module `mockito` pour python

`https://code.google.com/p/mockito-python`
`https://mockito-python.readthedocs.io/en/latest`

```
>>> import mockito
>>> help(mockito)
```

Mockito est un générateur automatique de doublures qui s'utilise ainsi dans un test :

- phase de création des objets du test : création des mocks ;
- toujours dans cette phase, on décrit leur comportement (phase de *stubbing*) ;
- lors de l'exécution du code à tester, toutes les interactions avec les mocks sont mémorisées par mockito ;
- l'oracle peut interroger les mocks pour savoir comment ils ont été utilisés.

```
def test_avec_des_mocks(self):
    # creation des mocks
    # creation du SUT + injection des dep
    # stubbing - when(monmock)...
    # exercice du SUT : sut.foo()
    # oracle - verify(monmock)...
```

Création d'un mock Dans l'esprit du duck typing, on crée un mock sans préciser de typage. Le mock est en l'état capable d'accepter n'importe quel appel de méthode, et retourne la valeur par défaut de Python (ce qui peut surprendre dans le cas d'un test genre `if not carte.valide():`. On peut aussi créer des mocks « stricts » qui lèvent une erreur en cas d'interaction inattendue.

```
>>> carte = mock()
>>> print(carte.foo())
None
```

```
>>> carte_s = mock(strict=True)
>>> carte_s.foo()
Traceback (most recent call last):
...
AttributeError: 'Dummy' has no attribute 'foo' configured
```

Stubbing, dire au mock comment se comporter
Retour d'une valeur, éventuellement fonction de paramètres :

```
>>> when(carte).solde().thenReturn(56)
>>> when(carte).foo(3).thenReturn(33)
>>> when(carte).foo(4).thenReturn(44)
>>> carte.solde()
56
```

```
>>> carte.foo(3)
33
>>> carte.foo(4)
44
>>> when(carte).foo(4).thenReturn(444)
>>> carte.foo(4)
444
```

Levée d'exception : `when(carte).debiter(100).thenRaise(Exception)`.
Chaque définition de comportement écrase la définition précédente (ou celle par défaut).
Comportement non utilisé par le test : si le comportement décrit dans la phase de stubbing n'est pas appelé lors de l'exécution du test, mockito ne se plaint pas.
Stubbing avancé : Il est possible d'enchaîner les `thenReturn` : avec `when(carte).solde().thenReturn(3).thenReturn(4)` le premier appel à `carte.solde())` retournera 56 et tous les suivants retourneront 3. Il est aussi possible d'enchaîner des `thenReturn` et des `thenRaise`.

Vérification que les interactions avec le mock sont celles attendues : `verify`

`verify(carte).debiter(100)` lève une exception si la méthode `debiter` n'est pas appelée exactement une fois avec `100` en paramètre.
Vérifications avancées :

- `verify(monmock, times=2).methode()` (le défaut est `times=1`)
- `verify(monmock, atleast=1).methode()`
- `verify(monmock, atmost=3).methode()`
- `verify(monmock, never).methode()`
- `verify(monmock, between=range(1,10)).methode()`
- `verifyNoMoreInteractions(monmock)`
- `verifyZeroInteractions(monmock)`

Les Argument Matchers permettent de ne pas préciser une valeur d'argument, ce qui sert à la fois en phase de *stubbing* et de vérification : `any` (tout objet ou `None`), `any(int)` (tout entier non None), etc.

```
verify(carte, never).debiter(any(int))
```