# UTM Coordinate Calculations with Distance and Bearing

## Abstract

The purpose of this exercise is to calculate the bearing and distance between two sets of UTM Coordinates. With the distance calculated into kilometers and the bearing as a compass bearing from 0 degrees to 359 degrees.

## Problem Statement

We are demonstrating the use of variables in python to calculate the bearing and distance in kilometers between two UTM coordinates. The following variables are needed to calculate the bearing and distance are e1, n1, e2, n2. Additional useful variables are eastings, northings, totaldist and quadrant. The following equations were used for the problem with variations of the variables e1, n1, e2, n2 to calculate bearing and distance.

## Methodology

Using Python to process the problem at hand with the included math library as well as custom functions to handle all calculations and outputs. A series of if statements were created to calculate the correct compass bearing between the various UTM Coordinates provided. The following code is the final product used within the problem.

'''

Chris Oakley

Distance/Bearing

08/24/2020

EGN3214 - Assignment 1

Calculate the Distance in kilometers and return the angle of the bearing in degrees.

Variables:

e1, n1 - Starting Point

e2, n2 - End Point

b - Bearing in degrees

d - Distance in kilometers

eastings - Distance traveled or measured eastward

northings - Distance traveled or measured northward

quadrant - What quadrant the bearing is in

Quad Setup for Bearings

360° 0°

(4) E-, N+ | (1) E+, N+

360+Bearing | No Change

|

270° --------------------------------- 90°

(3) E-, N- | (2) E+, N-

Bearing-180 | 180+Bearing

|

'''

import math

def calculateDB(e1, n1, e2, n2):

b = 0

d = math.sqrt((n2-n1)\*\*2+(e2-e1)\*\*2)/1000

eastings = e2-e1

northings = n2-n1

if eastings > 0 and northings > 0:

quadrant = 'First Quadrant E+, N+'

b = math.degrees(math.atan((e2-e1)/(n2-n1)))

elif eastings > 0 and northings < 0:

quadrant = 'Second Quadrant E+, N-'

b = 180+math.degrees(math.atan((e2-e1)/(n2-n1)))

elif eastings < 0 and northings < 0:

quadrant = 'Third Quadrant E-, N-'

b = 180+math.degrees(math.atan((e2-e1)/(n2-n1)))

elif eastings < 0 and northings > 0:

quadrant = 'Fourth Quadrant E-, N+'

b = 360 + math.degrees(math.atan((e2-e1)/(n2-n1)))

return d, b, eastings, northings, quadrant

'''

Either calls all coordinates listed in the assignment with a traveled distance or allows the user to input two sets of coordinates. Then prints the output to the terminal.

'''

def outputAnswers(question, des):

if question == 'A':

print(des)

print(quadrant)

print(f'Eastings {eastings}')

print(f'Northings {northings}')

print(f'Distance: {d:.3f} kilometers\nBearing: {b:.3f}\xb0 Degrees')

print()

if question == 'M':

print()

print('#'\*50)

print()

print(quadrant)

print(f'Eastings {eastings}')

print(f'Northings {northings}')

print(f'Distance: {d:.3f} kilometers\nBearing: {b:.3f}\xb0 Degrees')

print()

print('#'\*50)

while True:

question = input('Enter Eastings and Northings (M)anually or use (A)ssignment Coordinates (M or A)?').upper()

if question == 'A':

coordlist = [

[511005, 3208379, 509938, 3208346, 'Browns Bay Canoe Launch to Sign 3/8'],

[509938, 3208346, 509948, 3208025, 'Sign 3/8 to Sign 4/7'],

[509948, 3208025, 510284, 3208171, 'Sign 4/7 to Sign 5/6'],

[510284, 3208171, 510397, 3208018, 'Sign 5/6 to Sign 6/5'],

[510397, 3208018, 511005, 3208379, 'Sign 6/5 to Browns Bay Canoe Launch'],

]

totaldist = []

print() # Blank line to Start

print('#'\*50)

for item in coordlist:

d, b, eastings, northings, quadrant = calculateDB(item[0], item[1], item[2], item[3])

question = outputAnswers('A', item[4])

totaldist.append(d)

print(f'Total Traveled Distance is {sum(totaldist):.3f} kilometers')

print('#'\*50)

break

elif question == 'M':

e1 = int(input('Enter Eastings Starting Point: '))

n1 = int(input('Enter Northings Starting Point: '))

e2 = int(input('Enter Eastings Ending Point: '))

n2 = int(input('Enter Northings Ending Point: '))

d, b, eastings, northings, quadrant = calculateDB(e1, n1, e2, n2)

question = outputAnswers('M', None)

break

else:

print('Enter a valid option.')

## Solution

Using the following UTM Coordinates provided below in the table with the bearing and distance provided by the Python code above. When the code is run you will be asked to either Manually enter the Coordinates or run the Assignment Coordinates using the choice ‘M’ or ‘A’.

Table 1: Assignment Eastings and Northings with Distance and Bearing Calculated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Easting** | **Northing** | **Easting** | **Northing** | **Distance** | **Bearing** |
| Start to 3/8 | 511005 | 3208379 | 509938 | 3208346 | 1.068 km | 268.229° |
| 3/8 to 4/7 | 509938 | 3208346 | 509948 | 3208025 | 0.321 km | 178.216° |
| 4/7 to 5/6 | 509948 | 3208025 | 510284 | 3208171 | 0.366 km | 66.514° |
| 5/6 to 6/5 | 510284 | 3208171 | 510397 | 3208018 | 0.190 km | 143.552° |
| 6/5 to Start | 510397 | 3208018 | 511005 | 3208379 | 0.707 km | 59.300° |

## Conclusion

Calculating the bearing created a unique problem where we had to calculate the compass bearing. The following formulas are used to create the compass bearing depending on which quadrant it fell in too. The formula for quadrant 2 & 3 remained the same due to the bearing either being a negative or positive number. With quadrant two being a negative number and thus the bearing is subtracted from 180 degrees. While quadrant three was a positive number and added to 180 degrees.

The distance formula can be used as is for a distance in meters, so the final formula must be divided by 1000 to output the distance as kilometers. Instead of checking to see if the northing or easting were greater than or less than each other, such as, . I opted for calculating the difference and checking to see if it was greater or less than zero. A minor change for code readability.