HOW TO WRITE PYLINT PLUGINS

Alexander Todorov

http://kiwitcms.org
https://github.com/PyCQA/pylint-django/

FOSDEM 2019
Brussels

WHY DO WE NEED *MORE* LINTERS ?

Example: doc-string quotes

```
def this_is_documented():
    """
     like this
    """

def this_is_documented():
    '''
     or like this
    '''

def this_is_documented():
    ' or maybe like this '

def this_is_documented():
    " or even like this "
```

Kiwi TCMS

Example: hard-coded auth.User

Instead of:

```
from django.contrib.auth.models import User
User.objects.get()

class Component(models.Model):
    name = models.CharField()
    initial_owner = models.ForeignKey('auth.User')
```

Django recommends:

```
from django.contrib.auth import get_user_model
get_user_model().objects.get()

class Component(models.Model):
    name = models.CharField()
    initial_owner = models.ForeignKey(
                                  settings.AUTH_USER_MODEL)
```

Example: missing permissions

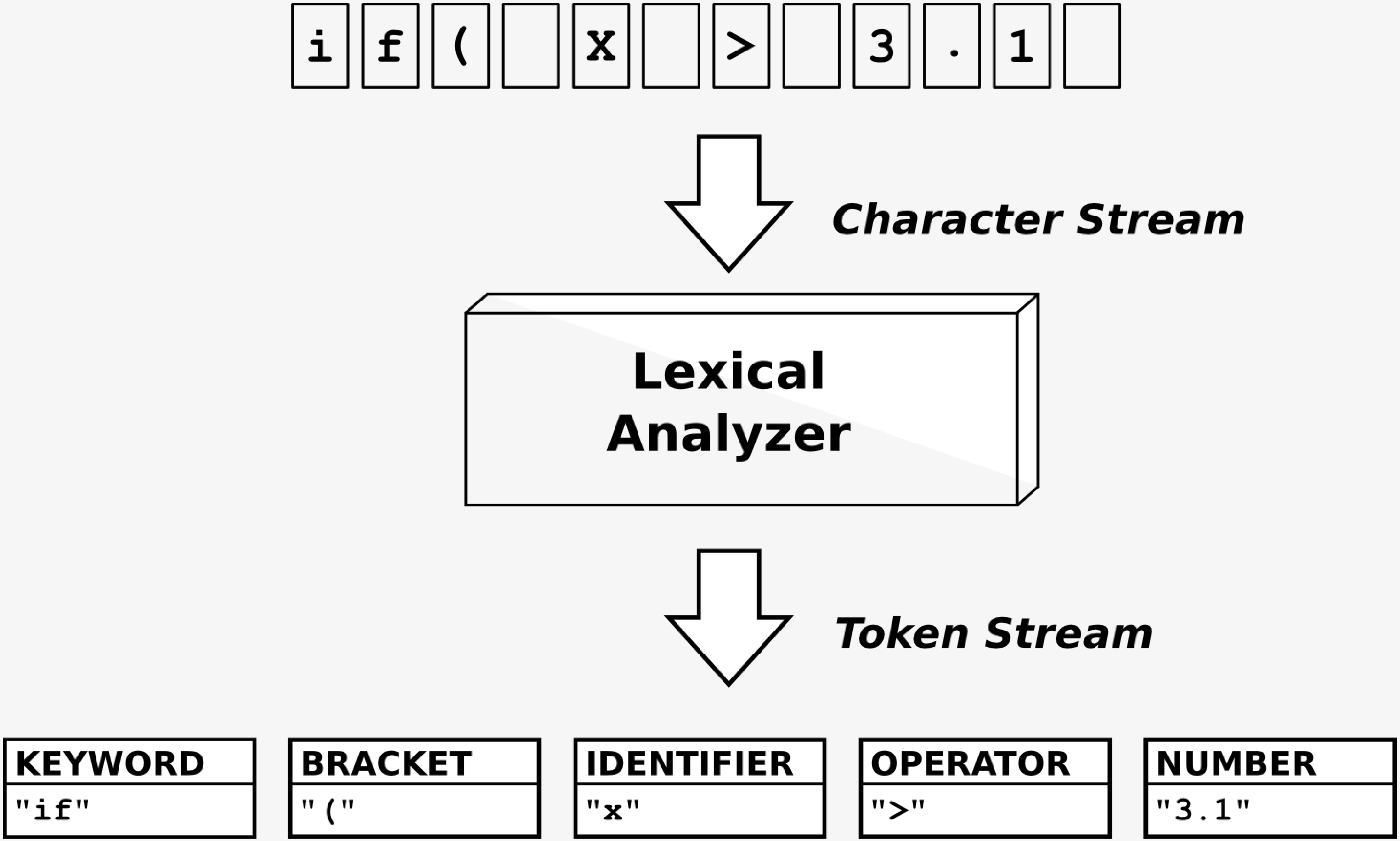
```
@permission_required('testplans.add_testplan')
def new_test_plan(request):
    pass


@permission_required('testplans.change_testplan')
def edit_test_plan(request):
    pass
```

