Algolab
(index.html#Chapters)
Out[1]:

Module 2: Algorithm and Data Structures Lab

David Leoni
teaching assistant
david.leoni [AT] unitn.it

University of Trento

21 Dic 2016
v0.11.1

Today

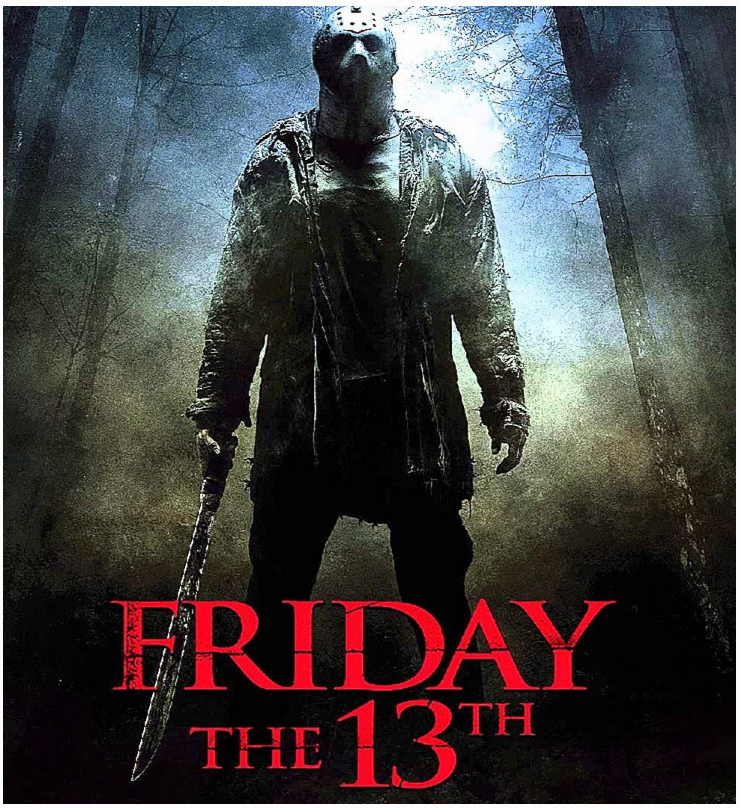
Exam simulation (exam-sim-2016-12-21.html) !

See you

Date	Hour	Topic
Thursday 22 Dec	9:00-11:00	Graph algos / Exam simulation discussion

Midterm

Friday 13th Jan 14:00-17:00



Course website

Theory

See Alberto Montresor website:

http://cricca.disi.unitn.it/montresor/teaching/scientific-programming (http://cricca.disi.unitn.it/montresor/teaching/scientific-programming)

Lab

davidleoni.github.io/algolab (https://davidleoni.github.io/algolab)

Chapters

Worksheets are meant to be used online - pdf quality is not very good, if they result unreadable please tell me

Commandments

> **WARNING: If you don't follow the Commandments, bad things happen!**

> **1) You shall test!**

To run tests, enter the following command in the terminal:

```
python -m unittest my-file
```

> **WARNING: In the call above, DON'T append the extension *.py* to *my-file***
> **WARNING: Still, on the hard-disk the file MUST be named with a *.py* at the end, like *my-file.py***

> **2. You shall also write on paper!**

If staring at the monitor doesn't work, help yourself and draw a representation of the state sof the program. Tables, nodes, arrows, all can help figuring out a solution for the problem.

> **3. You shall copy *exactly the same* function definitions as in the exercises!**

For example don't write :

```
def MY_selection_sort(A):
```

> **4. You shall never ever reassign function parameters:**

```
def myfun(i, s, L, D):

        # You shall not do any of such evil, no matter what the type of the p
arameter is:
        i = 666            # basic types (int, float, ...)
        s = "666"          # strings
        L = [666]          # containers
        D = {"evil":666}   # dictionaries

        # For the sole case of composite parameters like lists or dictionarie
s,
        # you can write stuff like this IF AND ONLY IF the function specifica
tion
        # requires you to modify the parameter internal elements (i.e. sortin
g a list
        # or changing a dictionary field):

        L[4] = 2          # list
        D.my_field = 5    # dictionary
```

This also applies to class methods. Never ever write horrors such as:

```
class MyClass
    def my_method(self, x, y):
        self = {a:666}  # since self is a kind of dictionary, you might be te
mpted to do like this
                        # but to the outside world this will bring no effect.
                        # For example, let's say somebody from outside makes
a call like this:
                        #    mc = MyClass()
                        #    mc.my_method()
                        # after the call mc will not point to {a:666}
        self = ['666']  # self is only supposed to be a sort of dictionary an
d passed from outside
        self = 6        # self is only supposed to be a sort of dictionary an
d passed from outside
```

> **5. You shall never ever assign values to method calls:**

WRONG WRONG STUFF

```
my_fun() = 666
my_fun() = '666'
my_fun() = [666]
```

CORRECT STUFF

With the assignment operator we want to store in the left side a value from the right side, so all of these are valid operations:

```
x = 5
y = my_fun()
z = []
z[0] = 7
d = {}
d["a"] = 6
```

Function calls such as `my_fun()` return instead results of calculations in a box that is created just for the purpose of the call and Python will just not allow us to reuse it as a variable. So whenever you see 'name()' at the left side, it *can't* be possibly followed by one equality `=` sign (but it can be followed by two equality signs `==` if you are performing a comparison).

