

Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code. Please read the Rules and Regulations from the U of T Governing Council (especially the Code of Behaviour on Academic Matters).

Please don't copy. We want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

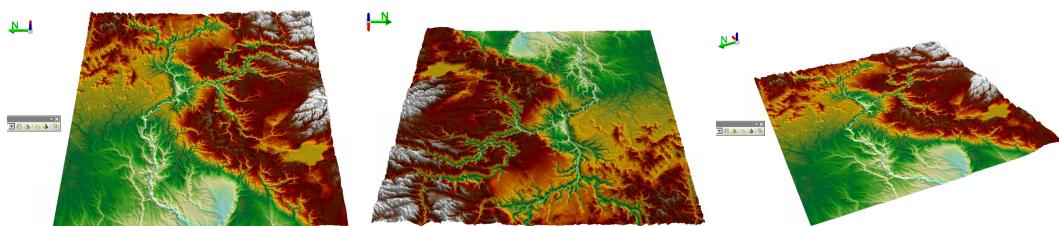
Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in CS involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too. The easiest way to avoid plagiarism is to only discuss a piece of work on the course discussion board, with the CSC108H TAs in the open labs, or the CSC108H instructors in office hours.

Assignment 2

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to the course Piazza page.

This assignment is due Nov. 12th, by 10:00pm

Consider a section of the Earth's surface from a birds-eye-view. The elevation may vary greatly across this section. See the Figures below as an example. If one were interested in analysing this area based on elevation, one could represent the section of Earth as a matrix of numerical values, where each cell of the matrix would indicate the elevation of the Earth at that particular location. For example, an area the size of 1 square kilometer of Earth could be encoded as a 1000x1000 matrix, where each cell is the elevation of a single square meter; here, the value at cell (500,500) of the matrix, would be the elevation at approximately the middle of the original 1 square kilometer section.



For this assignment, you will implement several functions which will allow someone with such a matrix of values to perform meaningful analysis on the area of the Earth's surface the matrix represents. You are given a file, `assignment2.py`, with six incomplete functions. For this assignment, **you're required to complete these functions**. A description regarding the intended behaviour of these functions is given later in this document. Further documentation and examples for these functions are given in the docstrings within the starter code: `assignment2.py`.

For the purposes of this assignment we will use the following definitions.

1. An *elevation map* is of the type `List[List[int]]`, and moreover, the length of an elevation map equals the length of all elements within the elevation map. An elevation map will only contain positive numbers. An example of an elevation map is:

```
valid_map = [[1,2,3],[4,5,6],[7,8,9]]
```

It may be more intuitive to view the map as:

```
valid_map = [[1,2,3],  
            [4,5,6],  
            [7,8,9]]
```

The following two examples are not elevation maps (note the length of all the lists in each):

```
invalid_map1 = [[1,2,3],[4,5],[6,7,8,9]]  
invalid_map2 = [[1,2,3],[4,5,6],[7,8,9],[10,11,12]]
```

2. A *cell* is of the type `List[int]` and has a length of 2. All values in a cell will be greater than or equal to 0. Within an elevation map m , we say cell $[i, j]$ as a shorthand for $m[i][j]$. We also say cell $[i, j]$ is adjacent to cell $[n, m]$ if and only if $[n, m]$ equals one of:
 $[i + 1, j + 1], [i + 1, j], [i + 1, j - 1], [i, j - 1], [i, j + 1], [i - 1, j + 1], [i - 1, j],$ or $[i - 1, j - 1]$.
3. Within an elevation map m , cell $[i, j]$ is a *sink* if for all adjacent cells $[n, m]$, $m[i][j] \leq m[n][m]$. With the physical interpretation of an elevation map in mind, water would collect in sinks, since there is no less elevated area in the immediate vicinity for the water to flow to.

Functions

You are required to implement all of the functions below. Pay attention to parameters of each function, for example, if it is said an input will be an elevation map, you can trust your function will never be tested on input which isn't an elevation map. For further examples of how these functions are intended to operate, view the docstrings of the starter code for this assignment.

1. `get_average_elevation(List[List[int]]) -> float`

The first parameter is an elevation map, m . Returns the average elevation across all the land in m .

2. `find_peak(List[List[int]]) -> List[int]`

The first parameter is an elevation map, m . Returns the cell which contains the highest elevation point in m . For the purposes of us testing this function, you may assume that all values of m are unique (no two locations have equal elevations).