PARSING & AST CRASH COURSE

```
while b ≠ 0
    if a > b
        a := a − b
    else
        b := b − a
return a
```

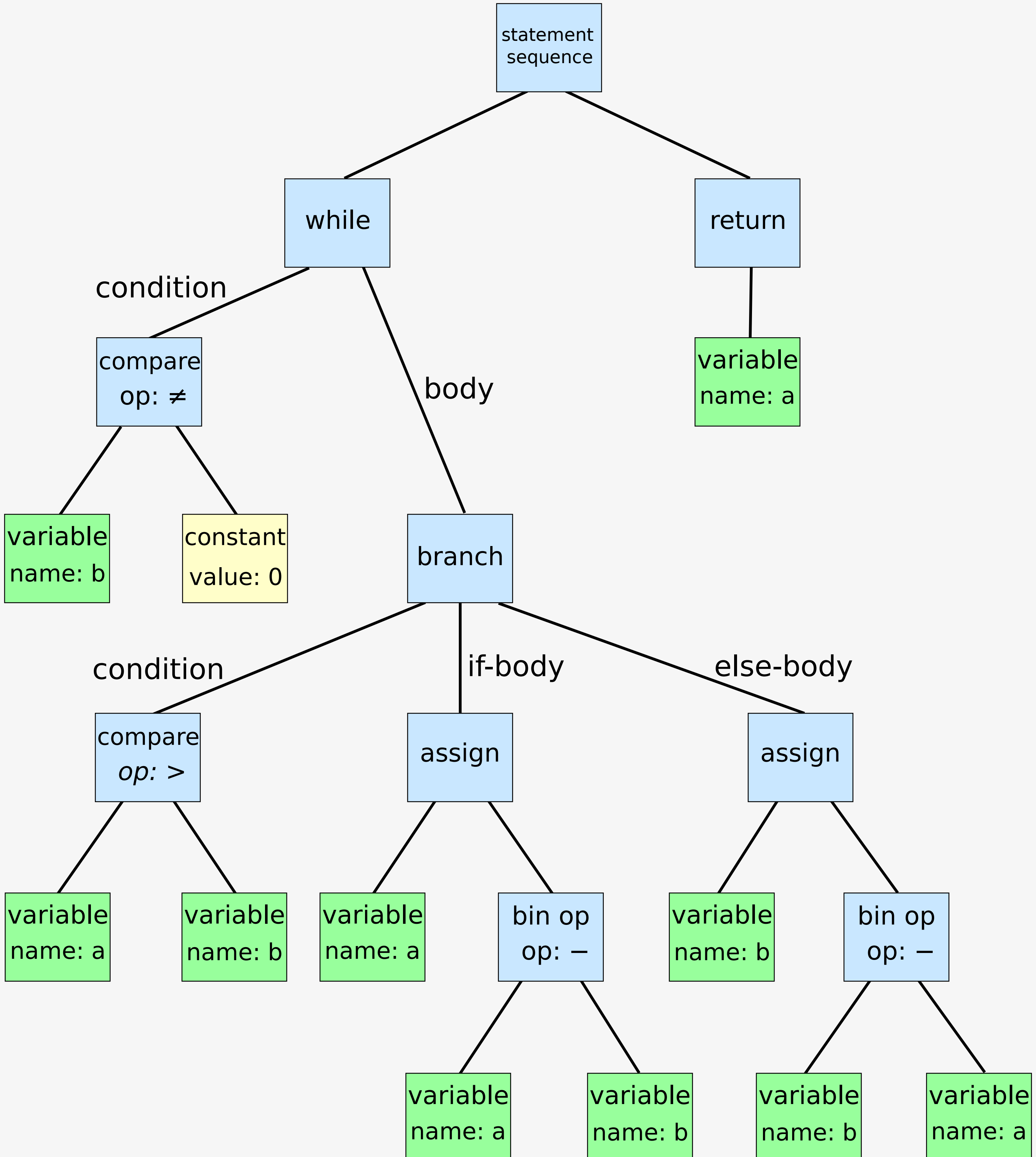
https://en.wikipedia.org/wiki/Parsing#Computer_languages

