

LOG3430 - Méthodes de test et de validation du logiciel

Laboratoire 1

Couverture de test

Hisham Boulifa - 2085232

Pungtzé-Sy D. Kamdem - 2139333

Gr. : 02

11 février 2024

Question 1

Après avoir fait la commande *'pytest in tests' with Coverage*, on peut apercevoir que la couverture totale du code pour *Requests* est de 87%.

Module	statements	missing	excluded	coverage
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm__jb_pytest_runner.py	33	3	0	91%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm__jb_runner_tools.py	173	48	0	72%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm__jb_serial_tree_manager.py	57	8	0	86%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm__jb_utils.py	71	34	0	52%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity__init__.py	6	0	0	100%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity\common.py	98	60	0	39%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity\diff_tools.py	62	36	0	42%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity__jb_local_exc_store.py	11	3	0	73%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity\messages.py	154	58	0	62%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teamcity\pytest_plugin.py	294	123	0	58%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests__init__.py	68	25	0	63%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests__version__.py	10	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests_internal_utils.py	21	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\adapters.py	194	12	0	94%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\api.py	19	3	0	84%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\auth.py	173	21	0	88%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\certs.py	4	1	0	75%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\compat.py	38	2	0	93%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\cookies.py	239	53	0	78%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\exceptions.py	37	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\help.py	62	19	0	69%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\hooks.py	14	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\models.py	456	33	0	93%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\packages.py	17	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\sessions.py	268	11	0	96%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\status_codes.py	14	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\structures.py	39	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\utils.py	483	66	0	86%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests__init__.py	6	1	0	83%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\compat.py	12	2	0	83%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\conftest.py	37	18	0	51%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_help.py	13	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_hooks.py	9	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_lowlevel.py	227	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_packages.py	7	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_requests.py	1684	49	0	97%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_structures.py	42	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_testserver.py	107	14	0	87%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_utils.py	309	5	0	98%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\testserver__init__.py	0	0	0	100%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\testserver\server.py	87	4	0	95%
C:\Users\hisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\utils.py	13	1	0	92%
Total	5660	713	0	87%

Figure 1. Couverture de code de la question 1

Question 2


En prenant en compte les branches, la couverture de code diminue de 2% pour tomber à 85%. Ceci peut s'expliquer par le fait qu'il manque des scénarios possibles à tester. Par exemple, il se peut qu'une condition *if* soit exécutée par les tests sans qu'on puisse avoir un aperçu de toutes les issues de la condition.

Module	statements	missing	excluded	branches	partial	coverage
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\fb_pytest_runner.py	33	3	0	17	7	80%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\fb_runner_tools.py	173	48	0	50	10	70%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\fb_serial_tree_manager.py	57	8	0	26	8	78%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\fb_utils.py	71	34	0	18	0	44%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity__init__.py	6	0	0	0	0	100%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity\common.py	98	60	0	43	3	33%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity\diff_tools.py	62	36	0	17	2	35%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity\fb_local_exc_store.py	11	3	0	0	0	73%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity\messages.py	154	58	0	30	9	60%
C:\Program Files\JetBrains\PyCharm 2022.2.3\plugins\python\helpers\pycharm\teacity\pytest_plugin.py	294	123	0	124	32	54%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests__init__.py	68	25	0	12	5	60%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests_version_.py	10	0	0	0	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests_internal_utils.py	21	0	0	2	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\adapters.py	194	12	0	70	7	93%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\api.py	19	3	0	2	0	86%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\auth.py	173	21	0	60	15	84%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\certs.py	4	1	0	2	1	67%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\compat.py	30	2	0	2	1	91%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\cookies.py	239	53	0	96	9	72%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\exceptions.py	37	0	0	2	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\help.py	62	19	0	18	5	60%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\hooks.py	14	0	0	10	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\models.py	456	33	0	188	19	92%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\packages.py	17	0	0	10	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\sessions.py	268	11	0	100	4	96%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\status_codes.py	14	0	0	10	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\structures.py	39	0	0	8	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\utils.py	483	66	0	220	12	85%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests__init__.py	6	1	0	0	0	83%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\compat.py	12	2	0	0	0	83%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\conftest.py	37	18	0	0	0	51%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_help.py	13	0	0	4	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_hooks.py	9	0	0	2	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_lowlevel.py	227	0	0	44	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_packages.py	7	0	0	0	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_requests.py	1684	49	0	206	10	96%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_structures.py	42	0	0	0	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_testserver.py	107	14	0	26	0	88%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\test_utils.py	309	5	0	34	4	97%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\testserver__init__.py	0	0	0	0	0	100%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\testserver\server.py	87	4	0	18	2	94%
C:\Users\hishal\POLYMTL\HIVER 2024\LOG3430\TP\requests\tests\utils.py	13	1	0	4	1	88%
Total	5660	713	0	1479	166	85%

Figure 2. Couverture de code de la question 2

Question 3

Nous avons d'abord modifié *.coveragerc* afin d'omettre le fichier *compat.py* des tests.

