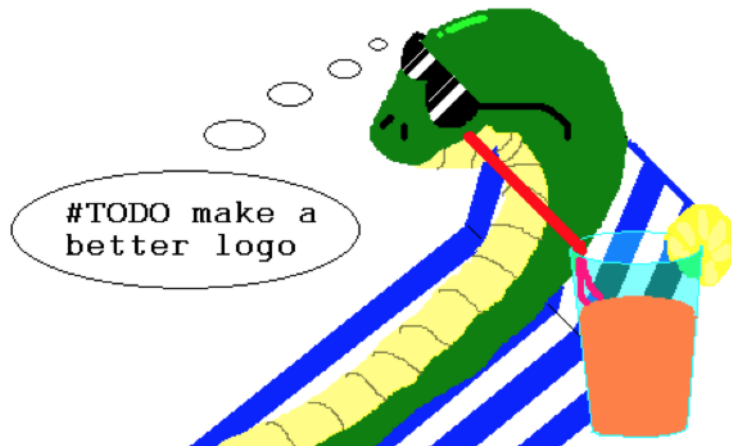


Pycrastinate

TODO less, DO more

by *Isaac Bernat*

May 20 - 2014



[github.com/
isaacbernat](https://github.com/isaacbernat)



PYCON
S W E D E N

Concepts

Pycrastiwhat!?

Find, list and analyse TODOs & os.

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Python version 3.10.0 (tags/v3.10.0:5086c62, Feb 22, 2022)

Why I hunt TODOs

HOWTO

Clone it, run it

git clone https://github.com/...

cd ..

cd pycrastinate

python3 setup.py install

python3 setup.py test

python3 setup.py clean

python3 setup.py build

python3 setup.py sdist

python3 setup.py bdist_wheel

python3 setup.py upload

python3 setup.py register

python3 setup.py upload_docs

python3 setup.py upload_source

python3 setup.py upload_binary

python3 setup.py upload_data

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

python3 setup.py upload_docs

config.py

Use Cases

Take control

Avoid bad practices

Code cleanup

Details

Error types

Exceptions in the pycrastinate package

• **MultiHypothesis** - no relation found (file?)

• **Type I** - incorrect rejection of a true file (file?)

• **Type II** - failure to reject a false file (file?)

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

• **Clone policy** - alpha (or a later) version

pycrastinate.py

Modules called through enclose

Initial results

Modules ordered by priority

Why use pycrastinate?

- **Consistency** - Use of a single Python version
- **Flexibility** - Customizable configuration
- **Modularity** - Customizable configuration
- **Speed** - Customizable configuration

Pycrastinate

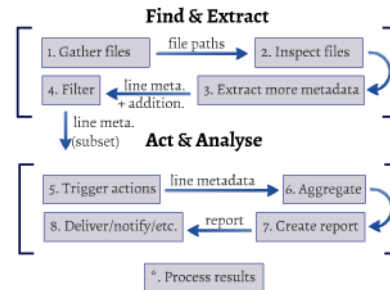
TODO less, DO more

Concepts

Pycrastiwhat!?

Find, act and analyse TODOs & co.

Pipeline: "chain of processing elements arranged so that the output of each element is the input of the next" - en.wikipedia.org



Why I hunt TODOs

Definition *WTF is a "TODO"?*
TODOs are jobs that the programmer thinks should be done, but for some reason can't do at the moment. [...] Whatever else a TODO might be, it is not an excuse to leave bad code in the system.

www.khan.com/Mark-Chan/.../todo.html

Analogy: What's a python function?