https://en.wikipedia.org/wiki/Abstract_syntax_tree


i
f
(
x
>
3
.
1
Character Stream
Lexical Analyzer
Token Stream
KEYWORD
"if"
BRACKET
"("
IDENTIFIER
"x"
OPERATOR
">"
NUMBER
"3.1"

TOKENIZING IN PYTHON

```
>>> import io, tokenize
>>> for token_info in tokenize.tokenize(io.BytesIO(
            b"""print('Hello World')""").readline):
...     print(token_info)

TokenInfo(type=59 (ENCODING), string='utf-8', start=(0, 0), end=(0, 0),
line='')

TokenInfo(type=1 (NAME), string='print', start=(1, 0), end=(1, 5),
line="print('Hello World')")

TokenInfo(type=53 (OP), string='(', start=(1, 5), end=(1, 6),
line="print('Hello World')")

TokenInfo(type=3 (STRING), string="'Hello World'", start=(1, 6),
end=(1, 19), line="print('Hello World')")

TokenInfo(type=53 (OP), string=')', start=(1, 19), end=(1, 20),
line="print('Hello World')")

TokenInfo(type=0 (ENDMARKER), string='', start=(2, 0), end=(2, 0),
line='')
```


statement sequence
while
return
condition
body
compare op: ≠
variable name: a
variable name: b
constant value: 0
branch
condition
if-body
else-body
compare op: >
assign
assign
variable name: a
variable name: b
variable name: a
bin op op: −
variable name: b
bin op op: −
variable name: a
variable name: b
variable name: b
variable name: a

AST WITH ASTROID

```
>>> import ast          # built-in, not used by pylint
>>> import astroid      # dependency of pylint, very similar to ast

>>> print(astroid.parse('print("Hello World")').repr_tree())
Module(
   name='',
   doc=None,
   file='<?>',
   path=['<?>'],
   package=False,
   pure_python=True,
   future_imports=set(),
   body=[Expr(value=Call(
            func=Name(name='print'),
            args=[Const(value='Hello World')],
            keywords=None))])
```

ASTROID CHEAT SHEET

- https://astroid.readthedocs.io/en/latest/api/astroid.nodes.html
- https://astroid.readthedocs.io/en/latest/api/base_nodes.html

- `.as_string()`
- `.repr_tree()`
- `.parent - the parent node in the syntax tree`
- `.frame(self) - the first parent frame node(Module, FunctionDef, or ClassDef)`

