



Міністерство освіти і науки, молоді та спорту України Національний Технічний Університет України  
«Київський Політехнічний Інститут» Навчально-науковий комплекс  
«Інститут прикладного системного аналізу» Кафедра системного проектування

## **«ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ СИСТЕМ»**

### **Лабораторная работа №4,5,6**

Виконав студент

групи ДА-21

Осадчий Дмитро

Київ 2015

## ЛАБОРАТОРНА РОБОТА №4

### Модульне тестування (Unit-тести) та рефакторинг.

**Мета роботи:** оволодіти навичками створення програмного забезпечення за методологією

TDD та ознайомитися з процедурами рефакторинга.

```
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

:Example:

>>> import testDocsPy
>>> a = testDocsPy.MainClass1()
>>> a.function1(1,1,1)
2

.. note:: can be useful to emphasize
        important feature
.. seealso:: :class:`MainClass2`
.. warning:: arg2 must be non-zero.
.. todo:: check that arg2 is non zero.
"""
return arg1/arg2 + arg3 + 1

if __name__ == "__main__":
    import doctest
    doctest.testmod()

*****
File "/Users/dimoo/testDocsPy.py", line 110, in __main__.MainClass1.function1
Failed example:
  a.function1(1,1,1)
Expected:
  2
Got:
  3
*****
1 items had failures:
  1 of 3 in __main__.MainClass1.function1
***Test Failed*** 1 failures.
[Finished in 0.2s]
```

```
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

:Example:

>>> import testDocsPy
>>> a = testDocsPy.MainClass1()
>>> a.function1(1,1,1)
2

.. note:: can be useful to emphasize
        important feature
.. seealso:: :class:`MainClass2`
.. warning:: arg2 must be non-zero.
.. todo:: check that arg2 is non zero.
"""
return arg1/arg2 + arg3

if __name__ == "__main__":
    import doctest
    doctest.testmod()

[Finished in 0.1s]
```

**Conclusion:** As you can see Unit Testing is a core of Test Driven Development, in order to decrease bugs amount it is a good solution to write tests for each function.

## ЛАБОРАТОРНА РОБОТА №5

### Система автоматичного створення довідника користувача та оформлення коду за допомогою Coding Convention.

#### Table Of Contents

[Welcome to lab5's documentation!](#)  
[Indices and tables](#)

#### This Page

[Show Source](#)

#### Quick search

Go

Enter search terms or a module, class  
or function name.

## Welcome to lab5's documentation!

This is a project Documentation for Laboratory Work number 4 and 5 it includes tests and auto Documentation

Contents:

- [Auto Generated Documentation](#)

## Indices and tables

- [Index](#)
- [Module Index](#)
- [Search Page](#)

©2015, Dima Ovsadchy. | Powered by Sphinx 1.3.1 & Alabaster 0.7.6 | [Page source](#)

#### This Page

[Show Source](#)

#### Quick search

Go

Enter search terms or a module,  
class or function name.

## Auto Generated Documentation

This module illustrates how to write your docstring in OpenAlma and other projects related to OpenAlma.

class testDocsPy.MainClass1

This class docstring shows how to use sphinx and rst syntax

The first line is brief explanation, which may be completed with a longer one. For instance to discuss about its methods. The only method here is `function1()`'s. The main idea is to document the class and method's arguments with

- parameters, types, return and return types:

```
:param arg1: description
:param arg2: description
:type arg1: type description
:type arg2: type description
:return: return description
:rtype: the return type description
```

- and to provide sections such as `Example` using the double commas syntax:

```
:Example:
followed by a blank line !
```

which appears as follow:

Example:

followed by a blank line

- Finally special sections such as `See Also`, `Warnings`, `Notes` use the sphinx syntax (paragraph directive):

```
.. seealso:: blabla
.. warnings also:: blabla
.. note:: blabla
.. todo:: blabla
```

#### Note:

There are many other Info fields but they may be redundant:

- param, parameter, arg, argument, key, keyword: Description of a parameter.
- rpe: Type of a parameter.
- raises, raise, except, exception: That (and when) a specific exception is raised.
- var, ivar, cvar: Description of a variable.
- returns, return: Description of the return value.
- rtype: Return type.

**funcTest**(arg1, arg2)  
returns arg1 + arg2

Parameters: • arg1 (int, float,...) - the first value  
• arg2 (int, float,...) - the second value

Returns: arg1 + arg2

Return type: int, float

Example:

```
>>> import testDocsPy
>>> a = testDocsPy.MainClass1()
>>> a.funcTest(1,1)
2
```

Note:

can be useful to emphasize important feature

(ПИС) Проектування інформаційних систем

See also:

**MainClass2**

Warning:

arguments should be not string

Todo:

check that arg1 or arg2 is not strings

**function1**(arg1, arg2, arg3)  
returns (arg1 / arg2) \* arg3

Drawbacks:

- Just looking at the docstring, the parameter, type and return sections do not appear nicely

Parameters: • arg1 (int, float,...) - the first value  
• arg2 (int, float,...) - the second value  
• arg3 (int, float,...) - the third value

Returns: arg1/arg2 \*arg3

Return type: int, float

Example:

```
>>> import testDocsPy
>>> a = testDocsPy.MainClass1()
>>> a.function1(1,1,1)
2
```

Note:

can be useful to emphasize important feature

See also:

**MainClass2**

Warning:

arg2 must be non-zero.

