Geo1000 – Python Programming – Assignment 1

Due: Monday, September 16, 2024 (18h00)

Introduction

You are expected to create 2 programs. All program files have to be handed in.

Start with the files that are distributed via Brightspace. It is sufficient to modify the function definitions inside these files (replace `pass` with your own implementation). **Do not change the function signatures. This means that you keep their names, which (data types) & how many arguments the functions take and in which order the functions take arguments as given. Also, in case of a fruitful function, make sure to return the values with their expected type.**

This assignment is *preferably* made in groups of 2 (enroll with your group or individually in Brightspace), and your mark will count for your final grade of the course. Helping each other is fine. However, make sure that your implementation is your *own*.

This assignment in total can give you 100 points. Your assignment will be marked based on whether your implementation does the correct things (as described in the assignment), whether your code is decent (e.g. use of proper variable names, indentation, etc), and whether your files are submitted as required (e.g. on time).

It is due: **Monday, September 16, 2024 (18h00)**.

Note that if you submit your assignment after the deadline, some points will be removed. For the first day that a submission is late, 10 points will be removed before marking. For every day after that, another 30 points will be removed. An example: Assume you deliver the assignment at Monday, September 16, 2024, 18h15, the maximum amount of points that then can be obtained for the assignment is 90 $(100 - 10 = 90)$.

Submit the resulting program files (`solve.py`, `segment.py`) via Brightspace (your last submission will be taken into account, also for determining whether you are late). Upload **a zip file** that contains just the program files (with no folders/hierarchy inside)!

Make sure that each Python file handed in starts with the following comment (augment `Authors` and `Studentnumbers` with your own names and numbers):

```
# GEO1000 -- Assignment 1
# Authors:
# Studentnumbers:
```

ABC formula – solve.py (40 points)

A quadratic equation can be written in the form:

$$ax^2 + bx + c = 0$$

Make a function that solves the real roots of a quadratic equation (the x where $y = 0$). These roots can be found by using the ABC-formula:

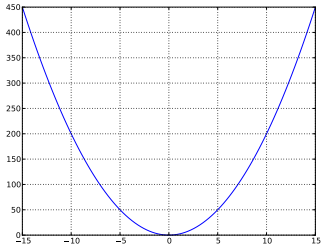
$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$

Here, the discriminant D is:

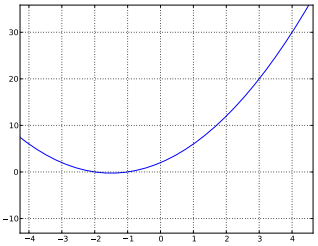
$$D = b^2 - 4ac$$

If the discriminant is positive, then there are two real solutions. When the discriminant is zero, then there is exactly one real solution. If the discriminant is negative, then there will be no real roots (only imaginary ones, which we will not consider for this program).

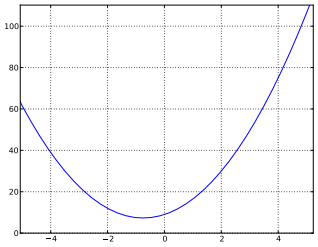
The program should read 3 values per equation.



(a) $2x^2 = 0$



(b) $1.0x^2 + 3.0x + 2.0 = 0$



(c) $3.0x^2 + 4.5x + 9.0 = 0$

Figure 1: Three sample equations

When we give the 3 equations of Figure 1, the output that is printed to the user should be as follows:

```
The roots of 2.0 x^2 + 0.0 x + 0.0 are:
x = 0.0
The roots of 1.0 x^2 + 3.0 x + 2.0 are:
x1 = -1.0 , x2 = -2.0
The roots of 3.0 x^2 + 4.5 x + 9.0 are:
not real
```

In your program, try to replicate the form the output has as best as possible.

The program should have 2 functions: `abc` and `discriminant`.

When the function `abc` is called it determines whether there are 2, 1, or no solution(s) and prints the output. For this, the function `abc` uses the `discriminant` function. The `discriminant` function is a fruitful function.

Do *not* put any print statements in the `discriminant` function. Do not use any other imports than:

```
from math import sqrt
```

Use the following skeleton for `solve.py`, in which you replace the `pass` statement with your implementation:

```
# GEO1000 - Assignment 1
# Authors:
# Studentnumbers:
```

```
from math import sqrt


def abc(a, b, c):
    pass


def discriminant(a, b, c):
    pass


abc(2.0, 0.0, 0.0)
abc(1.0, 3.0, 2.0)
abc(3.0, 4.5, 9.0)
```

Distance between a point & a finite line segment in 2D — segment.py (60 points)

Make a program that computes the distance of a point to a finite line segment. The line segment is given by a pair of 2D points (you can assume that the two points defining the segment are not equal).

The steps you will need to take are:

1. Make a fruitful `distance` function that calculates and returns the Cartesian distance between two points.
2. Make a fruitful function `angle` that calculates and returns the angle (in *radians*) given by three sides of a triangle, using the law of cosines (as illustrated in Figure 2):

   $$c^2 = a^2 + b^2 - 2ab\cos(\gamma) \implies \gamma = \arccos(\frac{a^2 + b^2 - c^2}{2ab})$$

   where side c is opposite of angle γ.

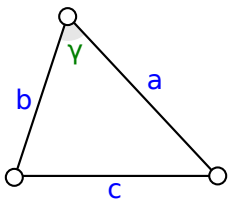


Figure 2: Angle γ can be computed if the lengths of the three sides a, b and c are known.

3. Make a fruitful function `heron` that calculates and returns the area A of a triangle based on the lengths of the triangle's sides (using Heron's formula):

   $$A = \sqrt{s(s-a)(s-b)(s-c)}$$

   where:

   $$s = \frac{a+b+c}{2}$$

   and a, b and c are the lengths of the triangle's sides.

4. Make a fruitful function `segment_point_dist` that calculates and returns the distance of a point to a line segment.

   Note that the distance you should calculate in this case is dependent on the angles α and β (see Figure 3). If both these angles are smaller than or equal to 90 degrees, you can use the height of the triangle as distance. For determining the height of the triangle you can use: area $= \frac{1}{2} \times$ base $\times$ height. If one of the two is bigger than 90 degrees, you should calculate the distance to either one of the two end points of the segment (to which one?).

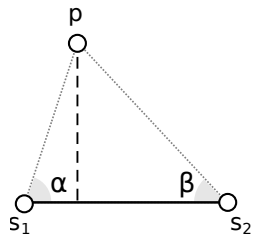


Figure 3: Compute the distance of p to the line segment defined by s_1 and s_2

5. Make sure you handle the cases when p is equal to s_1 or s_2 or when p is positioned on the line segment properly.

Start from the following skeleton:

```
# GEO1000 - Assignment 1
# Authors:
# Studentnumbers:

import math

def distance(x1, y1, x2, y2):
    pass


def heron(a, b, c):
    pass


def angle(a, b, c):
    pass


def segment_point_dist(s1x, s1y, s2x, s2y, px, py):
    pass


print(segment_point_dist(0, 0, 10, 0, 5, 10))
print(segment_point_dist(0, 0, 10, 0, 20, 0))
print(segment_point_dist(0, 0, 10, 0, 0, 0))
print(segment_point_dist(0, 0, 10, 0, 5, 0))
print(segment_point_dist(0, 0, 10, 10, 0, -12))
```

Do not use any other imports than `import math`.