A screenshot of a terminal window with a dark background. The prompt is [run]. The command being entered is omit = src/requests/compat.py. The cursor is at the end of the line.

```
[run]
omit = src/requests/compat.py
|
```

Figure 3. Contenu du fichier *.coveragerc* une fois modifié

Ensuite, nous avons écrit ces trois lignes de commandes :

- *coverage erase*
- *coverage run -m pytest*
- *coverage html*

ce qui a généré le fichier de couverture html suivant : (page suivante)

<i>Module</i>	<i>statements</i>	<i>missing</i>	<i>excluded</i>	<i>coverage</i>
src\requests__init__.py	68	25	0	63%
src\requests__version__.py	10	0	0	100%
src\requests_internal_utils.py	21	0	0	100%
src\requests\adapters.py	194	12	0	94%
src\requests\api.py	19	3	0	84%
src\requests\auth.py	173	21	0	88%
src\requests\certs.py	4	1	0	75%
src\requests\cookies.py	239	53	0	78%
src\requests\exceptions.py	37	0	0	100%
src\requests\help.py	62	19	0	69%
src\requests\hooks.py	14	0	0	100%
src\requests\models.py	456	34	0	93%
src\requests\packages.py	17	0	0	100%
src\requests\sessions.py	268	11	0	96%
src\requests\status_codes.py	14	0	0	100%
src\requests\structures.py	39	0	0	100%
src\requests\utils.py	483	66	0	86%
tests__init__.py	6	1	0	83%
tests\compat.py	12	2	0	83%
tests\conftest.py	37	18	0	51%
tests\test_help.py	13	0	0	100%
tests\test_hooks.py	9	0	0	100%
tests\test_lowlevel.py	227	0	0	100%
tests\test_packages.py	7	0	0	100%
tests\test_requests.py	1684	35	0	98%
tests\test_structures.py	42	0	0	100%
tests\test_testserver.py	107	14	0	87%
tests\test_utils.py	309	5	0	98%
tests\testserver__init__.py	0	0	0	100%
tests\testserver\server.py	87	4	0	95%
tests\utils.py	13	1	0	92%
Total	4671	325	0	93%

Figure 4. Couverture de code de la question 3

Nous pouvons apercevoir que la couverture de code est passé à 93% après avoir omis *compat.py*.

Question 4

Tout d'abord, si ce n'est pas déjà fait, il faut installer le module *coverage* en faisant *pip install coverage*.

Ensuite, il suffit d'exécuter la commande *coverage run -m --branch -m pytest*.

Enfin, pour visualiser le tableau de la couverture de code, nous avons exécuté la commande *coverage html*.

- La commande *coverage run* dit à *coverage.py* de surveiller et de mesurer la couverture du code Python lors de son exécution, ce qui inclut le suivi des appels de fonction, des lignes de code, des branches, et d'autres éléments du programme.
- L'option *-m* informe Python d'exécuter un module particulier comme s'il s'agissait d'un script autonome. Dans ce contexte, *-m* est employé pour exécuter le module *pytest* comme un script indépendant.
- *--branch* spécifie que la couverture de branche doit être calculée. Cela signifie que la couverture sera mesurée non seulement pour les lignes de code exécutées, mais aussi pour les branches de code, telles que les instructions conditionnelles, les boucles, etc.
- *pytest* est le module de test Python que nous utilisons pour exécuter les tests. Il recherche automatiquement les fichiers de test dans le répertoire courant et les sous-répertoires et exécute les tests qu'il trouve.