Example: Compute arithmetic mean of numbers

```
def mean(s):
    return sum(s)/len(s)
```

Modern C++: What You Need to Know - April 6, 2014 - Herb Sutter
source: <http://ericniebler.com/2014/04/06/lambda/>

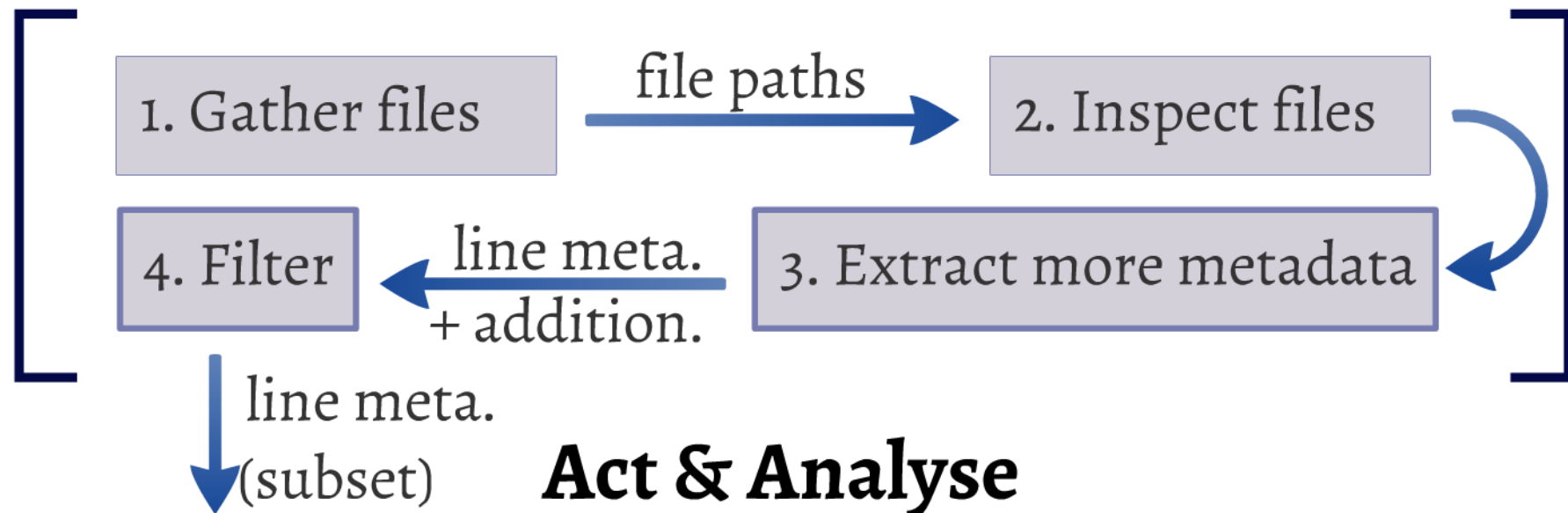
```
lambda s: sum(s)/len(s)
```

Pycrastin

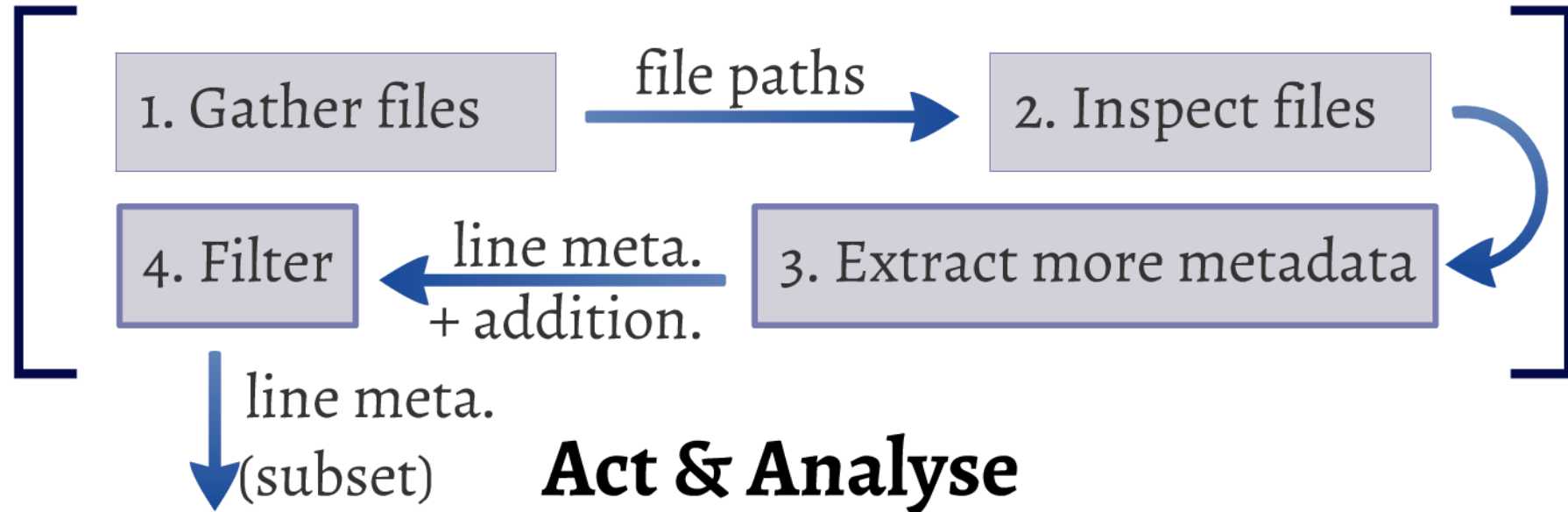
Find, act and analyse TODOs & co.

