

## **OMC codes – part 2**

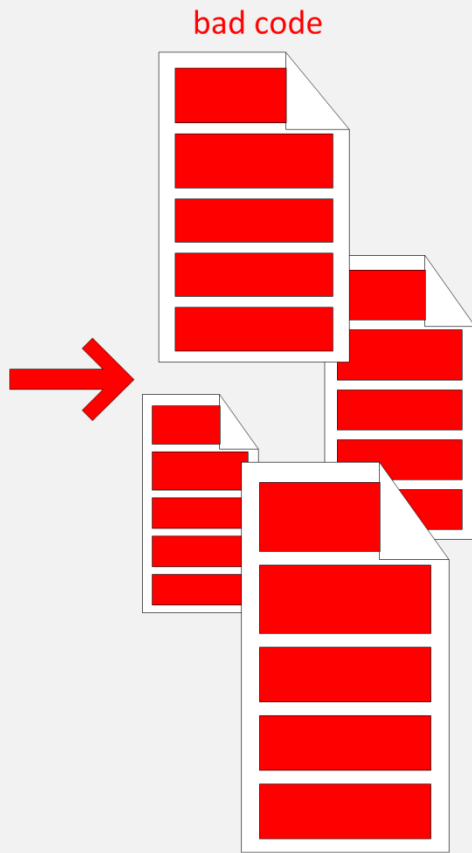
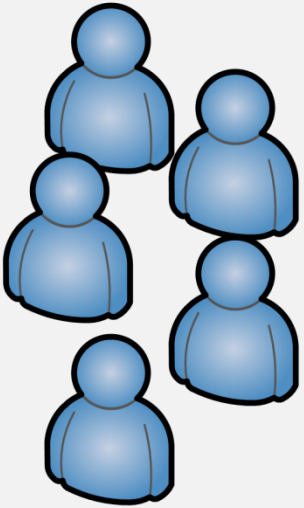
Viktor Maier

[viktor.maier@cern.ch](mailto:viktor.maier@cern.ch)

# Table of content

- Why do we have bad code?
- We want good code!
- We do
  - Version control system
  - Better coding
  - Testing

physicists





```

if "LHC" in accel:
    tfs_file = files_dict['getIP.out']
    tfs_file.add_column_names(["NAME", "BETASTARH", "BETASTARHMDL", "H", "PHIH", "PHIXH", "PHIHMDL", "BETASTARV", "BETASTARVMDL", "V", "PHIV", "PHIVV", "PHIVMDL"])
    tfs_file.add_column_datatypes(["%s", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le"])
    ips = ["1", "2", "3", "4", "5", "6", "7", "8"]
    try:

```

```
def function(x):
```

```
    fterm=4*abs(x[0])
```

```
    main=2*sinh(fterm)
```

```
    single=sinh(fterm)
```

```
    first=cosh(fterm)*sin(x[1])
```

```
    second=cosh(fterm)*sin(x[1]+2*phi12_nlinear)
```

```
    fun1=float((main*(single+first))-bb[0])
```

```
    fun2=float((main*(single+second))-bb[1])
```

```
    fun=[fun1, fun2]
```

```
    return fun
```

```

tfs_file_y.add_column_names(["NAME", "BETY", "BETYSTD", "BETYMDL", "ALFY", "ALFYSTD", "ALFYMDL", "BETY*", "BETY*STD", "BETY*MDL", "SY*", "SY*STD", "SY*MDL", "rt(2JYD)", "rt(2JYD)STD"])
tfs_file_y.add_column_datatypes(["%s", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le", "%le"])
for bpm_name in 'IP1', 'IP5', 'IP8', 'IP2':
    try:

```

```

        list_row_entries = ['*' + bpm_name + '*']
        for k in IPxf[bpm_name]:
            list_row_entries.append(k)