3. `is_sink(List[List[int]], List[int]) -> bool`

The first parameter is an elevation map, m , the second parameter is a cell, c . Returns True if and only if c is a sink in m . Note if c does not exist in m (the values are outside m 's dimensions), this function returns False. See the previous section for the definition of a sink.

4. `find_local_sink(List[List[int]], List[int]) -> List[int]`

The first parameter is an elevation map, m , the second parameter is a cell, c , which exists in m . Returns the local sink of c . A local sink of c is the cell which water would flow to if it started at c . Assume if the current location isn't a sink, water will always flow to the adjacent cell with the lowest elevation. You may also assume for the purposes of us testing this function, that all values of m are unique (no two locations have equal elevations). See the docstring for some examples.

5. `can_hike_to(List[List[int]], List[int], List[int], int) -> bool`

The first parameter is an elevation map, m , the second is start cell, s which exists in m , the third is a destination cell, d , which exists in m , and the forth is the amount of available supplies. Under the interpretation that the top of the elevation map is north, you may assume that d is to the south-east of s (this means it could also be directly south, or directly east). The idea

is, if a hiker started at s with a given amount of supplies could they reach f if they used the following strategy. The hiker looks at the cell directly to the south and the cell directly to the east, and then travels to the cell with the lower change in elevation. They keep repeating this stratagem until they reach d (return True) or they run out of supplies (return False). Assume to move from one cell to another takes an amount of supplies equal to the change in elevation between the cells. See the docstring for some examples. If the change in elevation is the same between going East and going South, the hiker will always go East. Also, the hiker will never choose to travel South, or East of d (they won't overshoot their destination). That is, if d is directly to the East of them, they will only travel East, and if d is directly South, they will only travel South.

6. `rotate_map(List[List[int]]) -> None`

The parameter is an elevation map, m . Under the interpretation that the top of m is north, the function mutates m such that the top of m would now be viewed as east. See the docstring for some examples.

Submitting and Grading

This assignment will be submitted electronically via MarkUs. Please find the MarkUs link on the course website. Note, to avoid potential confusion and submitting to the wrong location, there will be nowhere on MarkUs to submit Assignment 2 until the Assignment 1 resubmit is past due.

This assignment is worth 10% of your final grade. Grading is done completely automatically. That is, a program calls your function, passes it certain arguments, and checks to see if it returns the expected output. Each function is worth 20% of assignment grade, with the exception of `get_average_elevation` and `find_peak` which are each worth 10% of the assignment's grade. For any one function, if you pass n of the m tests we run on that function, your grade for that function will be n/m .

Shortly after the deadline, you will receive your grade. If you are not content with this grade, you may resubmit your assignment up to 48 hours after the original deadline with a 20% penalty. If you choose to resubmit, your final grade on the assignment will be the higher of the two grades (the original submission, and the re-submission with a 20% penalty). Good luck!

Additional Material

You will note the starter code also has a `create_real_map()` function. This will allow you to create an elevation map from the real world data found in the `data.csv` file. To properly generate the map, make sure `data.csv` is in the same directory as `assignment2.py` when you run the function.

This assignment was developed with the aid of the GIS department here at UTM. They've been kind enough to give a little background regarding the applications and source of the data we're using. If this kind of stuff interests you, I highly suggest you look into the program here at UTM; it is very popular with students who take 108.

Near the turn of the millennium, an international research effort was undertaken to acquire the most complete and high-resolution digital topographic database of Earth. During an 11 day mission, the Space Shuttle Endeavor was fitted with a synthetic aperture altimeter capable of resolving the elevation of the Earth's surface at 30m resolution. The data collected during the Shuttle Radar Topography Mission (SRTM) formed the first high resolution digital elevation model (DEM) of the globe that was homogeneous in data quality and freely available. The SRTM DEM has been used by over 1 million users from 221 countries in applications ranging from agricultural planning, dam breakage/flooding

risk assessment, natural hazard assessment, and countless others. The DEM data in `data.csv`, and seen in the figures previously given in this document is a 1200x1200 grid where each pixel (80mx80m) represents the elevation in meters. The area is the north-eastern US near the Wallowa-Whitman National Forest which straddles Washington and Idaho.

The space around us shapes our daily lives in more ways than we can imagine. Hospitals route ambulances through network analysis, accounting for the slope/curvature of roads to save time and reduce the strain on paramedics as they treat patients. Many institutions are incorporating spatial analysis and geographic information systems (GIS) into their strategic and everyday business planning. If you would like to know more about GIS, please see the courses offered by the Department of Geography and Programs in Environment (<http://geog.utm.utoronto.ca/gis>).

Good luck!