Pipeline: "chain of processing elements arranged so that the output of each element is the input of the next" - en.wikipedia.org

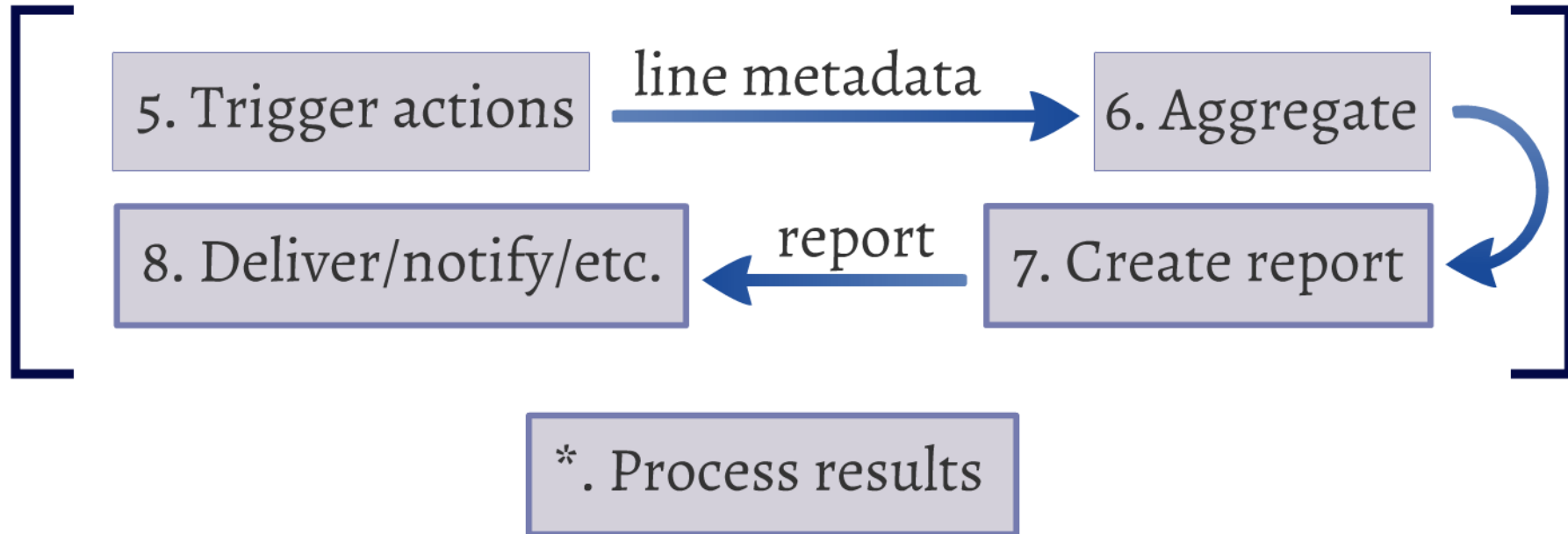
Find & Extract



Find & Extract



Act & Analyse



Definition

WTF is a "TODO"?