```

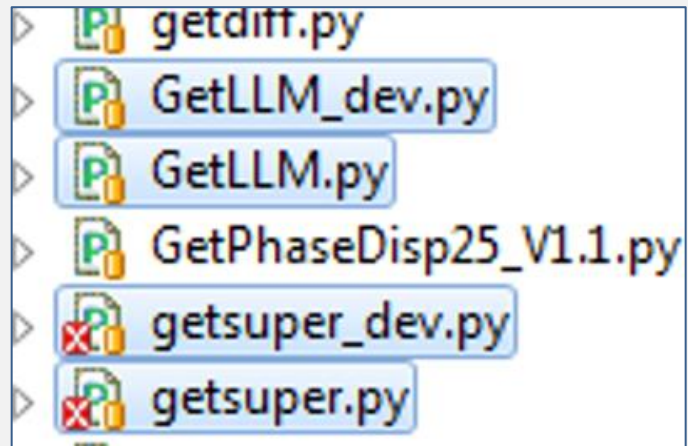
```

        tfs_file_x.add_table_row(list_row_entries)

```

## GetLLM.py

- 5800 LOC,
- 2000 main()
- 55 functions
- PyLint:
  - 128 errors
  - 959 warnings
  - 3822 infos

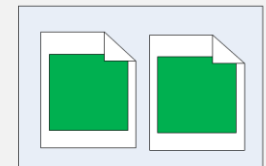
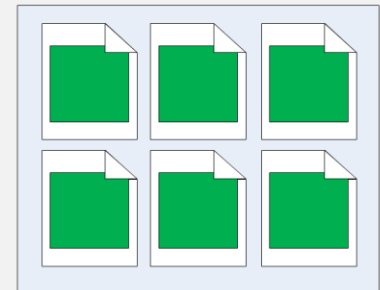
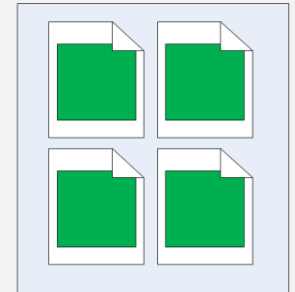


