



Presented at the FIG Working Week 2023.  
28 May - 1 June 2023 in Orlando, Florida, USA



# Visual Traverse: an open Source Python Program to assess, post-process, and visualize Polygon Data

Yanli Zhang, Matthew McBroom and Daniel R. Unger  
*Arthur Temple College of Forestry and Agriculture  
Stephen F. Austin State University*

# Outline

- Visual Traverse
  - Traverse
  - Python
  - Visual Traverse

# Texas HB 1523 Bill

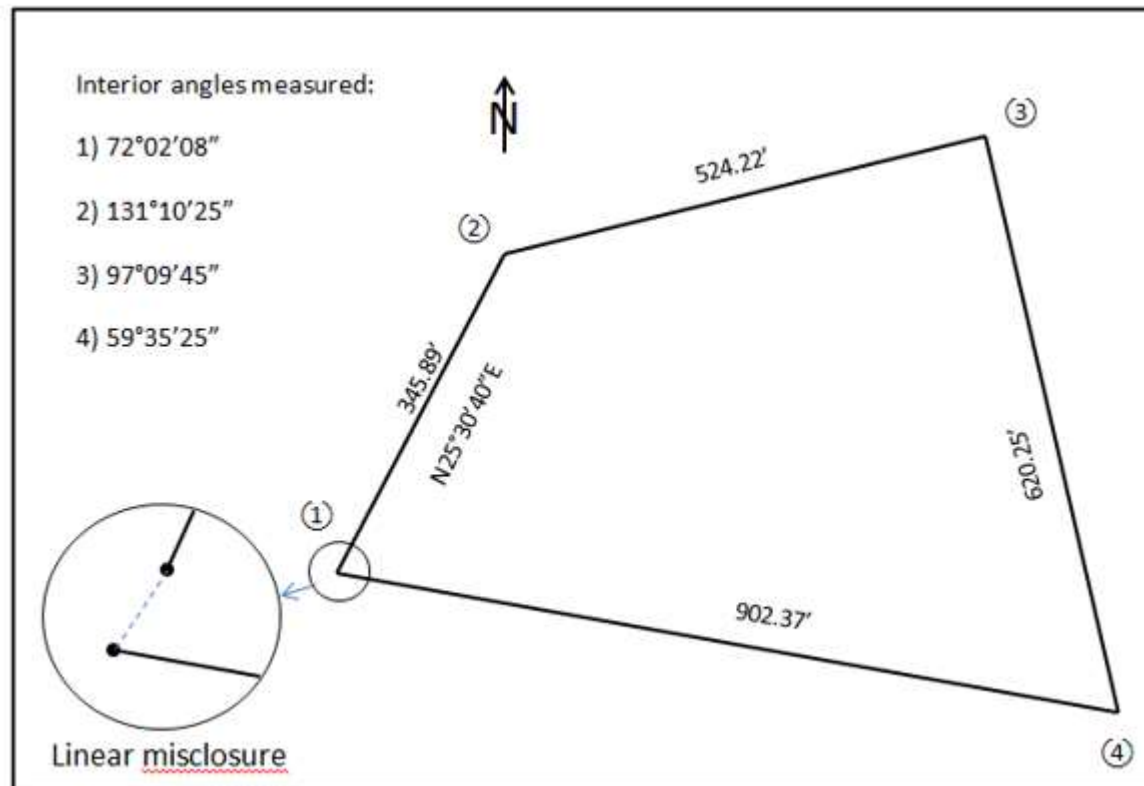
## ■ Effective on 9.1.2019

read as follows:

- (a) An applicant for registration as a registered professional land surveyor must:
- (1) hold a certificate as a surveyor-in-training;
  - (2) have at least two years of experience satisfactory to the board as a surveyor-in-training in performing surveying in delegated responsible charge as a subordinate to a surveyor registered or licensed to engage in the practice of surveying in this state or in another state having registration or licensing requirements equivalent to the requirements of this state; and
  - (3) ~~[if the application is filed after January 1, 2003,~~ have earned an **associate** or [a] bachelor's degree from an accredited institution of higher education that included at least 32 semester hours in a combination of courses acceptable to the board in:
    - (A) civil engineering;
    - (B) land surveying;
    - (C) mathematics;
    - (D) photogrammetry;
    - (E) forestry;
    - (F) land law; or
    - (G) the physical sciences.

