Inf-1400 - Objektorientert Programmering

Introduction

John Markus Bjørndalen

Department of Computer Science
University of Tromsø
Norway

2017-01-17

Course staff

Responsible / **lecturer**
John Markus Bjørndalen

Vitass/TA
Isak Singh
Raymon Hansen
TBA

Topics

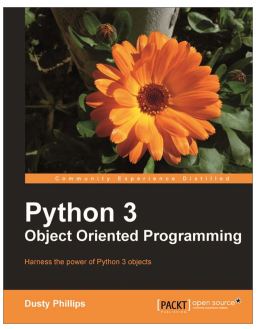
- About the course
- Practical things
- Introduction to Object-Oriented Programming

Course description - INF-1400

Innhold
Emnet gir en innføring i objektorientert programmering. Dette inkluderer en innføring i sentrale begreper som innkapsling, klasser, arv og polymorfi. Metoder, grensesnitt og samspill mellom objekter er også sentralt i emnet.

`http://uit.no/studiekatalog/emner/2017/var/inf-1400-1`

Textbook



Python 3 Object Oriented Programming, Dusty Phillips
`http://shop.oreilly.com/product/9781849511261.do`

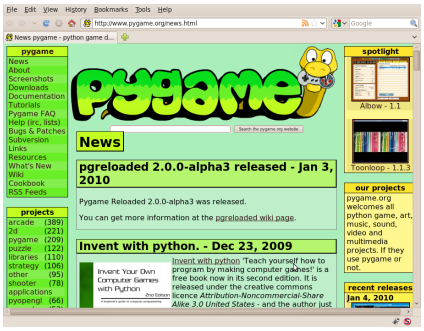
(Some) Python advantages

- High Level programming language, also useful for scripting
- Described as “executable pseudo code”
- Readable (and often short!) code
- Plenty of libraries
- Interpreted

One of Python’s drawbacks

- Interpreted (slow, but not always – see recent optimizations of NumPy)

PyGame



Used for the mandatory exercises. Based on SDL.
`http://www.pygame.org/`

Mandatory exercises

Last year:

- Breakout / Arkanoid (game)
- Boids (simulation of flocking birds)
- Mayhem / XPilot (game)

Fronter

- Make sure you register for the course and have access to Fronter (**`http://fronter.com/uit`**).
- We do not have a good tool for tracking students with mandatory assignments from previous years. To avoid confusion when we approve students for the exam, please hand in something on the mandatory assignments. An empty file named `I_DID_THIS_IN_2016.txt` would be fine (we can then check that you did it).

Fronter vs. GitLab

Course material provided through GitHub

- Project URL `https://github.com/uit-inf-1400-2017/uit-inf-1400-2017.github.io`
- Can be used as a normal web page from a browser (mobile, laptop, desktop). No log-in required.
- A full snapshot of the current course materials: look for “Clone or download”

Any changes (additions, updates etc) to the gitlab repository will be visible in the Fronter room (look for the RSS box).
You can also use that RSS url in your own RSS reader if you want to:
`https://github.com/uit-inf-1400-2017/uit-inf-1400-2017.github.io/commits/master.atom`

Git - the super-easy crash course

For those that want a fully updated copy of the course material without using a browser: use Git.

Initial checkout (get access to current files):

```
git clone git@github.com:uit-inf-1400-2017/uit-inf-1400-2017.github.io.git
```

If you don’t have a GitHub account, use https:

```
git clone https://github.com/uit-inf-1400-2017/uit-inf-1400-2017.github.io
```

Updating your copy:

```
# cd somewhere inside the course folder, then:
git pull
```

We will talk more about git later. It’s very useful for group projects.

Other messages

- Tutorials from Friday (10:15-12 this Friday. In the IfI lab.)
- First mandatory exercise handed out early this year.

Quick OO-introduction: lists

- List abstraction from inf-1100: a data structure and a set of functions.
- Want to provide multiple list types:
  - General list implementation
  - Optimized for adding items at the end
  - Optimized for accessing the nth item
  - A sorted list that stays sorted when it is modified
- Need multiple data structures and a set of functions for each data structure.

Quick OO-introduction: lists

Given a list pointer, which list_add() function do we call?

```
...
  list_add_last(list, item); ?
  sorted_list_add(list, item); ?
  append_list_add(list, item); ?
...
```


Quick OO-introduction: lists

One solution test for list type
- test everywhere you use a list?

```
...
if (list->type == LIST_TYPE_DEFAULT)
   list_add_last(list, item);
else if (list->type == LIST_TYPE_SORTED)
   sorted_list_add(list, item);
else if (list->type == LIST_TYPE_APPEND)
   append_list_add(list, item);
...
```

And modify all locations when you introduce a new type?

Quick OO-introduction: lists

One solution test for list type
- test everywhere you use a list?

```
...
if (list->type == LIST_TYPE_DEFAULT)
   list_add_last(list, item);
else if (list->type == LIST_TYPE_SORTED)
   sorted_list_add(list, item);
else if (list->type == LIST_TYPE_APPEND)
   append_list_add(list, item);
...
```

And modify all locations when you introduce a new type?

Quick OO-introduction: lists

OO-solution: add the functions (methods) to the objects.

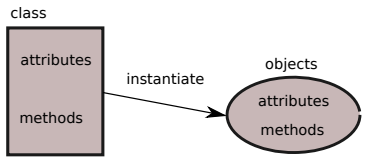
```
// C++ class
class List {
   ListNode *head;
   int numItems;
   ...
   int add(void *item) { return addLast(item); }
   int addLast(void *item);
};

int List::addLast(void *item)
{
...
```


Quick OO-introduction

Basic idea:

- The object has a known interface
- Allows the user/programmer to handle objects without caring about the implementation (for instance how data is stored and managed)
- Can use multiple implementations (types/classes) as long as they have the same interface

Quick OO-introduction



Instantiate: in practice similar to the allocate + initialize methods from the ADTs you made before Christmas.

Classes define how objects work (both implementation and interface) and give them an identity (as an object of that class).

Quick OO-introduction

Basic concepts:

- The object abstraction includes methods, or operations on the objects.
- Objects are created (instantiated) based on classes. A single object is called an instance of its class.
  - Example: a string object is an instance of a String class.
- Data in a class or object is called an attribute (ex: int numItems).
- Operations (functions) in a class are called methods.
- Signature: parameters and return value of a method.

Next part

- More on objects
- Introduction to Python