

Testes unitários como ferramentas de design de código

...

I GruPy Grande Rio - 2 de abril de 2016

Paula Grangeiro

Programadora por profissão,
desenhista nas horas vagas e
coleccionadora de gatos.



Sobre mim

**Os fatos relatados durante esta apresentação são
baseados em experiências pessoais.
Utilize-os com moderação.**

Ministério da procrastinação adverte:

**Por que pensar em
Design de Código?**

```
832
833 [] def get_report_overall(self, filter_args):
834     from collections import Counter
835
836     objEleitor = Eleitor.objects.filter(**filter_args)
837
838     graph_data = {}
839     graph_data['temp_count'] = objEleitor.count()
840
841     graph_data['sexo'] = Counter(objEleitor.values_list('sexo', flat=True))
842     graph_data['sexo'] = {
843         'null': (
844             float(graph_data['sexo'][None]) / graph_data['temp_count']
845         ) * 100,
846         'F': (
847             float(graph_data['sexo']['F']) / graph_data['temp_count']
848         ) * 100,
849         'M': (
850             float(graph_data['sexo']['M']) / graph_data['temp_count']
851         ) * 100,
852     }
853
854     temp_escolaridade = Counter(
855         objEleitor.exclude(
856             escolaridade__isnull=True
857         ).values_list('escolaridade', flat=True)
858     )
859     count_escolaridade = len(
860         Eleitor.objects.filter(
861             **filter_args
862         ).values('escolaridade').exclude(escolaridade__isnull=True)
863     )
864
865     if count_escolaridade != 0:
866         graph_data['escolaridade'] = {
867 +----- 11 linhas: 1: (float(temp_escolaridade[1]) / count_escolaridade) * 100,-----
878         }
879     else:
880 +---- 14 linhas: graph_data['escolaridade'] = {-----
894
895     graph_data['aprova_o_governo'] = Counter(
896         objEleitor.values_list('aprova_o_governo', flat=True)
```

custom_modelmanager.py

833:3[Syntax: line:1 (26)]

4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

12%

```
894  
895     graph_data['aprova_o_governo'] = Counter(
896         objEleitor.values_list('aprova_o_governo', flat=True)
897     )
898
899     graph_data['aprova_o_governo'] = {
900 +---- 35 linhas: 2: (-----
901         }
902
903     temp_graph_data = Counter(objEleitor.values_list('idade', flat=True))
904
905     idades = dict(
906 +---- 9 linhas: dict.fromkeys(-----
907         )
908
909     for item in range(15, 25):
910         idades['idade16_24'] += temp_graph_data[item]
911
912     for item in range(25, 35):
913         idades['idade25_34'] += temp_graph_data[item]
914
915     for item in range(35, 45):
916         idades['idade35_44'] += temp_graph_data[item]
917
918     for item in range(45, 60):
919         idades['idade45_59'] += temp_graph_data[item]
920
921     for item in range(60, 100):
922         idades['idade60'] += temp_graph_data[item]
923
924     count_idade = sum(idades.values())
925
926     graph_data['idade'] = {
927 +---- 6 linhas: 0: (float(idades['idade16_24']) / count_idade) * 100,-----
928         }
929
930     temp_prioridade = Counter(objEleitor.values_list(
931         'qual_a_sua_prioridade', flat=True)
932     )
933
934     prioridade = dict(dict.fromkeys(
935         ['agua', 'asfalto', 'educacao', 'emprego', 'saude', 'seguranca'], 0
936     ))
```

custom_modelmanager.py

894:0[Syntax: line:1 (26)]