# Traverse

## ■ misclosure



# Traverse computation

- Different methods

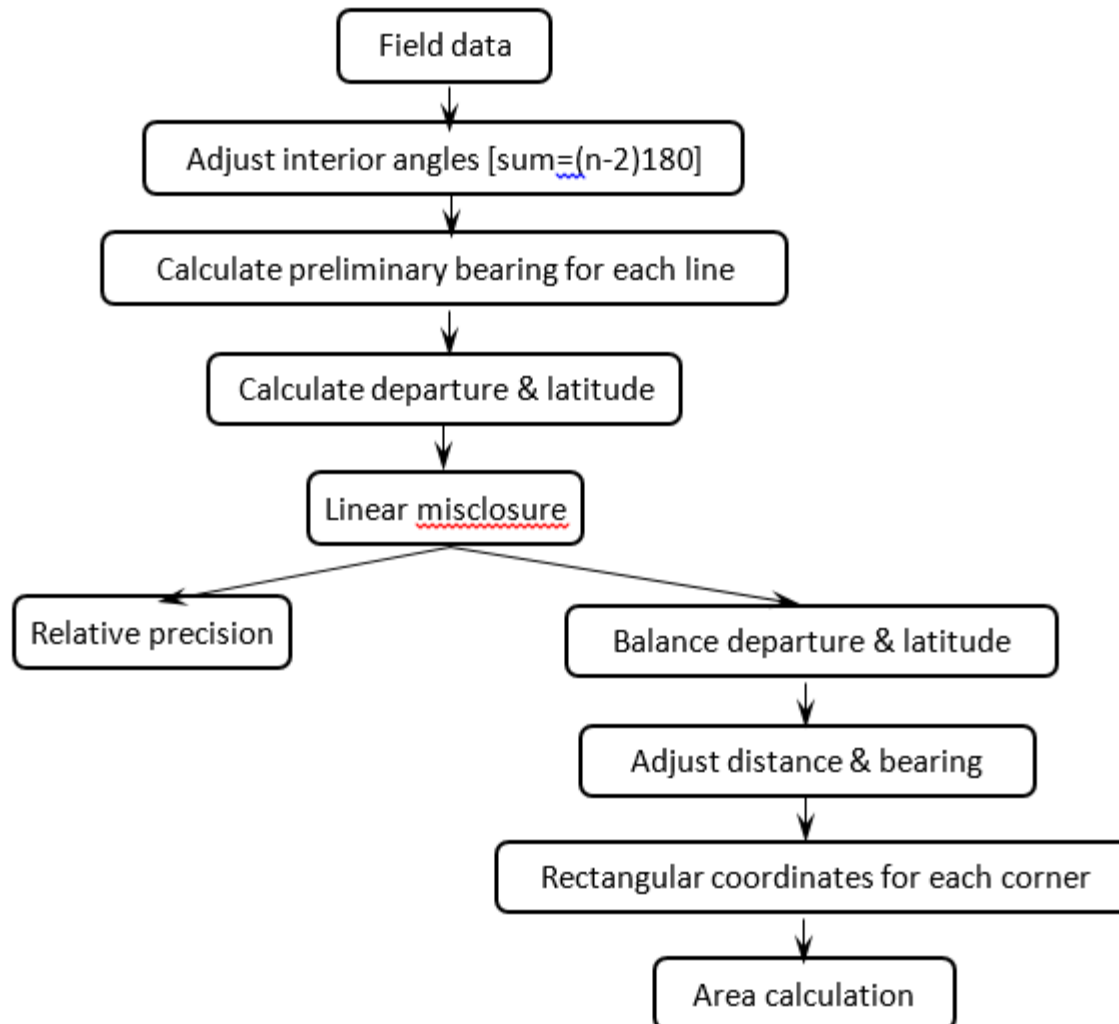
- Compass rule
- Least square
- ...

