# Odoo Coding Guidelines

Pham Thi Ngoc Mai

August 18th, 2021

Onnet - AHT

## I. Introduction

## 1 - Why these?

Those aim to improve the **quality** of Odoo Apps code. Indeed proper code improves **readability**, **eases maintenance**, helps **debugging**, **lowers complexity** and promotes **reliability**.

## 2 - What these include?

- Structure and naming conventions
- Formating rule
- Python standard
- Programming in Odoo
- Javascript & CSS
- Git

## 3 - How?

Learn, Learn more, Noob forever?

## II. Module structure

## Directories

```
crm
|-- data: demo and data xml
|-- models: models def
|-- controllers: HTTP routes
|-- views: views and templates
|-- static: web assets
|-- security: access rights and record rules
|-- report
|-- security
|-- tests
|-- wizard
|-- i18n: translations
|-- __init__.py
|-- __manifest__.py
```

## File naming - models

```
models
|-- crm_lead.py
|-- crm_lost_reason.py
|-- crm_stage.py
|-- crm_team.py
|-- res_partner.py
|-- res_users.py
...
```

Split the business logic by sets of models belonging to a same main model

## File naming - security

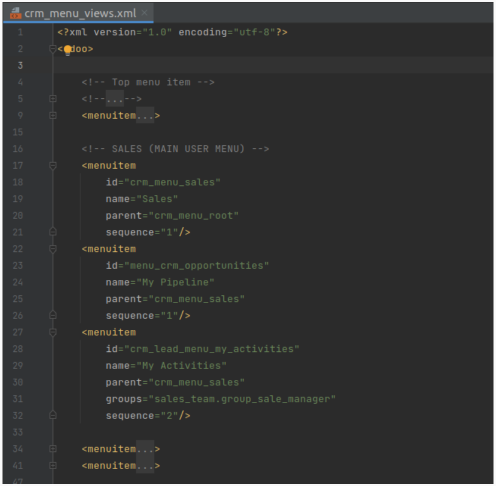
```
security
|-- crm_security.xml
|-- ir.model.access.csv
```

## File naming - views

```
views
|-- assets.xml
|-- crm_lead_views.xml
|-- crm_lost_reason_views.xml
|-- crm_menu_views.xml
|-- crm_stage_views.xml
|-- crm_team_views.xml
|-- res_partner_views.xml
```

- backend views: `<model>_views.xml`
- menus: `<module>_menus.xml`
- templates: `<model>_template.xml`
- bundles: `assets.xml`

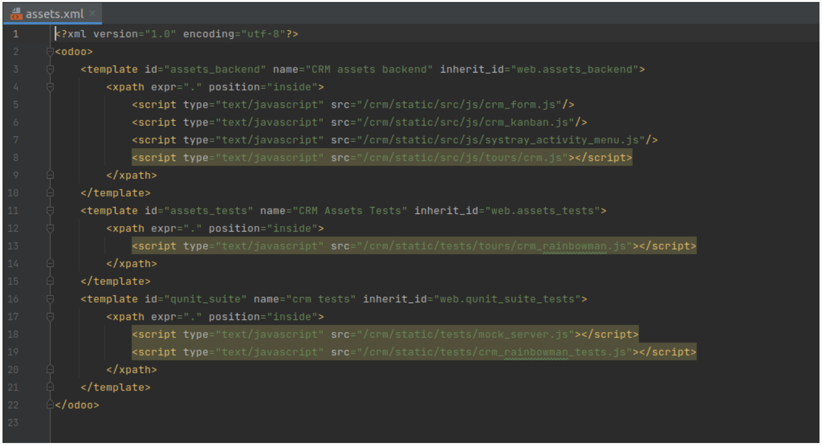
## Views menu - CRM



```
<?xml version="1.0" encoding="utf-8"?>
<odoo>

    <!-- Top menu item -->
    <!--...-->
    <menuitem...>

    <!-- SALES (MAIN USER MENU) -->
    <menuitem
        id="crm_menu_sales"
        name="Sales"
        parent="crm_menu_root"
        sequence="1"/>
    <menuitem
        id="menu_crm_opportunities"
        name="My Pipeline"
        parent="crm_menu_sales"
        sequence="1"/>
    <menuitem
        id="crm_lead_menu_my_activities"
        name="My Activities"
        parent="crm_menu_sales"
        groups="sales_team.group_sale_manager"
        sequence="2"/>

    <menuitem...>
    <menuitem...>
```

