Algorithms

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

As of yesterday, only 11 people have subscribed to the mailing list

Please subscribe to the mailing list as soon as possible

1. Open a browser and go to https://www.dsa.unibo.it/
2. Log in with your University email
3. Look for the mailing list (“*Liste docenti-studenti*”) - the name of the list is “*compthink1718*”
4. Subscribe to it (password needed, please ask it to me)

Then, you can send directly emails to silvio.peroni.compthink1718@studio.unibo.it

Communication 2

Exam sessions dates (informal):

- 23 January 2018
- 26 February 2018
- 10 May 2018
- 22 June 2018
- 25 September 2018
- 23 October 2018

For each date:

- written examination: 10-12
- project oral colloquium: 13-16

Any question about the previous lecture?

Historic hero: Ada Lovelace

English mathematicians

English translator of the first article about Babbage's Analytical Engine, written by Luigi Federico Menabrea in French

She enriched the English translation of the article with several annotations

Original article: ~8000 words

Notes by Ada: additional ~19000 words

Ada’s heritage

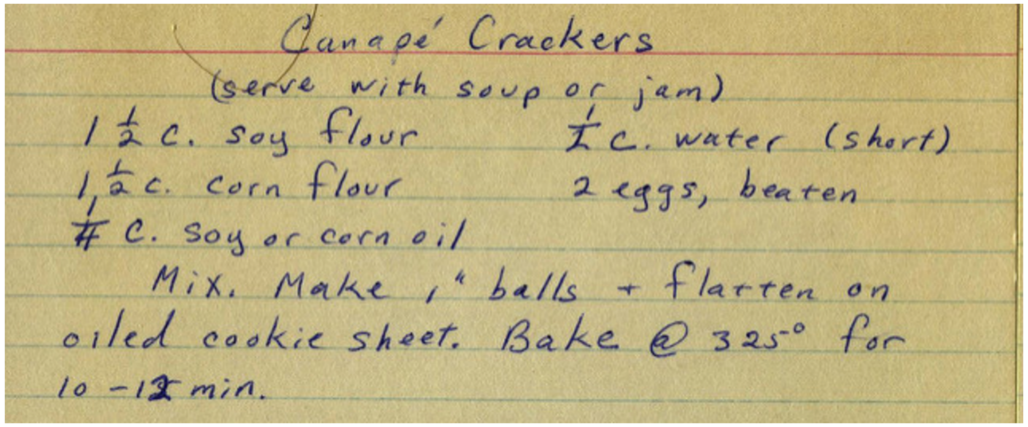
First mechanical computer programmer: in the article notes, there is a description of how to use the Analytical Engine to calculate the Bernoulli numbers

The operating mechanism can even be thrown into action independently of any object to operate upon (although of course no result could then be developed). Again, it might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent.

Science of Operations = Computer Science

Common features

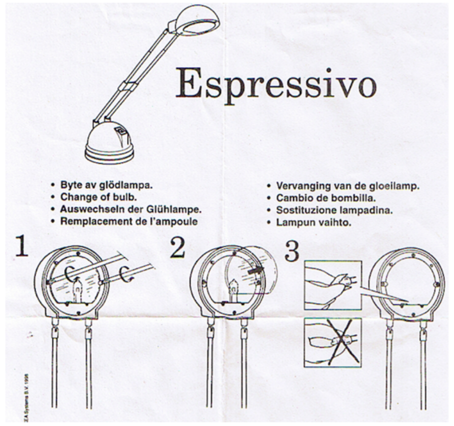
What do recipes and instructions for assembling objects have in common?



Canapé Crackers
(serve with soup or jam)

1 ½ c. soy flour — ½ c. water (short)
1 ½ c. corn flour — 2 eggs, beaten
¼ c. soy or corn oil

Mix. Make 1" balls + flatten on oiled cookie sheet. Bake @ 325° for 10 - 12 min.