4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

12%

```
983     ))
984
985     prioridade['agua'] = (temp_prioridade[u'ÁGUA'] + temp_prioridade[u'AGUA'] + temp_prioridade[u'Água'] + temp_prioridade[u'água'] + temp_prioridade[u'agua'])
986     prioridade['asfalto'] = (temp_prioridade[u'ASFALTO'] + temp_prioridade[u'Asfalto'] + temp_prioridade[u'asfalto'])
987     prioridade['educacao'] = (temp_prioridade[u'EDUCAÇÃO'] + temp_prioridade[u'EDUCACAO'] + temp_prioridade[u'Educação'] + temp_prioridade[u'Educacao'] + temp_
prioridade[u'educação'] + temp_prioridade[u'educacao'])
988     prioridade['emprego'] = (temp_prioridade[u'EMPREGO'] + temp_prioridade[u'Emprego'] + temp_prioridade[u'emprego'])
989     prioridade['saude'] = (temp_prioridade[u'SAUDE'] + temp_prioridade[u'SAUDE'] + temp_prioridade[u'Saúde'] + temp_prioridade[u'Saude'] + temp_prioridade[u'sa
úde'] + temp_prioridade[u'saude'])
990     prioridade['seguranca'] = (temp_prioridade[u'SEGURANÇA'] + temp_prioridade[u'SEGURANCA'] + temp_prioridade[u'Segurança'] + temp_prioridade[u'Seguranca'] +
temp_prioridade[u'segurança'] + temp_prioridade[u'seguranca'])
991
992     count_prioridade = sum(prioridade.values())
993
994     graph_data['prioridade'] = {
995 +---- 7 linhas: 0: (float(prioridade['agua']) / count_prioridade) * 100,-----
1002     }
1003
1004
S>1005     temp_contato = {
1006 +---- 6 linhas: 0: objEleitor.values_list('numero_1', flat=True).count(),-----
1012     }
1013
1014     count_contato = (
1015 +---- 7 linhas: float(objEleitor.values_list('numero_1', flat=True).exclude(numero_1_isnull=True).count() +-----
1022     )
1023
1024     graph_data['telefone'] = {
1025 +---- 7 linhas: 0: float(float(temp_contato[0]) / count_contato * 100),-----
1032     }
1033
1034     email_informed = len(objEleitor.values_list('email', flat=True).exclude(email_isnull=False))
1035     email_uninformed = len(objEleitor.values_list('email', flat=True).exclude(email_isnull=True))
1036
1037     graph_data['email'] = {
1038 +---- 2 linhas: 0: float((float(email_uninformed) / graph_data['temp_count']) * 100),-----
1040     }
1041
1042     objRenda = objEleitor.values_list('renda_mensal_presumida', flat=True)
1043
1044     renda = dict(dict.fromkeys(['renda_mais_1356', 'renda_mais_2034', 'renda_mais_3390', 'renda_nao_informada'], 0))
1045
```

custom_modelmanager.py


983:3[Syntax: line:1 (26)]

4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

12%


```
1004 
S>1005     temp_contato = {
1006 +---- 6 linhas: 0: objEleitor.values_list('numero_1', flat=True).count(),-----
1012     }
1013
1014     count_contato = (
1015 +---- 7 linhas: float(objEleitor.values_list('numero_1', flat=True).exclude(numero_1_isnull=True).count() +-----
1022     )
1023
1024     graph_data['telefone'] = {
1025 +---- 7 linhas: 0: float(float(temp_contato[0]) / count_contato * 100),-----
1032     }
1033
1034     email_informed = len(objEleitor.values_list('email', flat=True).exclude(email_isnull=False))
1035     email_uninformed = len(objEleitor.values_list('email', flat=True).exclude(email_isnull=True))
1036
1037     graph_data['email'] = {
1038 +---- 2 linhas: 0: float((float(email_uninformed) / graph_data['temp_count']) * 100),-----
1040     }
1041
1042     objRenda = objEleitor.values_list('renda_mensal_presumida', flat=True)
1043
1044     renda = dict(dict.fromkeys(['renda_mais_1356', 'renda_mais_2034', 'renda_mais_3390', 'renda_nao_informada'], 0))
1045
1046     for item in range(len(objRenda)):
1047 +---- 8 linhas: if objRenda[item] > 1356.00 and objRenda[item] < 2033.99:-----
1055
1056     count_renda = sum(renda.values())
1057
1058     graph_data['renda'] = {
1059 +---- 5 linhas: 0: (float(renda['renda_nao_informada']) / count_renda * 100,-----
1064     }
1065
1066     return graph_data
```

custom_modelmanager.py

1004:0[Syntax: line:1 (26)]

4 custom_modelmanager.py|145 col 13 error| continuation line missing indentation or outdented [E122]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

12%



Código Mogwai

“Escrever código limpo é o que você deve fazer para que possa se intitular como profissional. Não existem desculpas plausíveis para fazer menos do que o seu melhor.” - Uncle Bob em Código Limpo