- Different software

- Carlson Survey
- AutoCAD
- ...



# Compass rule



# Traverse computation

- Teaching challenge
  - First land surveying class



# Outline

- Visual Traverse
  - Traverse
  - Python
  - Visual Traverse



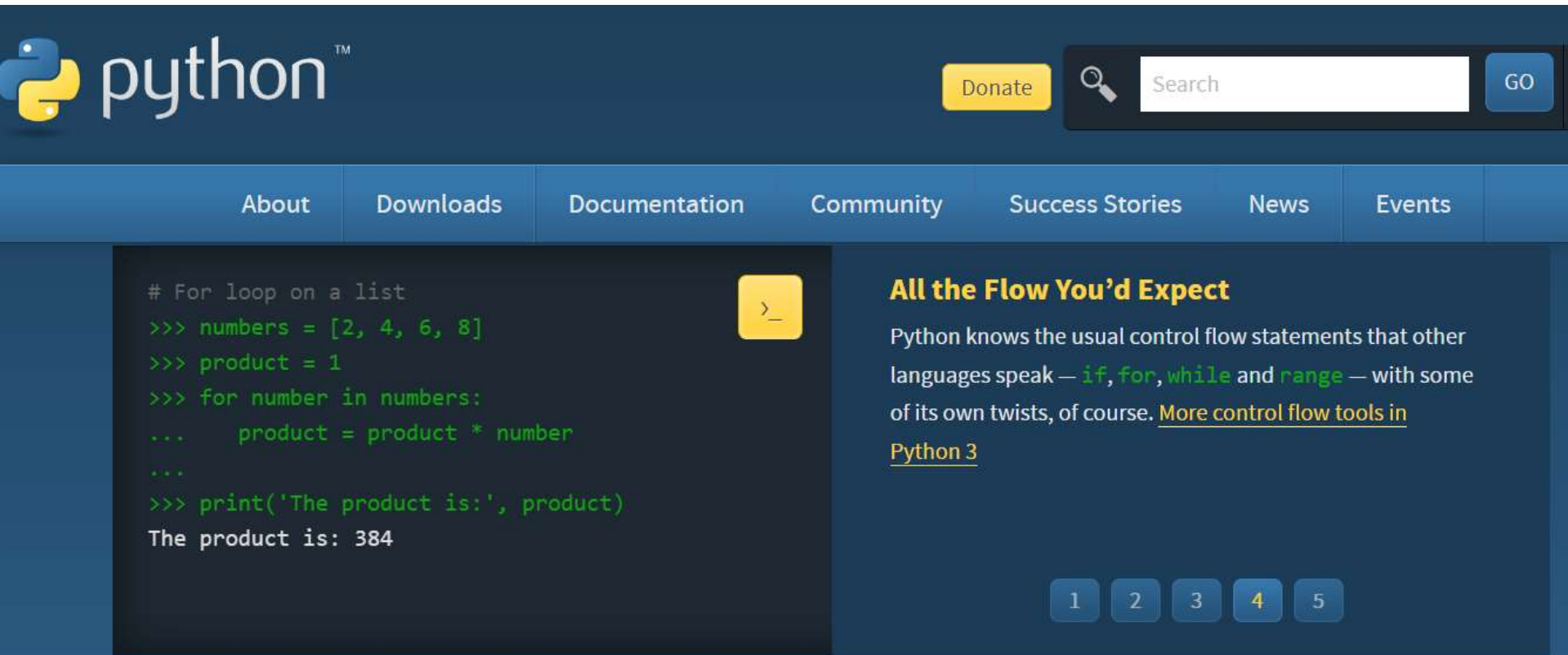
# Python

- Python is open source
  - Free to use, even for commercial products.
- Python is cross platform
  - It runs on Windows, Linux/unix, Mac OS X and have been ported to the Java and .NET virtual machines.



