Geo1000 – Python Programming – Assignment 1

Due: Friday September 15, 2017 (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, i.e. their names, which & how many and in which order the functions take arguments**.

This assignment is *preferably* made in groups of 2 (enroll with your group or individually in Brightspace) and the mark you will obtain 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 provides a program that runs, does the correct things (as described in the assignment), your code is decent (e.g. use of proper variable names / indentation / etc) and submitted as required (e.g. on time).

It is due: **Friday September 15, 2017 (18h00)**.

Note that if you submit your assignment after the deadline, 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 20 points will be removed. An example: Assume you deliver the assignment at Friday September 15, 2017, 18h15, the maximum amount of points that then can be obtained for the assignment is 90 $(100 - 10 = 90)$.

Submit the resulting program files (`units.py`, `tienstra.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 that is handed in, starts with the following comment (augment `Authors` and `Studentnumbers` with your own names and numbers):

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

1 Unit Conversion – units.py (20 points)

Make 4 functions. These functions should:

1. print the result of converting a value that represents (nautical) miles to kilometers: `mi_km`
2. print the result of converting a value that represents kilometers to (nautical) miles: `km_mi`

3. print the result of converting a value that represents a speed in kilometer/hour to meter/second: `kmh_ms`
4. print the result of converting a value that represents a speed in meter/second to kilometer/hour: `ms_kmh`

For the first two functions: 1 mile is 1609.344 meter and 1 nautical mile is 1852 meter. The nautical parameter is a boolean that specifies whether to convert to/from land miles (when `nautical` is set to `False`) or sea miles (`nautical` is set to `True`).

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

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

def mi_km(value, nautical):
    pass

def km_mi(value, nautical):
    pass

def kmh_ms(value):
    pass

def ms_kmh(value):
    pass

mi_km(10, True)
mi_km(10, False)
km_mi(10, True)
km_mi(10, False)
kmh_ms(10)
ms_kmh(10)
```

2 Resection by means of the Tienstra method – tienstra.py (80 points)

Make a program that determines the coordinates of an unknown point from angle measurements to 3 known points by using the Tienstra formula and prints the coordinates to the user.

Before working on this part of the assignment, you should have read the book upto and including Chapter 7.

Given the coordinates of 3 known points and 2 angle measurements at a position, the coordinates of this position can be determined. In surveying this is known as *resection* (in Dutch: *achterwaartse insnijding*): 'In surveying work, the most common methods of computing the coordinates of a point by resection are Cassini's Method and the Tienstra formula, though the first known solution was given by Willebrord Snellius.' https://en.wikipedia.org/wiki/Position_resection

The steps you need to make:

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

$$a^2 = b^2 + c^2 - 2bc\cos(\alpha)$$

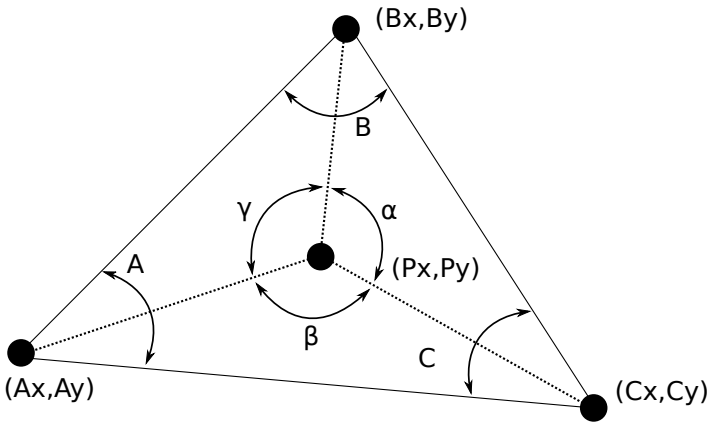

Figure 1: Resection

$$\alpha = \arccos(\frac{a^2 - b^2 - c^2}{-2bc})$$

where side a is opposite of angle α.

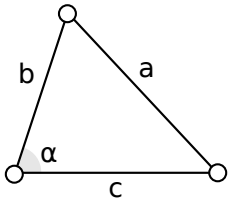

Figure 2: Angle α can be computed if the length of the three sides a, b and c is known.

- Make a function `tienstra` that takes 8 arguments: the coordinates of the known points A, B and C and the two angles α and γ that were measured (in degrees). The function prints as result the coordinates of the unknown point.

  Within this function:

  - Transform the given angles (α and γ) in radians (the math module provides a `radians` function for that).
  - Compute the missing angle β (in radians, a circle of $360^\circ = 2\pi$ radians).
  - Calculate the side lengths (`AB`, `BC`, `CA`) of the triangle (use the `distance` function).
  - Based on the side lengths compute the angles: `A`, `B` and `C` using the cosine law (use the `angle` function).
  - The unknown point coordinates can now be calculated by the following formulas:

$$P_x = \frac{(K_1 \times A_x) + (K_2 \times B_x) + (K_3 \times C_x)}{K_1 + K_2 + K_3}$$

$$P_y = \frac{(K_1 \times A_y) + (K_2 \times B_y) + (K_3 \times C_y)}{K_1 + K_2 + K_3}$$

    where:

$$K_1 = \frac{1}{\cot(A) - \cot(\alpha)}$$

$$K_2 = \frac{1}{\cot(B) - \cot(\beta)}$$

$$K_3 = \frac{1}{\cot(C) - \cot(\gamma)}$$

For this make a function `cot`. Hint:

$$\cot(\theta) = \frac{1}{\tan(\theta)}$$

Start from the following skeleton:

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

import math

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

def angle(a, b, c):
    pass

def cot(x):
    pass

def tienstra(ax, ay, bx, by, cx, cy, alpha, gamma):
    pass

tienstra(
    1000.0, 5300.0,
    2200.0, 6300.0,
    3100.0, 5000.0,
    115.052, 109.3045)
```

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