Université catholique de Louvain
École polytechnique de Louvain
Pôle d'ingénierie informatique

LINGI2252

Software Engineering : measures and maintenance

Assignment step 2: Assessing program quality

November 3, 2014

Professor
Kim Mens

Group 23
Michael Heraly
Thibault Gerondal

Presentation of Reddit

Reddit is a popular social networking service that allows its members to share content as text posts or links. Members can vote "up" or "down" on a content to promote it to other members. The content is organized in subcategories, called *subreddits*. Reddit is an open-source project, and its code is available on Github[1].

1 Methodology

Reddit & Pylint

There was already a `pylintrc` file in the project. This file disables some errors that can be generated by Pylint. We decided to keep their `pylintrc` file because it filters the output that is already consequent.

More precisely, here are the errors that Reddit disabled for Pylint:

Error code	Signification
E1103	X has no Y member (but some types could not be inferred)
W0212	Access to a protected member of X class
W0223	Method Y is abstract in class X but not overridden
C0103	Invalid name "%s" (should match %s)
C0111	Missing docstring
W0142	Used * or ** magic
R0201	Method could be a function
R0915	Too many statements
I0011	Locally disabling ... (don't need to see things we've explicitly disabled)

Particularly, we found that disabling the C0103 error is not a good idea. Respecting naming conventions is a good practice, but it appears that Reddit doesn't really care about that.

Analysis

After installing Reddit and its dependencies[2], we have run Pylint on the source code. We used the *parseable* output feature that Pylint offers, and created a script to create a CVS file format from the parseable file. This way, we created a spreadsheet containing the informations about errors/warnings/refactors/conventions/fatal detected by Pylint: the file concerned, at which line, code of the error, and the message associated. This was a practical help to analyse the Pylint output, and compute the statistics.

[1] https://github.com/reddit/reddit
[2] https://github.com/reddit/reddit/wiki/Install-guide

2 Type of errors

Error	Occ[3]	RFP[4]	Message
E1101	225	17.78%	An object (variable, function, ...) is accessed for a non-existent member.
W0201	144	0.00%	Attribute X defined outside __init__
W0613	98	0.00%	Unused argument X
W0612	91	8.79%	Unused variable X
R0913	91	8.79%	Too many arguments X
W0621	88	9.09%	Redefining name X from outer scope
E0611	49	0.00%	No name X in module Y
R0914	49	4.08%	Too many local variables
W0622	40	2.50%	Redefining built-in X
W0231	38	2.63%	__init__ method from base class X is not called
C0324	34	0.00%	Comma not followed by a space
W0311	27	0.00%	Bad indentation
C0321	26	3.85%	More than one statement on a single line
R0902	24	4.17%	Too many instance attributes
W0221	23	0.00%	Arguments number differs from X method
W0403	22	0.00%	Relative import X, should be Y
E0602	21	80.95%	Undefined variable X
W0232	21	19.05%	Class has no __init__ method
W0102	20	0.00%	Dangerous default value X as argument
R0911	12	0.00%	Too many return statements
W0404	10	0.00%	Reimport X
R0904	9	22.22%	Too many public methods
W0233	8	0.00%	__init__ method from a non direct base class X is called
W0702	8	0.00%	No exception type(s) specified
E0211	7	100.00%	Method has no argument
C0203	7	0.00%	Metaclass method X should have mcs as first argument
E1123	6	83.33%	Passing unexpected keyword argument X in function call
C0322	6	0.00%	Operator not preceded by a space
C0202	5	0.00%	Class method X should have cls as first argument
W0105	5	0.00%	String statement has no effect
R0912	5	0.00%	Too many branches
E1120	4	0.00%	No value passed for parameter X in function call
E0213	4	0.00%	Method should have 'self' as first argument

[3]Number of occurences
[4]Rate of false positive(s)

W0631	4	50.00%	Using possibly undefined loop variable X
E0710	3	0.00%	Raising a new style class which doesn't inherit from BaseException
W0703	3	0.00%	Catching too general exception X
F0401	2	50.00%	Unable to import X
W0106	2	0.00%	Expression X is assigned to nothing
R0901	2	0.00%	Too many ancestors
W0602	2	100.00%	Using global for X but no assigment is done
E1002	1	100.00%	Use of super on an old style class
W0122	1	100.00%	Use of exec
E0702	1	100.00%	Raising X while only classes, instances or string are allowed
W0101	1	0.00%	Unreachable code
E0711	1	0.00%	NotImplemented raised - should raise NotImplementedError
W0701	1	0.00%	Raising a string exception
E0102	1	0.00%	X already defined line Y
E0101	1	0.00%	Explicit return in __init__
E0203	1	100.00%	Access to member X before its definition line Y

Table 1: Analyzed messages

3 Statistics

Precision

We checked the source code to determine if these were true or false-positives. We computed the precision on all the messages given by Pylint (as seen in Table 1). This way, we can have a good idea of the precision of Pylint.

We have computed the precision for each category of the informations given by Pylint :

	Precision	**Messages analyzed**
Precision on Warning	95.89%	657
Precision on Error	77.84%	325
Precision on Refactoring	93.22%	192
Precision on Convention	98.71%	78
Precision on Fatal	50.00%	2
Total Precision	90.90%	1254

As we can see, Pylint is really good at detecting conventions, refactor-

ing and warning issues and is a little bit less accurate for advices about errors.