Testes unitários


Pequena função Python com dois
possíveis fluxos a partir da
validação do argumento

```
def foo(arg):  
    if arg:  
        return 'It has an argument!'  
    return 'Nope!'
```

Testes unitários que cobrem os possíveis fluxos da função *foo*

```
class FooTestCase(TestCase):  
    def test_foo_returns_correct_value_when_arg_is_true(self):  
        message = foo(True)  
        self.assertEqual('It has an argument!', message)  
    def test_foo_returns_correct_value_when_arg_is_false(self):  
        message = foo(False)  
        self.assertEqual('Nope!', message)
```

```
def foo(arg):  
    if arg:  
        return 'It has an argument!'  
    return 'Nope!'
```



```
class FooTestCase(TestCase):  
    def test_foo_returns_correct_value_when_arg_is_true(self):  
        message = foo(True)  
        self.assertEqual('It has an argument!', message)  
    def test_foo_returns_correct_value_when_arg_is_false(self):  
        message = foo(False)  
        self.assertEqual('Nope!', message)
```

```
>> 1 class CustomTestCase(TestCase):
>> 2     def setUp(self):
>> 3         print('Initializing Graph test case')
>> 4         usuarioTeste = coreModels.Usuario.objects.create(
>> 5             username='teste',
>> 6             nome='nome',
>> 7         )
>> 8
>> 9         campanhaTeste = coreModels.Campaign.objects.create(
>>10             name='CampanhaTeste',
>>11             description='Description',
>>12             start_date=datetime.datetime(2014, 01, 01),
>>13             end_date=datetime.datetime(2014, 01, 01),
>>14             coordinator=usuarioTeste,
>>15             manager=usuarioTeste,
>>16         )
>>17
>>18         perfilTeste = coreModels.Profile.objects.create(
>>19             name='perfilTeste',
>>20             description='Description',
>>21             contact_type=1,
>>22             goal=100,
>>23             profile_type=1,
>>24             target_filter={}
>>25         )
>>26
>>27         taskTeste = coreModels.Task.objects.create(
>>28             name='Task Teste',
>>29             goal=100,
>>30             description='',
>>31             contact_type=1,
>>32             task_type=1,
>>33             start_date=datetime.datetime(2014, 01, 01),
>>34             end_date=datetime.datetime(2014, 01, 01),
>>35             operator=usuarioTeste,
>>36             coordinator=usuarioTeste,
>>37             campaign=campanhaTeste,
>>38             profile=perfilTeste,
>>39             status=1,
>>40             visible_fields={},
>>41             fields={},
```

custom_modelmanager_test.py

41:6[Syntax: line:1 (112)]

4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]

5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

2%


```
40     visible_fields={},
41     fields={},
42 )
43 eleitorTesteM = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste', sexo="M")
44 eleitorTesteF = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste', sexo="F")
45 eleitorTesteNull = coreModels.Eleitor.objects.create(nome_completo='Eleitor Teste')
46
47 def test_gender_graph_data_two_days_same_distribution_m_and_f(self):
48     #Required data
49     eleitorTesteM = coreModels.Eleitor.objects.filter(sexo="M")[0]
50     eleitorTesteF = coreModels.Eleitor.objects.filter(sexo="F")[0]
51     taskTeste = coreModels.Task.objects.all()[0]
52
53     testeM = coreModels.TaskResult.objects.create(
54         elector = eleitorTesteM,
55         task = taskTeste,
56         done_at = datetime.date(2014, 01, 01),
57         status = 1
58     )
59     #unfortunally done at is auto_now_add so we need to do this litle hack
60     testeM.done_at = datetime.date(2014, 01, 01)
61     testeM.save()
62
63     testeF = coreModels.TaskResult.objects.create(
64         elector = eleitorTesteF,
65         task = taskTeste,
66         done_at = datetime.date(2014, 01, 01),
67         status = 1
68     )
69     #unfortunally done at is auto_now_add so we need to do this litle hack
70     testeF.done_at = datetime.date(2014, 01, 01)
71     testeF.save()
72
73     start_date = datetime.datetime(2014, 01, 01, 0, 0, 0)
74     end_date = datetime.datetime(2014, 01, 02, 23, 59, 59)
75     lookup_field = 'elector__sexo'
76
77     result = coreViews.generateGraphData(start_date, end_date, lookup_field)
78
79     self.assertEqual(result, [[["2014-01-01", 50.0], ["2014-01-02", 50.0]], [{"2014-01-01", 50.0}, {"2014-01-02", 50.0}], [{"2014-01-01", 0.0}, {"2014-01-02", 0.0}]])
```

custom_modelmanager_test.py

41:6[Syntax: line:1 (112)]

