Question 4 – Polygon Function

- I have a function that takes a point and a polygon and returns True if the point lies inside the polygon, and False if it does not. The function implementation is shown on the next page.
- In a white box testing implement your test cases in python class and consider each test in a separate method name starts with “test_”.
- Justify each test case implemented (“Why is this test case important?”) and document it.

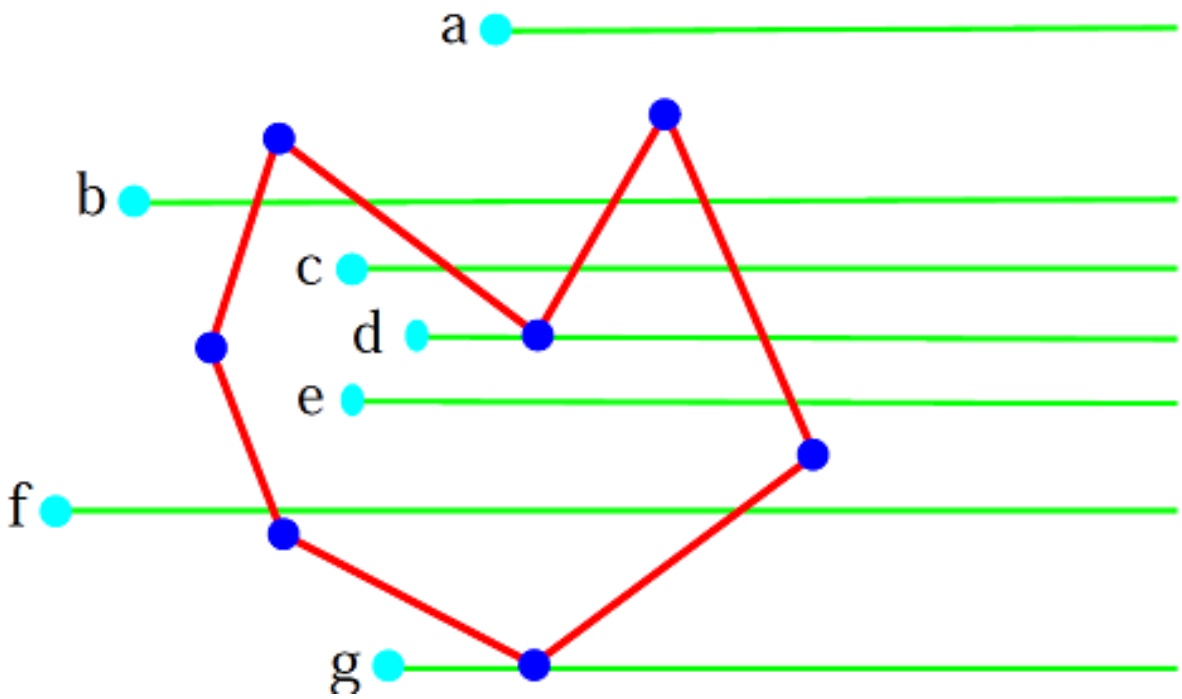
Solution:

Following is one of the simple ideas found in my study about the given problem, To check whether a point is inside or outside

1) Draw a horizontal line to the right of each point and extend it to infinity

2) Count the number of times the line intersects with polygon edges.

3) A point is inside the polygon if either count of intersections is odd or point lies on an edge of polygon. If none of the conditions is true, then point lies outside.



