AZION

Python Anti-Patterns

What we should **NOT** do in our code

July, 2021

Vinicius Gubiani Ferreira

QA / Edge Services SWE

A little bit about myself

Vinícius Gubiani Ferreira

Summary

- Motivation
- Generic anti-patterns
- The Little Book of Python Anti-Patterns

1

Motivation

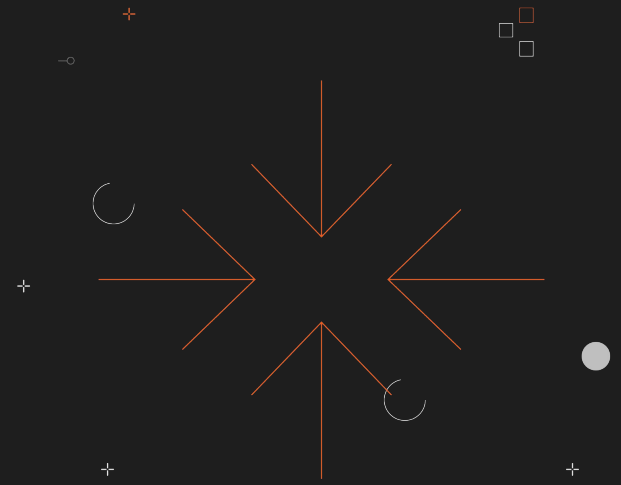
Help you reach the **next level**

2

Generic anti-patterns
(apply to any language)

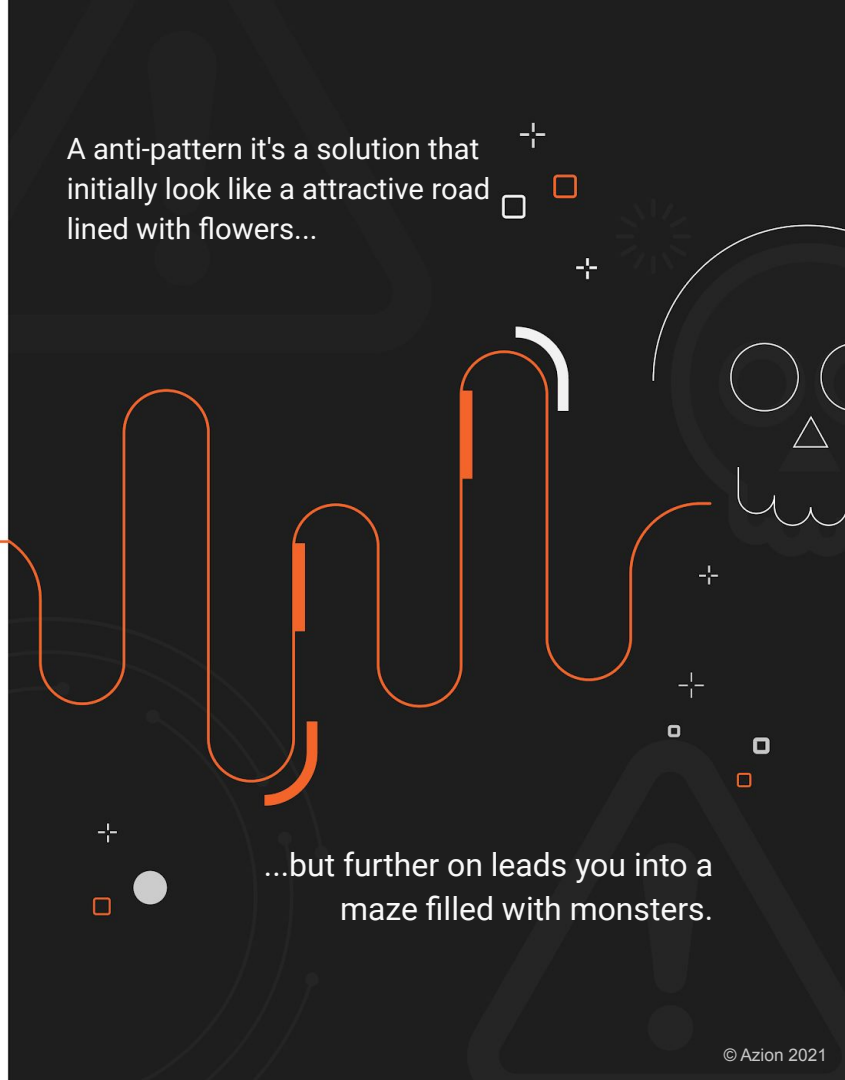
What exactly is a (design) pattern?