TODOs are jobs that the programmer thinks should be done, but for some reason can't do at the moment. [...]

*Whatever else a TODO might be, **it is not an excuse to leave bad code in the system.***

source: Robert C. Martin, "Clean Code – A Handbook of Agile Software Craftsmanship"

Analogy: What's a python function?

Example: Compute arithmetic mean of numbers

Python

```
def mean(seq):  
    n = 0.0  
    for x in seq:  
        n += x  
    return n / len(seq)
```

C++14 + concepts

```
auto mean(const Sequence& seq) {  
    auto n = 0.0;  
    for (auto x : seq)  
        n += x;  
    return n / seq.size();  
}
```

using a concept
(note: not yet VC++)

automatic return
type deduction

Modern C++: What You Need to Know - April 3, 2014 - Herb Sutter
source: <http://channel9.msdn.com/Events/Build/2014/2-661> (at 10:48)

Python

```
def mean(seq):  
    n = 0.0  
    for x in seq:  
        n += x  
    return n / len(seq)
```

C++14 + concepts

```
auto mean(const Sequence& seq) {  
    auto n = 0.0;  
    for (auto x : seq)  
        n += x;  
    return n / seq.size();  
}
```

using a concept
(note: not yet VC++)

automatic return
type deduction

Modern C++: What You Need to Know - April 3, 2014 - Herb Sutter
source: <http://channel9.msdn.com/Events/Build/2014/2-661> (at 10:48)

```
lambda s : sum(s) / len(s)
```


, run it

Try it out, tune it and master it!

```
(backend)localhost:pycrastinate ec$ python pycrastinate.py
```

```
=====
```

```
token > date > author > line > source
```

```
Generated at: 2014-05-09 23:11:47.779524
```

```
=====
```

```
FIXME
```

```
./TESTS/TEST_RAISE_IF_PRESENT.PY
```

```
FIXME 2014-03-09 bernat@wrapp.com 6 ./tests/test_raise_if_present.py {'c  
fixme 2014-03-09 bernat@wrapp.com 17 ./tests/test_raise_if_present.py "t  
fixme 2014-03-09 bernat@wrapp.com 32 ./tests/test_raise_if_present.py nt  
fixme 2014-03-09 bernat@wrapp.com 41 ./tests/test_raise_if_present.py nt
```

```
TODO
```

```
./TESTS/AUX_FILES/TEST_PYTHON.PY
```

```
TODO 2014-03-09 bernat@wrapp.com 1 ./tests/aux_files/test_python.py #TOI
```

```
...
```

```
pipeline = {  
    100: gather_files,  
    200: git_blames_from_files,  
    600: aggregate_by,  
    700: text_summary,  
    800: print_summary,  
    900: process_results,  
}
```

config.py

```
data = {
```

config.py

```
data = {  
    "gather_files": {  
        "root_paths": ["/"],  
        "file_sufixes": [".py"],},  
    "git_blames_from_files": {  
        "tokens": ["todo", "fixme"],  
        "case-sensitive": False,},  
    "aggregate_by": {  
        "keys": ["token", "file_path"],  
        "case-sensitive": False,},}
```



Use Cases

Take control

Avoid bad practices

Code cleanup

Take control over creeping FIXMEs

- 1) Search code for FIXMEs*
- 2) Filter >90 and <15 days old
- 3) Email report to assess severity

Add *exclude* and *send_email* modules

```
from datetime import date, timedelta
pipeline = dict(list(pipeline.items()) + [(400, exclude), (810, send_email)])
data = dict(list(data.items()) + list({
    "exclude": {
        "date": [{
            "values": [timedelta(15)],
            "functions": [lambda data, value: date.today() - data < value]
        }, {
            "values": [timedelta(90)],
            "functions": [lambda data, value: date.today() - data > value]
        }
    ], },
    "send_email": {
        "to": ["hello@email.com"], "from": "example@email.com",
        "username": "example@email.com", "password": "1234",
        "smtp_name": "smtp.email.com", }, }.items()))
```

```
"exclude": {
    "date": [{
        "values": [timedelta(15)],
        "functions": [lambda data, value: date.today() - data < value],
    }, {
        "values": [timedelta(90)],
        "functions": [lambda data, value: date.today() - data > value],
    }
], },
```