4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]

5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

2%

```
79 self.assertEqual(result, [[["2014-01-01", 50.0], ["2014-01-02", 50.0]], [{"2014-01-01", 50.0}, {"2014-01-02", 50.0}], [{"2014-01-01", 0.0}, {"2014-01-02", 0.0}]]
80
81 def test_gender_graph_two_days_same_distribution_m_and_f_and_null(self):
82     #Required data
83     eleitorTesteM = coreModels.Eleitor.objects.filter(sexo="M")[0]
84     eleitorTesteF = coreModels.Eleitor.objects.filter(sexo="F")[0]
85     eleitorTesteNull = coreModels.Eleitor.objects.filter(sexo__isnull=True)[0]
86
87     taskTeste = coreModels.Task.objects.all()[0]
88
89     #Elector day 1 Masc
90     test1m = coreModels.TaskResult.objects.create(
91         eleitor = eleitorTesteM,
92         task = taskTeste,
93         done_at = datetime.date(2014, 01, 01),
94         status = 1
95     )
96     #unfortunatly done at is auto_now_add so we need to do this little hack
97     test1m.done_at = datetime.date(2014, 01, 01)
98     test1m.save()
99
100     #Elector day 1 Fem
101     test1f = coreModels.TaskResult.objects.create(
102         eleitor = eleitorTesteF,
103         task = taskTeste,
104         done_at = datetime.date(2014, 01, 01),
105         status = 1
106     )
107     #unfortunatly done at is auto_now_add so we need to do this little hack
108     test1f.done_at = datetime.date(2014, 01, 01)
109     test1f.save()
110
111     #Elector day 1 Null
112     test1Null = coreModels.TaskResult.objects.create(
113         eleitor = eleitorTesteNull,
114         task = taskTeste,
115         done_at = datetime.date(2014, 01, 01),
116         status = 1
117     )
118     #unfortunatly done at is auto_now_add so we need to do this little hack
119     test1Null.done_at = datetime.date(2014, 01, 01)
```

custom_modelmanager_test.py

79:6[Syntax: line:1 (112)]

4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]

5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

2%

whitespace after '[' [E201]

```
S> 94         status = 1
95     )
96     #unfortunatly done at is auto_now_add so we need to do this litle hack
>> 97     test1m.done_at = datetime.date(2014, 01, 01)
98     test1m.save()
99
100     #Elector day 1 Fem
>> 101     test1f = coreModels.TaskResult.objects.create(
S> 102         elector = eleitorTesteF,
S> 103         task = taskTeste,
S> 104         done_at = datetime.date(2014, 01, 01),
S> 105         status = 1
106     )
107     #unfortunatly done at is auto_now_add so we need to do this litle hack
>> 108     test1f.done_at = datetime.date(2014, 01, 01)
109     test1f.save()
110
111     #Elector day 1 Null
>> 112     test1Null = coreModels.TaskResult.objects.create(
S> 113         elector = eleitorTesteNull,
S> 114         task = taskTeste,
S> 115         done_at = datetime.date(2014, 01, 01),
S> 116         status = 1
117     )
118     #unfortunatly done at is auto_now_add so we need to do this litle hack
>> 119     test1Null.done_at = datetime.date(2014, 01, 01)
120     test1Null.save()
121
122     start_date = datetime.datetime(2014, 01, 01, 0, 0, 0)
>> 123     end_date = datetime.datetime(2014, 01, 02, 23, 59, 59)
124     lookup_field = 'elector__sexo'
125
>> 126     result = coreViews.generateGraphData(start_date, end_date, lookup_field)
127     print result
S> 128     self.assertEqual(result, [[["2014-01-01", 33.33], ["2014-01-02", 33.33]], [{"2014-01-01", 33.33}, {"2014-01-02", 33.33}], [{"2014-01-01", 33.33}], [{"2014-01-02", 33.33}]]])
S> 129
```

custom_modelmanager_test.py

109:6[Syntax: line:1 (112)]

4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]

5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]

