Organising information: ordered structures

Silvio Peroni
silvio.peroni@unibo.it 0000-0003-0530-4305 @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)
Second Cycle Degree in Digital Humanities and Digital Knowledge
Alma Mater Studiorum - Università di Bologna

Communication 1

Take it as a warm suggestion: please start doing the exercises and send them in the mailing list, so as to start a discussion about them

Communication 2

There are no other communications so far

Any question about the previous lecture?

Historic hero: Donald Knuth

Donald Knuth is a Computer Scientist

Main contributions: theoretical and practical development of the analysis of the computational complexity of algorithms, The Art of Computer Programming series of monographs, TeX typesetting system for writing academic documents

The Art of Computer Programming is an ongoing work (3 volumes and an half out of 7 have been published so far)

Data structures

The first volume of Knuth's series of books is entirely dedicated to a comprehensive introduction of all the basic data structures

A *data structure* is a way in which we can organise the information to process and returned by a computer, so as it can be accessed and modified in an efficient and computational manner

Broadly speaking, it is a sort of bucket where we can place some information, that provides some methods to add and retrieve pieces of such information

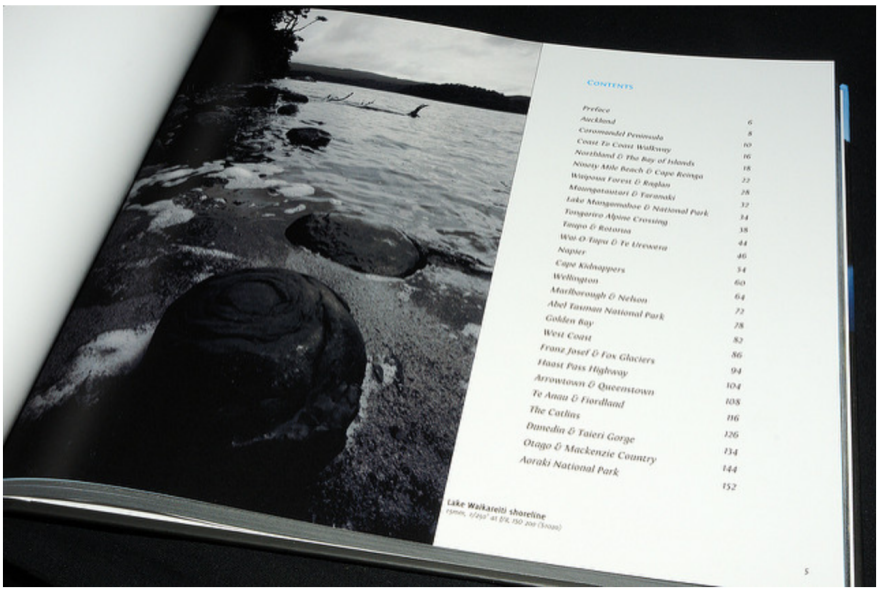
Order and repeatability

Among the simplest data structures, there are those ones that organise their element in a specific order and allow the repeatability of the values they contain

Order: the sequence in which the elements are added to these data structures matters

Repeatability: the same value can appear twice or more times in the same data structure

List: example



Research Articles in Simplified HTML: a Web-first format for HTML-based scholarly articles

Silvio Peroni[1], Francesco Osborne[2], Angelo Di Iorio[1], Andrea Giovanni Nuzzolese[3], Francesco Poggi[1], Fabio Vitali[1] and Enrico Motta[2]

[1] Digital and Semantic Publishing Laboratory, Department of Computer Science and Engineering, University of Bologna, Bologna, Italy
[2] Knowledge Media Institute, Open University, Milton Keynes, United Kingdom
[3] Semantic Technologies Laboratory, Institute of Cognitive Sciences and Technologies, Italian National Research Council, Rome, Italy

ABSTRACT

Purpose. This paper introduces the Research Articles in Simplified HTML (or RASH), which is a Web-first format for writing HTML-based scholarly papers; it is accompanied by the RASH Framework, a set of tools for interacting with RASH-based articles. The paper also presents an evaluation that involved authors and reviewers of RASH articles submitted to the SAVE-SD 2015 and SAVE-SD 2016 workshops.