> **6. You shall use *return* command only if you see written *return* in the pseudocode!**

If there is no `return` in the pseudocode, the function is intended to return `None`. In this case you don't even need to write `return None`, as Python will do it implicitly for you.

Slides

Lab 1 Slides

3 Nov 2016

Lab Goals

- Going from theory taught by Prof. Alberto Montresor to implementation
- Pseudo code --> Python

How

- Hands-on approach
- Performance considerations
- Focus on correct code
- Few Python functions

Lab Midterm?

Probably not. Still, will provide exam example.

Lab 2 Slides

Date: Nov 11th, 2016

- More practical than last time!
- Finish selection_sort and gap implementation
- midlab pause ;-)
- implement a Python class

Lab 3 Slides

Nov 17th, 2016

- Recursion
  - `gap_rec`, `binary_search_rec`
- `binary_search_iter`
- Will give you more tests
- Write also on paper!
- Copy *exactly the same* function definitions!
  - For example don't write `def MY_selection_sort(A):`
- use `return` command *only* if you see written `return` in the pseudocode!
  - If there is no `return` in the pseudocode, the function is intended to return `None`. In this case you don't even need to write `return None`, as Python will do it implicitly for you.

Lab 4 Slides

Nov 18th, 2016

- Divide et Impera
  - binary_search_iter
- Implement ComplexNumber

New Commandment:

You shall never ever reassign function parameters:

```
def myfun(L, i, s):

    # You shall not do any of this evil:
    L = [666]
    i = 666
    s = "666"
```

Previous commandments:

- You shall also write on paper!
- You shall copy exactly the same function definitions as in the exercises!
- For example don't write def MY_selection_sort(A):
- You shall use return command only if you see Written return in the pseudocode!
- If there is no return in the pseudocode, the function is intended to return None. In this case you don't even need to write return None, as Python will do it implicitly for you.

Lab 5 slides

24 Nov 2016

- Implement ComplexNumber
- Implement Stack

Lab 6 slides

1 Dic 2016

- Implement `CappedStack`
- Implement `UnorderedList`

Lab 7 slides

- UnorderedList

Lab 8 slides

- will start writing tests (will be required during exams)
- invariants in particular
- `UnorderedList` (see Chapter 2 (data-structures.html)) until fast `size()` and `append()` included

Lab 9 slides

15 Dic 2016

- Implement `GenericTree`, see Trees chapter (trees.html)

Lab 10 slides

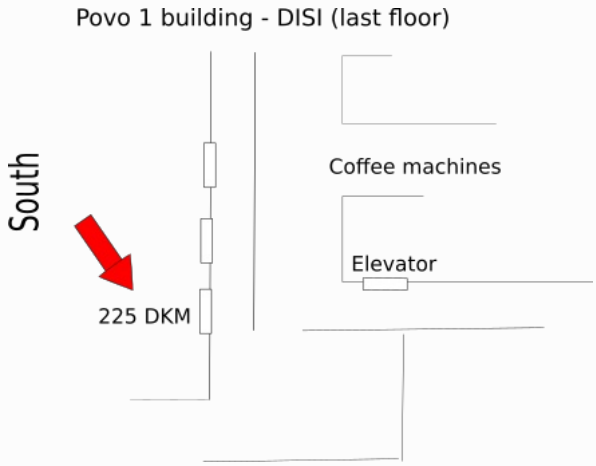
19 Dic 2016

- `GenericTree` see Graphs chapter (graphs.html)

Office hours

You can schedule a meeting by emailing me at david.leoni [AT] unitn.it , more or less I'm available every day until 19.00

Then you will find me in Povo 1 building at DISI, in room 225 DKM :



Resources

- Online book: Problem Solving with Algorithms and Data Structures using Python (http://interactivepython.org/runestone/static/pythonds/index.html) by Brad Miller and David Ranum
- Theory slides (http://cricca.disi.unitn.it/montresor/teaching/scientific-programming/slides/) by Alberto Montresor
- Will try to be consistent with other lab module notes (http://disi.unitn.it/~teso/courses/sciprog/index.html) of Stefano Teso and Toma Tebaldi
- PythonTutor (http://www.pythontutor.com/visualize.html#mode=edit), a visual virtual machine (*very useful*! can also be found in examples inside the book!)
- Source code (https://github.com/DavidLeoni/algolab) of these worksheets (download zip (https://github.com/DavidLeoni/algolab/archive/master.zip)), in Jupyter Notebook (http://jupyter.org) format.
- The internet ....

Editors

- Jupyter Notebook (http://jupyter.org): Nice environment to execute Python commands and display results like graphs. Allows to include documentation in Markdown format
- Spyder (https://pythonhosted.org/spyder/): Should be a fine editor, although I haven't used it in a long time