[Lista de locais] :SyntasticCheck flake8 (python)

4,1

2%

```
custom_modelmanager_test.py 109:6[Syntax: line:1 (112)]
4 custom_modelmanager_test.py|12 col 24 error| undefined name 'datetime' [F821]
5 custom_modelmanager_test.py|13 col 22 error| undefined name 'datetime' [F821]
[Lista de locais]: SyntasticCheck flake8 (python) 4,1 2%
```



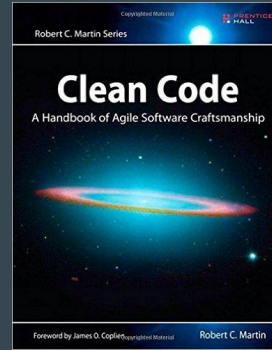
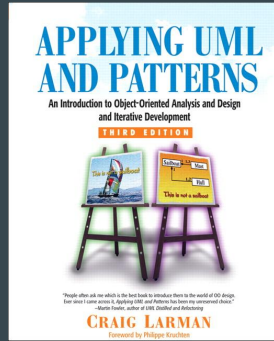
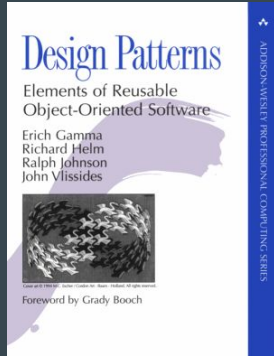
Então o que há de errado com o nosso código?

Padrões de Projeto

Design Patterns

Padrões de Projeto

- GoF
- GRASP
- SOLID



- Padrões de Projeto - Soluções Reutilizáveis de Software Orientado a Objetos
- Utilizando UML e Padrões
- Código limpo
- Curso Python Patterns - Luciano Ramalho

Indicação de estudo

Testes unitários como ferramenta de design

```
def create_income_tax(income):
    tax = 0
    if income < 25661.7:
        '''
            codigo calculo do imposto isento
        '''
    else:
        '''
            codigo calculo do imposto
        '''
    f = open('income_tax_file', 'w')
    f.write(tax)
    f.close()
```

```
class CrateIncomeTaxTestCase(TestCase):
    def
    test_method_calcs_exemption_corrctly(self):
        '''
            teste para rendimento < 25661.7
        '''
    def test_method_calcs_taxes_correctly(self):
        '''
            teste para rendimento > 25661.7
        '''
    def test_method_creates_file_correctly(self):
        '''
            teste que valida criacao do arquivo
        '''
```

Teste unitariamente o seu código

```
def create_income_tax(rendimento):  
    tax = calculate_tax(rendimento)  
    income_tax_file = create_file(valor_imposto)
```

Teste unitariamente o seu código

```
class CreateIncomeTaxTestCase(TestCase):  
    def test_method_calls_calculate_tax(self):  
        """  
        teste para chamada do calcular imposto  
        """  
  
    def test_method_calls_create_file(self):  
        """  
        teste para chamada do criar arquivo  
        """  
  
class CalculateTaxTestCase(TestCase):  
    def  
test_method_calcs_exemption_correctly(self):  
    """  
        teste para rendimento < 25661.7  
    """  
  
    def test_method_calcs_taxes_correctly(self):  
    """  
        teste para rendimento >= 25661.7  
    """  
  
class CreateFileTestCase(TestCase):  
    def test_method_creates_file_correctly(self):  
    """  
        teste que valida criacao do arquivo  
    """  
    _____
```

```
class PatientTestCase(TestCase):

    def test_save_patient_correctly(self):
        """
        código que testa se o paciente foi salvo corretamente
        """

    def test_save_raises_exception_if_patient_with_same_cpf_already_exists(self):
        """
        código que testa se foi levantada excecao quando paciente com cpf ja existe
        """

        patient_1 = Patient(cpf='12121212108')
        patient_1.save()

        patient_2 = Patient(cpf='12121212108')
        self.assertRaises(PatientAlreadyExistsException, patient_2.save)
```

Teste fluxos de sucesso e erro separadamente

```
class CalculateTaxTestCase(TestCase):

    def test_method_calcs_taxes_when_income_less_than_margin(self):
        expected_value = 0
        income = 25000

        taxes = calculate_tax(income)

        self.assertEqual(expected_value, taxes)

    def test_method_calcs_taxes_when_income_equal_or_greater_than_margin(self):
        expected_value = 432.6
        income = 26000

        taxes = calculate_tax(income)

        self.assertEqual(expected_value, taxes)
```

