acrolinx IQ Prüfbericht

Einstellungen und Übersicht der Ergebnisse ▼anzeigen | Ausblenden

	Status	Ergebnisse	Fehlerquote
Rechtschreibung	✓	37	321
Grammatik	✓	6	52
Stil	✓	38	329
Terminologie	✓	0	0
Wiederverwendung	✗		
Neue Terme	✓	135	

Administrative Informationen ▼anzeigen | Ausblenden

Benutzername	admin
Lizenz-ID	acrolinx Internal / admin
Lizenz-Edition	Enterprise
Versionsinformation	Plug-in : 2.2.0 (build : 1000) / Server : 2.2 (build : 2704)
Ressourcen	QA : 2.2 (build: 1016 / 2010-12-14)
Dateiname	file://C:/Dokumente und Einstellungen/melanie/Eigene Dateien/Lehre/Coli Saarbrücken/Student Texts/Magik/Magik.pdf
Dateiformat	application/pdf
processing_settings_extractor_type	PDF

Prüfungsinformationen ▼anzeigen | Ausblenden

Zeitpunkt der Prüfung	Tue Dec 14 16:41:47 CET 2010
Dokumentsprache	English
Regelset	Standard_US
Verwendete Termsets	acrolinx
Prüfumfang	1154 Wort/Wörter / 98 Satz/Sätze
Prüfstatus	● (Fehlerquote: 702; Grün < 100, Gelb 100 - 199, Rot ≥ 200)
Flesch Reading Ease	56.5

Rechtschreibung ▼anzeigen | Ausblenden

Wort	Anzahl	Vorschlag	Kontext
Magik	7	Magi Magic Magis Malik	CHAPTER ONE DEVELOPERS' GUIDE 1.1 Starting development This guide will serve as a basic introduction to installing **Magik** for development purposes. You must have at least Python 2.6 to run **Magik**. Bootstrapping creates needed directories and generates the buildout script, which downloads all the **Magik** dependencies and places command-line scripts to the bin/ directory. 1.2 Developing **Magik** 1.2.1 Running tests py.test is a testing toolkit that is used in Magik to run unit tests. 1.2 Developing Magik 1.2.1 Running tests py.test is a testing toolkit that is used in **Magik** to run unit tests. 1.2.2 Making changes Coding standarts In general, **Magik** is written with PEP 8 in mind. Developing **Magik** 3
magik	6	magi magic magis	./magik/ -- bootstrap.py -- buildout.cfg -- conftest.py -- doc -- **magik** 1 http://mercurial.selenic.com/wiki/QuickStart#Setting_a_username 1 bin/py.test **magik** 2 Chapter 1. A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit will run unit tests of the **magik** module and doctests in the doc/ directory. bin/py.test **magik** # Run unit tests bin/py.test doc # Run doctests bin/py.test magik/statement/_tests/test_select_parser.py # Run tests defined in

			magik/statement/_tests/test_select_parser.py Note: bin/py.test **magik** --pastebin=all There is another test runner: bin/py.test --cover=**magik** --cover-report=html File coverage/index.html contains the coverage report in the HTML format.
cread	1	bread Bread creak cream creed dread Dread read tread Tread	• setup.cfg and setup.py are files required to **cread** a Python egg.
Buildout	1	Wortwiederholung	1.1.3 Buildout **Buildout** Buildout is a Python-based build system for creating, assembling and deploying applications.
interpretator	1	interpretation	python2.6 bootstrap.py # Use Python 2.6 /usr/bin/python bootstrap.py # Explicitly define python **interpretator**.
bpython	1	python	• bin/buildout • bin/bpython **bpython** is a fancy interface to the Python interpreter.
them	1	Wortwiederholung	To execute all of them **them** run:
doctests	2	contests Contests detests dockets doclets octets	A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit will run unit tests of the magik module and **doctests** in the doc/ directory. bin/py.test magik # Run unit tests bin/py.test doc # Run **doctests** bin/py.test magik/statement/_tests/test_select_parser.py # Run tests defined in magik/statement/_tests/test_select_parser.py Note:
fiels	1	feels fiefs field fields files fills fuels Niels	You can specify which test to run by providing a directory or a file path which contains test **fiels** to py.test.
pastebin	2	wastebin Wastebin	py.test can submit its output to a **pastebin**, a hosting for text snippets. bin/py.test magik --**pastebin**=all There is another test runner:
pyflakes	1	flakes	It additionally to unit and doc tests performs the PEP 8 check and static analysis by **pyflakes**.
standarts	1	standards	1.2.2 Making changes Coding **standarts** In general, Magik is written with PEP 8 in mind.
Todo	2	Bodo Dodo Todd To-do Togo Too Toyo	**Todo** pep8 Actually, there still some work needs to be done to respect pep8 more. **Todo** codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the programmer.
ment	1	bent Bent cent dent gent lent meant meat meet melt men mend menu met mint pent rent sent tent vent went	Todo codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was **ment** by the programmer.