Module	statements	missing	excluded	branches	partial	coverage
src\requests__init__.py	68	25	0	12	5	60%
src\requests__version__.py	10	0	0	0	0	100%
src\requests_internal_utils.py	21	0	0	2	0	100%
src\requests\adapters.py	194	12	0	70	7	93%
src\requests\api.py	19	3	0	2	0	86%
src\requests\auth.py	173	21	0	60	15	84%
src\requests\certs.py	4	1	0	2	1	67%
src\requests\compat.py	30	2	0	2	1	91%
src\requests\cookies.py	236	53	0	98	9	72%
src\requests\exceptions.py	37	0	0	2	0	100%
src\requests\help.py	62	19	0	18	5	60%
src\requests\hooks.py	14	0	0	10	0	100%
src\requests\models.py	456	32	0	188	18	92%
src\requests\packages.py	17	0	0	10	0	100%
src\requests\sessions.py	268	11	0	100	4	96%
src\requests\status_codes.py	14	0	0	10	0	100%
src\requests\structures.py	39	0	0	8	0	100%
src\requests\utils.py	483	66	0	220	12	85%
tests__init__.py	6	1	0	0	0	83%
tests\compat.py	12	2	0	0	0	83%
tests\conftest.py	37	18	0	0	0	51%
tests\test_help.py	13	0	0	4	0	100%
tests\test_hooks.py	9	0	0	2	0	100%
tests\test_lowlevel.py	227	0	0	44	0	100%
tests\test_packages.py	7	0	0	0	0	100%
tests\test_requests.py	1684	35	0	208	5	97%
tests\test_structures.py	42	0	0	0	0	100%
tests\test_testserver.py	187	14	0	26	0	88%
tests\test_utils.py	309	5	0	34	4	97%
tests\testserver__init__.py	0	0	0	0	0	100%
tests\testserver\server.py	87	4	0	18	2	94%
tests\utils.py	13	1	0	4	1	88%
Total	4701	325	0	1154	89	91%

Figure 5. Couverture de code de la question 4

Question 5

Comme pour la question précédente, il faut installer le module *pytest-cov* en faisant *pip install pytest-cov*. Ensuite, il faut exécuter la commande suivante :

```
pytest --cov=requests --cov-branch --cov-report term --cov-report html
```

- *pytest* est la commande pour exécuter des tests en utilisant *pytest*.
- *--cov=requests* indique à *pytest-cov* de mesurer la couverture de code pour le paquet *requests*.
- *--cov-branch* active la mesure de la couverture de branches.
- *--cov-report term* indique à *pytest-cov* de montrer le rapport de couverture dans le terminal après l'exécution des tests.
- *--cov-report html* crée un rapport de couverture détaillé en HTML.

Module	statements	missing	excluded	branches	partial	coverage
src\requests__init__.py	68	68	0	12	0	0%
src\requests__version__.py	10	10	0	0	0	0%
src\requests_internal_utils.py	21	11	0	2	0	52%
src\requests\adapters.py	194	57	0	70	7	76%
src\requests\api.py	19	12	0	2	0	43%
src\requests\auth.py	173	53	0	60	15	70%
src\requests\certs.py	4	4	0	2	0	0%
src\requests\compat.py	30	30	0	2	0	0%
src\requests\cookies.py	239	116	0	98	9	53%
src\requests\exceptions.py	37	29	0	2	0	26%
src\requests\help.py	62	19	0	18	5	60%
src\requests\hooks.py	14	3	0	10	0	88%
src\requests\models.py	456	119	0	188	18	79%
src\requests\packages.py	17	17	0	10	0	0%
src\requests\sessions.py	268	60	0	100	3	82%
src\requests\status_codes.py	14	14	0	10	0	0%
src\requests\structures.py	39	18	0	8	0	62%
src\requests\utils.py	483	141	0	220	11	75%
Total	2148	781	0	814	68	68%

Figure 6. Couverture de code de la question 5

Question 6

Voici un exemple de couverture avec la méthode *coverage.py* :