- Common solution to recurring problem;
- Happens at least 3 times, with different teams, without contact among them;
- Ends up being widely adopted;
- Convergence methodology;
- Reliable and effective;



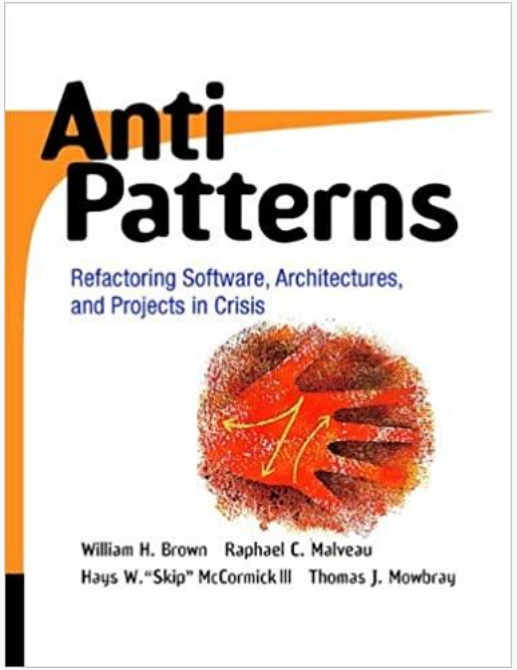
The AntiPattern on the other hand

- Looks great when we start ...
- until it's not anymore!
- Often causes more damage than the original problem itself;



Usually belong to 1 of 3 large categories

- Development;
- Architecture;
- Project Management;



A few AntiPatterns

Too many patterns to discuss in a 30 minute presentation, so we'll only discuss some of them, such as

- Boat anchor;
- Spaghetti code;
- God object;
- Vendor Lock-in;
- Cargo cult programming;
- Premature optimization;
- Magic numbers;
- Gold plating;

3

The Little Book of Python Anti-Patterns

Categories of the book

Correctness

Maintainability

Performance

Security

Readability

Migration

No exception type specified

Bad Code sample

```
1  try:
2      do_something()
3  except:
4      pass
```

Good Code sample

```
1  try:
2      do_something()
3  except ValueError:
4      logging.exception('Caught error')
5      pass
```

Ignore context managers to handle files

Bad Code sample

```
1  f = open("file.txt", "r")
2  content = f.read()
3  1 / 0
4  f.close()
```

Good Code sample

```
1  with open("file.txt", "r") as f:
2      content = f.read()
3      1 / 0
```

Return more than one variable type in function calls

Bad Code sample

```
1  def get_secret_code(password):
2      if password != "bicycle":
3          return None
4      return "42"
```

Good Code sample

```
1  def get_secret_code(password):
2      if password != "bicycle":
3          raise ValueError
4      return "42"
```

Accessing a protected member from outside the class

Bad Code sample

```
class Rectangle(object):
    def__init__(self, width, height):
        self._width = width
        self._height = heigh

tr = Rectangle(5, 6)
memberprint(
    "Width:{:d}".format(r._width)
)
```

Good Code sample

```
Make attributes part of the public
interface of the class (getters and
setters).
```