hg	5	Hg	**hg** diff shows which content of files was removed, added or changed. **hg** push sends changes sets (commits) to other repository. **hg** pull gets them from other repository. **hg** in shows commits that you are missing, hg out shows commits that other repository is missing. **hg** up allows to switch to desired commit.
diff	2	duff miff Miff tiff Tiff	hg **diff** shows which content of files was removed, added or changed. [extensions] color = hg st and hg **diff** are very handy in finding out what have been changed, what files are missing (were not added to the repository).
hgrc	1		**hgrc** line color =.
bugfix	1		it can be either a **bugfix**, a formatting fix, a new feature, a new unit test, or anything else, what is important that it is one item, not several.

Grammatik ▼anzeigen | Ausblenden

Zusammenfassung

noun_adjective_confusion	1
than_then_confusion	1
subject_verb_agreement	2
your_confusion	1
use_verb_with_object_and_to	1

Details
Nach Auftreten sortieren

noun_adjective_confusion

*• bin/py.test py.test is a **command line** tool to collect, run and report about automated tests.* → *• bin/py.test py.test is a **command-line** tool to collect, run and report about automated tests.*

than_then_confusion

*• Lines longer **then** 79 characters are fine.* → *• Lines longer **than** 79 characters are fine.*

subject_verb_agreement

***Test are** located in the _tests directory of the module.*

*Make sure that all **test are** passed before making a commit.*

your_confusion

*To enable it ad to the extensions sections of **your** .* → *To enable it ad to the extensions sections of **you are** .*
*To enable it ad to the extensions sections of **you're** .*

use_verb_with_object_and_to

*hg up **allows to switch** to desired commit.*

Stil ▼anzeigen | Ausblenden

Zusammenfassung

avoid_future_tense	3
use_comma_after_introductory_phrase	1
avoid_need_to	4
use_imperative	5
use_this_that_these_those_with_noun	1
avoid_contractions	1
sentence_too_long	6
avoid_user	1

use_serial_comma	3
avoid_modal_verbs	4
spell_out_numerals	1
avoid_possessives	2
avoid_passive	1
avoid_needless_word	2
missing_space	1
use_comma_with_parenthetical_expressions	1
unnecessary_space	1

Details
Nach Auftreten sortieren

avoid_future_tense
*CHAPTER ONE DEVELOPERS' GUIDE 1.1 Starting development This guide **will serve** as a basic introduction to installing Magik for development purposes.*

*hg clone https://magik.inf.unibz.it/hg/magik/ The command **will copy** the source code from the server to your local computer to folder ./magik with the following structure:*

*A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit **will run** unit tests of the magik module and doctests in the doc/ directory.*

use_comma_after_introductory_phrase
***Before starting the development process you** need to have Python installed on your system.*

avoid_need_to
*Before starting the development process you **need to** have Python installed on your system.*

*1.1.2 Getting the source code To get sources you **need to** clone the main repository:*

*First of all we **need to** "cd" to the folder with repository, bootstrap the project, and buildout it:*

*Having atomic commits is important in teams, where team members **need to** track how the source code changes.*

use_imperative
*1.1.1 Setting up Mercurial To use Mercurial **you need to** set up your username.*

*Todo pep8 Actually, there still some work **needs to be done** to respect pep8 more.*

*Committing changes Once a feature is implemented, code changes **have to be committed**.*

*If you have created new files, they **needed to be added** to the version control.*

*To share your changes wit the world, **you need to** push your changes either to the main repository, or to any other repository.*

use_this_that_these_those_with_noun
***This** is often meaningless.*

avoid_contractions
***It's** best to configure a proper email address in ~/.hgrc (or on a Windows system in %USERPROFILE%\Mercurial.ini) by adding lines such as the following 1:*

sentence_too_long

It's best to configure a proper email address in ~/.hgrc (or on a Windows system in %USERPROFILE%\Mercurial.ini) by adding lines such as the following 1:

A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit will run unit tests of the magik module and doctests in the doc/ directory.

