Install required package

```
1 pip install pycodestyle pep257 pytest
```

```
Collecting pycodestyle
  Downloading https://files.pythonhosted.org/packages/10/5b/88879fb861ab79aef45c
     |████████████████████████████████| 51kB 4.6MB/s
Collecting pep257
  Downloading https://files.pythonhosted.org/packages/ec/31/e432e1aa35f692e3f686
Requirement already satisfied: pytest in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.6/dis
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: pluggy<0.8,>=0.5 in /usr/local/lib/python3.6/dist
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3.6
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.6/dist-packag
Installing collected packages: pycodestyle, pep257
Successfully installed pep257-0.7.0 pycodestyle-2.6.0
```

Checking code style including pep8 pep257 by pycodestyle and pep257

```
1 !pycodestyle fisher.py
2 !pep257 fisher.py
```

Unit testing for the function

```
1 !pytest fisher.py
```

```
============================= test session starts ==============================
platform linux2 -- Python 2.7.17, pytest-3.6.4, py-1.8.0, pluggy-0.7.1
rootdir: /content, inifile:
collected 3 items

fisher.py .FF                                                            [100%]

=================================== FAILURES ===================================
_____________________________________ test_2 ______________________________________

    def test_2():
        """
        second test for Fisher acceptance.

        testing if G > N
        """
        try:
            fisher_accept2(N=10, G=11, n=6)
        except AssertionError as e:
>           pytest.fail(e, pytrace=True)
E           Failed: assert 10 > 11
```

```
fisher.py:176: Failed
_____________________________________ test_3 _____________________________________

    def test_3():
        """
        third test for Fisher acceptance.

        testing if input is a wrong data type
        """
        try:
            fisher_accept2('10', 6, 2)
        except AssertionError as e:
>           pytest.fail(e, pytrace=True)
E           Failed: assert False
E            +  where False = isinstance('10', int)

fisher.py:188: Failed
====================== 2 failed, 1 passed in 0.35 seconds ======================
```

In the first fisher_accept function. It does not consider the if the population size is smaller than "good" items and also smaller than sample size. Moreover the datatype of the input might also be wrong. Thus, in the second algorithm, adding two assertationsto avoid this situation.

Attached code

```
"""
Fish accept.

Test and check algorithms for Fisher accept
"""

import numpy as np
from scipy.stats import hypergeom
import pytest


def fisher_accept(N, G, n, alpha=0.05):
    """
    Acceptance region for randomized hypergeometric test.

    Find the acceptance region for a randomized, exact level alpha test of
    the null hypothesis X~Hypergeometric(N, G, n). The acceptance region is
    the smallest possible. (And not, for instance, symmetric.)

    If a non-randomized, conservative test is desired, use the union of I and J
    as the acceptance region.

    Parameters
    ----------
    N:  integer
```

```
    N:  integer
        population size
    G:  integer
        number of "good" items in the population
    n:  integer
        sample size
    alpha : float
        desired significance level

    Returns
    --------
    I:  list
        values for which the test never rejects
    J:  list
        values for which the test sometimes rejects
    gamma : float
        probability the test does not reject when the value is in J
    """
    x = np.arange(0, n+1)          # all possible values of X
    posout = list(x)
    # start with all possible outcomes, then remove some
    pmf = hypergeom.pmf(x, N, G, n)   # hypergeometric pmf
    bottom = 0                     # smallest outcome still in I
    top = n                        # largest outcome still in I
    J = []
    p_J = 0                        # probability of the randomized outcome
    p_tail = 0                     # probability of outcomes excluded from I
    while p_tail < alpha:
        # still need to remove outcomes from the acceptance region
        pb = pmf[bottom]
        pt = pmf[top]
        if pb < pt:             # the lower possibility has smaller probability
            J = [bottom]
            p_J = pb
            bottom += 1
        elif pb > pt:           # the upper possibility has smaller probability
            J = [top]
            p_J = pt
            top -= 1
        else:
            if bottom < top:    # the two possibilities have equal probability
                J = [bottom, top]
                p_J = pb+pt
                bottom += 1
                top -= 1
            else:               # there is only one possibility left
                J = [bottom]
                p_J = pb
                bottom += 1
        p_tail += p_J
        for j in J:
            posout.remove(j)
```

```
    gamma = (p_tail-alpha)/p_J
    # probability of accepting H_0 when X in J to get exact level alpha
    return posout, J, gamma


def fisher_accept2(N, G, n, alpha=0.05):
    """
    Acceptance region for randomized hypergeometric test.

    Find the acceptance region for a randomized, exact level alpha test of
    the null hypothesis X~Hypergeometric(N, G, n). The acceptance region is
    the smallest possible. (And not, for instance, symmetric.)

    If a non-randomized, conservative test is desired, use the union of I and J
    as the acceptance region.

    Parameters
    ----------
    N:  integer
        population size
    G:  integer
        number of "good" items in the population
    n:  integer
        sample size
    alpha : float
        desired significance level

    Returns
    --------
    I:  list
        values for which the test never rejects
    J:  list
        values for which the test sometimes rejects
    gamma : float
        probability the test does not reject when the value is in J
    """
    assert N > n
    assert N > G
    assert isinstance(N, int)
    assert isinstance(G, int)
    assert isinstance(n, int)
    assert isinstance(alpha, float)
    x = np.arange(0, n+1)          # all possible values of X
    posout = list(x)
    # start with all possible outcomes, then remove some
    pmf = hypergeom.pmf(x, N, G, n)   # hypergeometric pmf
    bottom = 0                     # smallest outcome still in I
    top = n                        # largest outcome still in I
    J = []
    p_J = 0                        # probability of the randomized outcome
    p_tail = 0                     # probability of outcomes excluded from I
    while p_tail < alpha:
```

```
        # still need to remove outcomes from the acceptance region
        pb = pmf[bottom]
        pt = pmf[top]
        if pb < pt:             # the lower possibility has smaller probability
            J = [bottom]
            p_J = pb
            bottom += 1
        elif pb > pt:           # the upper possibility has smaller probability
            J = [top]
            p_J = pt
            top -= 1
        else:
            if bottom < top:    # the two possibilities have equal probability
                J = [bottom, top]
                p_J = pb+pt
                bottom += 1
                top -= 1
            else:                   # there is only one possibility left
                J = [bottom]
                p_J = pb
                bottom += 1
        p_tail += p_J
        for j in J:
            posout.remove(j)
    gamma = (p_tail-alpha)/p_J
    # probability of accepting H_0 when X in J to get exact level alpha
    return posout, J, gamma


def test_1():
    """
    first test for Fisher acceptance.

    testing for a regular case
    """
    assert fisher_accept(N=10, G=2, n=5) == ([1], [0, 2], 0.8875000000000001)


def test_2():
    """
    second test for Fisher acceptance.

    testing if G > N
    """
    try:
        fisher_accept2(N=10, G=11, n=6)
    except AssertionError as e:
        pytest.fail(e, pytrace=True)


def test_3():
    """
```

```
    """
    third test for Fisher acceptance.

    testing if input is a wrong data type
    """
    try:
        fisher_accept2('10', 6, 2)
    except AssertionError as e:
        pytest.fail(e, pytrace=True)

```