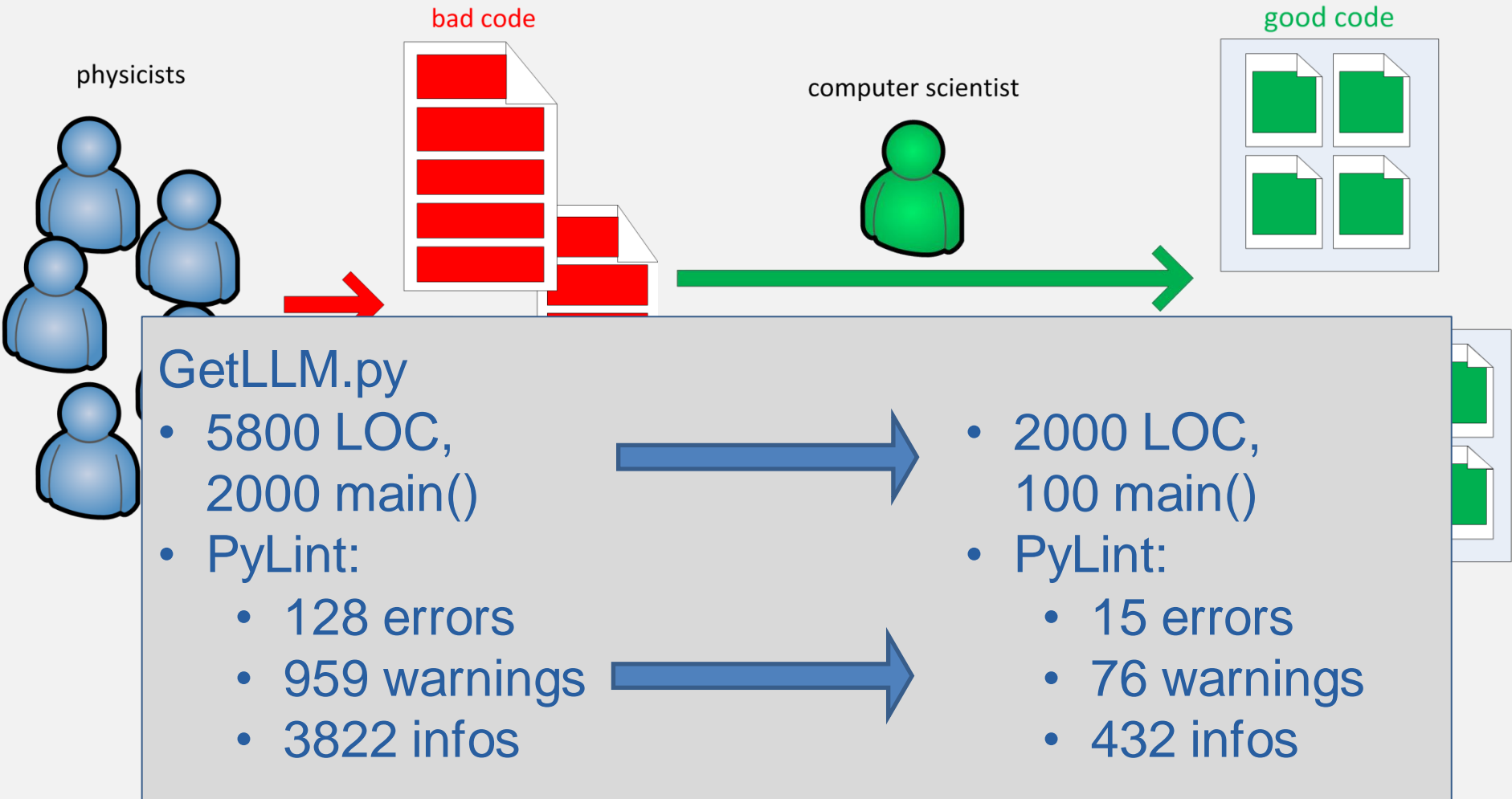
```
else:  
    if file.split('.')[-2][-4:] == '_dev':  
        import GetLLM_dev as GetLLM  
    else:  
        import GetLLM
```

