Exam Simulation

Scientific Programming Algolab

Wednesday 21 Dec 2016

Introduction

- This is just an exam simulation. If you don't ship it or do a poor work, you lose nothing. If you make a good work, you earn nothing.
- To make you feel more comfortable, all what you produce and submission process will be completely anonymous
- Anonymous doesn't imply private. In fact, in this simulation, the work will be made public and corrections will be public as well. The anonimity means your work and corrections won't be associated to your name or student id.
- So don't put your name or id number around in the files ;-)

Allowed material

Real exam will be in the lab with restricted internet access. You will only be able to access:

- Sciprog Algolab website (http://davidleoni.github.io/algolab)
- Alberto Montresor slides (http://cricca.disi.unitn.it/montresor/teaching/scientific-programming/)
- Stefano Teso slides (http://disi.unitn.it/~teso/courses/sciprog/index.html)
- Python 2.7 documentation (https://docs.python.org/2/)
  - In particular, Unittest docs (https://docs.python.org/2/library/unittest.html)
- The course book 'Problem Solving with Algorithms and Data Structures using Python' (http://interactivepython.org/runestone/static/pythonds/index.html)

You won't be able to use anything else, in particular:

- no Google
- no StackOverflow

So if you need to look up some Python function, please start today learning how to search documentation using the search functionality on Python website.

Grading

- We need still need to define a grading system for the exam. Here we just put provisional rules.
- Correct implementations with the required complexity grant you full grade.
- One extra point can be earned by writing stylish code. You got style if you:
  - do not infringe the Commandments (index.html#Commandments)
  - write pythonic code (http://docs.python-guide.org/en/latest/writing/style)
  - avoid convoluted code like i.e.

    ```
    if x > 5:
        return True
    else:
        return False
    ```

    when you could write just

    ```
    return x > 5
    ```

!!!!!!!!! WARNING !!!!!!!!!

!!!!!!!!! **ONLY IMPLEMENTATIONS OF THE PROVIDED FUNCTION SIGNATURES WILL BE EVALUATED !!!!!!!!!**

For example, if you are given to implement:

```
def cool_fun(x):
        raise Exception("TODO implement me")
```

and you ship this code:

```
def cool_fun_non_working_trial(x):
    # do some absurdity

def cool_fun_a_perfectly_working_trial(x):
    # a super fast, correct and stylish implementation

def cool_fun(x):
    raise Exception("TODO implement me")
```

We will assess only the latter one `cool_fun(x)`, and conclude it doesn't work at all :P !!!!!!!

Still, you are allowed to define any extra helper function you might need. If your `cool_fun(x)` implementation calls some other function you defined like `my_helper` here, it is ok:

```
def my_helper(y,z):
    # do something useful

def cool_fun(x):
    my_helper(x,5)

# this will get ignored:
def some_trial(x):
    # do some absurdity
```

What to do

1) Download this zip (exam-simulation/exam.zip) and extracts its contents on your disk. You will see a folder like this:

```
exam
|- exercise1.py
|- exercise2.py
```

2) Edit the files following the instructions in this worksheet for each exercise.

WARNING: *DON'T* modify function signatures! Just provide the implementation.

WARNING: *DON'T* change the existing test methods, just add new ones !!! You can add as many as you want.

WARNING: *DON'T* create other files. If you still do it, they won't be evaluated.

IMPORTANT: Pay close attention to the comments of the functions.

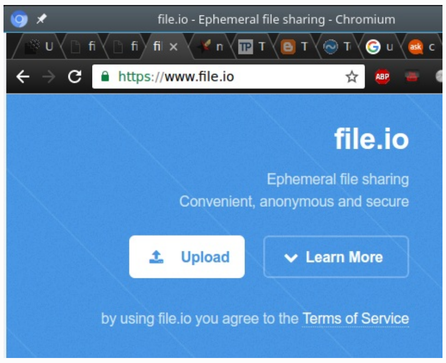
3) Every exercise should take max 25 mins. If it takes longer, leave it and try another exercise.

