Possible Python preparation

- Repeat basics in lectures and tutorials.
- Look through your solution to the assignment.
- Go through the previous exams.
- Go through the lectures in more detail, note keywords and go through some of the code examples.
Common mistakes

- Missing self-parameter
- Not reseting variables
- Mixing up index variables
- Start/end-cases in loops
Parameters are references..

- ..but usually doesn't change outside functions, unless changing the internal data of a mutable type:

```python
a = [0,1,2]
print "original: " + str(a)
def change_elem(foo):
    foo[1] = 3
change_elem(a)
print "after change_elem(a): " + str(a)
```

Output:
* original: [0, 1, 2]*
* after change_elem(a): [0, 3, 2]*

```python
def change(foo):
    foo = [4,5]
change(a)
print "after change(a): " + str(a)
```
Important for Python part

- Understanding the concepts.
- Knowing the terms covered in the lectures.
- Being able to read code, understand somewhat more difficult code pieces and spot bugs.
Python language check list

- Data storage: numeric types, sequences, mutable/immutable types, object referencing
- Flow control: if, while, for, yield, return, error handling
- Sequence processing: map, filter
- Partitioning code: functions, classes, modules
- String processing: regular expressions
- Tools and libraries: those mentioned
Functions & classes

def sqr(y):
    return y*y

class Person:
    def __init__(self, name):
        self.name = name
    def print_name(self):
        print self.name
Classes

- Attributes
- Constructor, destructor
- Class & instance variables
- Functions within classes
C++ part of the exam

- Focused on writing code.
- Small syntax errors not important.
- Being able to compare Python to C++ in different regards.
Some control questions

- What is the motivation behind the template?
- Why is templates not necessary in Python?
You should be able to..

- write classes
- write templates
- use STL containers
- handle pointers
- allocate and deallocate memory

and not only knowing how a class works but also a little bit about designing them. E.g. being able to identify class variables that are needed for a class to perform a specified task.
You are not expected to know..

- .. exactly how to set up a Python – C/C++ binding.
- .. all the functions that are available for the different STL containers.
Some advice

• Drawing the internal program structure on a piece of paper can be great for getting a bird-eye-view.

• Difficult to find the right organization right away. Challenging problems often requires code to be rewritten several times.

• Find the trade-offs