Todo codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the programmer.

A Tool for Managing Data Completeness, Release 0.2.2dev As a rule of thumb, try not to run the code manually to see if the feature you have written works, but write tests that implement such a run.

[extensions] color = hg st and hg diff are very handy in finding out what have been changed, what files are missing (were not added to the repository).

it can be either a bugfix, a formatting fix, a new feature, a new unit test, or anything else, what is important that it is one item, not several.

- avoid_user

The Definitive Guide or MoinMoin Mercurial guide The Hg tip has handy tips for Mercurial users.

- use_serial_comma

1.1.3 Buildout Buildout Buildout is a Python-based build system for creating, assembling and deploying applications.	→	*1.1.3 Buildout Buildout Buildout is a Python-based build system for creating, assembling, and deploying applications.*
• bin/py.test py.test is a command line tool to collect, run and report about automated tests.	→	*• bin/py.test py.test is a command line tool to collect, run, and report about automated tests.*
hg diff shows which content of files was removed, added or changed.	→	*hg diff shows which content of files was removed, added, or changed.*

- avoid_modal_verbs

To specify which Python should be used, run the bootstrap script with it.

In other words, after running the tests, you should pretty sure that the code works well.

Inside the test module (or directory) an __init__.py file should be located, and files that contains test classes named test_*.py.

Test class names should start with Test and test methods with test_.

- spell_out_numerals

bin/py.test magik 2 Chapter 1.	→	*bin/py.test magik two Chapter 1.*

- avoid_possessives

Developers' guide

Developers' guide

- avoid_passive

Todo codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the programmer.

- avoid_needless_word

In other words, after running the tests, you should pretty sure that the code works well.	→	*In other words, after running the tests, you should sure that the code works well.*

*[extensions] color = hg st and hg diff are **very** handy in finding out what have been changed, what files are missing (were not added to the repository).*	→	*[extensions] color = hg st and hg diff are handy in finding out what have been changed, what files are missing (were not added to the repository).*

missing_space

*Test class names should start with Test and test methods with **test_**.*	→	*Test class names should start with Test and test methods with **test _**.*

use_comma_with_parenthetical_expressions

*what files were changed, removed, deleted, **added and so on**.*	→	*what files were changed, removed, deleted, **added, and so on**.*

unnecessary_space

*To enable it ad to the extensions sections of **your .***	→	*To enable it ad to the extensions sections of **your.***

Terminologie ▼anzeigen | Ausblenden

Keine Einträge gefunden

Neue Terme ▼anzeigen | Ausblenden

(http://127.0.0.1:8031/output/TH/en/Magik_pdf_admin_07514746aa401bf5_23...)