REFERENCES

Alexander C. 1979. *The timeless way of building*. Oxford: Oxford University Press.

Atkins Jr T, Etemad EJ, Rivoal F. 2017. CSS Snapshot 2017. W3C Working Group Note 31 January 2017. World Wide Web Consortium. *Available at https://www.w3.org/TR/css3-roadmap/*.

Berjon R, Ballesteros S. 2015. What is scholarly HTML? *Available at http://scholarly.vernacular.io/*.

Bourne PE, Clark T, Dale R, De Waard A, Herman I, Hovy EH, Shotton D. 2011. FORCE11 White Paper: improving The Future of Research Communications and e-Scholarship. White Paper, 28 October 2011. FORCE11. *Available at https://www.force11.org/white_paper*.

Brooke J. 1996. SUS-A quick and dirty usability scale. *Usability Evaluation in Industry* **189(194)**:4–7.

Capadisli S, Guy A, Verborgh R, Lange C, Auer S, Berners-Lee T. 2017. Decentralised authoring, annotations and notifications for a read-write web with dokieli. In: *Proceedings of the 17th international conference on web engineering*. Cham: Springer, 469–481 DOI 10.1007/978-3-319-60131-1_33.

List: definition

A list is a countable sequence of ordered and repeatable elements

Countable: it is possible to know the length of the list (i.e. how many elements it contains) – in ThyMopani, we can use the support algorithm `def len(countable_object)`

Ordered: the elements are placed in the list in a specific order, which is preserved

Repeatable: the elements may appear more than one time in the list

List: methods

Create a new list: `list()`

Add new element: `<list>.append(<element>)`

Remove element: `<list>.remove(<element>)`

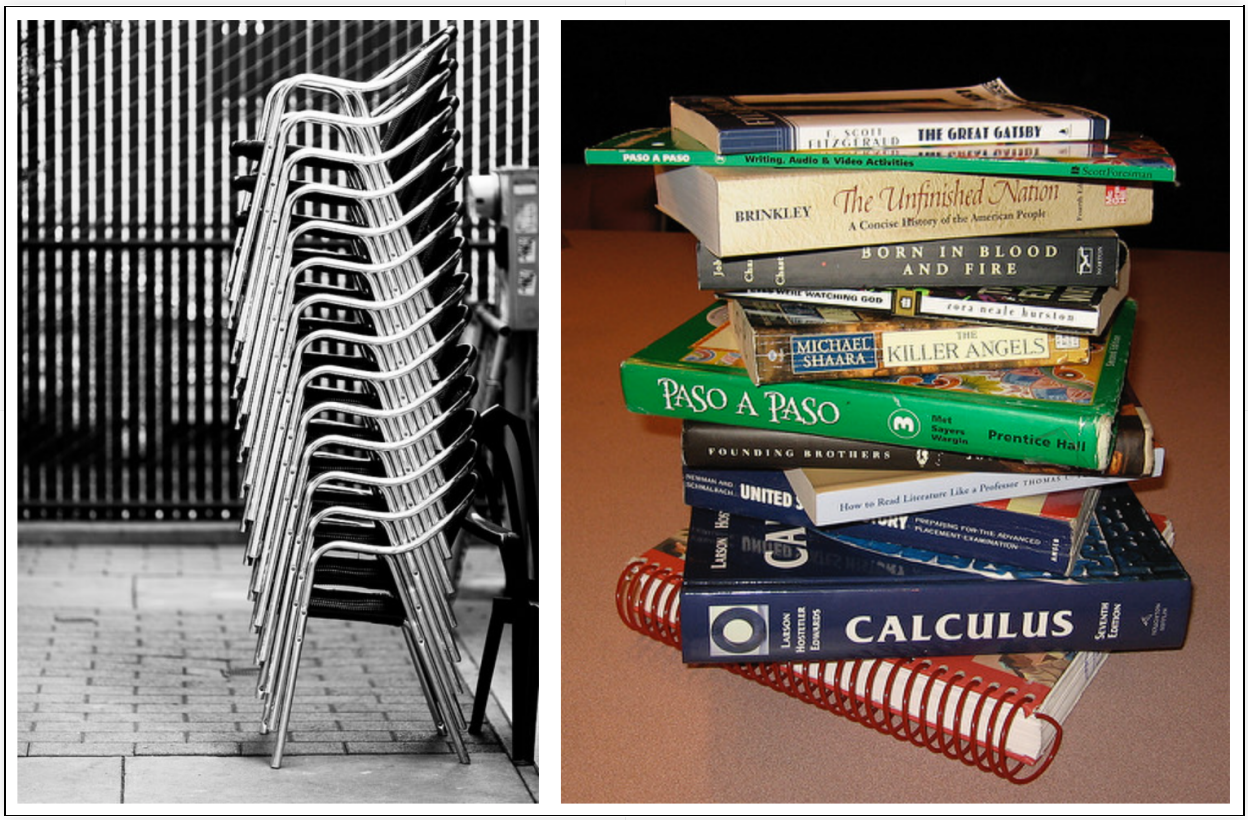
Add elements from another list:
`<list>.extend(<another_list>)`