Nomeie os testes de acordo com o fluxo testado

```
from mock import patch
from unittest import TestCase

from diretorio.arquivo import create_income_tax

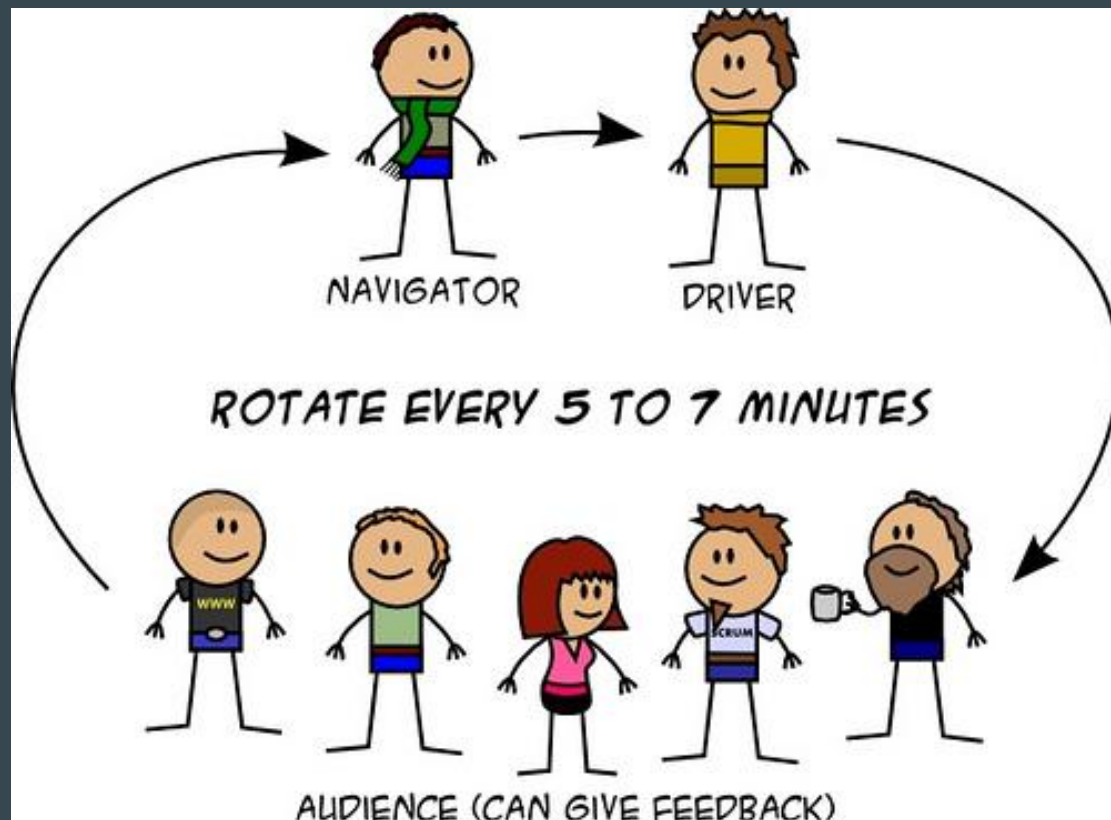
class CreateIncomeTaxTestCase(TestCase):

    @patch('diretorio.arquivo.calculate_tax')
    def test_method_calls_calculate_tax(self, mocked_calculate_tax):
        create_income_tax(1)
        self.assertTrue(mocked_calculate_tax.called)

    @patch('diretorio.arquivo.create_file')
    def test_method_calls_create_file(self, mocked_create_file):
        create_income_tax(1)
        self.assertTrue(mocked_create_file.called)
```

Testes devem ser isolados!

TDD



Funcionamento do Dojo

Dojos no Rio

- DTM todas as quartas
- Colworking todas as segundas
- dojo-rio@googlegroups.com

Obrigada!

<http://www.paulagrangoiro.com.br>

<https://twitter.com/paulagrangoiro>

<http://fb.me/paula.grangoiro>

<https://github.com/pgrangoiro>

pgrangoiro.dev@gmail.com