4) When you're done, proceed with the submission

Submission

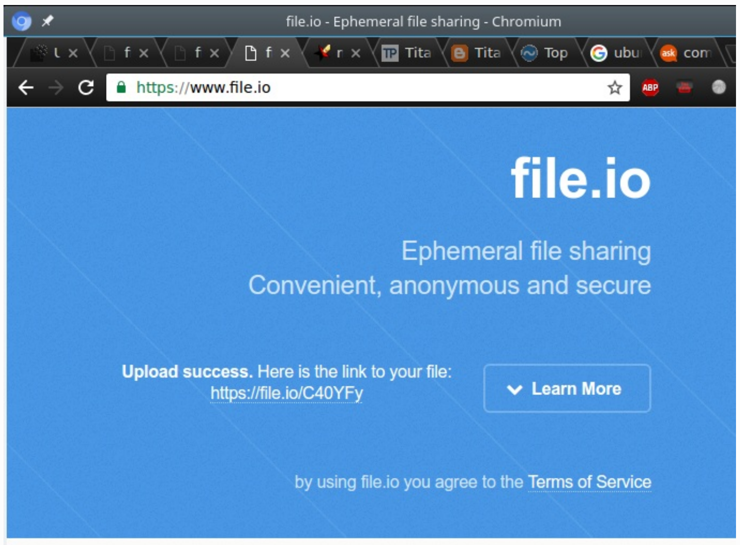
1) Zip your files into a file named `exam.zip`

2) Upload the zip to file.io (http://file.io)


file.io - Ephemeral file sharing - Chromium
https://www.file.io
file.io
Ephemeral file sharing
Convenient, anonymous and secure
Upload
Learn More
by using file.io you agree to the Terms of Service

(http://file.io)

2) After the upload you should see a download link (something like `https://file.io/C40YFy`)


file.io - Ephemeral file sharing - Chromium
https://www.file.io
file.io
Ephemeral file sharing
Convenient, anonymous and secure
Learn More
by using file.io you agree to the Terms of Service

Upload success. Here is the link to your file:
https://file.io/C40YFy

3) Copy paste the link into the common ethercalc here:

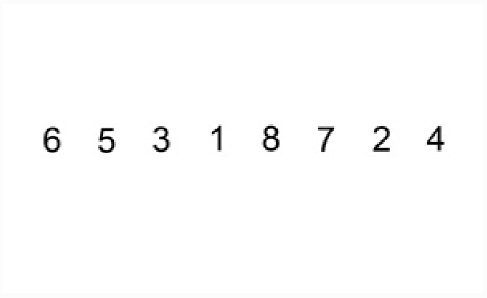
https://ethercalc.org/2sp752kuzqqy (https://ethercalc.org/2sp752kuzqqy)

> **WARNING: REMEMBER TO WRITE SOMEWHERE (IN YOUR COMPUTER OR ON PAPER) YOUR DOWNLOAD LINK, OTHERWISE IT WILL BE HARD FOR YOU TO UNDERSTAND WHICH ZIP WAS CORRECTED.**

Exercises

1) insertion sort

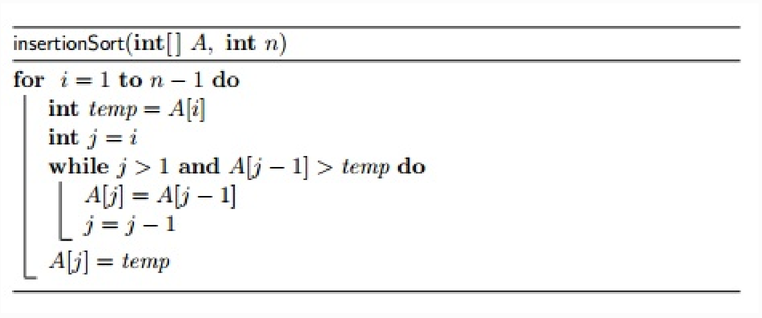
Insertion sort is a basic sorting algorithm. This animation gives you an idea of how it works:



Here is the pseudo code:

> **WARNING: The following pseudo code contains a bug (just one!).**

> **IMPORTANT:**
> **Array *A* in the pseudo code has indexes starting from zero included.**
> ***n* is the length of the input array.**



```
insertionSort(int[] $A$, int $n$)
for $i = 1$ to $n - 1$ do
    int $temp = A[i]$
    int $j = i$
    while $j > 1$ and $A[j-1] > temp$ do
        $A[j] = A[j-1]$
        $j = j - 1$
    $A[j] = temp$
```

Start editing the file `exercise1.py`:

1.1) insertion_sort

Implement in Python the pseudocode for `insertion_sort`. Make sure at least the provided tests pass (they won't check for the bug).

1.2) Fixing insertion_sort

We said the given pseudo code has a bug. Write additional test cases that show where the bug is, and then fix the code accordingly.

> **WARNING: *DON'T* modify function signatures! Just provide the implementation.**

> **WARNING: *DON'T* change the existing test methods, just add new ones !!! You can add as many as you want.**

> **IMPORTANT: Pay close attention to the comments of the functions.**

2) UnorderedList

We are going to have some more fun with good old `UnorderedList`, which is a monodirectional linked list.

Start editing the file `exercise2.py`:

2.1) rev(self)

Implement the method `rev(self)` that you find in the skeleton and check provided tests pass.

2.2) copy(self)

Implement the method `copy(self)` that you find in the skeleton and check provided tests pass.