Avoid bad practices

breaking Continuous Integration builds

- 1) Search public API for *kwargs**
- 2) Filter >180 days old
- 3) Raise `AssertionError`

Add `raise_if_present` and `filter_by_age` modules

```
pipeline = dict(list(pipeline.items()) +
                [(400, filter_by_age),
                 (500, raise_if_present)])
data = dict(list(data.items()) + list({
    "raise_if_present": {
        "case-sensitive": False,
        "token": ["kwargs"]},
    "filter_by_age": {
        "oldest": 180, "earliest": -1,
    }, }.items()))
```

```
"filter_by_age": {
    "oldest": 180, "earliest": -1, }
```

Code cleanup

remove TODOs from an old stable codebase

- 1) Search for TODOs, XXXs, HACKs...
- 2) Filter out those <6 months old*
- 3) Email reports to committers with "1-click-distance" links

New pipeline: *gather_git_blames_shell*, *aggregate_by*, and *send_aggregated_email* modules

```
pipeline = {100: gather_git_blames_shell, 600: aggregate_by,
            610: send_aggregated_email, 900: process_results}
data = {"send_aggregated_email": {
    "cc": ["me@email.com"], "from": "me@example.com",
    "username": "me@example.com", "password": "1234",
    "smtp_name": "smtp.email.com", "concurrent": True,
    "render_function": html_summary,
    "html_summary": {"css": ["td{font-family: monospace}"]},
    }, "gather_git_blames_shell": {
    "init_path": "./", "tokens": ["todo", "xxx", "hack"],
    "file_sufixes": [".py", ".rb"],
    }, "aggregate_by": {
    "keys": ["email", "token", "file_path"]}}
```

```
data = {"send_aggregated_email": {
    "render_function": html_summary,
    "html_summary": {"css": ["td{font-family: monospace}"]},
    "cc": ["me@email.com"], "from": "me@example.com",
    "username": "me@example.com", "password": "1234",
    "smtp_name": "smtp.email.com", "concurrent": True,
```


Pycrastinate HTML report

Generated at: 2014-05-16 15:51:03.083690

TODO

CONFIG.PY

token	date	email	line_count	file_path	code
todo	2014-05-16	not.committed.yet	15	config.py#L15	"tokens": ["todo", "xxx", "hack"],

TESTS/AUX_FILES/TEST_PYTHON.PY

token	date	email	line_count	file_path	code
TODO	2014-03-09	bernat@wrapp.com	1	tests/aux_files/test_python.py#L1	#TODO this is a test

Details

Error types

Comments on file parser policies

- **Null hypothesis** no relation found (H_0)
- **Type I** incorrect rejection of a true H_0 (*false +*)
- **Type II** failure to reject a false H_0 (*false -*)
- **Chosen policy** $\alpha \geq 0, \beta = 0$ (like bloom filters)

source: http://en.wikipedia.org/wiki/Type_I_and_type_II_errors

pycrastinate.py

```
import sys
from collections import defaultdict
from functools import wraps

def pipeline(func):
    """Decorator to wrap a function with pipeline logic"""
    @wraps(func)
    def wrapper(*args, **kwargs):
        results = defaultdict(list)
        for item in func(*args, **kwargs):
            results[item].append(item)
        return results
    return wrapper

if __name__ == '__main__':
    pass
```

*Modules called
through enclose*

Initial results

```
reduce(lambda results, func:
        enclose(func, (config, results)),
        OrderedDict(sorted(pipeline.items())).values(), [])
```