Fatal happens if an error occurred which prevented Pylint from doing further processing. We got two 'unable to import X' in our case because these modules are not mandatory to run Reddit.

Recall

As the project is too big to be analysed in details, we have selected 5 random files to compute the *recall* statistic.

We didn't manage to find much more than what Pylint did. In `r2/lib/base.py`, the following errors were not reported :

Listing 1: 3 useless functions

```
def try_pagecache(self):
    pass
def pre(self): pass
def post(self): pass
```

Pylint could generate three errors for these functions but didn't.
Another file analysed is `r2/lib/authorize/interaction.py` :

Listing 2: test data in the file..

```
# useful test data:
test_card = dict(AMEX       = ("370000000000002"  , 1234),
                 DISCOVER   = ("6011000000000012" , 123),
                 MASTERCARD = ("5424000000000015" , 123),
                 VISA       = ("4007000000027"    , 123),
                 # visa card which generates error codes based
                 # on the amount
                 ERRORCARD  = ("4222222222222"    , 123))

test_card = Storage((k, CreditCard(cardNumber=x,
                                   expirationDate="2011-11",
                                   cardCode=y)) for k, (x, y) in
                    test_card.iteritems())

test_address = Address(firstName="John",
                       lastName="Doe",
                       address="123␣Fake␣St.",
                       city="Anytown",
                       state="MN",
                       zip="12346")
```

These datas are not used in the code at all. It looks like a developer tested the code with these data, but did not remove them.

We haven't found any remarks to tell in the three others files. The recall we calculated is 0.82%.

4 Bad things

4.1 3 nested for-loops

We found 3 nested for-loops, 2 times in a row, in a single function. This is a good example of the kind of *bad smell* we can encounter in the Reddit project. There should be a refactoring of this function, and it would certainly make the code cleaner to decompose this function in some smaller functions. Pylint did not give a hint of this *bad smell*.

Listing 3: `r2/lib/inventory.py` at lines 261-292

```
for campaign in all_campaigns:
  camp_dates = set(get_date_range(campaign.start_date, \
                                  campaign.end_date))
  sr_names = tuple(sorted(campaign.target.subreddit_names))
  daily_impressions = campaign.impressions / campaign.ndays

  for location in locations:
    if location and not location.contains(campaign.location):
      # campaign's location is less specific than location
      continue

    for date in camp_dates.intersection(dates):
      booked_dict[date][location][sr_names] += daily_impressions

  # calculate inventory for each target and location on each date
  datekey = lambda dt: dt.strftime('%m/%d/%Y') if datestr else dt

ret = {}
for target in targets:
  name = make_target_name(target)
  subreddit_names = target.subreddit_names
  ret[name] = {}
  for date in dates:
    pageviews_by_location = {}
    for location in locations:
      # calculate available impressions for each location
      booked_by_target = booked_dict[date][location]
      pageviews_by_sr_name = pageviews_dict[location]
      pageviews_by_location[location] = get_maximized_pageviews( \
          subreddit_names, booked_by_target, pageviews_by_sr_name)
    # available pageviews is the minimum from all locations
    min_pageviews = min(pageviews_by_location.values())
    ret[name][datekey(date)] = max(0, min_pageviews)
```

4.2 Hard to read flow structure

The following code is really hard to read. First, it uses a really unusual control flow structure, the for-else structure. The Python documentation tell us : 'Loop statements may have an else clause; it is executed when the loop terminates through exhaustion of the list (with for) or when the condition becomes false (with while), but not when the loop is terminated by a break statement.'. The second trick, for understanding this code, is about the scope of variables in Python. The scope of a variable does not exist in `for`, `while` and `if` structures. So the scope of any variable is delimited by the scope of the function. So, bin still exists after the for-else structure.

Listing 4: `r2/lib/nymph.py` at lines 123-135

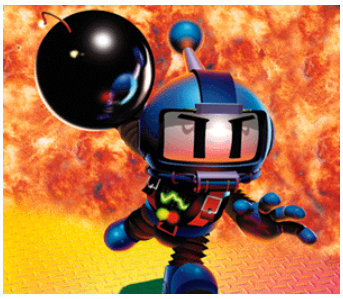
```
bins = []

for image in small_images:
  # find a bin to fit in
  for bin in bins:
    if bin.has_space_for(image):
      break
  else:
    # or give up and create a new bin
    bin = SpriteBin((0, sprite_height, sprite_width,\
                 sprite_height + image.height))
    sprite_height += image.height + SPRITE_PADDING
    bins.append(bin)
  bin.add_image(image)
```

Pylint sends us the following error for this part of the code 'Using possibly undefined loop variable bin'. This was a hard one.

5 Crazy if's

Pylint warned us about a case of "too many branches", and it was right. In the `r2/lib/pages/pages.py`, from line 460 to 661, there were 48 `if ... else`, in a single function. This shows that there is clearly a need to refactor and simplify this code, because adding comments would not make the code really more comprehensible.

6 Bomb



Listing 5: `r2/lib/utils/utils.py` at lines 1200-1215

```
class Hell(object):
    def __str__(self):
        return "boom!"

class Bomb(object):
    @classmethod
    def __getattr__(cls, key):
        raise Hell()

    @classmethod
    def __setattr__(cls, key, val):
        raise Hell()

    @classmethod
    def __repr__(cls):
        raise Hell()
```