PolyGon.py

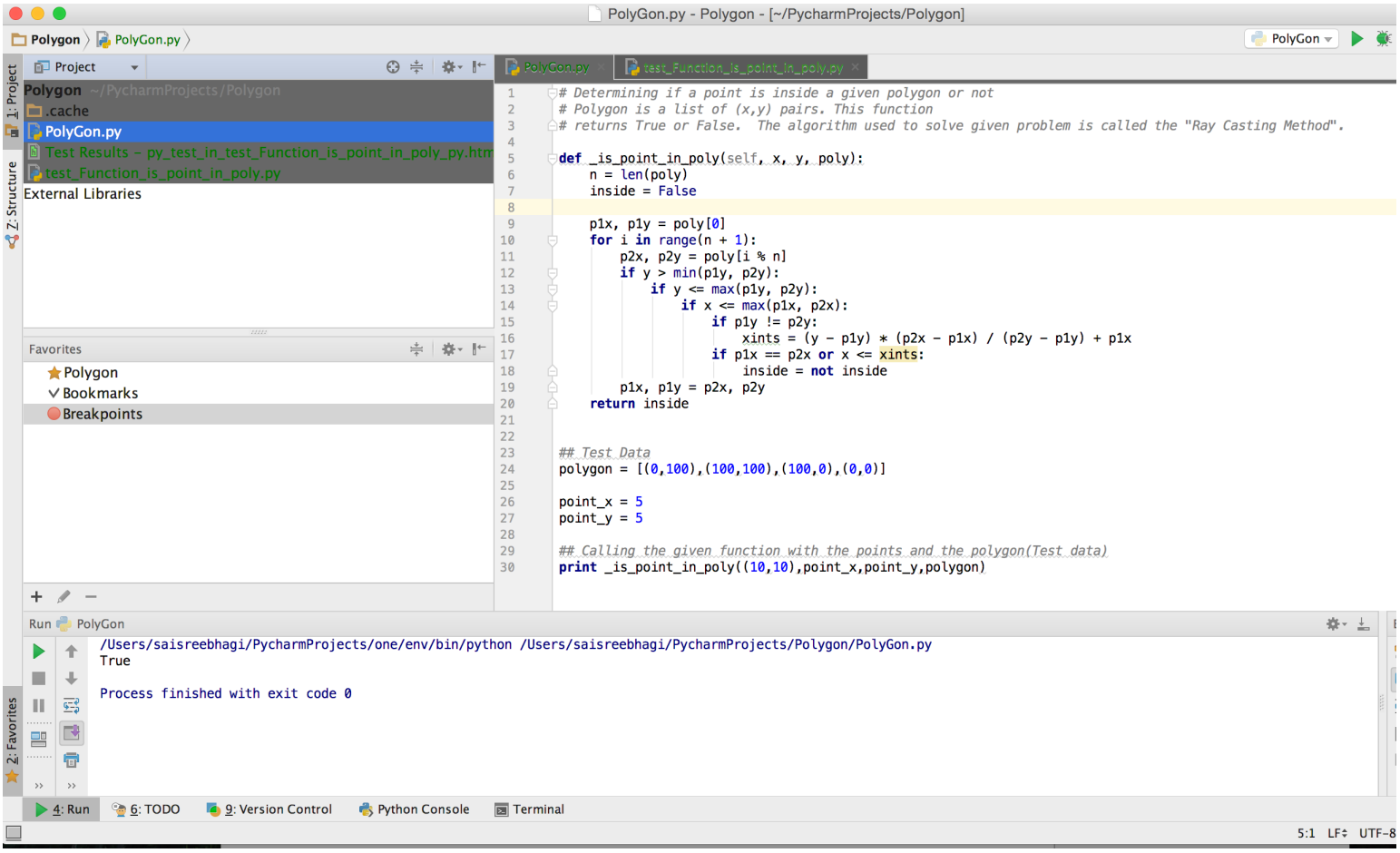
```
def _is_point_in_poly(self, x, y, poly):
    n = len(poly)
    inside = False

    p1x, p1y = poly[0]
    for i in range(n + 1):
        p2x, p2y = poly[i % n]
        if y > min(p1y, p2y):
            if y <= max(p1y, p2y):
                if x <= max(p1x, p2x):
                    if p1y != p2y:
                        xints = (y - p1y) * (p2x - p1x) / (p2y - p1y) + p1x
                    if p1x == p2x or x <= xints:
                        inside = not inside
        p1x, p1y = p2x, p2y
    return inside

## Test Data/Input:
#polygon = [(0,10),(10,10),(10,0),(0,0)]

#point_x = 5
#point_y = 5

## Calling the function with the points and the polygon
print _is_point_in_poly((5,5),point_x,point_y,polygon)
```

Output: True

Testing above function using White Box Testing method: White Box Testing (also known as Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Along with the suggested test approach for improving code structure, Can add unit tests to improve the test efficiency for a given function using python ‘unittest’ module for varying inputs of polygon types and points for covering all edge/cornered and negative cases based on the polygon types as seen in following tutorial

https://rosettacode.org/wiki/Ray-casting_algorithm

Screen Shot:

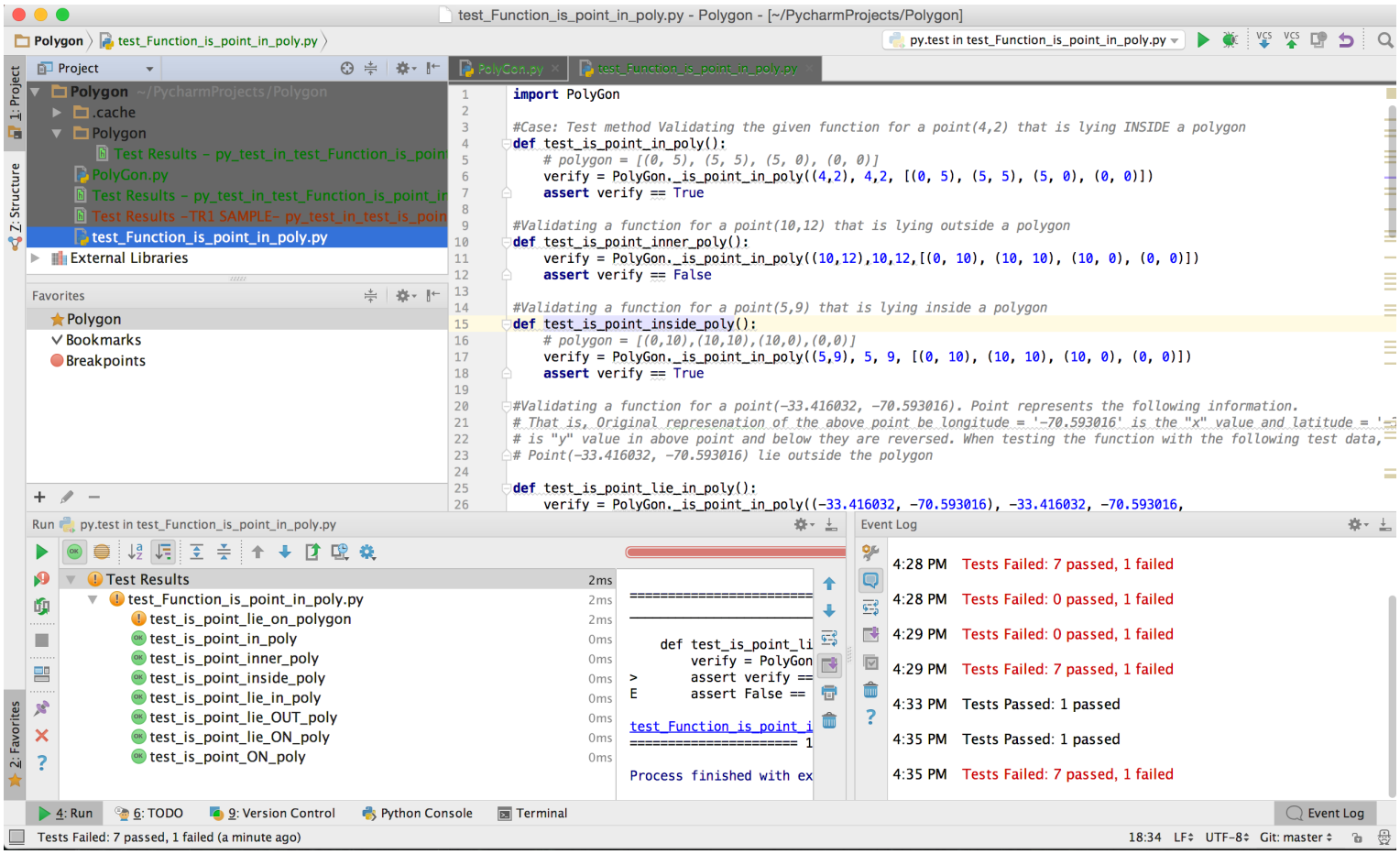


test_Function_is_point_in_poly.py

```
import PolyGon

def test_is_point_in_poly():
    verify = PolyGon._is_point_in_poly((4,2), 4,2, [(0, 5), (5, 5), (5, 0), (0, 0)])
    assert verify == True

def test_is_point_inner_poly():
    verify = PolyGon._is_point_in_poly((10,12),10,12,[(0, 10), (10, 10), (10, 0), (0, 0)])
    assert verify == False

def test_is_point_inside_poly():
    verify = PolyGon._is_point_in_poly((5,9), 5, 9, [(0, 10), (10, 10), (10, 0), (0, 0)])
    assert verify == True
```

```
def test_is_point_lie_in_poly():
    verify = PolyGon._is_point_in_poly((-33.416032, -70.593016), -33.416032, -70.593016,
                                       [(-33.416032, -70.593016), (-33.415370, -70.589604),
                                        (-33.417340, -70.589046), (-33.417949, -70.592351),
                                        (-33.416032, -70.593016)])
    assert verify == False

def test_is_point_lie_OUT_poly():
    verify = PolyGon._is_point_in_poly((-10,1),-10,1,[(0,10),(10,10),(10,0),(0,0)])
    assert verify == False

def test_is_point_lie_ON_poly():
    verify = PolyGon._is_point_in_poly((2,2),2,2,[(0,0), (0,2), (2,2), (2,0)])
    assert verify ==  True

def test_is_point_ON_poly():
    verify = PolyGon._is_point_in_poly((90,90),90,90,[(0,100),(100,100),(100,0),(0,0)])
    assert verify == True

def test_is_point_lie_on_polygon():
    verify = PolyGon._is_point_in_poly((9,0),9,0,[(0,10),(10,10),(10,0),(0,0)])
    assert verify == True
```