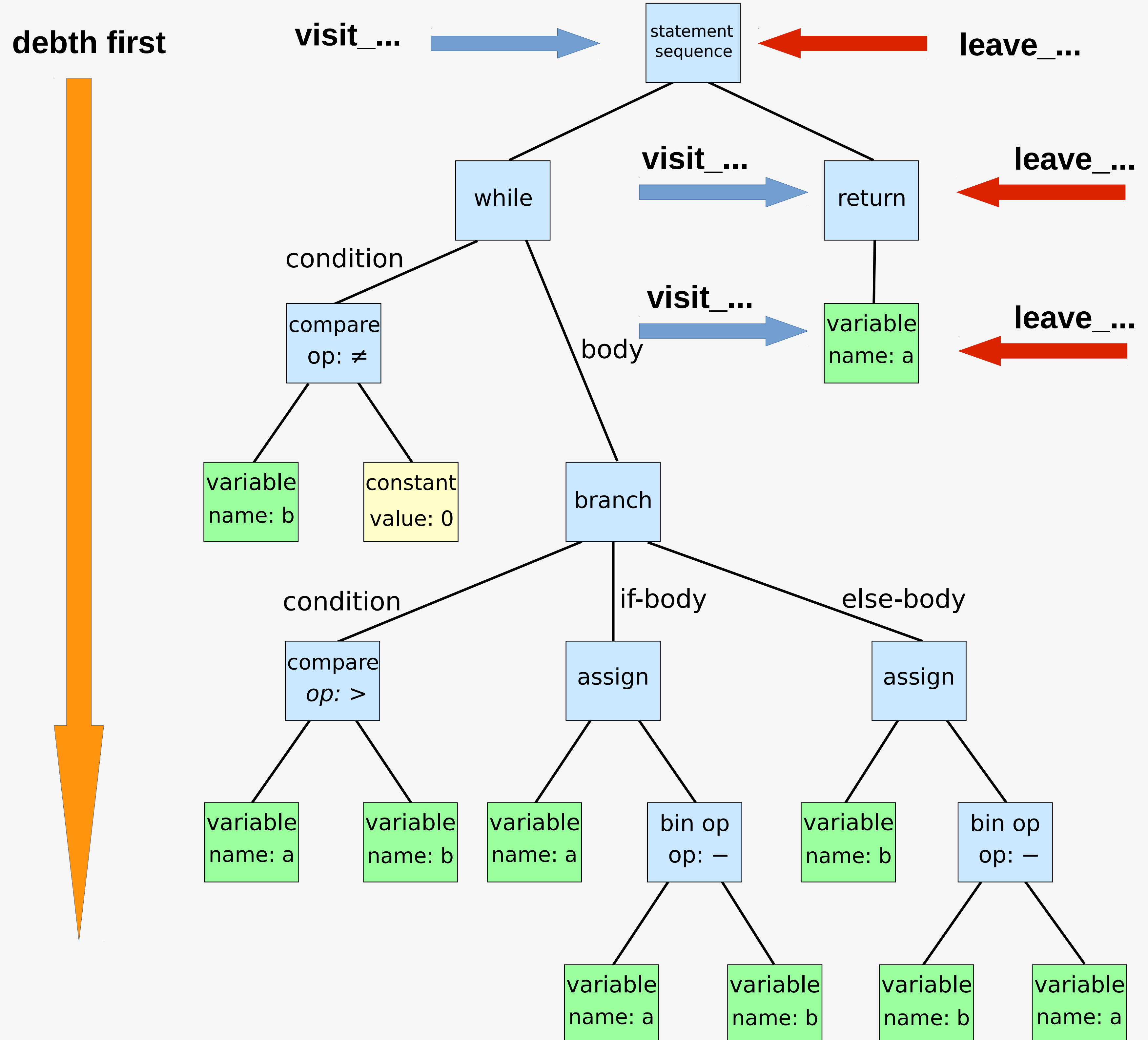
- `.parent_of()`
- `.get_children()`
- `.next_sibling()`
- `.previous_sibling()`

Kiwi TCMS

PYLINT CHECKER INTERFACES

```
IChecker:
  def open(self)
  def close(self)

IRawChecker(IChecker):
  def process_module(self, astroid.parse())

ITokenChecker(IChecker):
  def process_tokens(self, tokenize.tokenize())

IAstroidChecker(IChecker):
  def visit_...(self, node)
  def leave_...(self, node)
```

`... == astroid.__class__.__name__.lower()`



debth first
visit_...
statement sequence
leave_...
while
visit_...
return
leave_...
condition
visit_...
compare op: ≠
body
variable name: a
leave_...
variable name: b
constant value: 0
branch
condition
if-body
else-body
compare op: >
assign
assign
variable name: a
variable name: b
variable name: a
bin op op: −
variable name: b
bin op op: −
variable name: a
variable name: b
variable name: b
variable name: a

PLUGIN SKELETON

```
$ cat myplugin.py

from checkers.awesome import AwesomeChecker


def register(linter):
    print("Hello Pylint Plugins")

    linter.register_checker(AwesomeChecker(linter))
```

