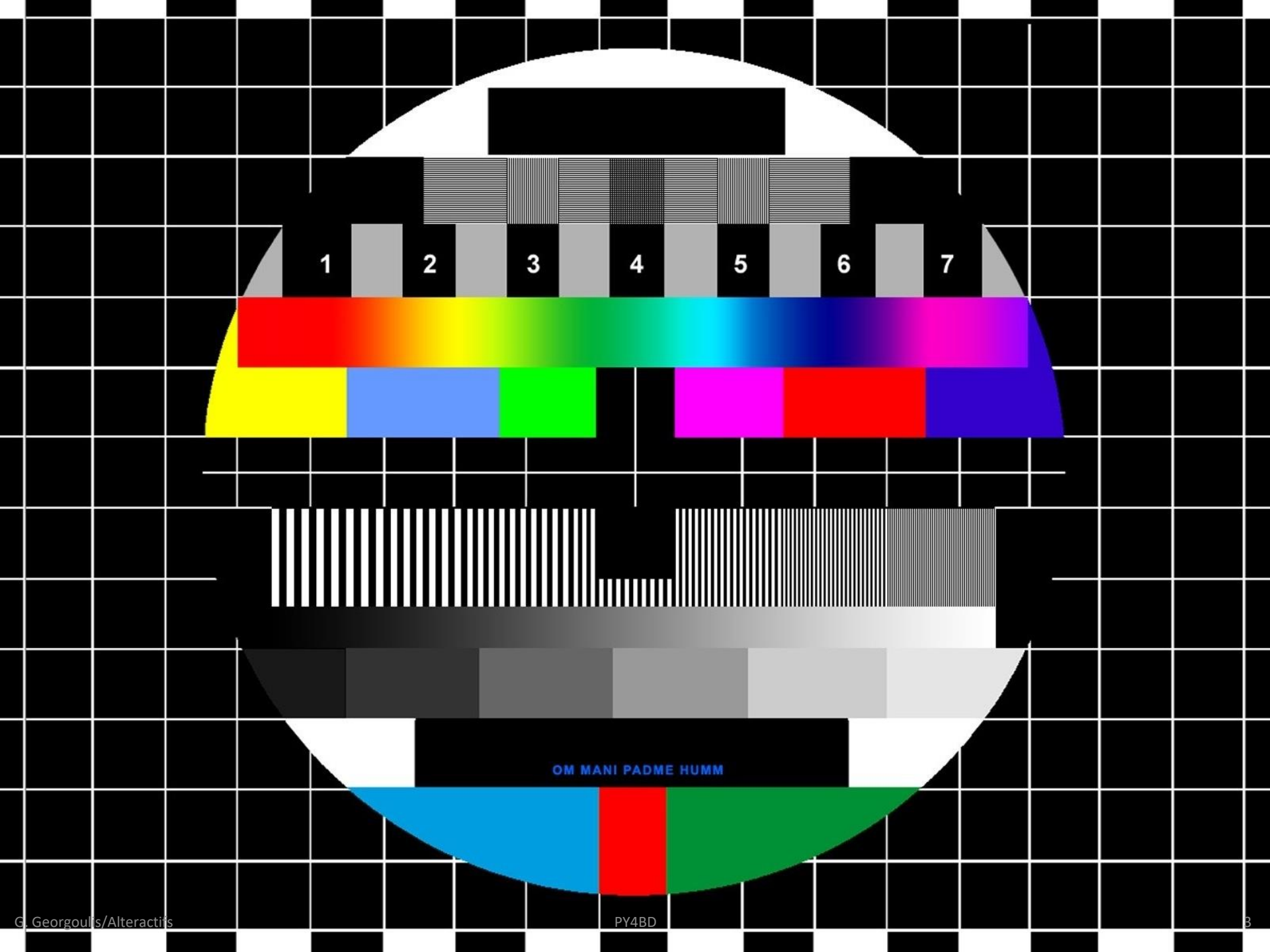
j8

J8: Objectifs

- Tests
- PEP8


1
2
3
4
5
6
7
OM MANI PADME HUMM

Tests

- Ce qui n'est pas testé, ne marche pas !
- Tests écrits en même temps que le code
- Développement gouverné par les tests
  Test Driven Development, TDD