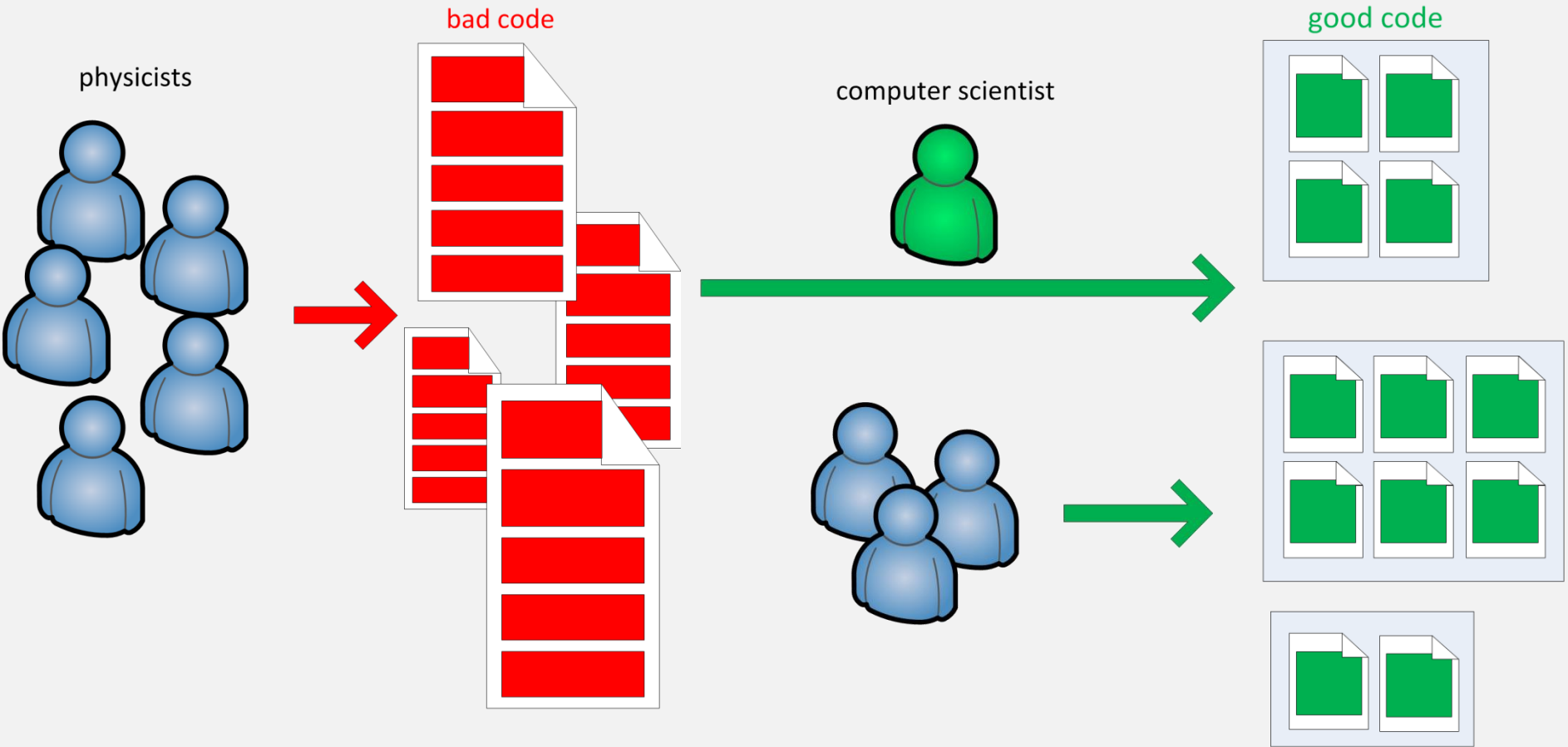
➔ restricts research and development

- Clean code
  - Easy to read, to understand, to extend...
- Testsuites and automatic tests
  - Is the code still correct?
- Reliable and stable software
  - No crashes; efficient usage...