Kiwi TCMS

CHECKER SKELETON

```
from pylint import interfaces
from pylint import checkers


class AwesomeChecker(checkers.BaseChecker):
    __implements__ = (interfaces.IAstroidChecker, )

    name = 'awesome-checker'

    # Convention, Warning, Error, Fatal & Refactoring
    msgs = {'R12345': ('Short message',
                       'awesome-checker-message',
                       'Longer help message!')}

    def close(self):
        print('Awesome checker finished working')
        self.add_message('awesome-checker-message',
                                  node=None)
```

INVOKING THE PLUGIN

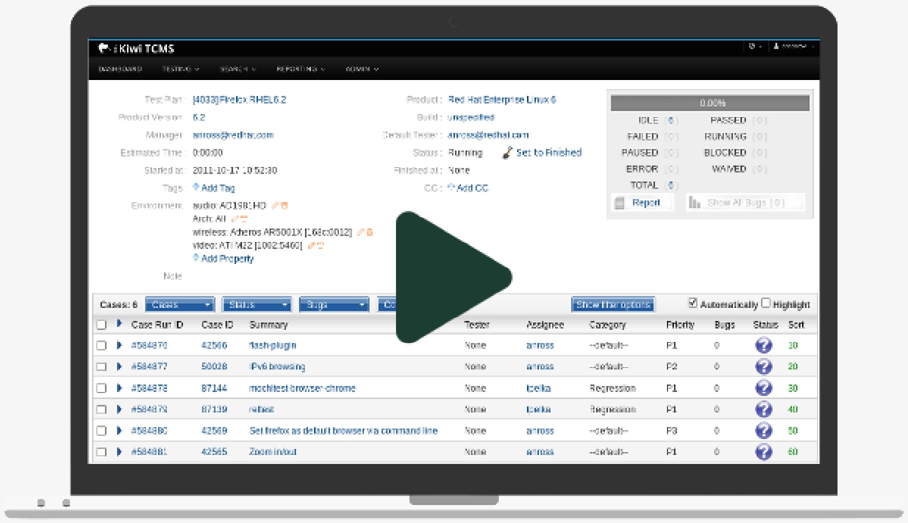
```
$ PYTHONPATH=. pylint --load-plugins=myplugin *.py

Hello Pylint Plugins

No config file found, using default configuration
************* Module myplugin
C:  1, 0: Missing module docstring (missing-docstring)
C:  2, 0: Missing function docstring (missing-docstring)
W:  2,13: Unused argument 'linter' (unused-argument)

Awesome checker finished working

---------------------------------------------------------
Your code has been rated at -5.00/10 (previous run:
-5.00/10, +0.00)
```

Kiwi TCMS

the leading open source test case management system

- Transform the testing process within your team
- Be more organized, transparent & accountable
- Boost engineering productivity & participation
- 10+ years of history
- GPL 2 licensed

GitHub Login | See Features



Test Plan : [4033]Firefox RHEL6.2
Product Version : 6.2
Manager : aross@redhat.com
Estimated Time : 0:00:00
Started at : 2011-10-17 10:52:30
Tags : Add Tag
Environment : audio: AD1981HD
Arch: All
wireless: Atheros AR5001X [168c:0012]
video: ATI M22 [1002:5460]
Add Property
Note :

Product : Red Hat Enterprise Linux 6
Build : unspecified
Default Tester : aross@redhat.com
Status : Running Set to Finished
Finished at : None
CC : Add CC

0.00%

IDLE	[6]	PASSED	[0]
FAILED	[0]	RUNNING	[0]
PAUSED	[0]	BLOCKED	[0]
ERROR	[0]	WAIVED	[0]
TOTAL	[6]		

Report Show All Bugs [0]

Cases: 6 Cases Status Bugs Co Show filter options ☑ Automatically ☐ Highlight

Case Run ID	Case ID	Summary	Tester	Assignee	Category	Priority	Bugs	Status	Sort
#584876	42566	flash-plugin	None	aross	--default--	P1	0	?	10
#584877	50028	IPv6 browsing	None	aross	--default--	P2	0	?	20
#584878	87144	mochitest-browser-chrome	None	tpelka	Regression	P1	0	?	30
#584879	87139	reftest	None	tpelka	Regression	P1	0	?	40
#584880	42569	Set firefox as default browser via command line	None	aross	--default--	P3	0	?	50
#584881	42565	Zoom in/out	None	aross	--default--	P1	0	?	60

Test case management system, with lots of great features, such as workflow wizards, automation plugins, 3rd party integrations, release reports and external API.