Todo:

check that arg2 is non zero.

1) Формат документування для Python sphinx на прикладі:

2. `def search(name):`

```
    """
    Searching items by name
    :param name: String query string(key)
    :return: Flask.Response -- JSON Item objects
    """
    searched = Item.objects(name__icontains=name).all()
    return searched.to_json()
```

1. 2) Команди для авто-генерації довідника користувача:

2. `sphinx-quickstart`

3. create RST file for your program( example Code.rst )

3.1 Your file body :

Auto Generated Documentation

=====

```
.. automodule:: testDocsPy
   :members:
```

`testDocsPy` - my program name (testDocsPy.py)

4. uncomment this line - `sys.path.insert(0, os.path.abspath('.'))` in file `conf.py` and specify your program name

```
8 #
9 # Note that not all possible configuration values are present in this
10 # autogenerated file.
11 #
12 # All configuration values have a default; values that are commented out
13 # serve to show the default.
14
15 import sys
16 import os
17 import shlex
18
19 # If extensions (or modules to document with autodoc) are in another directory,
20 # add these directories to sys.path here. If the directory is relative to the
21 # documentation root, use os.path.abspath to make it absolute, like shown here
22 sys.path.insert(0, os.path.abspath('.'))
23
24 # -- General configuration -----
25
26 # If your documentation needs a minimal Sphinx version, state it here.
27 #needs_sphinx = '1.0'
28
29 # Add any Sphinx extension module names here, as strings. They can be
30 # extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
```



5. make `clean` - in terminal

6. make `html`

7. review your work in `build/html`

**Conclusion: We learned how to use auto documentation using sphinx as an example. Also we learned how to set up it and configure**

## Системи контролю версій SVN, GIT.

```
[dimao-note:StockChart dimao$ git add .
[dimao-note:StockChart dimao$ git commit -m "documentation work"
[master 46dffbd] documentation work
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lab456.pages
[dimao-note:StockChart dimao$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
 (use "git push" to publish your local commits)
nothing to commit, working directory clean
[dimao-note:StockChart dimao$ git branch
* master
  newTestBranch
[dimao-note:StockChart dimao$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.36 MiB | 683.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0)
To https://github.com/dimaosa/StockChart.git
 2656cec..46dffbd master -> master
[dimao-note:StockChart dimao$ git pull
Already up-to-date.
dimao-note:StockChart dimao$ █
```

```
[dimao-note:StockChart dimao$ git branch
* master
[dimao-note:StockChart dimao$ git branch ntb
[dimao-note:StockChart dimao$ git branch
* master
  ntb
[dimao-note:StockChart dimao$ git checkout ntb
Switched to branch 'ntb'
[dimao-note:StockChart dimao$ git branch
  master
* ntb
[dimao-note:StockChart dimao$ touch newFileInNewBranch.py
[dimao-note:StockChart dimao$ git add .
[dimao-note:StockChart dimao$ git commit -m 'core file in ntb module'
[ntb fc1b75e] core file in ntb module
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newFileInNewBranch.py
```



```

dimao-note:StockChart dimao$ git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
dimao-note:StockChart dimao$ git merge ntb
Updating 46dffbd..fc1b75e
Fast-forward
 newFileInNewBranch.py | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 newFileInNewBranch.py
dimao-note:StockChart dimao$ git push
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/dimaosa/StockChart.git
 46dffbd..fc1b75e master -> master
dimao-note:StockChart dimao$ █