screenshot



Documentation: Describing the test_Function_is_point_in_poly() methods and related test data used

import PolyGon

Case 1: Testing the case when point lies **inside** the polygon

#Test Data - 1
Test method Validating the given function for a point(4,2) that is lying **INSIDE** a polygon

```
def test_is_point_in_poly():
    # polygon = [(0, 5), (5, 5), (5, 0), (0, 0)]
    verify = PolyGon._is_point_in_poly((4,2), 4,2, [(0, 5), (5, 5), (5, 0), (0, 0)])
    assert verify == True
```

#Test Data - 2
#Validating a given function for a point(5,9) that is lying inside a polygon

```
def test_is_point_inside_poly():
    # polygon = [(0,10),(10,10),(10,0),(0,0)]
    verify = PolyGon._is_point_in_poly((5,9), 5, 9, [(0, 10), (10, 10), (10, 0), (0, 0)])
    assert verify == True
```

Case 2: Testing the case when point lies **outside** the polygon

#Test Data - 1
Test method Validating a given function for a point(-10,1) in Quadrant(II)(in planar co-ordinate system/axis) which lie **OUTSIDE** the given polygon(text data) as shown below when running against given function returns 'False'(As expected)

```
def test_is_point_lie_OUT_poly():
    verify = PolyGon._is_point_in_poly((-10,1),-10,1,[(0,10),(10,10),(10,0),(0,0)])
    assert verify == False
```

#Test data - 2
#Test method Validating a given function for a point(10,12) that is lying outside a polygon

```
def test_is_point_inner_poly():
    verify = PolyGon._is_point_in_poly((10,12),10,12,[(0, 10), (10, 10), (10, 0), (0, 0)])
    assert verify == False
```

#Test Data -3
#Test method Validating a given function for a point(-33.416032, -70.593016). Point represents the following information. That is, Original representation of the above point be longitude = '-70.593016' is the "x" value and latitude = '-33.416032' is "y" value in above point and below they are reversed. When testing the function with the following test data, # Point(-33.416032, -70.593016) lie OUTSIDE the polygon returns false as expected

```
def test_is_point_lie_in_poly():
    verify = PolyGon._is_point_in_poly((-33.416032, -70.593016), -33.416032, -70.593016,
```

```
                                    [(-33.416032, -70.593016), (-33.415370, -70.589604),
                                     (-33.417340, -70.589046), (-33.417949, -70.592351),
                                     (-33.416032, -70.593016)])
    assert verify == False
```

Case 3: Testing the case when point lies **on** the polygon

#Test data - 1
Test method Validating a given function for a point(90,90) that coincides or **'ON'** the below polygon

```
def test_is_point_ON_poly():
    verify = PolyGon._is_point_in_poly((90,90),90,90,[(0,100),(100,100),(100,0),(0,0)])
    assert verify == True
```

#Test Data-2
Test method Validating a given function for a point(2,2) that lies 'ON' the polygon

```
def test_is_point_lie_ON_poly():
    verify = PolyGon._is_point_in_poly((2,2),2,2,[(0,0), (0,2), (2,2), (2,0)])
    assert verify ==  True
```

Test Data - 3
#polygon = [(0,100),(100,100),(100,0),(0,0)], POINT(100,100)
#point_x = 100, point_y = 100 -- Given function returns 'True' as expected

Observation for the case:
Changing the order of points in a POLYGON can have significant impact on current algorithm. For example, When running the given function for the following order of vertices of a below convex polygon(test data) while testing this case,
#[(0,0), (0,100), (100,0), (100,100)] against a point(90,90) - Given Function is returning 'False'