**Figure 1:** crm_menu_views.xml

## Views assets - CRM



```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <template id="assets_backend" name="CRM assets backend" inherit_id="web.assets_backend">
        <xpath expr="." position="inside">
            <script type="text/javascript" src="/crm/static/src/js/crm_form.js"/>
            <script type="text/javascript" src="/crm/static/src/js/crm_kanban.js"/>
            <script type="text/javascript" src="/crm/static/src/js/systray_activity_menu.js"/>
            <script type="text/javascript" src="/crm/static/src/js/tours/crm.js"></script>
        </xpath>
    </template>
    <template id="assets_tests" name="CRM Assets Tests" inherit_id="web.assets_tests">
        <xpath expr="." position="inside">
            <script type="text/javascript" src="/crm/static/tests/tours/crm_rainbowman.js"></script>
        </xpath>
    </template>
    <template id="qunit_suite" name="crm tests" inherit_id="web.qunit_suite_tests">
        <xpath expr="." position="inside">
            <script type="text/javascript" src="/crm/static/tests/mock_server.js"></script>
            <script type="text/javascript" src="/crm/static/tests/crm_rainbowman_tests.js"></script>
        </xpath>
    </template>
</odoo>
```

**Figure 2:** crm/views/assets.xml

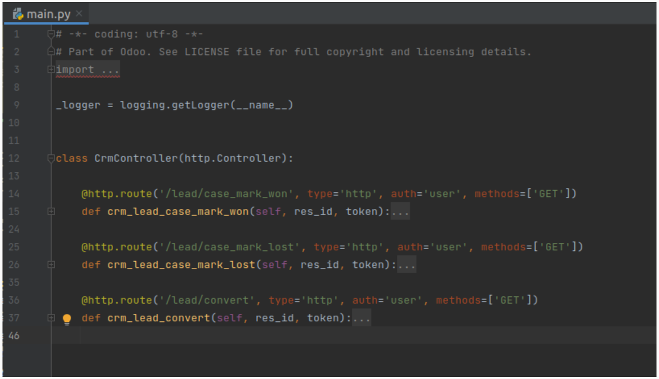
## File naming - data

```
data
|-- crm_lead_demo.xml
|-- crm_lost_reason_data.xml
|-- crm_stage_data.xml
|-- crm_team_data.xml
|-- crm_team_demo.xml
...
```

- Split them by purpose:
  - demo: `<model>_demo.xml`
  - data: `<model>_data.xml`

## File naming - controllers

- outdated: `main.py`
- now: `<module_name>.py`
- inherit: `<inherited_module_name>.py`



main.py

```python
# -*- coding: utf-8 -*-
# Part of Odoo. See LICENSE file for full copyright and licensing details.
import ...

_logger = logging.getLogger(__name__)


class CrmController(http.Controller):

    @http.route('/lead/case_mark_won', type='http', auth='user', methods=['GET'])
    def crm_lead_case_mark_won(self, res_id, token):...

    @http.route('/lead/case_mark_lost', type='http', auth='user', methods=['GET'])
    def crm_lead_case_mark_lost(self, res_id, token):...

    @http.route('/lead/convert', type='http', auth='user', methods=['GET'])
    def crm_lead_convert(self, res_id, token):...
```

**Figure 3:** CRM Controllers

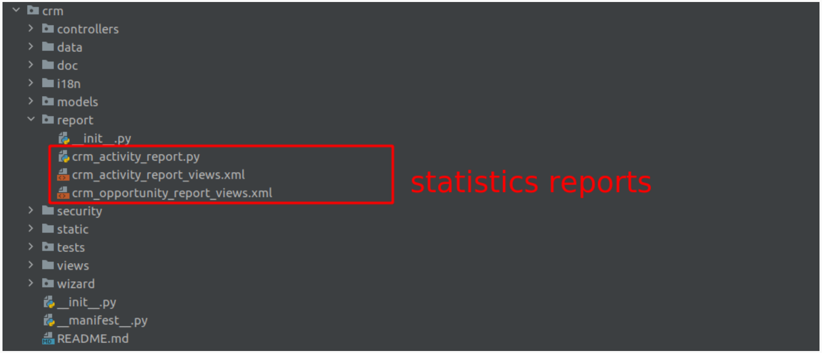
## File naming - static

skip (->_____->) =============>

## File naming - wizard

```
wizard
|-- crm_lead_lost.py
|-- crm_lead_lost_views.xml
|-- crm_lead_to_opportunity.py
|-- crm_lead_to_opportunity_views.xml
|-- crm_merge_opportunities.py
|-- crm_merge_opportunities_views.xml
```