la méthode manuelle

- Insérer des tests

```
assert f(2,7) == 3, "Devrait faire 3"
```

unittest

- Tests unitaires, *unit testing*
  - composants, testés séparément
    - une classe , une méthode, une fonction
- Mettre les tests dans des cellules spécifiques d'un classeur jupyter

```
import unittest

class TestThing(unittest.TestCase):
    def test_thing1(self):
        self.assertEqual(1+2,3)
    def test_thing2(self):
        self.assertTrue(2+2==4)

res = unittest.main(argv=[''],
     exit=False)
```

doctest

- Inclure des commentaires
  - commandes de l'interpréteur
  - sortie de l'interpréteur
- Le test consiste en une comparaison du résultat du calcul avec la chaîne doctest, caractère par caractère
- -v pour la version bavarde (*verbose*)
- Test possible dans une cellule d'un classeur Jupyter en incluant
- Tests réunis dans un fichier texte séparé

```
def mul(a,b):
    """
    >>>mul(3,7)
    21
    >>>mul('bla',3)
    'blablabla'
    """
    return a*b
```

```
$ python3 –m doctest mon_script.py
```

```
import doctest
doctest.testmod()
```

```
$ python3 –m doctest mon_script.py
    -v tests.txt
```

PEP-8 -- Style Guide for Python Code

Recommandations de présentation

- Tabulations
- Longueur de ligne
- Indentation
- Ligne blanches
- Chaines quotées
- Espaces
- Virgule finale
- Anglais
- Commentaires
- Conventions de nommage
  - voir plus loin

Recommandations de codage

- in
- is /is not
- expressions booléennes
- fonctions de chaînes
- with
- hiérarchie des exceptions
- cohérence des returns
- annotations des fonctions et des variables par des indications de types

Conventions de nommage

Noms	ASCII pur et pas d'accents
Package, modules	minuscules et courts
Classes	CapWords
Exceptions	nom qui termine en "Error"
Variables et fonctions	minuscules et underscore _
Arguments	self, cls, résoudre les collisions par ajout de _
Méthodes	comme les fonctions , ajouts de _, __, __ __
Constantes	tout en majuscules

pylint

- Installer pylint,
  - le lint de Python

```
$ python3 –m venv pylintenv
$ source pylintenv/bin/activate
$ pip install pylint
```

- Question : Qu'est ce que lint ?

- Passer pylint sur un de vos scripts validés et éliminer l'un après l'autre tous les messages d'erreur !

```
$ pylint monscript.py
```

- pylint vérifie la PEP-8
- Les AGL (IDE) Python intègrent une telle fonctionnalité.
  - C'est le cas de pyCharm (et +)

Exercice pylint

flake8

- Installer

```
$ pip install flake8
```

- Utiliser

```
$ flake8 mon_script.py
```

- Configurer via un fichier
  ~./flake8 ou bien
  ~./config/flake8

- Paramètres
  - règles à ignorer
  - fichiers à exclure
  - complexité au sens de McCabe (nombre cyclomatique)

Indications de types

- *Type hints*
- x: type
- def ... -> type

et plus.

- Vérifier le typage statique est optionnel
- Outil mypy

- D'autres checkers existent :
  - intégré à PyCharm
  - pytype, pyright, pyre

```
def divide(a:float, b:float)
           -> float:
  try:
    return a/b
  except Exception as e:
    raise ValueError('Invalid input')
```

```
$ pip install mypy
```

```
...
divide("a",3)      #line 9
```

```
$ mypy ectypagestat.py
ec.py:9: error: Argument 1 to "divide" has
incompatible type "str"; expected "float"
Found 1 error in 1 file (checked 1 source file)
```

Merci !

- Restons en contact :
  - Georges Georgoulis – ggeorgoulis@alteractifs.org – 06 12 68 40 06

  - Alteractifs

    Coopérative d'activité et d'entrepreneurs www.alteractifs.org