INDEXING & SLICING

9.11.2020

INDEXING

* sequence[index]

  * for indexes between 0 and len(sequence)-1, this returns the corresponding element of sequence

INDEXING

* sequence[index]

  * for indexes between -1 and -len(sequence), this *counts backward from the end of the sequence*

SLICING

* sequence[start:end]

  * returns a list containing the elements of sequence between start and end (**including start**, but **not end**!)

  * either start or end can be negative

SLICING

* sequence = [0, 1, 2, 3, 4, 5]

* sequence[2:-2] => ?

SLICING

* sequence = [0, 1, 2, 3, 4, 5]

* sequence[-5:3] => ?

SLICING

* sequence[start:]
    * returns all elements after (and including) start

SLICING

```
* sequence = [0, 1, 2, 3, 4, 5]

* sequence[4:] => ?
```

SLICING

* sequence[:end]
  * returns all elements before end

SLICING

* sequence = [0, 1, 2, 3, 4, 5]
* sequence[:-1] => ?

SLICING

* sequence[start:end:step]
  * slice from start to end, taking every step’th element
  * either start or end (or both) can be omitted

SLICING

* sequence = [0, 1, 2, 3, 4, 5]
* sequence[::2] => ?
* sequence[1::2] => ?

SLICING

* sequence[start:end:step]
    * negative step sizes => negative steps

SLICING

* sequence = [0, 1, 2, 3, 4, 5]

* sequence[::-1] => ?

NESTED INDEXING

* If you have a **nested** data structure (e.g. a list of lists) then you can stack indexing operations together

NESTED INDEXING

* list_of_lists[2][0]

    * *gives you the 0’th element of the 2’th list in* list_of_lists

MEMBERSHIP

* for lists, the **in** keyword tells you whether an object is in the list

MEMBERSHIP

* my_list = [1, 9, “rat”, “bingo”, -2.7]

* “rat” in my_list => True

* “bango” in my_list => False

MEMBERSHIP

* for dictionaries, the **in** keyword tells you whether a key is in the dictionary

MEMBERSHIP

* my_dict = {1: “one”, 2: “two”, 3: “lol nine”}

* 1 in my_dict => True

* “one” in my_dict => False

END