Termkandidat	Kontext
atomic commits	Having **atomic commits** is important in teams, where team members need to track how the source code changes.
automated test	• bin/py.test py.test is a command line tool to collect, run and report about **automated tests**.
automatic project deployment	It is used for **automatic project deployment** and dependency installation in an isolated environment.
bin/bpython	• bin/buildout • **bin/bpython** bpython is a fancy interface to the Python interpreter.
bin/buildout	cd magik/ python bootstrap.py **bin/buildout** Note: • **bin/buildout** • bin/bpython bpython is a fancy interface to the Python interpreter.
bootstrap script	To specify which Python should be used, run the **bootstrap script** with it.
Bootstrapping	**Bootstrapping** creates needed directories and generates the buildout script, which downloads all the Magik dependencies and places command-line scripts to the bin/ directory.
bpython	• bin/buildout • bin/bpython **bpython** is a fancy interface to the Python interpreter.
bugfix	it can be either a **bugfix**, a formatting fix, a new feature, a new unit test, or anything else, what is important that it is one item, not several.
buildout	First of all we need to “cd” to the folder with repository, bootstrap the project, and **buildout** it: You can find more information about **buildout** at Buildout documentation or A brief introduction to Buildout.
Buildout	A Tool for Managing Data Completeness, Release 0.2.2dev -- setup.cfg -- setup.py • bootstrap.py and buildout.py are files required by **Buildout** to build the development environment. You can find more information about buildout at Buildout documentation or A brief introduction to **Buildout**.
Buildout documentation	You can find more information about buildout at **Buildout documentation** or A brief introduction to Buildout.
buildout script	Bootstrapping creates needed directories and generates the **buildout script**, which downloads all the Magik dependencies and places command-line scripts to the bin/ directory.
Buildout Buildout Buildout	1.1.3 **Buildout Buildout Buildout** is a Python-based build system for creating, assembling and deploying applications.
changes set	hg push sends **changes sets** (commits) to other repository.
changes wit	To share your **changes wit** the world, you need to push your changes either to the main repository, or to any other repository.
CHAPTER ONE DEVELOPERS	**CHAPTER ONE DEVELOPERS**’ GUIDE 1.1 Starting development This guide will serve as a basic introduction to installing Magik for development purposes.
ci	Once you prepared the code base for a new commit, run hg **ci** to make a commit to the local repository.
code base	Once you prepared the **code base** for a new commit, run hg ci to make a commit to the local repository.
code change	Committing changes Once a feature is implemented, **code changes** have to be committed.

codereview	Todo codereview It would be nice to setup **codereview** at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the programmer.
coding style	Read MoinMoin **coding style** to get inspiration.
color extension	To make output of Mercurial more readable, the **color extension** can be enabled.
command line tool	• bin/py.test py.test is a **command line tool** to collect, run and report about automated tests.
coverage report	bin/py.test --cover=magik --cover-report=html File coverage/index.html contains the **coverage report** in the HTML format.
cover-report	bin/py.test --cover=magik --**cover-report**=html File coverage/index.html contains the coverage report in the HTML format.
Data Completeness	A Tool for Managing **Data Completeness**, Release 0.2.2dev -- setup.cfg -- setup.py • bootstrap.py and buildout.py are files required by Buildout to build the development environment. A Tool for Managing **Data Completeness**, Release 0.2.2dev In this case the toolkit will run unit tests of the magik module and doctests in the doc/ directory. A Tool for Managing **Data Completeness**, Release 0.2.2dev As a rule of thumb, try not to run the code manually to see if the feature you have written works, but write tests that implement such a run.
Definitive Guide	The **Definitive Guide** or MoinMoin Mercurial guide The Hg tip has handy tips for Mercurial users.
dependency installation	It is used for automatic project deployment and **dependency installation** in an isolated environment.
development process	Before starting the **development process** you need to have Python installed on your system.
development purpose	CHAPTER ONE DEVELOPERS' GUIDE 1.1 Starting development This guide will serve as a basic introduction to installing Magik for **development purposes**.
development environment	A Tool for Managing Data Completeness, Release 0.2.2dev -- setup.cfg -- setup.py • bootstrap.py and buildout.py are files required by Buildout to build the **development environment**.
diff	[extensions] color = hg st and hg **diff** are very handy in finding out what have been changed, what files are missing (were not added to the repository).
doctests	A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit will run unit tests of the magik module and **doctests** in the doc/ directory. bin/py.test magik # Run unit tests bin/py.test doc # Run **doctests** bin/py.test magik/statement/_tests /test_select_parser.py # Run tests defined in magik/statement/_tests/test_select_parser.py Note:
documentation Documentation	1.3 Writing **documentation Documentation** is located in the doc/ directory.
executable	The most important **executables** in the bin/ directory are:
extensions section	To enable it ad to the **extensions sections** of your .
fancy interface	• bin/buildout • bin/bpython bpython is a **fancy interface** to the Python interpreter.
file path	You can specify which test to run by providing a directory or a **file path** which contains test fiels to py.test.
formatting fix	it can be either a bugfix, a **formatting fix**, a new feature, a new unit test, or anything else, what is important that it is one item, not several.
GUIDE	CHAPTER ONE DEVELOPERS' **GUIDE** 1.1 Starting development This guide will serve as a basic introduction to installing Magik for development purposes.
handy tip	The Definitive Guide or MoinMoin Mercurial guide The Hg tip has **handy tips** for Mercurial users.
hg	**hg** in shows commits that you are missing, hg out shows commits that other repository is missing. **hg** up allows to switch to desired commit.
hg clone	**hg clone** https://magik.inf.unibz.it/hg/magik/ The command will copy the source code from the server to your local computer to folder ./magik with the following structure:
hg diff	**hg diff** shows which content of files was removed, added or changed.
hg pull	**hg pull** gets them from other repository.
hg push	**hg push** sends changes sets (commits) to other repository.
Hg tip	The Definitive Guide or MoinMoin Mercurial guide The **Hg tip** has handy tips for Mercurial users.
HTML	bin/py.test --cover=magik --cover-report=html File coverage/index.html contains the coverage report in the **HTML** format.
label tip	The most recent commit has **label tip**.
least Python	You must have at **least Python** 2.6 to run Magik.
line color	hgrc **line color** =.
Linux	The steps have been tested on **Linux** and Mac OS X.
magik	./magik/ -- bootstrap.py -- buildout.cfg -- conftest.py -- doc -- **magik** 1 http://mercurial.selenic.com /wiki/QuickStart#Setting_a_username 1
Magik	CHAPTER ONE DEVELOPERS' GUIDE 1.1 Starting development This guide will serve as a basic introduction to installing **Magik** for development purposes. You must have at least Python 2.6 to run **Magik**. 1.2 Developing **Magik** 1.2.1 Running tests py.test is a testing toolkit that is used in Magik to run unit