```



dimaosa / **StockChart**

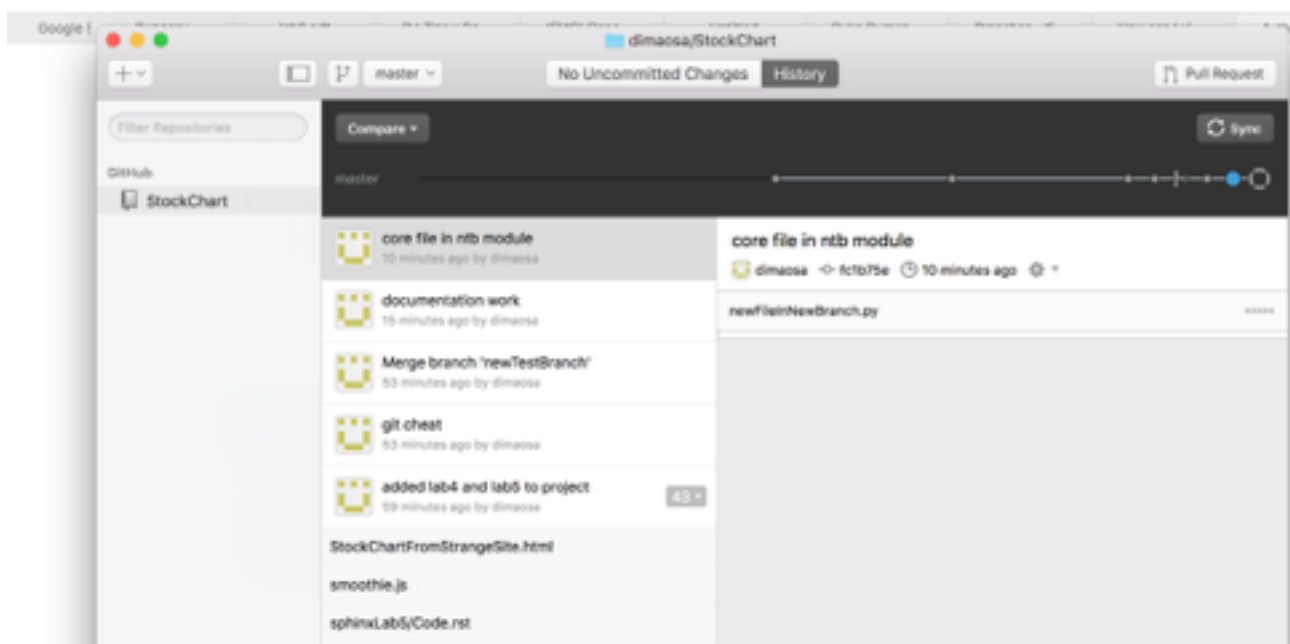
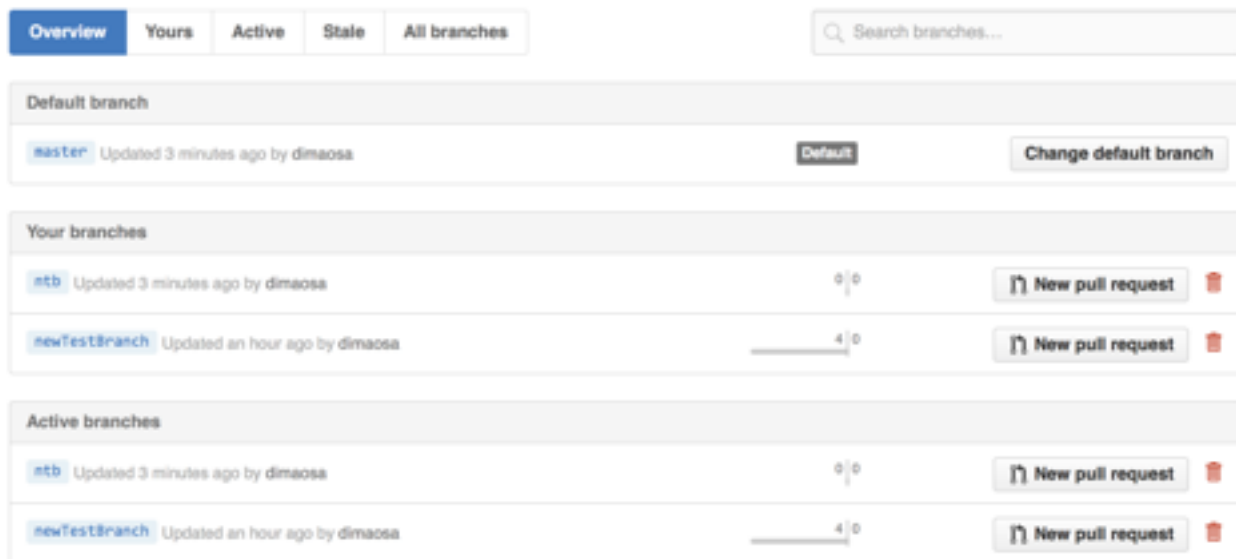
Unwatch 1

The application will provide Charts for different stocks. Chart will depends on user choice. — Edit

8 commits
3 branches
0 releases
1 contributor

Branch: master StockChart / +

dimaosa core file in ntb module		Latest commit fc1b75e 3 minutes ago
sphinxLab5	added lab4 and lab5 to project	an hour ago
README.md	Initial commit	a month ago
	added lab4 and lab5 to project	an hour ago
fileForNewBranch	created new branch added useless file	an hour ago
github-git-cheat-sheet.pdf	git cheat	an hour ago
lab1PIS.docx	add 1 lab	a month ago
lab456.pages	documentation work	8 minutes ago
newFileInNewBranch.py	core file in ntb module	3 minutes ago
smoothie.js	added lab4 and lab5 to project	an hour ago



**Conclusion: This lab work gave me the basic knowledges how to use version control in a project.**