# Python

- 3.\* and 2.\* versions



The screenshot shows the Python.org website. At the top left is the Python logo and the word "python" with a trademark symbol. To the right is a "Donate" button, a search bar with a magnifying glass icon, and a "GO" button. Below the header is a navigation menu with links for "About", "Downloads", "Documentation", "Community", "Success Stories", "News", and "Events". The main content area is split into two columns. The left column contains a code snippet in a dark background with a yellow prompt icon on the right. The right column contains an article preview with the title "All the Flow You'd Expect" and a paragraph of text with a link to "More control flow tools in Python 3". At the bottom right of the article preview are five numbered buttons (1-5).

```
# For loop on a list
>>> numbers = [2, 4, 6, 8]
>>> product = 1
>>> for number in numbers:
...     product = product * number
...
>>> print('The product is:', product)
The product is: 384
```

### All the Flow You'd Expect

Python knows the usual control flow statements that other languages speak — `if`, `for`, `while` and `range` — with some of its own twists, of course. [More control flow tools in Python 3](#)

1 2 3 4 5

# Python application

- Web and internet development
- Scientific computing
- Education
- Desktop GUI (graphical user interface) design
- Software development
- E-commerce system

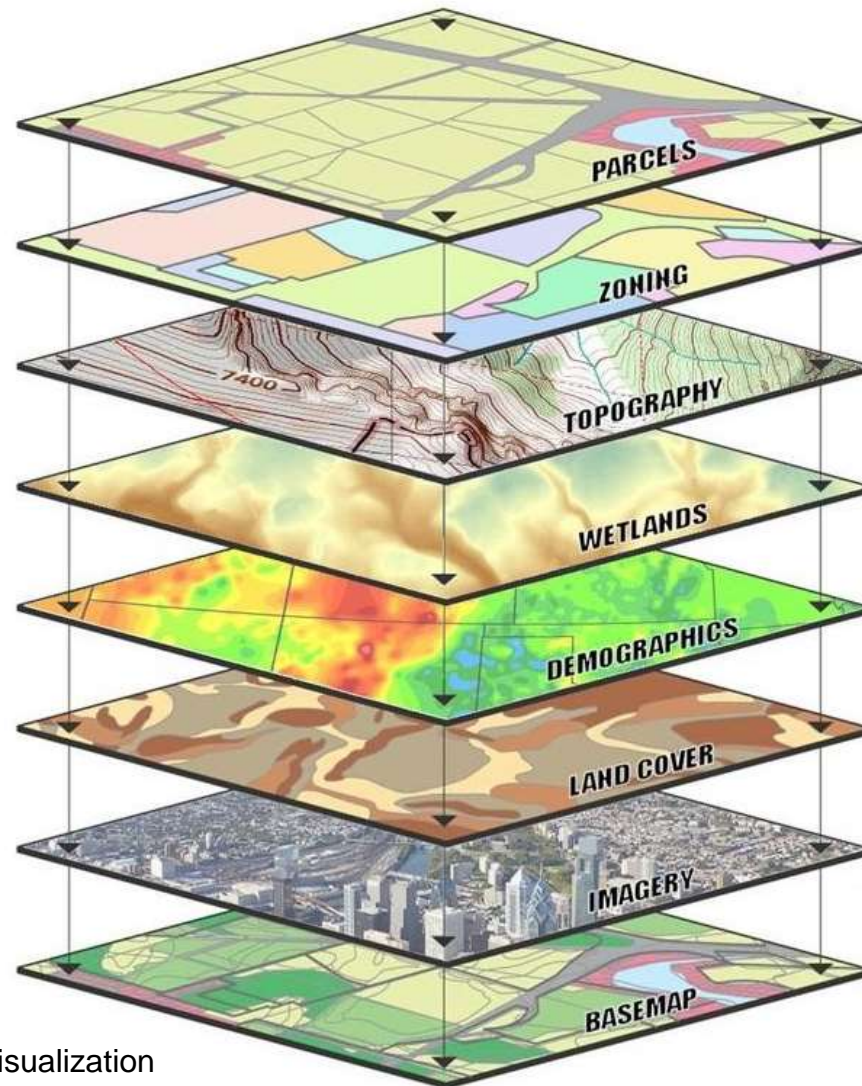
# Python and GIS



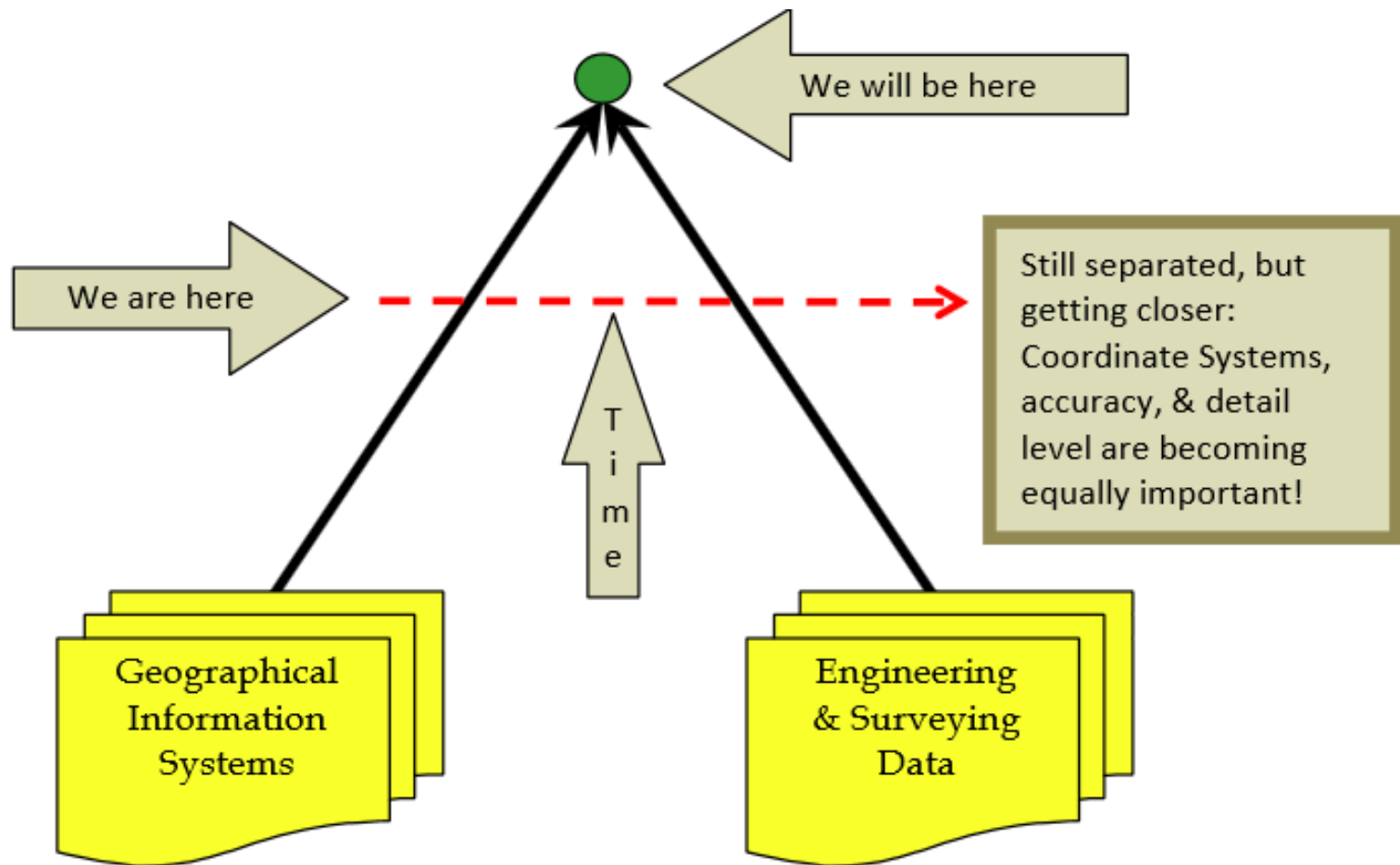
- It is integrated into ArcGIS Desktop
  - Python was introduced to the ArcGIS community with ArcGIS 9.0. Since then, it has been accepted as the scripting language of choice for geoprocessing users and continues to grow. Each release has furthered the Python experience, providing more capabilities and a richer Python-friendly experience.
    - Easy to learn and suitable for both beginners and experts
    - Highly scalable, suitable for large projects or small one-off programs known as scripts
    - Portable, cross-platform
    - Embeddable (making ArcGIS scriptable)
    - Stable and mature
    - A large user community