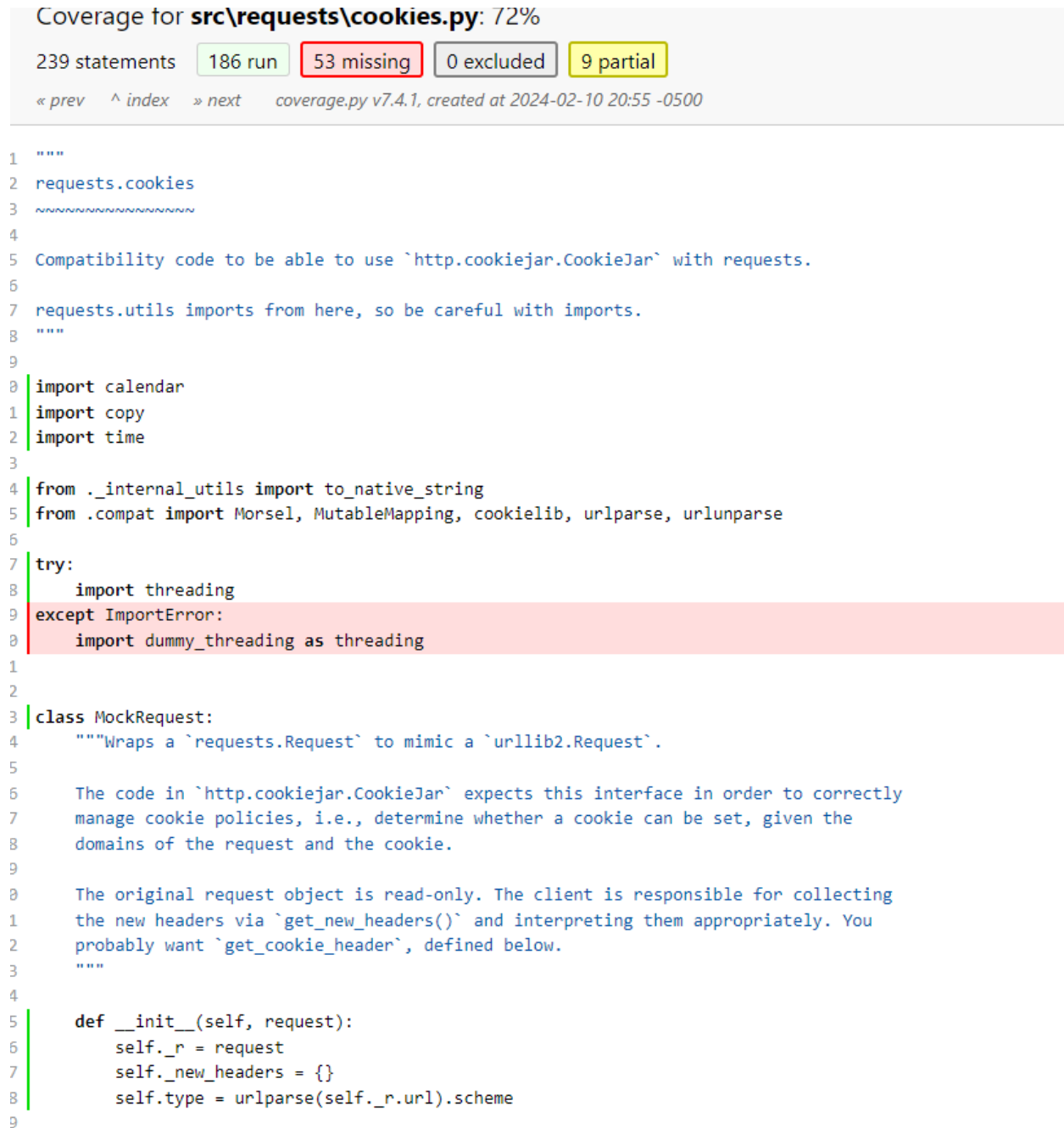


Figure 7. Couverture de code de la question 6 avec la méthode *coverage.py*

Pour le même fichier, voici la couverture avec *pytest-cov* :