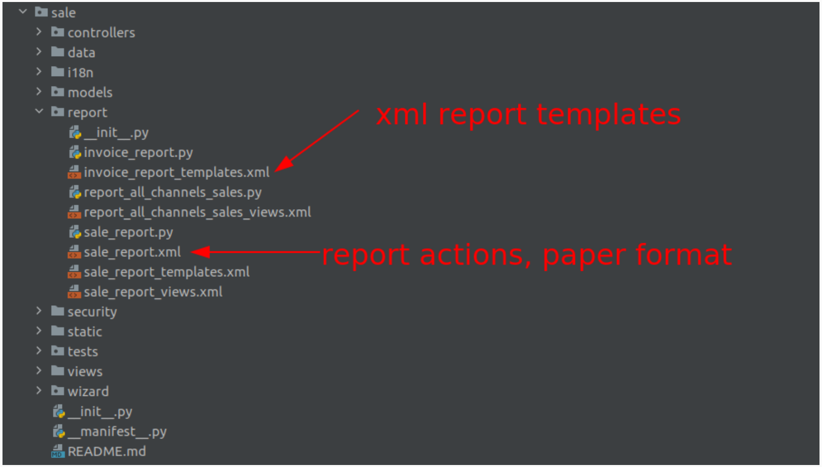
## File naming - report



```
crm
  controllers
  data
  doc
  i18n
  models
  report
    __init__.py
    crm_activity_report.py
    crm_activity_report_views.xml
    crm_opportunity_report_views.xml
  security
  static
  tests
  views
  wizard
  __init__.py
  __manifest__.py
  README.md
```

**Figure 4:** Statistics Reports

## File naming - report



```
sale
  controllers
  data
  i18n
  models
  report
    __init__.py
    invoice_report.py
    invoice_report_templates.xml
    report_all_channels_sales.py
    report_all_channels_sales_views.xml
    sale_report.py
    sale_report.xml
    sale_report_templates.xml
    sale_report_views.xml
  security
  static
  tests
  views
  wizard
  __init__.py
  __manifest__.py
  README.md
```

**Figure 5:** Printable Reports

## File naming - Notes

***NOTE:*** File names should only contain [a-z0-9_] (lowercase alphanumerics and _)

## III. XML files

## Format

- `id` before `model`
- fields: `name` then `eval` then others (widgets, options, . . . )
- group records by model **except** dependencies between action/menu/views
- naming convention (later)

## Format

```
<record id="view_id" model="ir.ui.view">
  <field name="name">view.name</field>
  <field name="model">object_name</field>
  <field name="priority" eval="16"/>
  <field name="arch" type="xml">
    <tree>
      <field name="my_field_1"/>
      <field name="my_field_2" string="My Label"
        widget="statusbar"
        statusbar_visible="draft,sent,progress,done" />
      </tree>
  </field>
</record>
```

## Format

- syntactic sugar:
    - `<menuitem>`: `ir.ui.menu`
    - `<template>`: `arch` section of qweb view
    - `<report>`: report action (old)
    - `<act_window>`: action window (old)

## XML IDs and naming

### Security, View and Action

- menu: `<model_name>_menu.xml`
- submenu: `<model_name>_menu_do_stuff.xml`

```
<!-- menus and sub-menus -->
<menuitem
    id="model_name_menu_root"
    name="Main Menu"
    sequence="5"
/>
<menuitem
    id="model_name_menu_action"
    name="Sub Menu 1"
    parent="module_name.module_name_menu_root"
    action="model_name_action"
    sequence="10"
```

## Inheriting XML

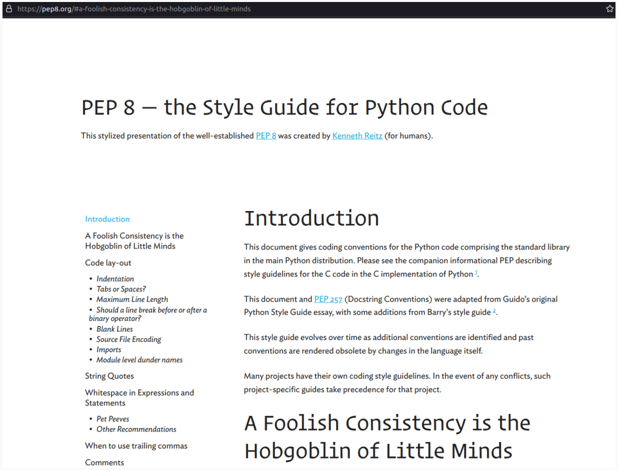
- name: suffix .inherit.{detail}

```
<record id="model_view_form" model="ir.ui.view">
  <field name="name">model.view.form.inherit.module2</fiel
  <field name="inherit_id" ref="module1.model_view_form"/>
  ...
</record>
<record id="module2.model_view_form" model="ir.ui.view">
  <field name="name">model.view.form.module2</field>
  <field name="inherit_id" ref="module1.model_view_form"/>
  <field name="mode">primary</field>
  ...
</record>
```

