Lists

Thomas Schwarz, SJ

Lists

- Python is a high-level programming language with built-in sophisticated data structures
  - E.g.: Containers
- The simplest of these data structures is the list.
- A list is just an ordered collection of other objects
  - The type of the objects is not restricted

- Let’s start unpacking this a bit.

Lists

- We create a list by using the square brackets.
  - `alist = [1, 3.5, "hello"]`
    - A list with three elements of three different types
  - `blist = [1, 3.5, "hello", 1]`
    - A list with four elements, where one element is repeated
  - `clist = [1, "hello", 3.5]`
    - A different list than alist, but with the same elements
    - The order is different

Lists

- Accessing elements in a list
  - We access elements in a list by using the square brackets and an index
  - Indices start at 0
- Example:
  - `lista = ['a', 'b', 'c', 'd']`
  - `lista[0]` is ‘a‘
  - `lista[1]` is ‘b’
  - `lista[2]` is ‘c’

Lists

- Python uses negative numbers in order to count from the back of the list
  - `lista = ['a', 'b', 'c', 'd']`
  - `lista[-1]` is the last object, namely the character ‘d’
  - `lista[-2]` is the second-last object, namely the character ‘c’
  - `lista[-4]` is the first object, namely the character ‘a’

Lists

- To check whether an object is part of the list, we use the *in* keyword
  - `x in a_lista`
- is true if x is in the list and false if it is not

```
lista_a = [1,2,3]
lista_b = [2,3,4]
if 1 in lista_a:
   print('true')
if 1 in lista_b:
   print('false')
```

prints out

```
true
false
```

Lists

- The in-function is very fast, because of the way membership test is arranged.

Lists and for loops

- The for-loop in Python iterates through a list (or more generally an iterator)
  - `for x in lista:`
  - `x` takes on all values in `lista`