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 робота №4,5,6

Виконав студент

групи ДА-21

Осадчий Дмитро

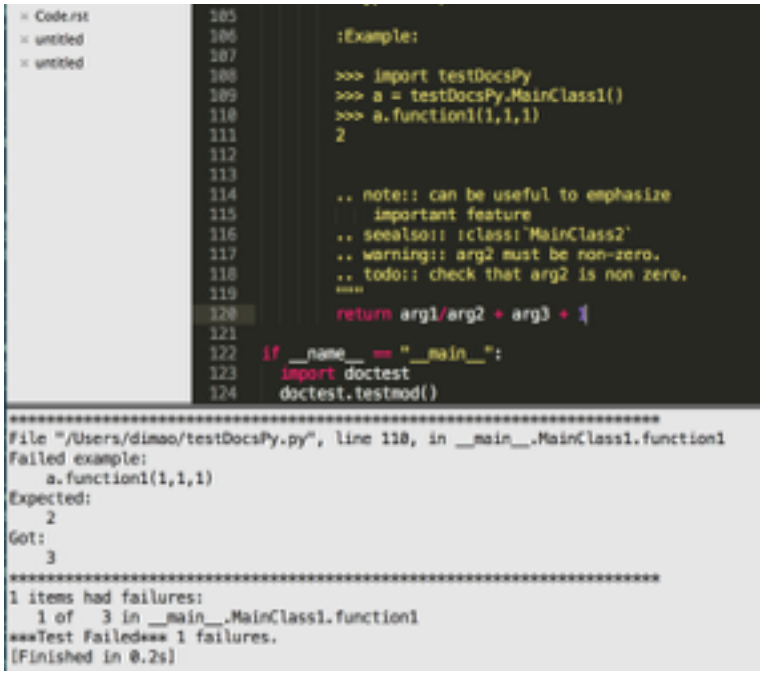
Київ 2015

ЛАБОРАТОРНА РОБОТА No4

Модульне тестуваня (Unit-тести) та рефакторинг.

Мета роботи: оволодіти навичками створення програмного забезпечення за метолологією

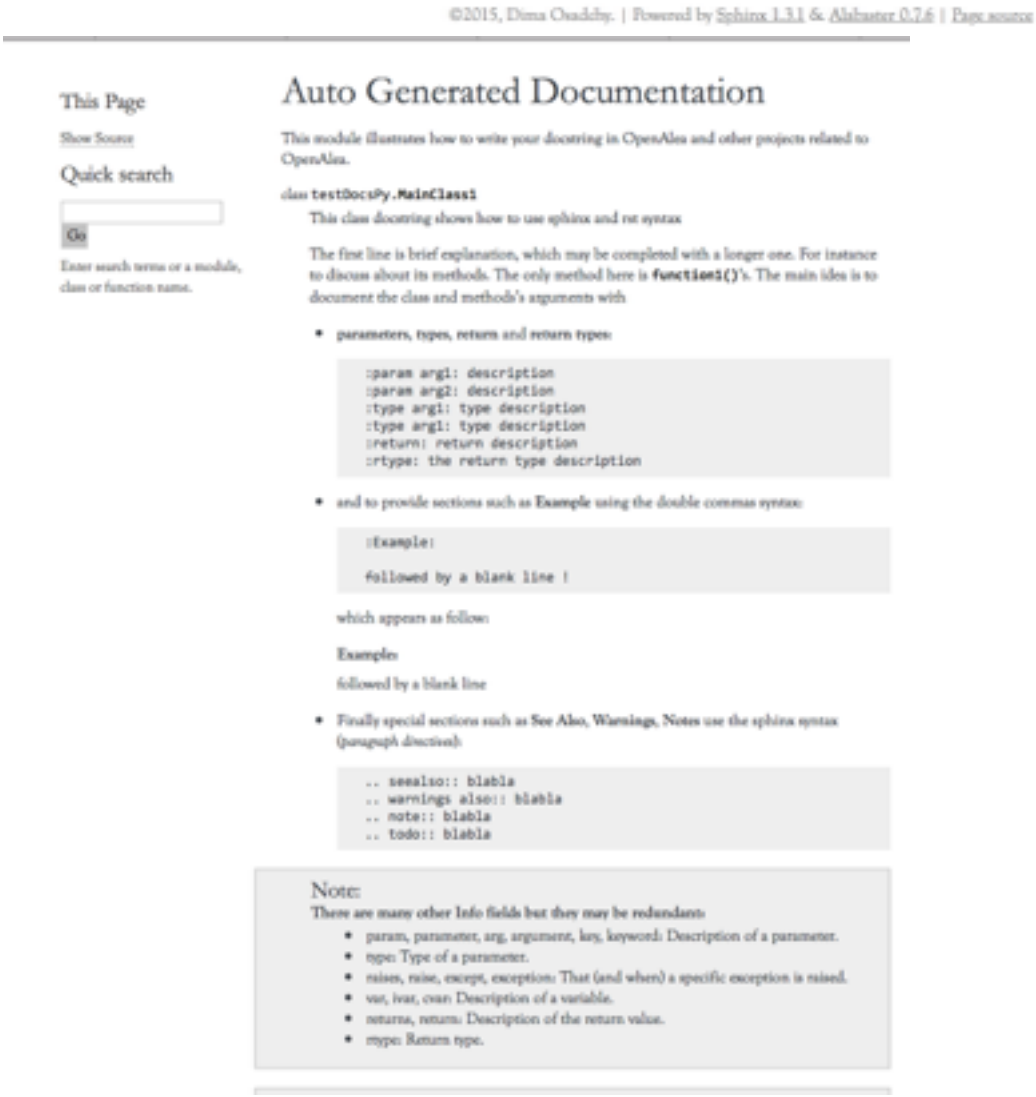
TDD та ознайомитися з процедурами рефакторинга.