## IV. Python

## PEP8 options

Odoo source code tries to respect Python standard, but some of them can be ignored.


https://pep8.org/#a-foolish-consistency-is-the-hobgoblin-of-little-minds

PEP 8 — the Style Guide for Python Code

This stylized presentation of the well-established PEP 8 was created by Kenneth Reitz (for humans).

Introduction

A Foolish Consistency is the Hobgoblin of Little Minds

Code lay-out

- *Indentation*
- *Tabs or Spaces?*
- *Maximum Line Length*
- *Should a line break before or after a binary operator?*
- *Blank Lines*
- *Source File Encoding*
- *Imports*
- *Module level dunder names*

String Quotes

Whitespace in Expressions and Statements

- *Pet Peeves*
- *Other Recommendations*

When to use trailing commas

Comments

Introduction

This document gives coding conventions for the Python code comprising the standard library in the main Python distribution. Please see the companion informational PEP describing style guidelines for the C code in the C implementation of Python [1].

This document and PEP 257 (Docstring Conventions) were adapted from Guido's original Python Style Guide essay, with some additions from Barry's style guide [2].

This style guide evolves over time as additional conventions are identified and past conventions are rendered obsolete by changes in the language itself.

Many projects have their own coding style guidelines. In the event of any conflicts, such project-specific guides take precedence for that project.

A Foolish Consistency is the Hobgoblin of Little Minds

Figure 6: PEP8

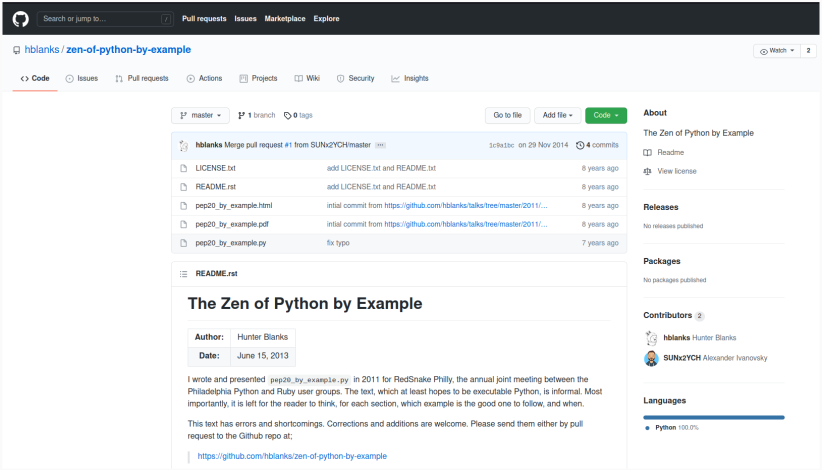
## Imports

```
# 1 : imports of python lib
import base64
import re
import time
from datetime import datetime
# 2 : imports of odoo
import odoo
from odoo import api, fields, models, _
from odoo.tools.safe_eval import safe_eval as eval
# 3 : imports from odoo addons
from odoo.addons.website.models.website import slug
```