# GIS and land survey

- Two complementary disciplines



# GIS and land survey



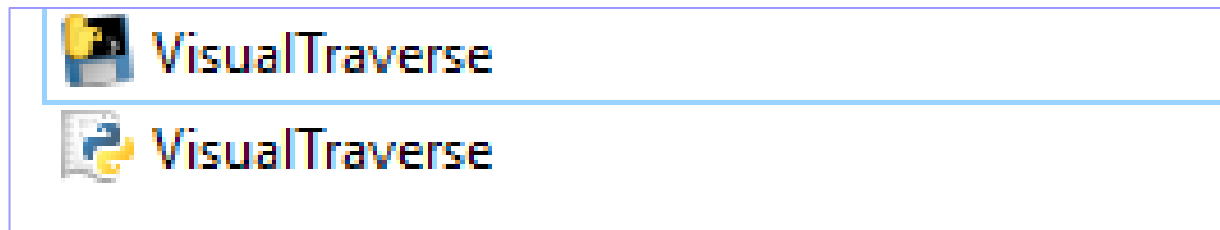


# Outline

- Visual Traverse
  - Traverse
  - Python
  - Visual Traverse

# Visual Traverse

- Open source code
  - For programmers
- Exe file available
  - For non-programmers





# Visual Traverse

Visual Traverse 1.0

File

N  S  
 E  W

Degrees   
Minutes   
Seconds

Reference Bearing

Interior  Exterior  
 Clockwise  Counterclock

Degrees   
Minutes   
Seconds   
Distance

Record Station

Station 1 coordinates X and Y:  
10000  5000

Run Computation  Clear Report

Clear canvas

Report

Canvas to visualize the boundary after traverse computation

Report area to show traverse computation outputs

# Visual Traverse

- Thorough consideration for polygon

N                       S  
 E                       W

Degrees   
Minutes   
Seconds

Reference Bearing

Interior                       Exterior  
 Clockwise                       Counterclock

Degrees   
Minutes   
Seconds   
Distance

Record Station

Station 1 coordinates X and Y:

10000  5000

Run Computation      Clear Report

Clear canvas

Report

# Python code

## ■ Code example

```
def in_quad(bearing):|
    # Expects string with bearing "N:67:11:00:E"
    (b_maj, b_degree, b_minutes, b_seconds, b_minor) = bearing.split(":")

    if b_maj.upper() == "N" and b_minor.upper() == "W":
        quad = "NW"
    elif b_maj.upper() == "N" and b_minor.upper() == "E":
        quad = "NE"
    elif b_maj.upper() == "S" and b_minor.upper() == "E":
        quad = "SE"
    elif b_maj.upper() == "S" and b_minor.upper() == "W":
        quad = "SW"
    else:
        quad = "ERROR - in_quad(bearing) expected form N:67:11:00:E"
    return quad
```