Everyday testing

Use the dashboard to see pending work. Execute tests, mark results and report bugs.

Test management

Create test plans and cases, track progress and assign work across multiple teams. Perform peer reviews.

Process organization

Workflow wizards guide you through requirements gathering, user story definition and test planning.

Reporting

See who's doing what and provide status report to stakeholders before release. Centralize your acceptance books!

Integration

Integrated bug reporting with Bugzilla, JIRA, GitHub and GitLab. Planned system integration with Trello and GitHub.

Test runner plugins

Use Kiwi TCMS plugins for JUnit, TestNG, py.test and other popular test runners to collect test automation results!

External API

Provides full access so you can get creative. Available via JSON and XML RPC with API client in Python.

Fast

Performance baseline at 7.5 req/sec or 130 msec/req which is comparable to giants like GitHub!

https://github.com/kiwitcms/Kiwi/tree/master/kiwi_lint

doc-string checker

```
_string_tokens = {}

def process_tokens(self, tokens):
    for (tok_type, token, _, _, _) in tokens:
        if tok_type == tokenize.STRING:
            token_text = token.strip('"').strip("'")
            self._string_tokens[token_text] = token

def visit_{module|classdef|functiondef}(self, node):
    self._check_docstring(node)

def _check_docstring(self, node):
    if node.doc in self._string_tokens:
        token = self._string_tokens[node.doc]
        if not token.startswith('"""'):
            self.add_message('use-triple-double-quotes',
                             node=node)
```

Kiwi TCMS

auth.User checker

```
def visit_const(self, node):
    if node.value == 'auth.User':
        self.add_message('hard-coded-auth-user', node=node)

def visit_importfrom(self, node):
    if node.modname == 'django.contrib.auth.models':
        for imported_names in node.names:
            if imported_names[0] in ['*', 'User']:
                self.add_message('imported-auth-user',
                                 node=node)
                break
```

Missing permissions checker

```
def visit_module(self, node):
    self.inside_views_module = node.name.endswith('views')

def visit_functiondef(self, node):
    if not self.inside_views_module:
        return
    arg0 = None
    if node.args.args:
        arg0 = node.args.args[0]
    if arg0 and arg0.name != 'request':
        return
    self._check_for_missing_decorator(node)

def visit_classdef(self, node):
    if not self.inside_views_module:
        return
    if not node.bases:
        return
    self._check_for_missing_decorator(node)
```

Missing permissions checker

```
def _check_for_missing_decorator(self, node):
    if not node.decorators:
        self.add_message('missing-permission-required')
        return

    found_permissions_required = False

    for decorator in node.decorators.nodes:
        if isinstance(decorator, astroid.Call):
            if decorator.func.name == 'permission_required':
                found_permissions_required = True
                break
            elif decorator.func.name == 'method_decorator' and
          decorator.args[0].func.name == 'permission_required':
                found_permissions_required = True
                break

    if not found_permissions_required:
        self.add_message('missing-permission-required')
```

Other checkers in Kiwi TCMS

- empty_module.py – discovers empty modules
- nested_definition.py – discovers nested functions and classes
- bulk_create.py – Model.objects.bulk_create() b/c history
- objects_update.py – Model.objects.update() b/c history
- raw_sql.py – Model.objects.extra(select=“raw sql“)
- tags.py – Tag.objects.get_or_create() -> Tag.get_or_create()
- `https://github.com/PyCQA/pylint/pull/1823 - useless return`
- `https://github.com/PyCQA/pylint/pull/1939 - try-except-raise`
- `https://github.com/PyCQA/pylint-django/pull/158`
  - Use `JsonResponse() instead of HttpResponse(json.dumps())`

- `https://github.com/kiwitcms/Kiwi/issues?utf8=%E2%9C%93&q=is%3Aissue+is%3Aopen+new+linter`

THANK YOU !

Building K, Level 2, Stand 10
Open Source Test Management

Kiwi TCMS