## Idiomatics of Programming (Python)

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
...
```

## Idiomatics of Programming (Python)



hblanks Merge pull request #1 from SUNx2YCH/master ··· 1c9a1bc on 29 Nov 2014 4 commits

| | | |
|---|---|---|
| LICENSE.txt | add LICENSE.txt and README.txt | 8 years ago |
| README.rst | add LICENSE.txt and README.txt | 8 years ago |
| pep20_by_example.html | intial commit from https://github.com/hblanks/talks/tree/master/2011/... | 8 years ago |
| pep20_by_example.pdf | intial commit from https://github.com/hblanks/talks/tree/master/2011/... | 8 years ago |
| pep20_by_example.py | fix typo | 7 years ago |

README.rst

The Zen of Python by Example

| | |
|---|---|
| **Author:** | Hunter Blanks |
| **Date:** | June 15, 2013 |

I wrote and presented `pep20_by_example.py` in 2011 for RedSnake Philly, the annual joint meeting between the Philadelphia Python and Ruby user groups. The text, which at least hopes to be executable Python, is informal. Most importantly, it is left for the reader to think, for each section, which example is the good one to follow, and when.

This text has errors and shortcomings. Corrections and additions are welcome. Please send them either by pull request to the Github repo at;

> https://github.com/hblanks/zen-of-python-by-example

About

The Zen of Python by Example

Readme

View license

Releases

No releases published

Packages

No packages published

Contributors 2

hblanks Hunter Blanks

SUNx2YCH Alexander Ivanovsky

Languages

● Python 100.0%

**Figure 7:** The Zen of Python by Examples

## Idiomatics of Programming (Python)

- Use meaningful variable/class/method names
- Useless variable
- Know your builtins
- Use list comprehension, dict comprehension, and basic manipulation using map, filter, sum, ... They make the code easier to read.
- Collections are booleans too

## Idiomatics of Programming (Python)

You can’t learn to write good code only by following the rules.

**To learn to write good code you have to write a shit-metric-ton of bad code.** —- Going beyond the idiomatic Python

## Programming in Odoo

- Avoid to create generators and decorators
- Use filtered, mapped, sorted, … methods to ease code reading and performance.

## Programming in Odoo

Make your method work in batch

```
@api.depends('user_id')
def _compute_date_open(self):
  for lead in self:
    lead.date_open = fields.Datetime.now() if lead.user_id
```

## Programming in Odoo

Propagate the context

- Passing parameter in context can have dangerous side-effects.
- If you need to create a key context influencing the behavior of some object, choice a good name, and eventually prefix it by the name of the module to isolate its impact.

## Programming in Odoo

Keep it **Simple** and **Stupid**

- Split the method as soon as it has more than one responsibility

Never commit the transaction

- You should **NEVER** call cr.commit() yourself, **UNLESS**...

Use translation method correctly

## Symbols and Conventions

**Variables**

- model name: singular form
- suffix your variable name with _id or _ids if it contains a record id or list of id

```
Partner = self.env['res.partner']
partners = Partner.browse(ids)
partner_id = partners[0].id
```

## Symbols and Conventions

**Variables**

- `One2Many` and `Many2Many` fields should always have _ids as suffix
- `Many2One` fields should have _id as suffix

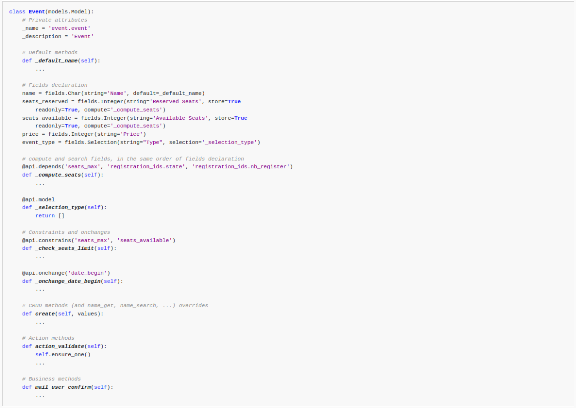
## Symbols and Conventions

**Method conventions**

- compute field: `_compute_<field_name>`
- onchange method: `_onchange_<field_name>`
- constraint method: `_check_<constraint_name>`

## Symbols and Conventions

### Model attribute order



```python
class Event(models.Model):
    # Private attributes
    _name = 'event.event'
    _description = 'Event'

    # Default methods
    def _default_name(self):
        ...

    # Fields declaration
    name = fields.Char(string='Name', default=_default_name)
    seats_reserved = fields.Integer(string='Reserved Seats', store=True
        readonly=True, compute='_compute_seats')
    seats_available = fields.Integer(string='Available Seats', store=True
        readonly=True, compute='_compute_seats')
    price = fields.Integer(string='Price')
    event_type = fields.Selection(string="Type", selection='_selection_type')

    # compute and search fields, in the same order of fields declaration
    @api.depends('seats_max', 'registration_ids.state', 'registration_ids.nb_register')
    def _compute_seats(self):
        ...

    @api.model
    def _selection_type(self):
        return []

    # Constraints and onchanges
    @api.constrains('seats_max', 'seats_available')
    def _check_seats_limit(self):
        ...

    @api.onchange('date_begin')
    def _onchange_date_begin(self):
        ...

    # CRUD methods (and name_get, name_search, ...) overrides
    def create(self, values):
        ...

    # Action methods
    def action_validate(self):
        self.ensure_one()
        ...

    # Business methods
    def mail_user_confirm(self):
        ...
```

**Figure 8:** Attribute order in a model