good code







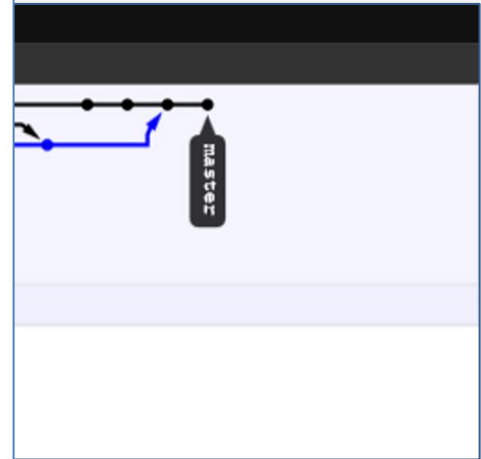
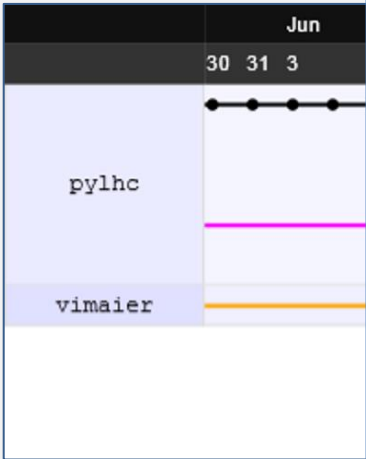
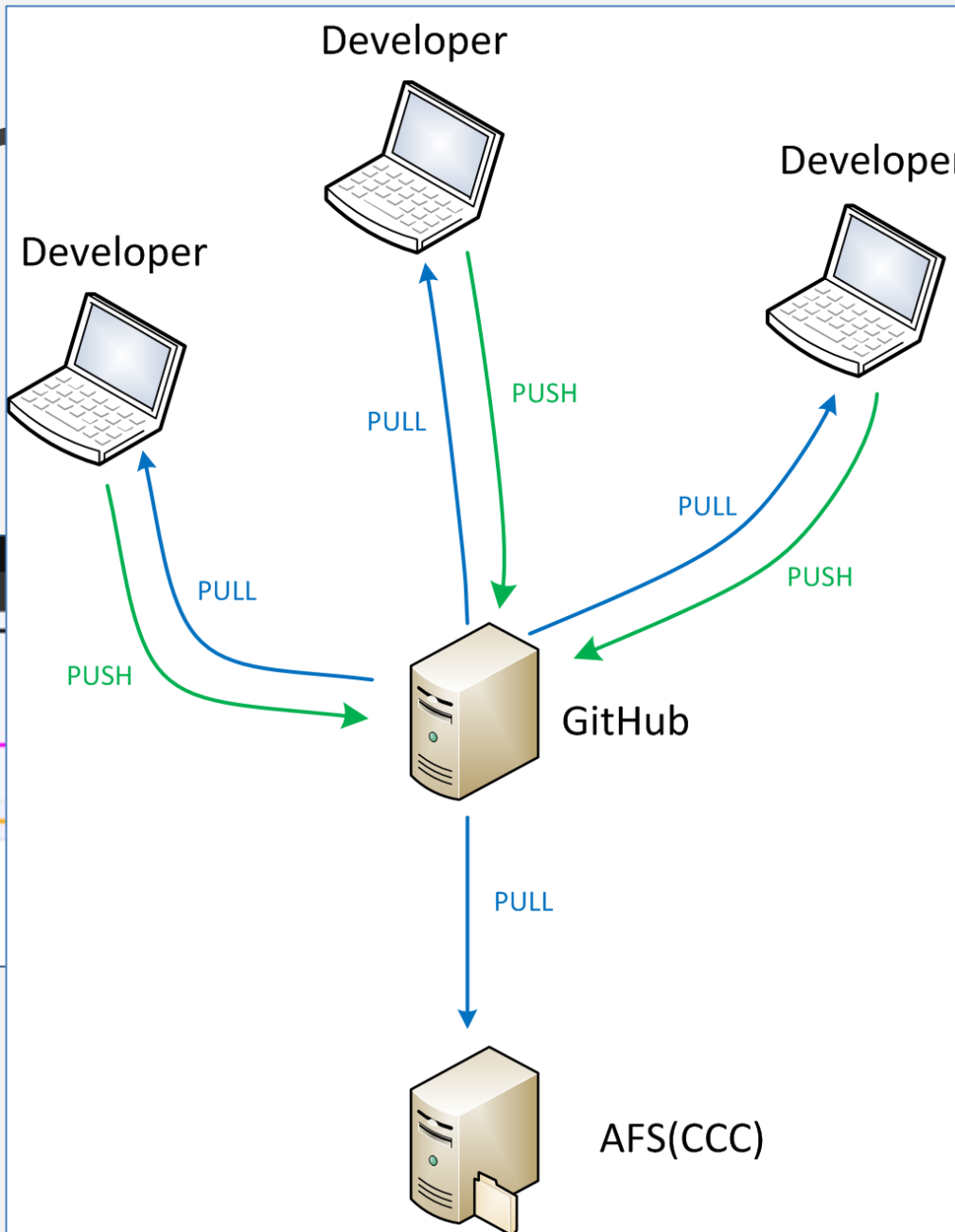


# Version

# System

- Git

- Github



# Version control system

- Git

- IssueTracker

- Github

The screenshot displays a GitHub Issues page with the following components:

- Filters:** 11 Open, 13 Closed, Sort: Newest.
- Issue List:**
  - BPM ignoring in Drive** (TODO) #24, opened by vimaier a day ago.
  - Invocation GetLLM.GetPhases for SPS and RHIC wrong** (bug) #23, opened by vimaier 4 days ago.
  - can we use argparse?** (enhancement, question) #22, opened by Eothred 5 days ago, 9 comments.
  - GetLLM: BPMs next to Ac dipole** (question) #20, opened by vimaier 8 days ago, 2 comments.
  - Wrong source file names in "getCOy\_dpp\_'+str(k+1)+'.out"?** (bug, question) #18, opened by vimaier 12 days ago, 3 comments.
  - Wrong column names in "getphasey\_dpp\_'+str(k+1)+'.out"** (bug, question) #17, opened by vimaier 12 days ago, 3 comments.
  - Renaming SegmentBySegment\_0.33.py ?** (question) #14, opened by vimaier 19 days ago, 7 comments.
- Left Sidebar:**
  - Everyone's Issues: 11
  - Assigned to you: 0
  - Created by you: 9
  - Mentioning you: 2
  - No milestone selected (gear icon)
  - Labels:**
    - TODO: 4
    - bug: 4
    - enhancement: 1
    - question: 5
    - Info: 0
    - duplicate: 0
    - invalid: 0
    - wontfix: 0

# Better coding

## Eclipse IDE

- Static analysis
- Debugger
- Auto completion
- (Quick) Outline
- Navigation
- TODO's
- Highlighting occurrences
- ...

The screenshot displays the Eclipse IDE environment. The main editor shows Python code with syntax highlighting. The Package Explorer on the left shows a project structure with various sub-packages. The Tasks/Outline view on the right shows a hierarchical view of the code structure. The Console/Problems view at the bottom shows a list of errors and warnings.

Description	Resource	Path	Location	Type
<b>2 errors, 16 warnings, 0 others</b>				
<b>Errors (2 items)</b>				
Undefined variable: dictionary	GetLLM.py	/Beta-Beat.src/GetLLM	line 404	PyDev Problem
ID:E0602 Undefined variable 'i_will_be_exported'	test2.py	/externalProjects/test	line 11	PyLint Problem
<b>Warnings (16 items)</b>				
Unused variable: bpmsx	GetLLM.py	/Beta-Beat.src/GetLLM	line 1658	PyDev Problem
Unused variable: bpmsx	GetLLM.py	/Beta-Beat.src/GetLLM	line 1890	PyDev Problem
Unused variable: bpmsy	GetLLM.py	/Beta-Beat.src/GetLLM	line 1659	PyDev Problem
Unused variable: bpmsy	GetLLM.py	/Beta-Beat.src/GetLLM	line 1892	PyDev Problem
Unused variable: invx	GetLLM.py	/Beta-Beat.src/GetLLM	line 1790	PyDev Problem
Unused variable: invly	GetLLM.py	/Beta-Beat.src/GetLLM	line 1852	PyDev Problem
Unused variable: MUX	GetLLM.py	/Beta-Beat.src/GetLLM	line 1658	PyDev Problem
Unused variable: MUX	GetLLM.py	/Beta-Beat.src/GetLLM	line 1755	PyDev Problem
Unused variable: MUX	GetLLM.py	/Beta-Beat.src/GetLLM	line 1890	PyDev Problem
Unused variable: MUY	GetLLM.py	/Beta-Beat.src/GetLLM	line 1659	PyDev Problem
Unused variable: MUY	GetLLM.py	/Beta-Beat.src/GetLLM	line 1818	PyDev Problem

# Better coding

## HowTo: Eclipse with PyDev and EGit

[Home](#)

### Table of content

1. [Install Eclipse](#)
2. [Install PyDev](#)
3. [Configure Editor](#)
4. [Install EGit](#)
5. [Abstract workflow with Git\(Hub\)](#)
6. [Clone Git repositories](#)
7. [Git: Useful links and sheets](#)
8. [Working with Git](#)
9. [Hotkeys](#)
10. [Advantages of Eclipse](#)
11. [Python Style Guide](#)