Use methods on list

```
my_first_list = list()
my_first_list.append(34)
my_first_list.append(15)
my_first_list.append("Silvio")
my_first_list.remove(34)
my_first_list.extend(my_first_list)
```

`my_first_list =` 15 "Silvio" 15 "Silvio"

Stack: example



Stack: definition

A stack is a is a kind of list seen from a particular perspective, i.e. from bottom to top, and with a specific set of operations

The elements follow a *last in first out* strategy (LIFO) for addition and removal

The last element inserted in the structure is placed in the top of the stack and, thus, it is also the first one that will be removed when requested

For removing the element in the middle of the stack one has to remove all the elements that have been added after such middle element

Stack: methods

Create a new stack: `deque()`

Add new element: `<stack>.append(<element>)`

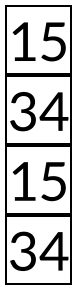
Remove (and return) latest element: `<stack>.pop()`

Add elements from another stack: `<stack>.extend(<another_stack>)`

Use methods on stack

```
my_first_stack = deque()
my_first_stack.append(34)
my_first_stack.append(15)
my_first_stack.extend(my_first_stack)
my_first_stack.append("Silvio")
my_first_stack.pop()
```

my_first_stack =



Queue: example

Queue: definition

A queue is a is a kind of list seen from a particular perspective, i.e. from left to right, and with a specific set of operations

The elements follow a *first in first out* strategy (FIFO) for addition and removal

The first element is placed in the left-most part of the queue and, thus, it is also the first one that will be removed when requested

For removing the element in the middle of the queue one has to remove all the elements that have been added before such middle element

Queue: methods

Create a new queue: `deque()`

Add new element: `<queue>.append(<element>)`

Remove (and return) first element: `<queue>.leftpop()`

Add elements from another queue:
`<queue>.extend(<another_queue>)`

Use methods on queue

```
my_first_queue = deque()
my_first_queue.append(34)
my_first_queue.append(15)
my_first_queue.append("Silvio")
my_first_queue.leftpop()
my_first_queue.extend(my_first_queue)
```

`my_first_queue =` 15 "Silvio" 15 "Silvio"

END

Organising information: ordered structures

Silvio Peroni

silvio.peroni@unibo.it 0000-0003-0530-4305 @essepuntato

Computational Thinking and Programming (A.Y. 2017/2018)

Second Cycle Degree in Digital Humanities and Digital Knowledge

Alma Mater Studiorum - Università di Bologna