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 No5

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



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

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)

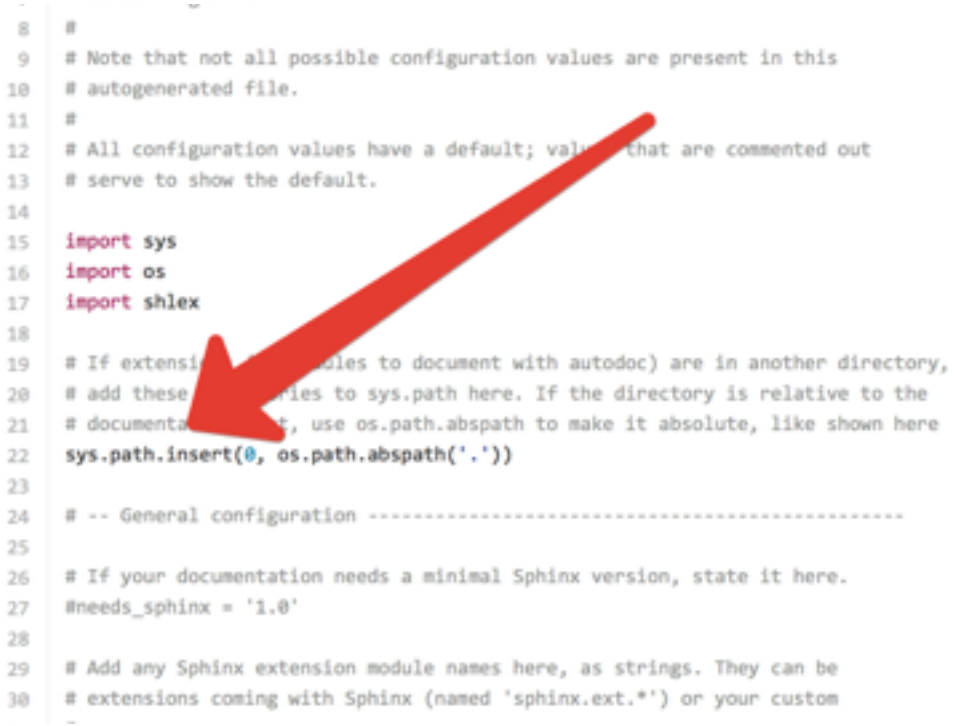
3.1 Your file body :

```
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



```
#
# Note that not all possible configuration values are present in this
# autogenerated file.
#
# All configuration values have a default; values that are commented out
# serve to show the default.

import sys
import os
import shlex

# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here
sys.path.insert(0, os.path.abspath('.'))