# Python code

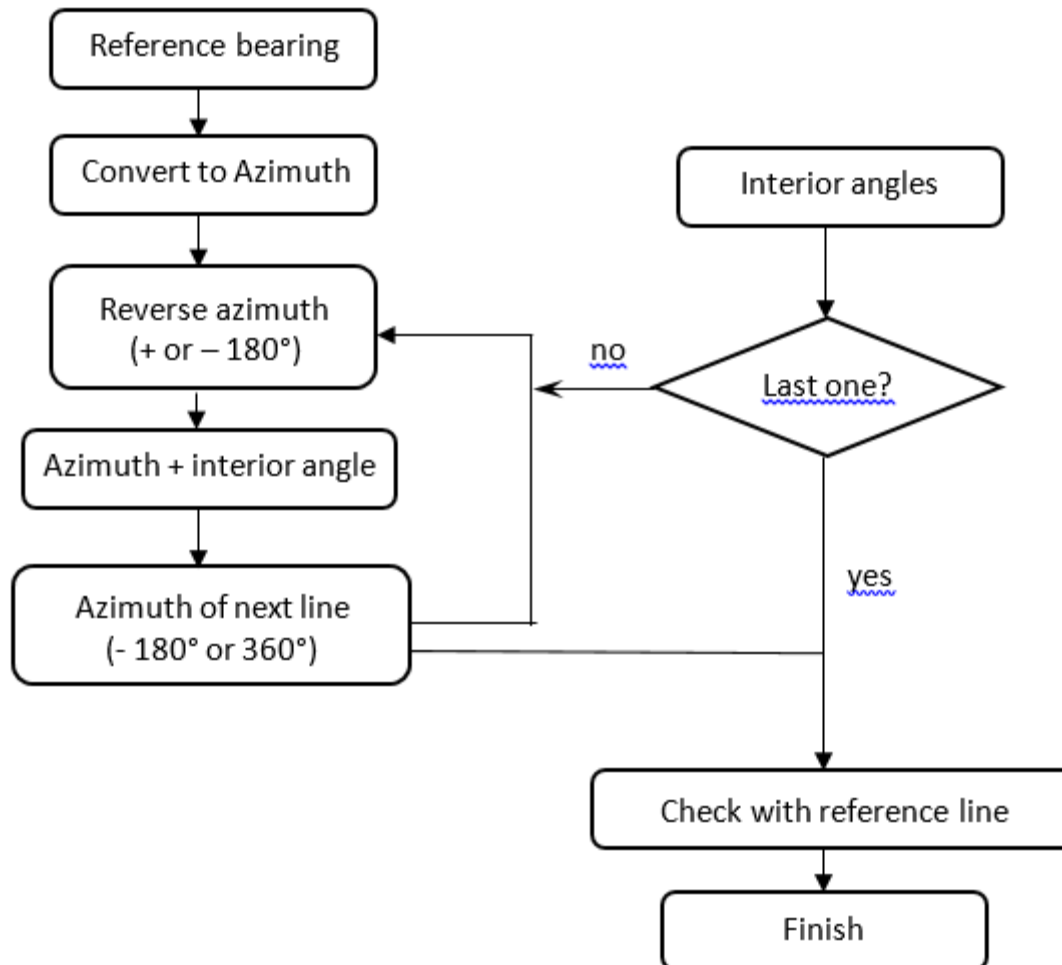
## ■ Code example

```
def bearing_2_azimuth(bearing):
    # Expects string with bearing as "N:67:11:00:E"
    # Returns float Digital Degrees dd.dddd
    (b_maj, b_degree, b_minutes, b_seconds, b_minor) = bearing.split(":")

    if in_quad(bearing) == "NE":
        myAzimuth = dms_2_dd(b_degree + ":" + b_minutes + ":" + b_seconds)
    elif in_quad(bearing) == "SE":
        myAzimuth = 180 - dms_2_dd(b_degree + ":" + b_minutes + ":" + b_seconds)
    elif in_quad(bearing) == "SW":
        myAzimuth = 180 + dms_2_dd(b_degree + ":" + b_minutes + ":" + b_seconds)
    elif in_quad(bearing) == "NW":
        myAzimuth = 360 - dms_2_dd(b_degree + ":" + b_minutes + ":" + b_seconds)
    else:
        myAzimuth = "Error in bearing_2_azimuth. Check input"

    return myAzimuth
```