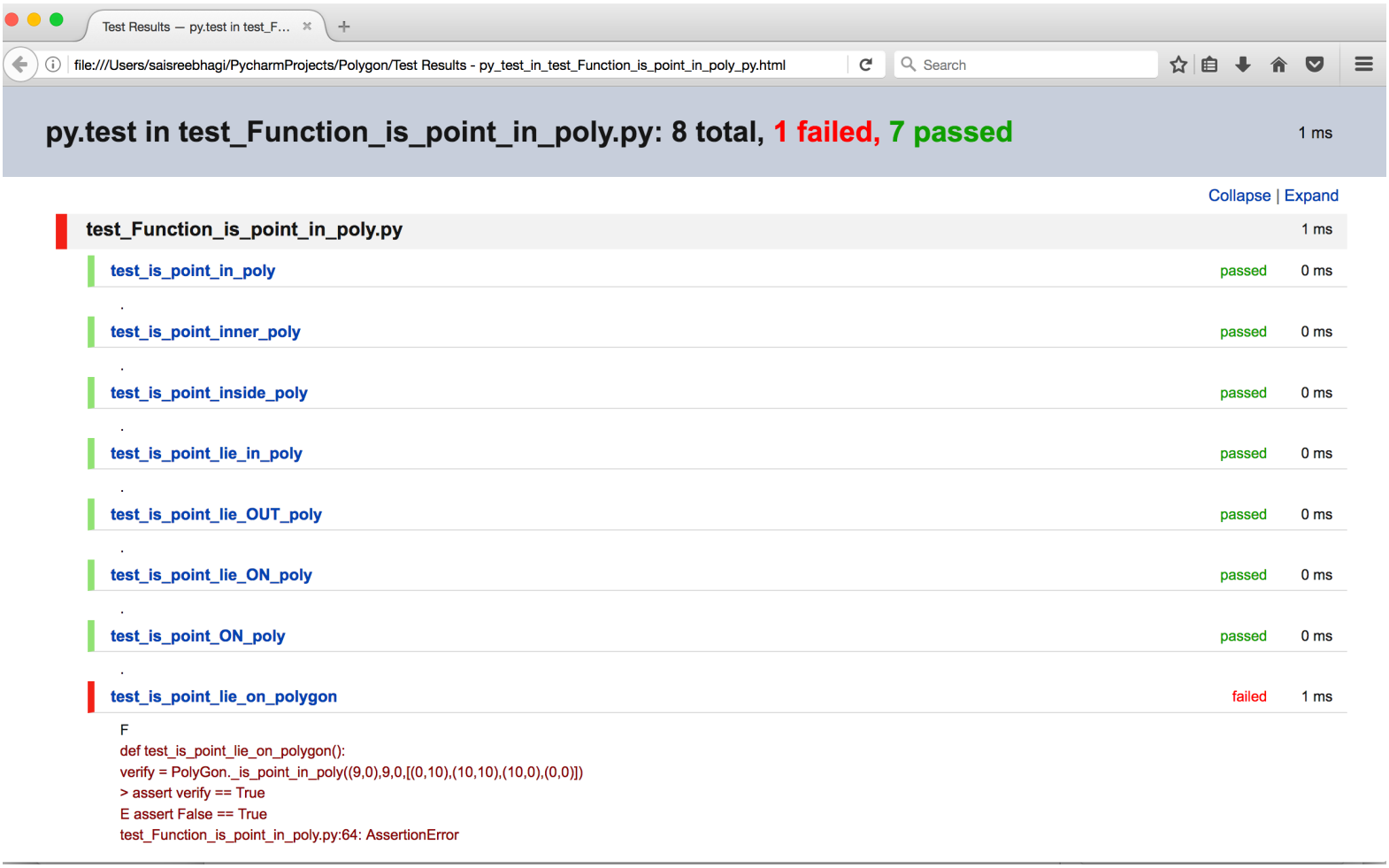
Case 4: Testing the case when point lies **on the side or on the edge** of a polygon

#Test Data - 1
Test method Validating a given function for a point(9,0) that lies '**ON the edge/side**' of the polygon

```
def test_is_point_lie_on_polygon():
    verify = PolyGon._is_point_in_poly((9,0),9,0,[(0,10),(10,10),(10,0),(0,0)])
    assert verify == True
```

Observation for the case:
So, Given function is not solving the above case. Even though, The point 9,0 is not inside the polygon #[(0,10),(10,10),(10,0),(0,0)] It's on the edge. Points exactly on the edge can be considered IN or OUT depending on the current algorithm for the function. Also, Type of the polygon convex/concave matters when using current approach of solving this problem using given function. Debugging this case to improve test passing rate using given function can be improved in a better way using different test modules in python test libraries

Test Execution report



References:

http://erich.realtimerendering.com/ptinpoly/
https://www.google.com
https://www.stackoverflow.com
http://docs.python-guide.org/en/latest/writing/tests/
http://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/
https://rosettacode.org/wiki/Ray-casting_algorithm