Lecture 7

Exercises

Programming in geoinformatics

Autumn 2017

Exercise 1

With a list such as:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

- create a function `smallerThan(list, number)` which will return a new list with numbers from the original `list` smaller than `number`
- if no `number` is specified when the function is called, ask the user for the `number`

Example:

```
print smallerThan(a, 5) # [1, 1, 2, 3]
print smallerThan([4, 8, 93, 763, 34], 50) # [4, 8, 34]
```

Exercise 2

- create a function `checkPalindrome(word)`, which will check whether a `word` (or text) is a palindrome (is symmetrical, the first letter is the same as last, etc.)

```
print checkPalindrome("racecar") # True
print checkPalindrome("programming") # False
```

- you don't have to (but can) use loops

Homework 1

- create a function `generatePassword(length, specialChars=False)`, which will generate a random password of specified length (using the same password on more accounts is dangerous) from lowercase and uppercase letters and digits
- if special characters are provided when calling the function, include these characters as well
- use the following as baseline:

```
import random
import string
characters = string.ascii_letters
numbers = "".join([str(n) for n in range(10)]) # the same as
    ↪   "0123456789"
```

- **hint:** use `random.choice()` function and concatenate all the possible characters

Example:

```
print generatePassword(10) # kJmgcCBlTK
print generatePassword(10, ",.-=!?") # tne==X?OXN
```

Homework 1 Bonus

2 points

- modify the function to `generatePassword(length, specialChars=False, readable=False)`, which will also remove any visually similar characters from the set of possible characters (e.g. 0, O, l - lowercase L, I - uppercase I, 1, etc.) to avoid mistakes when copying and pasting passwords vs. reading them from the screen.
- make all sets of possible characters (lowercase letters, uppercase letters, digits, special characters) as likely to appear in the password (pick a category at random for each character first and then a random character from the category)
- modify the function so it's possible to choose categories from which the characters will be picked (so it's possible to generate passwords only from lowercase letters and special characters)
- check whether the generated password includes characters from all the categories provided. If it doesn't, generate a new one (until all categories are represented)

Homework 2

- create a function `getPosition()`, which will return a position (coordinates) of a the `"x"` character in a random board (some board game for example)
- use the following as baseline:

```
import random
size = random.randint(3,8)
x, y = random.randint(1,size-1), random.randint(1,size-1)
board = [["o"] * size for i in range(size)]
board[x][y] = "x"
for row in board:
print " ".join(row) # for checking visually
print ""

def getPosition():
# your code here
```

- **hint:** use `in` keyword and loops

Homework 3

- create a function `factorial(n)` which will return the factorial of specified number `n` (n! $\rightarrow$ factorial of n)

$$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$$

- **hint:** use loops if you want OR you can try using the function inside itself recursively!

Example:

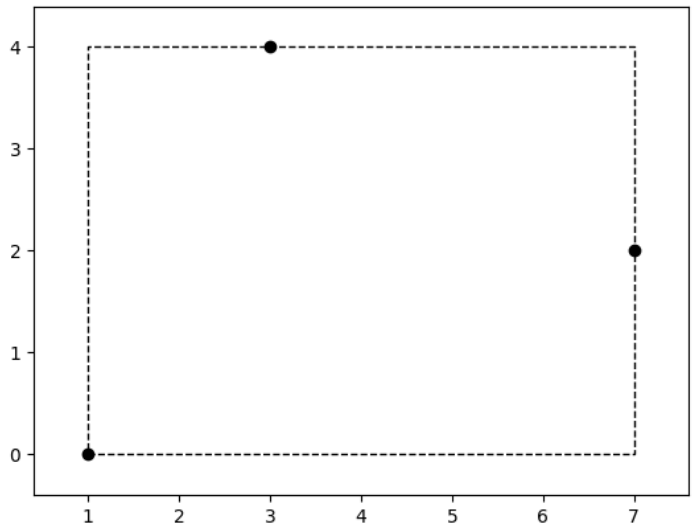
```
print factorial(8) # 40320
```

Homework 4

Create a function `boundingBox(geometry)`, which takes a geometry object as an argument (e.g. `[[1, 0], [3, 4], [7, 2]]`) and returns its bounding box (MBR – minimum bounding rectangle) – rectangle which contains the whole object.

Example:

```
print boundingBox([[1, 0], [3, 4], [7, 2]]) # [[1, 0], [7, 4]] →
    ↪   lower left, upper right corner coordinates
```

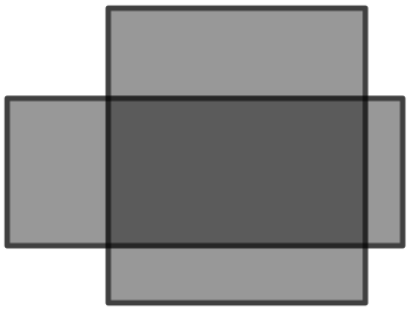


Bonus Homework

3 points

- create a function `BBoxOverlap(bbox1, bbox2)`, which will return whether two bounding boxes overlap.

- you should also consider the following situation:



- if you don't figure out how to check this type of situation, submit your code anyway for less points