	tests. 1.2 Developing Magik 1.2.1 Running tests py.test is a testing toolkit that is used in **Magik** to run unit tests. 1.2.2 Making changes Coding standarts In general, **Magik** is written with PEP 8 in mind.
Magik dependency	Bootstrapping creates needed directories and generates the buildout script, which downloads all the **Magik dependencies** and places command-line scripts to the bin/ directory.
Mercurial	By default **Mercurial** uses a username of the form user@localhost for commits. [ui] username = John Doe <john@example.com> For more detail refer to the Mercurial wiki, **Mercurial**:
Mercurial user	The Definitive Guide or MoinMoin Mercurial guide The Hg tip has handy tips for **Mercurial users**.
Mercurial wiki	[ui] username = John Doe <john@example.com> For more detail refer to the **Mercurial wiki**, Mercurial:
Mercurial documentation	Refer to the **Mercurial documentation** for more details.
Mercurial guide	The Definitive Guide or MoinMoin **Mercurial guide** The Hg tip has handy tips for Mercurial users.
MoinMoin	The Definitive Guide or **MoinMoin** Mercurial guide The Hg tip has handy tips for Mercurial users. Read **MoinMoin** coding style to get inspiration.
package documentation	• doc/ contains **package documentation**.
pastebin	py.test can submit its output to a **pastebin**, a hosting for text snippets. bin/py.test magik --**pastebin**=all There is another test runner:
PEP	It additionally to unit and doc tests performs the **PEP** 8 check and static analysis by pyflakes. 1.2.2 Making changes Coding standarts In general, Magik is written with **PEP** 8 in mind.
programmer	Todo codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the **programmer**.
pyflakes	It additionally to unit and doc tests performs the PEP 8 check and static analysis by **pyflakes**.
python	cd magik/ **python** bootstrap.py bin/buildout Note:
Python	Before starting the development process you need to have **Python** installed on your system. To specify which **Python** should be used, run the bootstrap script with it. python2.6 bootstrap.py # Use **Python** 2.6 /usr/bin/python bootstrap.py # Explicitly define python interpretator.
Python egg	• setup.cfg and setup.py are files required to cread a **Python egg**.
python interpretator	python2.6 bootstrap.py # Use Python 2.6 /usr/bin/python bootstrap.py # Explicitly define **python interpretator**.
Python interpreter	• bin/buildout • bin/bpython bpython is a fancy interface to the **Python interpreter**.
readable	To make output of Mercurial more **readable**, the color extension can be enabled.
repository	1.1.2 Getting the source code To get sources you need to clone the main **repository**: First of all we need to “cd” to the folder with **repository**, bootstrap the project, and buildout it: hg st shows the status of the **repository**: [extensions] color = hg st and hg diff are very handy in finding out what have been changed, what files are missing (were not added to the **repository**). Once you prepared the code base for a new commit, run hg ci to make a commit to the local **repository**. To share your changes wit the world, you need to push your changes either to the main **repository**, or to any other repository. To share your changes wit the world, you need to push your changes either to the main repository, or to any other **repository**. hg push sends changes sets (commits) to other **repository**. hg pull gets them from other **repository**. hg in shows commits that you are missing, hg out shows commits that other **repository** is missing.
source code	Also Mercurial, a version control system, is required to get the **source code**. 1.1.2 Getting the **source code** To get sources you need to clone the main repository: hg clone https://magik.inf.unibz.it/hg/magik/ The command will copy the **source code** from the server to your local computer to folder ./magik with the following structure: • magik/ is, actually, the **source code** of the program.
source code change	Having atomic commits is important in teams, where team members need to track how the **source code changes**.
Sphinx documentation	Read **Sphinx documentation** for more details.
standarts	1.2.2 Making changes Coding **standarts** In general, Magik is written with PEP 8 in mind.
static analysis	It additionally to unit and doc tests performs the PEP 8 check and **static analysis** by pyflakes.
team member	Having atomic commits is important in teams, where **team members** need to track how the source code changes.
Test class name	**Test class names** should start with Test and test methods with test_.