Espressivo
• Byte av glödlampa.
• Change of bulb.
• Auswechseln der Glühlampe.
• Remplacement de l'ampoule
• Vervanging van de gloeilamp.
• Cambio de bombilla.
• Sostituzione lampadina.
• Lampun vaihto.
1
2
3

Shared abstract notion: step-by-step procedure for producing something starting from some initial material we have = algorithm

Algorithm

The word *“algorithm”* is a combination of the Latin word *algorismus* (that is the Latinization of the name Al-Khwarizmi, who was a great mathematician from Persia in the 8th century) and the Greek word *arithmos*, meaning *number*

Each algorithm is written in a specific language which is functional to communicate its instruction to a computer

Computer programmer: anyone that creates algorithms that can be interpreted by any computer

Pseudocode

A pseudocode in an informal language that could be interpreted easily by any computer

It is used by Computer Scientists for sketching an algorithm before to implement it in a particular programming language

Checking if an algorithm can be interpretable by a computer: provide a colleague with the pseudocode of the algorithm and ask her to execute it using a particular input, writing down all the passages of the execution on a piece of paper

ThyMopani

It is a pseudocode enough expressive for allowing one to describe any kind of algorithm that can be implemented in any programming language

Our first algorithm: taking in input three different strings, i.e. two words and a bibliographic entry of a published paper, return 2 if both the words are contained in the bibliographic entry, 1 if at least one of the words is contained in the bibliographic entry, and 0 otherwise

E.g.: input `"Berners-Lee"`, `"web"` and `"Tim Berners-Lee: Designing the web for an open society. WWW 2011: 3-4"`; output `2`.

Defining an algorithm

```
def <algorithm_name>(<parameter_1>, <parameter_2>, ...)
```

`<algorithm_name>` and `<parameter_i>` cannot contain spaces and cannot start with a number

```
def contains_word(first_word, second_word, bibliographic_entry):
    ...
    ...
    ...
```

All the instructions of the algorithm must be specified in the following lines, as an indented block

A *variable* is a symbolic name that contains some information referred to as a value (e.g. `first_word`, which is a particular kind of variable, called parameter)

A first step

Partial algorithm: if the first input word is contained in the bibliographic entry, then the number 1 is returned, otherwise 0 is returned

We need:

- a mechanism to return a particular value if a specific condition is true
- a way for checking if the input word is contained in the bibliographic reference
- a command for returning the result

Conditional block

It allows one to execute a particular instruction if a condition is true (the `if` statement), while an alternative set of instructions is executed instead if the condition specified is false (the `else` statement)

```
if <condition>:
     ...
     ...
else:
     ...
     ...
```

Specify the condition

The command `in` ise used to check if a certain string is contained in another one

`<string1> in <string2>` would be true if the value <string1> is contained in <string2>

A *string* is a particular type of value that records a sequence of characters, and it is usually defined by using the quotes – e.g. `"Berners-Lee"`

Return statement

```
return <value_to_return>
```

The execution of a return statement finishes the whole execution of an algorithm, and all the instructions that follow that statement are not processed anymore

E.g.: `return 1`

A *number* is defined by writing it down as it is – e.g. 42 and -42 for positive/negative integers, 1.625 and -1.625 for positive/negative decimals

`"42"` is different from `42`

Partial algorithm

```python
def contains_word(first_word, second_word, bibliographic_entry):
    if first_word in bibliographic_entry:
        return 1
    else:
        return 0
```

Boolean values

The description of the algorithm says to return 2 if both the input words are contained in the bibliographic entry

A *boolean* can be assigned to one out of two distinct and disjoint values, `True` and `False`

E.g.: `first_word in bibliographic_entry` returns a boolean value

Logical operations

`<operator> <B1>`, where `<operator>` can be only `not`

`<B1> <operator> <B2>`, where `<operator>` can be either `or` or `and`

B1	**B2**	**not B1**	**B1 and B2**	**B1 or B2**
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Comparisons

Strings: `<S1> <operator> <S2>`

S1	S2	S1 < S2	S1 <= S2	S1 > S2	S1 >= S2	S1 == S2	S1 != S2	S1 in S2	S1 not in S2
"Alice"	"Bob"	True	True	False	False	False	True	False	True
"Alice"	"Alice"	False	True	False	True	True	False	True	False