# Azimuth



# Outputs

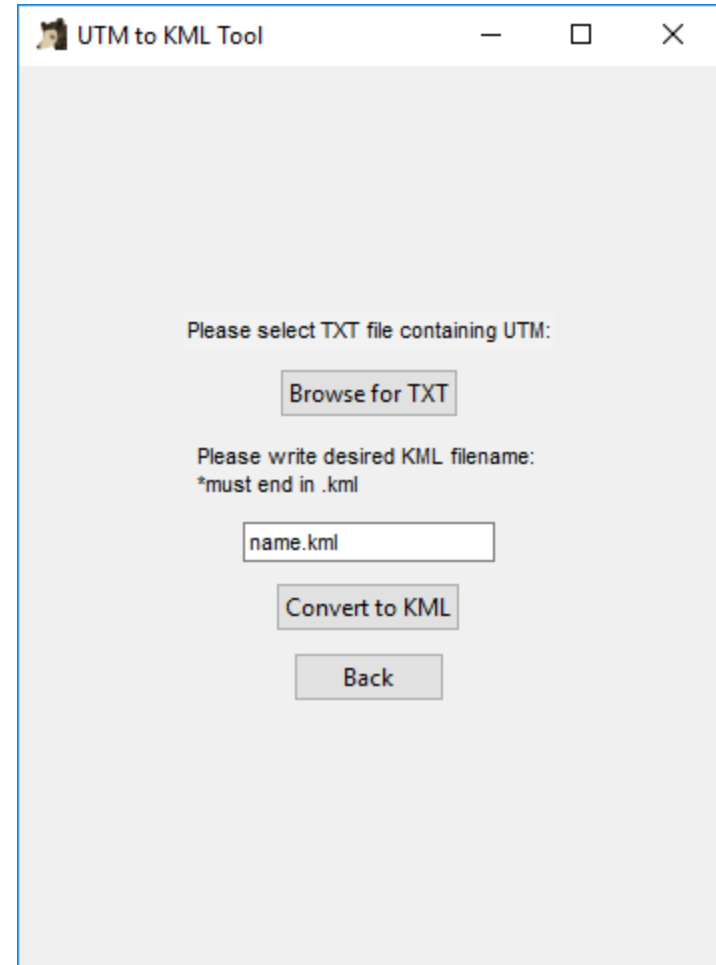
The screenshot displays the 'Visual Traverse 1.0' software interface. On the left, a diagram shows a traverse with four vertices and four sides, with an arrow indicating a clockwise direction. On the right, the control panel includes several sections:

- Reference Bearing:** Radio buttons for N, E, S, W. Input fields for Degrees, Minutes, and Seconds.
- Record Station:** Radio buttons for Interior and Exterior. Radio buttons for Clockwise and Counterclockwise. Input fields for Degrees, Minutes, Seconds, and Distance.
- Station 1 coordinates X and Y:** Input fields for 1000 and 500.
- Buttons:** Run Computation, Clear Report, and Clear canvas.
- Report:** A text area displaying the following data:

```
Reference Line: N:25:30:40:E
Station: 72:2:8 Distance: 345.89
Station: 131:10:25 Distance: 524.22
Station: 97:9:45 Distance: 620.25
Station: 59:35:25 Distance: 902.37

Begin Computations:
The survey is done clockwise
Total Uncorrected Angle: 359:57:43
Total err in seconds 137.0
Adjusted angle 1 -> 72:2:42
Adjusted angle 2 -> 131:11:10
```

# UTM to KML



# Future

- Open source
  - Community effort






# Acknowledgement

- A senior programmer who would like to keep anonymous.



# Thank You!



Questions?

Comments?

Suggestions?

[zhangy2@sfasu.edu](mailto:zhangy2@sfasu.edu)

Yanli Zhang, Matthew McBroom and Daniel R. Unger  
*Arthur Temple College of Forestry and Agriculture  
Stephen F. Austin State University*