Assigning to built-in function

Very Bad Code sample

```
1  list = [1, 2, 3]
2  cars = list()
```

Good Code sample

```
1  numbers = [1, 2, 3]
2  cars = list()
```

Using tabs or mixing tabs with spaces

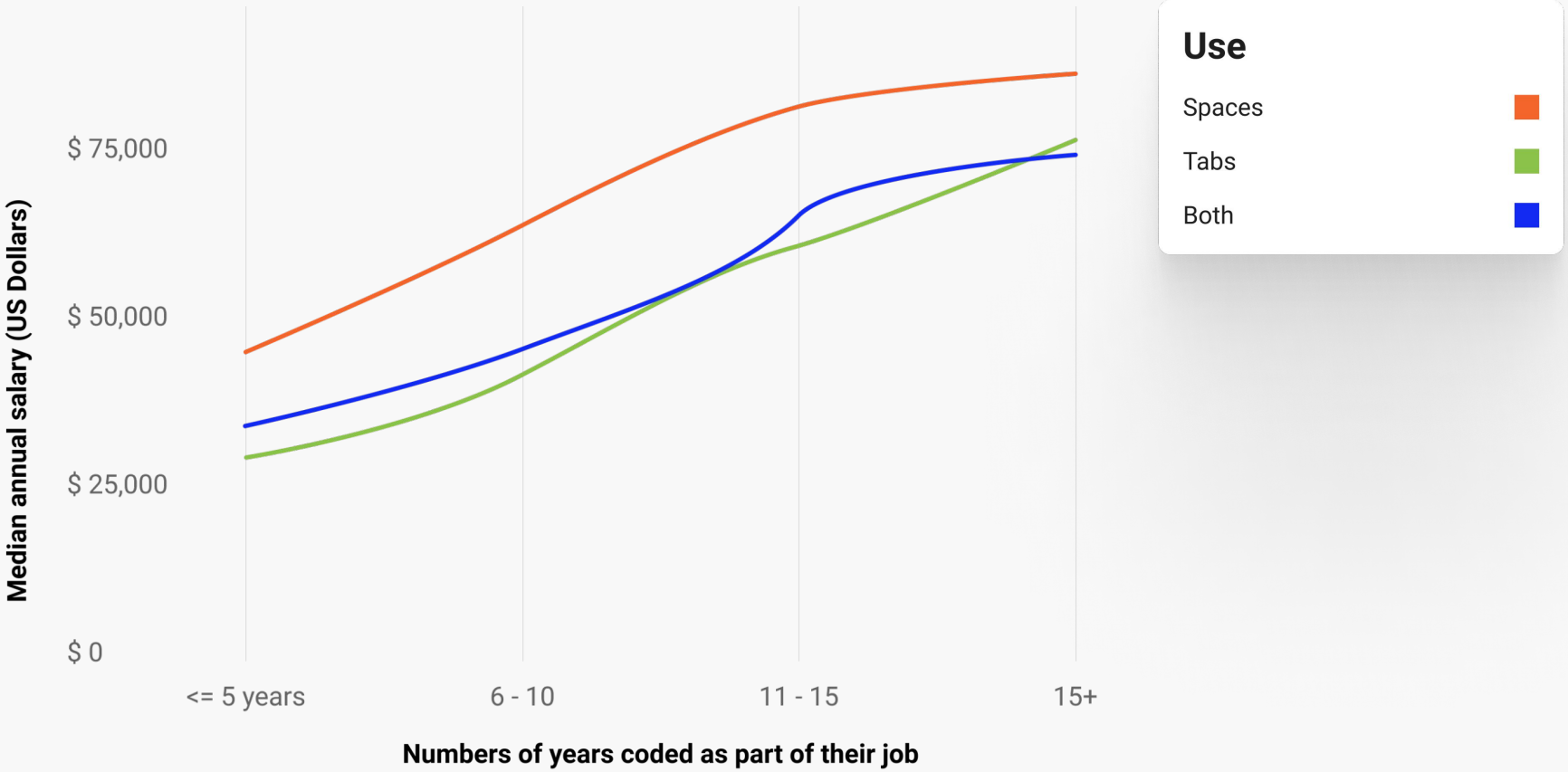
Bad sample

```
1	# Indentation with Tabs
```

Good Code sample

```
1    # Indentation with 4 spaces
```

Developers who use spaces make ***more*** money!