Numbers: `<N1> <operator> <N2>`

N1	N2	N1 < N2	N1 <= N2	N1 > N2	N1 >= N2	N1 == N2	N1 != N2
3	4	True	True	False	False	False	True
4	4	False	True	False	True	True	False

Complete algorithm

```
def contains_word(first_word, second_word, bibliographic_entry):
    if first_word in bibligraphic_entry and
       second_word in bibliographic_entry:
        return 2
    else:
        if first_word in bibliographic_entry or
           second_word in bibliographic_entry:
            return 1
        else:
            return 0
```

Avoiding repetitions

`first_word in bibliographic_entry` and `second_word in bibliographic_entry` used and evaluated twice

It can be avoided defining new variables: `<variable_name> = <variable_value>`

For instance:

- `contains_first_word = first_word in bibliographic_entry`
- `contains_second_word = second_word in bibliographic_entry`

Collapsing else-if blocks

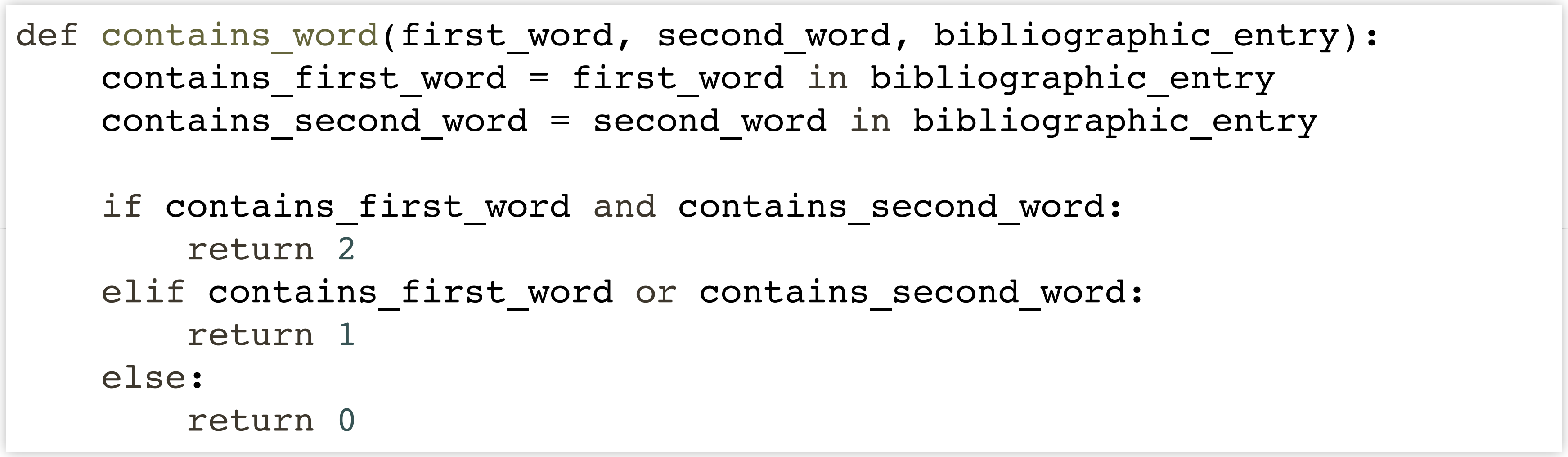
It is possible to collapse occurrences of `else` statements when these contain an `if` statement as their first instruction.

In this case, the *else-if* pair can be safely replaced by an `elif` statement

```
if <condition1>:
    ...
else:
    if <condition2>:
        ...
    else:
        ...
```

```
if <condition1>:
    ...
elif <condition2>:
    ...
else:
    ...
```

Final algorithm



```python
def contains_word(first_word, second_word, bibliographic_entry):
    contains_first_word = first_word in bibliographic_entry
    contains_second_word = second_word in bibliographic_entry

    if contains_first_word and contains_second_word:
        return 2
    elif contains_first_word or contains_second_word:
        return 1
    else:
        return 0
```

END

Algorithms

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