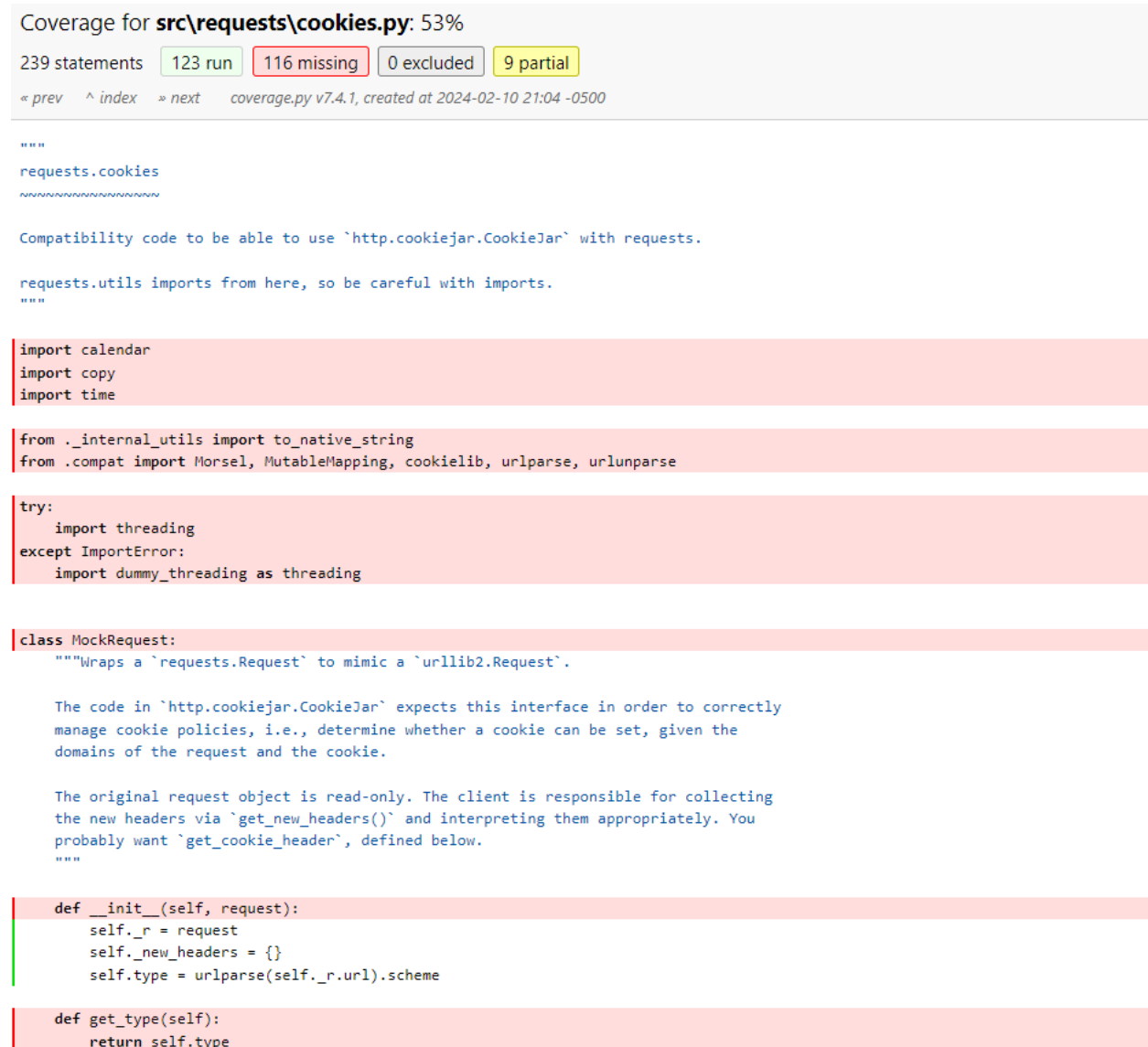


Figure 8. Couverture de code de la question 6 avec la méthode *pytest-cov*

À travers l'analyse de la couverture de ce fichier avec les deux méthodes différentes, nous pouvons remarquer que *pytest-cov* ne couvre pas les lignes d'importation de module (ex : `import calendar`), les lignes de déclaration de classe (ex : `class MockRequest`) ainsi que les lignes de déclaration de méthode (ex : `def __init__(self, request)`).

Ces résultats sont sensés, car *pytest-cov* couvre uniquement le code qu'on teste activement. Les lignes d'importation et de déclaration ne font pas partie de ces lignes désirées. En voici l'explication; les lignes d'importation servent plutôt à importer d'autres modules ou ressources nécessaires à l'exécution de votre code. Ainsi, inclure ces lignes dans les mesures de la couverture de code peut fausser les résultats, car elles ne représentent pas le fonctionnement réel de notre code. Cela s'applique également pour les lignes de déclaration.

Question 7

Voici les tests ajoutés pour augmenter la couverture :

```
class TestFromKeyValList:
    new *
    @pytest.mark.parametrize(
        "value, expected",
        (
            [
                (('key1', 'val1'), ('key2', 'val2'))
            ]
        ),
    )
    def test_valid(self, value, expected):
        assert from_key_val_list(value) == expected

    new *
    def test_invalid(self):
        with pytest.raises(ValueError):
            from_key_val_list("string")
```

Figure 9. Tests ajoutés

C:\Users\nisha\POLYMTL\HIVER 2024\LOG3430\TP\requests\src\requests\utils.py 483 63 0 87%

Figure 10. Couverture de code de la question 7

```
def from_key_val_list(value):
    """Take an object and test to see if it can be represented as a
    dictionary. Unless it can not be represented as such, return an
    OrderedDict, e.g.,

    ::

    >>> from_key_val_list([('key', 'val')])
    OrderedDict([('key', 'val')])
    >>> from_key_val_list('string')
    Traceback (most recent call last):
    ...
    ValueError: cannot encode objects that are not 2-tuples
    >>> from_key_val_list({'key': 'val'})
    OrderedDict([('key', 'val')])

    :rtype: OrderedDict
    """
    if value is None:
        return None

    if isinstance(value, (str, bytes, bool, int)):
        raise ValueError("cannot encode objects that are not 2-tuples")

    return OrderedDict(value)
```

Figure 11. Lignes couvertes grâce aux nouveaux tests