Modules ordered by priority

Error types

Comments on file parser policies

- ***Null hypothesis*** no relation found (Ho)
- ***Type I*** incorrect rejection of a true Ho (*false +*)
- ***Type II*** failure to reject a false Ho (*false -*)
- ***Chosen policy*** $\alpha \geq 0$, $\beta = 0$ (like bloom filters)

source: http://en.wikipedia.org/wiki/Type_I_and_type_II_errors

pycrastinate

```
import config
from collections import OrderedDict
from functools import reduce

def run(pipeline=config.pipeline, config=config.data, enclose=config.enclose):
    return reduce(lambda results, func: enclose(func, (config, results)),
                  OrderedDict(sorted(pipeline.items())).values(), [])

if __name__ == "__main__":
    run()
```

*Modules called
through enclose*

Initial results

```
reduce(lambda results, func:
        enclose(func, (config, results)),
        OrderedDict(sorted(pipeline.items())).values(), [])
```

Modules ordered by priority

Why use pycrastinate?

a summary of key arguments

- **Convenience** Use it out of the box. No weird formats.
- **No setups** The only dependency is python 2.7/3.3+
- **Modularity** Easily extensible to meet your needs.
- **Speed** (django) <3.5 sec. >240k lines, >1.7k .py, >60 hits

found 2 TODOs from 2006 (!)



github.com/
isaacbernat



PYCON
S W E D E N

TODO: ask for questions

by *Isaac Bernat*

May 20 - 2014

Pycrastinate

TODO less, DO more

Enclose

```
from datetime import datetime

def print_log(func, params):
    log_func_info = "pycrastinate module: " + func.__name__
    print("{} start {}".format(datetime.now(), log_func_info))
    results = func(*params)
    print("{} finish {}".format(datetime.now(), log_func_info))
    return results

def no_closure(func, params):
    return func(*params)
```

gather_git_blames_shell

```
def get_blames_lines():
    include_sufixes = u' " --include "*{}"'.format(u' " --include "*"'.join(
        config["file_sufixes"])) if config["file_sufixes"] else ""

    case_sensitive = config.get("case-sensitive", False)
    insensitive = "" if case_sensitive else "-i"
    grep = 'grep -E -n {} "{}{}" -R {}'.format(
        insensitive, regex["raw_tokens_re"],
        include_sufixes, config["init_path"])
    cut = 'cut -d: -f1,2'
    awk = " ".join([
        "awk '{",
        'FS= ":";',
        'split($1, path, "/");',
        'paths="";',
        'fn=path[length(path)];',
        'for(i=0; i<length(path)-1; i++){paths=paths "/" path[i]};',
        'print "-C " paths "/ blame --line-porcelain -L" $2, " $2 " " fn',
        "}'"]
    )
    sed = 'sed s"/\\//\\//; s/\\:.*//"'
    xargs = 'xargs -P4 -n6 git'
    cat = 'cat'
    one_liner = " | ".join([grep, cut, awk, sed, xargs, cat])

    return subprocess.Popen(one_liner, stdout=subprocess.PIPE, shell=True)\
        .stdout.read().decode("utf-8").split("\n")
```



```

cut = 'cut -d: -f1,2'
awk = " ".join([
    "awk '{",
    'FS= ":";',
    'split($1, path, "/");',
    'paths="";',
    'fn=path[length(path)];',
    'for(i=0; i<length(path)-1; i++){paths=paths "/" path[i]};',
    'print "-C " paths "/ blame --line-porcelain -L" $2," $2 " " fn',
    "}'"]
)
sed = 'sed s"/\//\//\//; s/\:.*//"'
xargs = 'xargs -P4 -n6 git'
cat = 'cat'
one_liner = " | ".join([grep, cut, awk, sed, xargs, cat])

return subprocess.Popen(one_liner, stdout=subprocess.PIPE, shell=True)\
    .stdout.read().decode("utf-8").split("\n")

```

"one-liner"

```

grep -E -n -i "todo|fixme" --include "*.py" --include "*.rb" -R ./ |
cut -d: -f1,2 |
awk '{ FS= ":"; split($1, path, "/"); paths=""; fn=path[length(path)];\
    for(i=0; i<length(path)-1; i++){paths=paths "/" path[i]};\
    print "-C " paths "/ blame --line-porcelain -L" $2," $2 " " fn }' |
sed s"/\//\//\//; s/\:.*//"' |
xargs -P4 -n6 git |
cat

```