# -- General configuration ------------------------------------------------

# If your documentation needs a minimal Sphinx version, state it here.
#needs_sphinx = '1.0'

# Add any Sphinx extension module names here, as strings. They can be
# 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

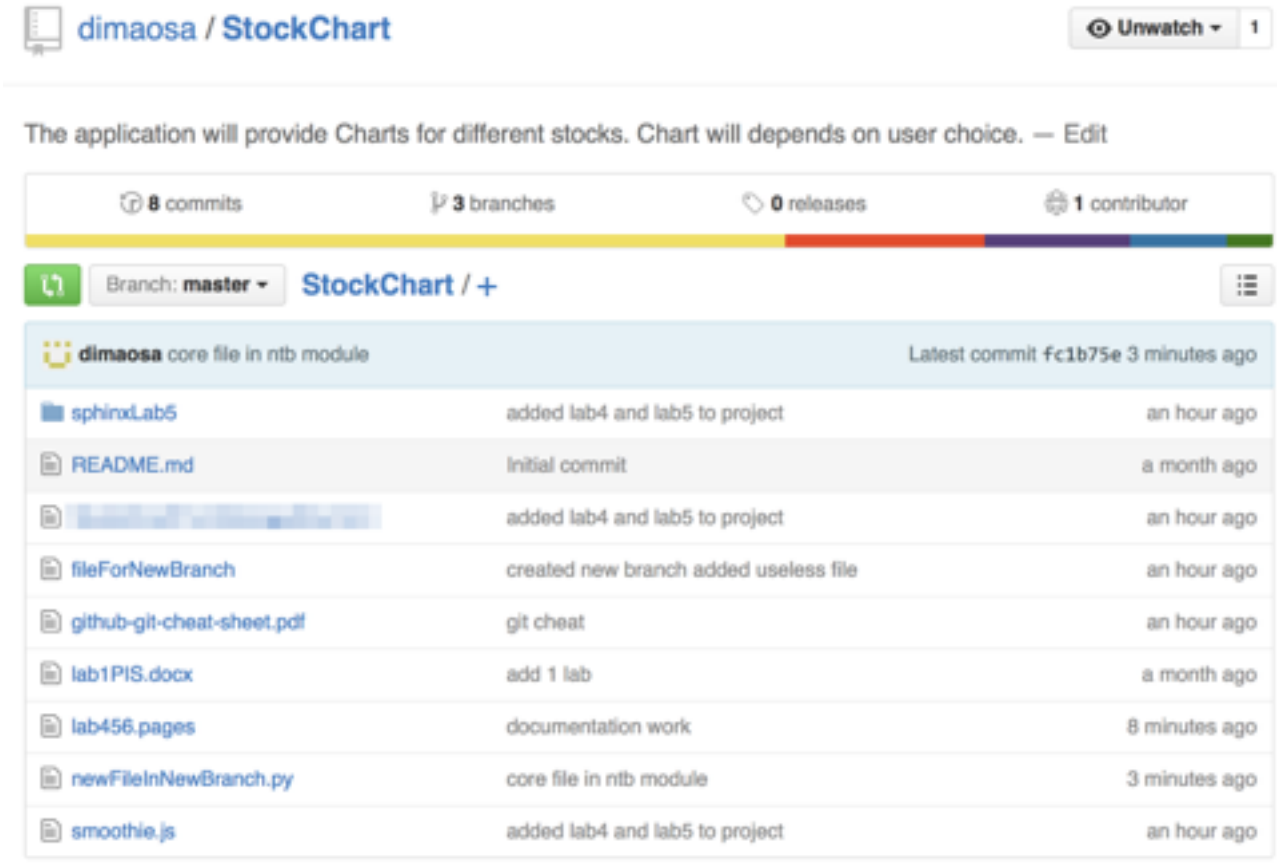
ЛАБОРАТОРНА РОБОТА No6

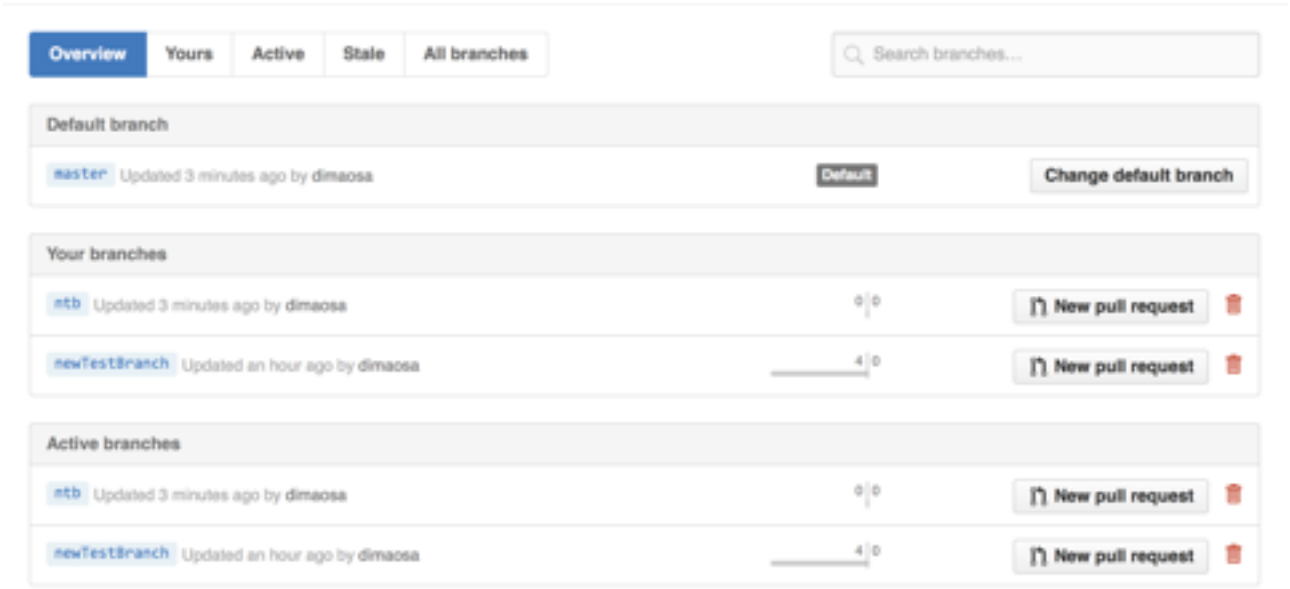
Системи контролю версій SVN, GIT.

