Lecture 2

Lists, conditions, loops

Programming in geoinformatics

Autumn 2017

Coordinate check

1. Ask the user for **latitude** and **longitude** with `raw_input()` and check if they are valid

Coordinate check

1. Ask the user for **latitude** and **longitude** with `raw_input()` and check if they are valid
2. If the value is out of bounds, we **change it** to the nearest valid one:

Coordinate check

1. Ask the user for **latitude** and **longitude** with `raw_input()` and check if they are valid
2. If the value is out of bounds, we **change it** to the nearest valid one:
    - latitude: (-90, 90)

Coordinate check

1. Ask the user for **latitude** and **longitude** with `raw_input()` and check if they are valid
2. If the value is out of bounds, we **change it** to the nearest valid one:
    - latitude: (-90, 90)
    - longitude: (-180, 180)

Coordinate check

1. Ask the user for **latitude** and **longitude** with `raw_input()` and check if they are valid
2. If the value is out of bounds, we **change it** to the nearest valid one:
    - latitude: (-90, 90)
    - longitude: (-180, 180)
3. Example: `lat = 98 → 90`

Do we have enough cars?

1. Assign numeric values to variables **carsAmount** and **peopleAmount**

Do we have enough cars?

1. Assign numeric values to variables **carsAmount** and **peopleAmount**
2. Assume that 5 people fit in a car

Do we have enough cars?

1. Assign numeric values to variables **carsAmount** and **peopleAmount**
2. Assume that 5 people fit in a car
3. Print if you have enough cars for all the people

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**
2. Terrain types:

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**
2. Terrain types:
   - $0 - 300 \rightarrow$ flats

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**
2. Terrain types:
    - 0 – 300 → flats
    - 300 – 800 → low hills

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**
2. Terrain types:
   - 0 – 300 $\rightarrow$ flats
   - 300 – 800 $\rightarrow$ low hills
   - 800 – 1500 $\rightarrow$ hills

Elevation classification

1. Write an **if statement** that will print how flat the terrain is based on the **relative elevation change**
2. Terrain types:
    - 0 – 300 $\rightarrow$ flats
    - 300 – 800 $\rightarrow$ low hills
    - 800 – 1500 $\rightarrow$ hills
    - 1500+ $\rightarrow$ mountains

Rock, paper, scissors

1. Create the **rock, paper, scissors** game

Rock, paper, scissors

1. Create the **rock, paper, scissors** game
2. Let the user make a choice (get a numerical value – 1 = rock, 2 = paper, 3 = scissors)

Rock, paper, scissors

1. Create the **rock, paper, scissors** game
2. Let the user make a choice (get a numerical value – 1 = rock, 2 = paper, 3 = scissors)
3. Create a bot variable with a random value:

   ```
   import random
   bot = random.randint(1,3)
   ```

Rock, paper, scissors

1. Create the **rock, paper, scissors** game
2. Let the user make a choice (get a numerical value – 1 = rock, 2 = paper, 3 = scissors)
3. Create a bot variable with a random value:
   ```
   import random
   bot = random.randint(1,3)
   ```
4. Check **who won** or if it's a **tie**

List sums

1. **Calculate the sum** of random values in a list that is always 4 or 5 items long:

   ```
   from random import random
   numbers = [random(), random(), random(),
   random(), random()]
   ```

   or

   ```
   numbers = [random(), random(), random(),
   random()]
   ```

List sums

1. **Calculate the sum** of random values in a list that is always 4 or 5 items long:

   ```
   from random import random
   numbers = [random(), random(), random(),
   random(), random()]
   ```

   or

   ```
   numbers = [random(), random(), random(),
   random()]
   ```

2. use: **if** condition, **len()** function and get the items with **[] notation**

List sums

1. **Calculate the sum** of random values in a list that is always 4 or 5 items long:

```
from random import random
numbers = [random(), random(), random(),
random(), random()]
```

or

```
numbers = [random(), random(), random(),
random()]
```

List sums

1. **Calculate the sum** of random values in a list that is always 4 or 5 items long:

   ```
   from random import random
   numbers = [random(), random(), random(),
   random(), random()]
   ```

   or

   ```
   numbers = [random(), random(), random(),
   random()]
   ```

2. use: **if** condition, **len()** function and get the items with **[] notation**

Line topology

1. Some terminology:

Line topology

1. Some terminology:
   - Point: [4, 16]

Line topology

1. Some terminology:
    - Point: [4, 16]
    - Line: [[1,1], [2,5], [6,8]] (list of points)

Line topology

1. Some terminology:
    - Point: [4, 16]
    - Line: [[1,1], [2,5], [6,8]] (list of points)
    - Multiline: [[[0, 0], [1, 2]], [[5, 1], [4, 8], [5, 10]], [[1, 1], [5, 4], [8, 10], [6, 1]]] (list of lines)

Line topology

1. Some terminology:
   - Point: [4, 16]
   - Line: [[1,1], [2,5], [6,8]] (list of points)
   - Multiline: [[[0, 0], [1, 2]], [[5, 1], [4, 8], [5, 10]], [[1, 1], [5, 4], [8, 10], [6, 1]]] (list of lines)
2. Points **a..e** are defined: `a = [47,18]; b = [49,19]; c = [48,18]; d= [44, 19]; e= [42, 17]`

LINE TOPOLOGY

1. Some terminology:
   - Point: [4, 16]
   - Line: [[1,1], [2,5], [6,8]] (list of points)
   - Multiline: [[[0, 0], [1, 2]], [[5, 1], [4, 8], [5, 10]], [[1, 1], [5, 4], [8, 10], [6, 1]]] (list of lines)
2. Points **a..e** are defined: `a = [47,18]; b = [49,19]; c = [48,18]; d= [44, 19]; e= [42, 17]`
3. Create three lines: `l1 = [a, d]; l2 = [c, d, b]; l3 = [e, a, c]`

LINE TOPOLOGY

1. Some terminology:
   - Point: [4, 16]
   - Line: [[1,1], [2,5], [6,8]] (list of points)
   - Multiline: [[[0, 0], [1, 2]], [[5, 1], [4, 8], [5, 10]], [[1, 1], [5, 4], [8, 10], [6, 1]]] (list of lines)
2. Points **a..e** are defined: `a = [47,18]; b = [49,19]; c = [48,18]; d= [44, 19]; e= [42, 17]`
3. Create three lines: `l1 = [a, d]; l2 = [c, d, b]; l3 = [e, a, c]`
4. Create a multiline that contains lines l1, l2, l3

Line topology

1. Some terminology:
   - Point: [4, 16]
   - Line: [[1,1], [2,5], [6,8]] (list of points)
   - Multiline: [[[0, 0], [1, 2]], [[5, 1], [4, 8], [5, 10]], [[1, 1], [5, 4], [8, 10], [6, 1]]] (list of lines)
2. Points **a..e** are defined: `a = [47,18]; b = [49,19]; c = [48,18]; d= [44, 19]; e= [42, 17]`
3. Create three lines: `l1 = [a, d]; l2 = [c, d, b]; l3 = [e, a, c]`
4. Create a multiline that contains lines l1, l2, l3
5. Calculate centerL1, centerL2, centerL3 with **average** values of **lat** and **lon** of the points in the line

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`
2. Check how many sides it has and print whether:

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`
2. Check how many sides it has and print whether:
   - it's not a valid polygon (has less than three sides)

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`
2. Check how many sides it has and print whether:
   - it's not a valid polygon (has less than three sides)
   - it's a triangle

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`
2. Check how many sides it has and print whether:
   - it's not a valid polygon (has less than three sides)
   - it's a triangle
   - it has four sides

Polygon sides

1. Polygon is defined just like a line:
   `polygon = [[1,1], [2,5], [6,8]]`
2. Check how many sides it has and print whether:
   - it's not a valid polygon (has less than three sides)
   - it's a triangle
   - it has four sides
   - it's a more complex polygon