- Comming and going
- Quick Getting started

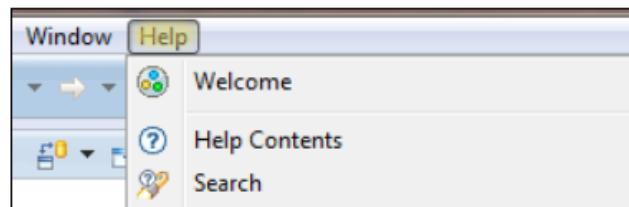
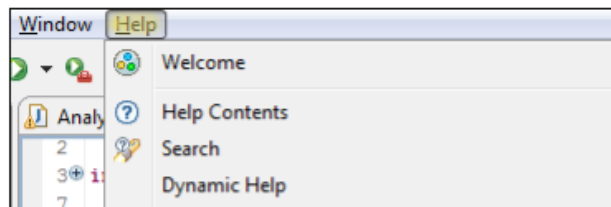
### Install PyDev

[PyDev](#) is a plugin for Eclipse which makes the Python development in the popular Eclipse IDE possible.

It is possible to install PyDev directly from Eclipse.

Go to "Help > [Install from Catalog...|or|Eclipse Marketplace...]"

If you are using not BE-CO Eclipse and there is no "Eclipse Marketplace" in the menu then you have to [install](#) it.



# Better coding

## Python Style Guide

By using this style guide you can improve the software quality of your code.

Every developer has his own coding style and is therefore used to read his style. Another style of the same code would probably consume more time to analyze. To avoid such problems and to make your code more readable you can use a set of common rules(style guide).

### Naming Conventions

- **module, function, method, attribute and local variable names**  
Lowercase and words divided by an underscore.  
Examples: *segment\_by\_segment.py, io\_utils.py, get\_phase.py*
- **class names**  
CamelCase  
Examples: *FileWriter, Point, MouseClickEventHandler*
- **module constants**  
Uppercase and words divided by an underscore.  
Examples: *ERROR\_MESSAGE, PI, DEBUG*

### Docstrings

There are several styles of docstrings. Use the following style to document your functions:

```
def func_x(param)
"""
<Function description>

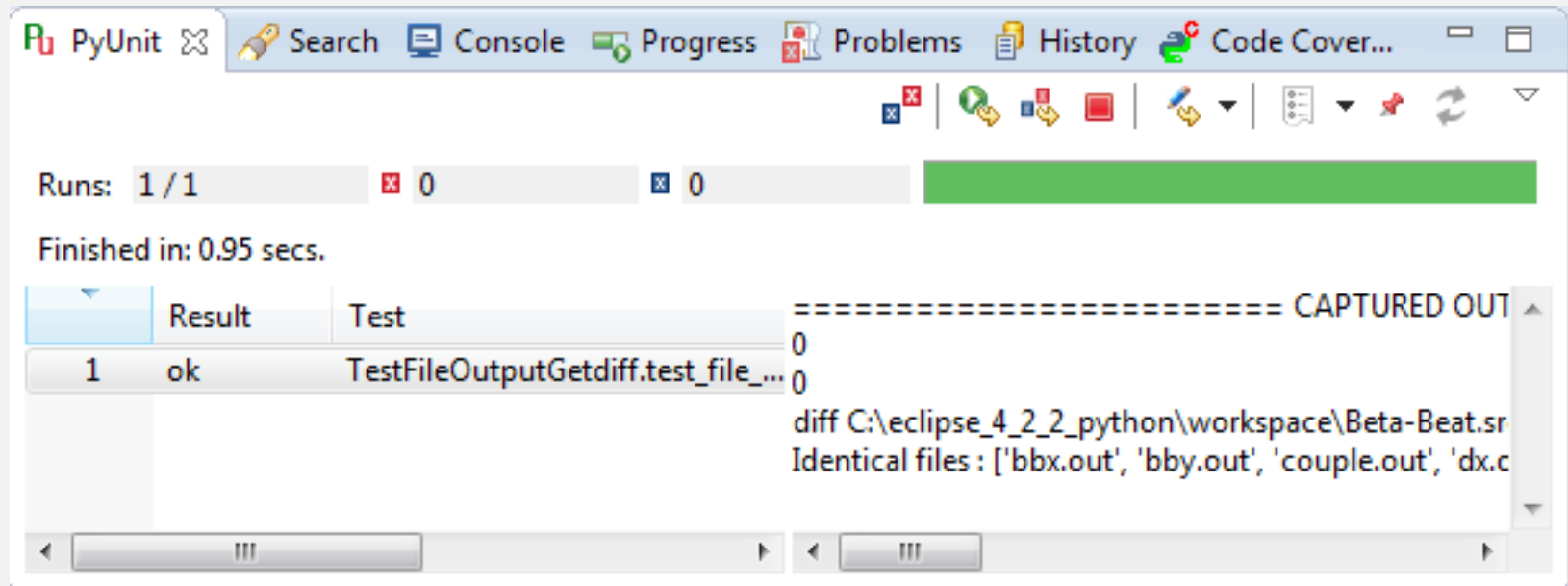
:Parameters:
  '<param_name>': <datatype>
  <description>

:Return: <datatype>
  <description>
"""
...
```