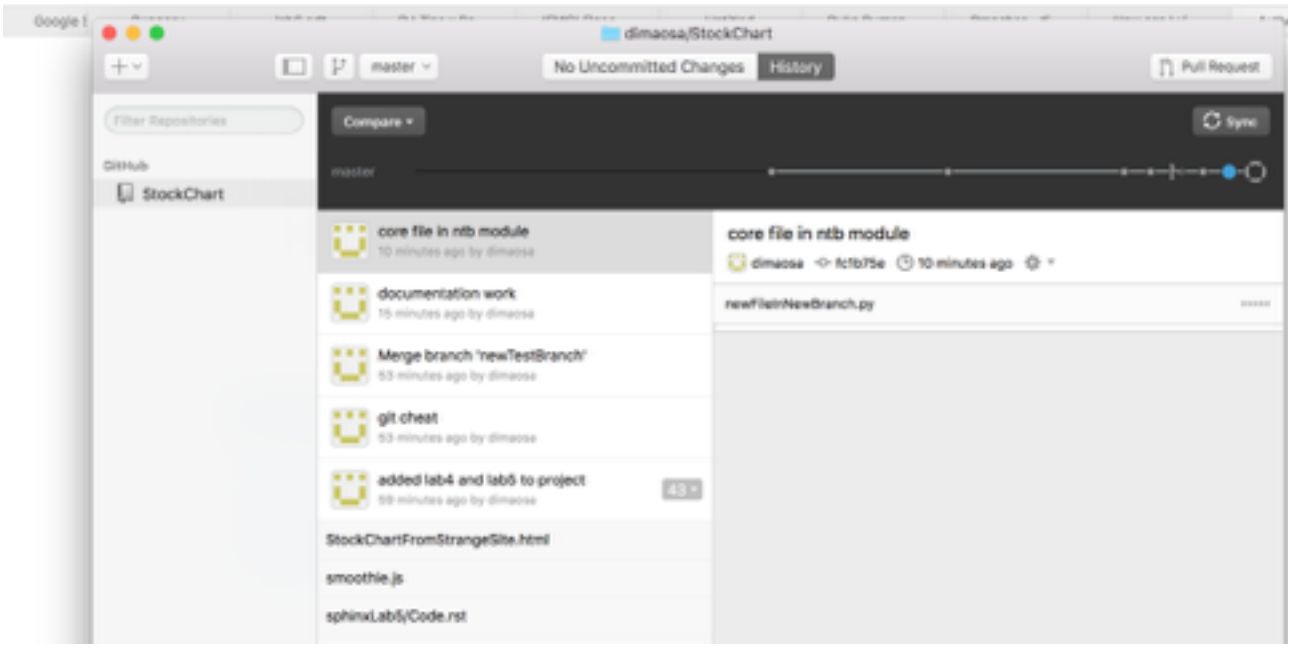
```
dimao-note:StockChart dimao$ git add .
dimao-note:StockChart dimao$ git commit -m "documentation work"
[master 46dffbd] documentation work
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 lab456.pages
dimao-note:StockChart dimao$ git status
On branch master
Your branch is ahead 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$
```







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