Source: Stackoverflow

Not using else where appropriate in a loop

Bad Code sample

```
1  my_list = [1, 2, 3]
2  magic_number = 4
3  found =False
4
5  for number in my_list:
6      if number == magic_number:
7          found =True
8          print("Magic number found")
9          break
10
11 if not found:
12     print("Magic number not found")
```

Good Code sample

```
1  my_list = [1, 2, 3]
2  magic_number = 4
3
4  for number in my_list:
5      if number == magic_number:
6          print("Magic number found")
7          break
8  else:
9      print("Magic number not found")
```

Not using get() to return a default value from a dict

Bad Code sample

```
1  dictionary = {"message": "Hello!"}
2  data = ""
3  if "message" in dictionary:
4      data = dictionary["message"]
5  print(data)
```

Good Code sample

```
1  dictionary = {"message": "Hello!"}
2  data = dictionary.get("message", "")
3  print(data)
```

Using wildcard imports

Bad sample

```
1  from math import *
```

Good Code sample

```
1  from math import ceil
```

Using the global statement

Bad Code sample

```
1  WIDTH = 0
2  HEIGHT = 0
3
4  def area(w, h):
5      global WIDTH
6      global HEIGHT
7      WIDTH = w
8      HEIGHT = h
9      return WIDTH * HEIGHT
```

Good Code sample

```
1  class Rectangle:
2      def __init__(self, width, height):
3          self.width = width
4          self.height = height
5      def area(self):
6          return self.width * self.height
```

Using single letter to name your variables

Very Bad sample

```
1  l = [1, 2, 3, 4, 5]
```

Good sample

```
1  car_ids = [1, 2, 3, 4, 5];
```

Comparing things to True the wrong way

Bad Code sample

```
1  flag = True
2  if flag == True:
3      print("This works!")
```

Good Code sample

```
1  flag = True
2  if flag:
3      print("This works!")
4
5  flag = True
6  if flag is True:
7      print("This works!")
```

Code*Tip*

```
>>> 1 is True
False

>>> 1 == True
True

>>> 1.0 == True
True

>>> -1 == True
False


>>> True is True
True
```

```
>>> id(1)
4495956272

>>> id(True)
4495126176

>>> id(1.0)
4497188208

>>> id(-1)
4495956208
```

Using type() to compare types

Bad Code sample

```
c = Circle(2)
r = Rectangle(3, 4)
if type(r) is not type(c):
    print("object types do not match")
```

Good Code sample

```
r = Rectangle(3, 4)
if isinstance(r, types.ListType):
    print("object r is a list")
```

Not using named tuples in function return

Bad Code sample

```
def get_name():
    return "Richard", "Jones"
name = get_name()
# no idea what these indexes map to!
print(name[0], name[1])
```

Good Code sample

```
from collections import namedtuple
def get_name():
    name = namedtuple(
        "name", ["first", "last"]
    )
    return name("Richard", "Jones")
name = get_name()
print(name.first, name.last)
```

AZION

References

- AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis
- https://sahandsaba.com/nine-anti-patterns-every-programmer-should-be-aware-of-with-examples.html
- https://martinfowler.com/bliki/AntiPattern.html
- https://sourcemaking.com/antipatterns
- https://en.wikipedia.org/wiki/Anti-pattern
- https://stackoverflow.blog/2017/06/15/developers-use-spaces-make-money-use-tabs/
- https://realpython.com/the-most-diabolical-python-antipattern/
- https://deepsource.io/blog/8-new-python-antipatterns/
- https://raw.githubusercontent.com/quantifiedcode/python-anti-patterns/master/docs/The-Little-Book-Of-Python-Anti-Patterns.pdf

AZION

Thank you

Obrigado

Gracias

Vielen Dank

Спасибо

谢谢啦

ありがと

AZION

Have a question?

Please contact me

 **vinigfer**

 **vinigfer**

 **vinicius-gubiani-ferreira**

 **vini.g.fer@gmail.com**