test class	Inside the test module (or directory) an __init__.py file should be located, and files that contains **test classes** named test_*.py.
test coverage	bin/py.test-friends It is also possible to check **test coverage**.
test fiels	You can specify which test to run by providing a directory or a file path which contains **test fiels** to py.test.
test method	Test class names should start with Test and **test methods** with test_.
test module	Inside the **test module** (or directory) an __init__.py file should be located, and files that contains test classes named test_*.py.
test run	Sometimes it is handy to share the output of a **test run**.
test runner	bin/py.test magik --pastebin=all There is another **test runner**:
testing toolkit	• conftest.py configures the **testing toolkit**. 1.2 Developing Magik 1.2.1 Running tests py.test is a **testing toolkit** that is used in Magik to run unit tests.
tests Unit test	Todo codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing **tests Unit tests** is an important part of the project, they help to ensure that the program works as was ment by the programmer.
text snippet	py.test can submit its output to a pastebin, a hosting for **text snippets**.
Todo	**Todo** pep8 Actually, there still some work needs to be done to respect pep8 more. **Todo** codereview It would be nice to setup codereview at http://magik.inf.unibz.it/rb Writing tests Unit tests is an important part of the project, they help to ensure that the program works as was ment by the programmer.
toolkit	A Tool for Managing Data Completeness, Release 0.2.2dev In this case the **toolkit** will run unit tests of the magik module and doctests in the doc/ directory.
ui	[**ui**] username = John Doe <john@example.com> For more detail refer to the Mercurial wiki, Mercurial:
unit test	it can be either a bugfix, a formatting fix, a new feature, a new **unit test**, or anything else, what is important that it is one item, not several.
unit test	1.2 Developing Magik 1.2.1 Running tests py.test is a testing toolkit that is used in Magik to run **unit tests**. A Tool for Managing Data Completeness, Release 0.2.2dev In this case the toolkit will run **unit tests** of the magik module and doctests in the doc/ directory. bin/py.test magik # Run **unit tests** bin/py.test doc # Run doctests bin/py.test magik/statement/_tests /test_select_parser.py # Run tests defined in magik/statement/_tests/test_select_parser.py Note:
username	1.1.1 Setting up Mercurial To use Mercurial you need to set up your **username**. By default Mercurial uses a **username** of the form user@localhost for commits. [ui] **username** = John Doe <john@example.com> For more detail refer to the Mercurial wiki, Mercurial:
USERPROFILE	It's best to configure a proper email address in ~/.hgrc (or on a Windows system in %**USERPROFILE**%\Mercurial.ini) by adding lines such as the following 1:
version control	If you have created new files, they needed to be added to the **version control**.
version control system	Also Mercurial, a **version control system**, is required to get the source code.