a),id(b))
85992520 85992520
```



No Side Effect

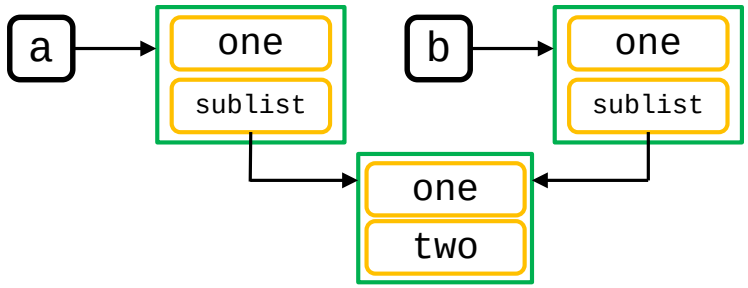
```
>>> b[1] = 'three'
>>> print(id(a),id(b))
85992520 85995336
```



Excursus: Copying in Python (cont.)

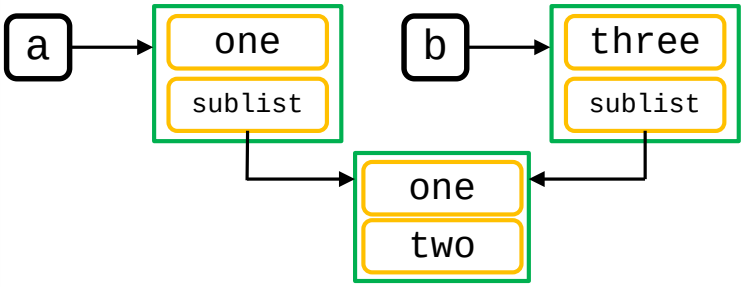
Shallow Copy

```
>>> a = ['one',['one','two']]
>>> b = a[:]
>>> print(id(a),id(b))
85992520 85995336
```



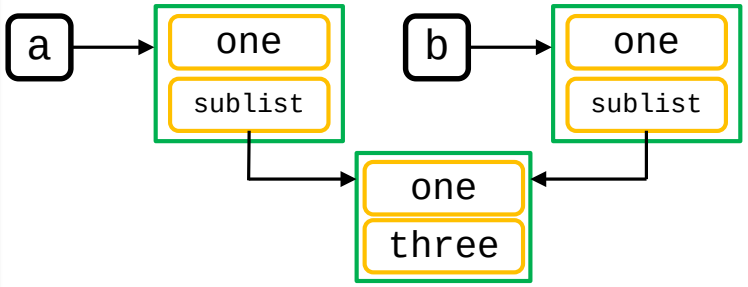
No Side Effect

```
>>> b[0] = 'three,
>>> print(id(a),id(b))
85992520 85995336
```



Side Effect

```
>>> b[1][1] = 'three'
>>> print(id(a),id(b))
85992520 85995336
```



Solution: the method deepcopy from the module copy

```
>>> from copy import deepcopy
>>> b = deepcopy(a)
```

Data Structures: Tuples


```
('A tuple with', 3, 'entries')
```

- A tuple is a sequence of comma separated values
- Values can have different types
- Tuples are immutable (but can contain mutable values)

```
>>> t = 1, [2], 'tuple' #tuple packing
>>> t[2]
'tuple'
>>> t[0] = 3
TypeError
>>> t[1][0] = 3
>>> t
(1, [3], 'tuple')
>>> x, y, z = t #sequence unpacking
```

Data Structures: Dictionaries


```
{'Munich': 1.5, 'Berlin': 3.5, 'Hamburg': 1.8 }
```

- Dictionaries are collections of (key,value) pairs
- Dictionaries are unordered
- Dictionaries are not sequence types like strings, lists or tuples
- Keys must be immutable, values can be of arbitrary type
- The types of keys, resp. values, must not be consistent

Data Structures: Dictionaries (cont.)

- Each key must be unique, since values are obtainable via the key

- Dictionaries also support comprehension

```
>>> d = { i**2: i for i in range(7) }
>>> d
{0: 0, 1: 1, 4: 2, 9: 3, 16: 4, 25: 5, 36: 6}
>>> d[4]
2
>>> for entry in d.items():
        if entry[0] == 4:
            print(entry)

(4,2)
>>> [key for key in d.keys()] #iterating over values is supported, too
[0, 1, 4, 9, 16, 25, 36]
>>> d[49] = 7 #delete values by using del key word, e.g. del d[36]
>>> d
{0: 0, 1: 1, 4: 2, 49: 7, 9: 3, 16: 4, 25: 5, 36: 6}
```

Conditional Statements and Loops

- Conditional Statements:

```
>>> if <condition1>:
        <block 1>
    elif <condition2>:
        <block 2>
    else:
        <block 3>

>>> a = 1 if (b > 2) else 0
```

- Loops:

```
>>> while <condition1>:
        <block 1>
    else: #else case can be avoided by using break or simply be omitted
        <block 2>

>>> for <variable> in <sequence>:
        <block 1>
    else:
        <block 2>
```

Functions and Lambda Functions

- Example of a simple Python function:

```
>>> def mult(a, b):
        return a*b

>>> mult(2, 3)
6
```

- Lambda functions are the anonymous throw-away equivalent
- Syntax: `lambda argument_list: expression`

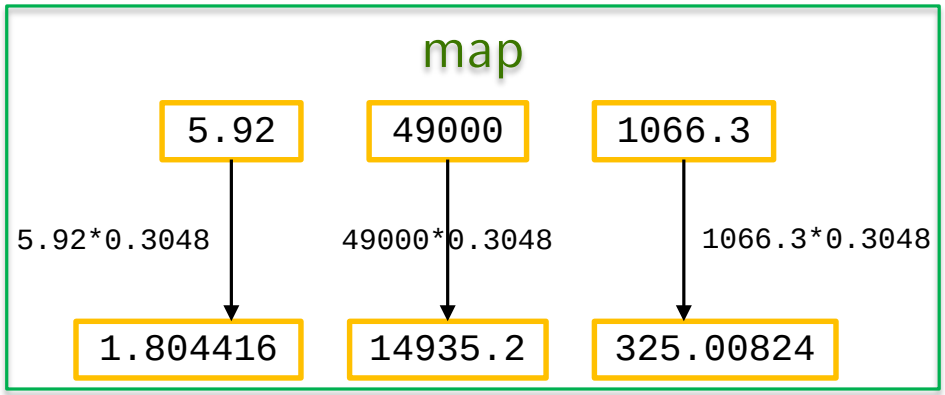
```
>>> mult = lambda x, y : x*y

>>> mult(2, 3)
6
```

- The lambda operator is mainly used for a special group of functions, i.e. `map()`, `filter()` and `reduce()`

map(), filter() and reduce()

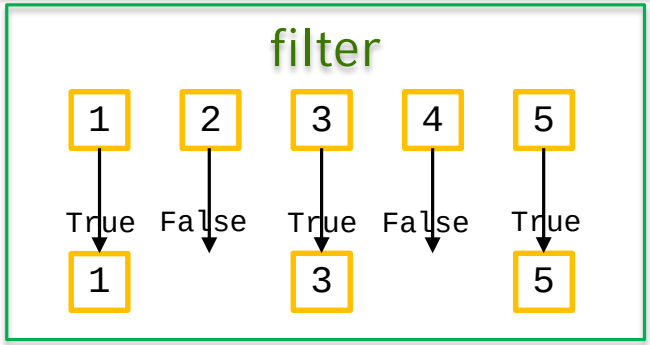
- map(func,seq)

```
>>> def feet_to_meter(x):
        return x*0.3048

>>> feet = [5.92, 49000, 1066.3]
>>> list(map(feet_to_meter, feet))
[1.804416, 14935.2, 325.00824]

>>> list(map(lambda x: x*0.3048, feet))
[1.804416, 14935.2, 325.00824]
```



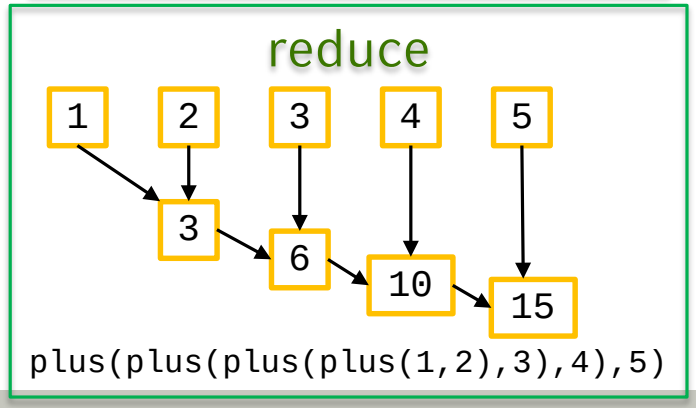
- filter(func,seq)

```
>>> list(filter(lambda x: x%2==1, [1,2,3,4,5]))
[1, 3, 5]
```



- reduce(func,seq)

```
>>> from functools import reduce
>>> reduce(lambda x,y: x+y, [1,2,3,4,5])
15
```



NumPy

- The fundamental package for scientific computing and core part of the SciPy stack
- Homogeneous multidimensional arrays as main objects
- Provides many arithmetic operations on arrays

```
>>> import numpy as np
>>> A = np.array([[1,2],[1,1]])
>>> A
array([[1, 2],
       [1, 1]])
>>> B = np.array([[0,1],[2,1]])
>>> A*B
array([[0, 2],
       [2, 1]])
>>> np.dot(A,B)
array([[4, 3],
       [2, 2]])
```

Further Information

Courses:

- *http://www.python-course.eu/python3_course.php*
  (english and german)
- *https://www.codecademy.com/en/tracks/python* (interactive course)

Downloads:

- *https://www.python.org/downloads/* (Python)
- *http://ipython.org/notebook.html* (Web Browser Interface)
- *http://continuum.io/downloads* (Full Distribution)

SciPy ecosystem:

- *http://www.scipy.org/*
  (containing Python, SciPy library, NumPy, Matplotlib, …)