- Agreement of programmers
- Easier to read ,foreign' code

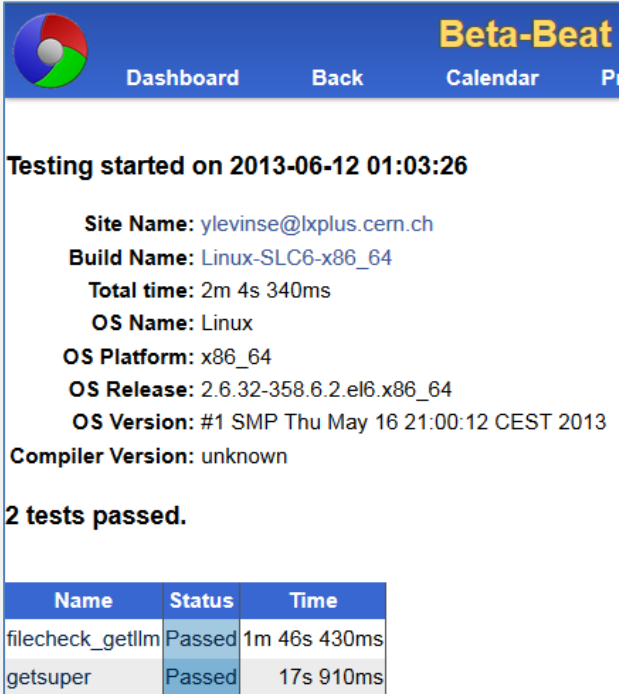
# Testing

- PyUnit testing framework



# Testing

- PyUnit testing framework
- Nightly tests(CDash)



**Beta-Beat**

Dashboard Back Calendar Pr

Testing started on 2013-06-12 01:03:26

Site Name: ylevinse@lxplus.cern.ch  
Build Name: Linux-SLC6-x86\_64  
Total time: 2m 4s 340ms  
OS Name: Linux  
OS Platform: x86\_64  
OS Release: 2.6.32-358.6.2.el6.x86\_64  
OS Version: #1 SMP Thu May 16 21:00:12 CEST 2013  
Compiler Version: unknown

**2 tests passed.**

Name	Status	Time
filecheck_getllm	Passed	1m 46s 430ms
getsuper	Passed	17s 910ms



<http://www.cdash.org/cdash/img/software-process.png>

# Testing

- PyUnit testing framework
- Nightly tests(CDash)
- Tests at CCC



[http://public.web.cern.ch/public/Objects/Spotlight/0603037\\_03.jpg](http://public.web.cern.ch/public/Objects/Spotlight/0603037_03.jpg)



# Questions?

