Lecture 3

Loops, dictionaries

Programming in geoinformatics

Autumn 2017

For loops

- When we need to repeat a code for *n* elements

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
for letter in alphabet:
    print letter
```

For loops

- When we need to repeat a code for *n* elements

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
for letter in alphabet:
    print letter
```

- We can use **range()** to create a list of numbers:

```
range(5) # → [0, 1, 2, 3, 4]
range(2,8) # → [2, 3, 4, 5, 6, 7]
range(len(alphabet)) # → [0, 1, 2, … 24, 25]
```

For loops

- When we need to repeat a code for *n* elements

```
alphabet = 'abcdefghijklmnopqrstuvwxyz'
for letter in alphabet:
    print letter
```

- We can use **range()** to create a list of numbers:

```
range(5) # → [0, 1, 2, 3, 4]
range(2,8) # → [2, 3, 4, 5, 6, 7]
range(len(alphabet)) # → [0, 1, 2, … 24, 25]
```

- That way we can use the item's index too!

```
for position in range(len(cities)):
print "City no.", position + 1, cities[position]
```

Conditionals in loops

The last homework in a loop:

```
polygons = [
[[1,7], [1,3], [2,3], [2,7]], # polygon 1
[[1,1], [1,5], [3,3]], # polygon 2
[[0,0], [0,5], [2,10], [4,7], [3,10], [8,2]] # polygon 3
]

for polygon in polygons:
    if len(polygon) < 3:
        print "This is not a polygon!"
    elif len(polygon) == 3:
        print "This is a triangle"
    elif len(polygon) == 4:
        print "This has four sides (might be a square?)."
    else:
        print "This is a more complex polygon."
```

Line with random coordinates

Some useful stuff. Need a random line for testing?

```
import random
length = 10
line = []
for i in range(length):
    coordinates = [random.random() * 1000, random.random() * 1000]
    line.append(coordinates)
```

While loop

While loop runs until a condition is no longer true:

```
# Blackjack! Dealer draws until he has 17 points.
total = 0
while total < 17:
    total += 2 * random.randint(1,5) # blackjack cards values are
        ↪  2, 4, 6, 8, 10 (also 1 and 11, whatever)
if total > 21:
    print "The dealer lost!"
else:
    print "The dealer has " + str(total) + " points."
```

While loop

Beware **infinite loops!**

```
i = 5
while i < 6:
    print i
```

This will never end!

Exercise 1

Two lists are defined:

```
hats = ["red", "black", "blue", "yellow"]
ids = [2, 7, 15, 22, 25, 34]
```

1. Create a new list **idsNew**, where
   - you add 1 to even numbers
   - you subtract 1 from odd numbers
   - e.g. $3 \rightarrow 2$; $8 \rightarrow 9$
2. Print pairs in **idsNew** and **hats**, such as: `"person with id 5 has a red hat"`

Exercise 2

Cards in a card game have set values:

- points are set:
  - 7, 8, 9, 10 cards $\rightarrow$ 7, 8, 9, 10 points
  - J, Q, K $\rightarrow$ 10 points
  - A $\rightarrow$ 15 points
- suite of the card multiplies its points:
  - hearts $\rightarrow$ 4$\times$
  - diamonds $\rightarrow$ 3$\times$
  - spades $\rightarrow$ 2$\times$
  - clubs $\rightarrow$ 1$\times$

1. Print each card combination
2. Calculate the value of every card and `if value > 30` print it to console

Dictionaries

- Defined as: `dict = {}`
- Has its own keys compared to list's indexes:

```
# great way to store coordinates
coords = { "lat": 49.1876, "lon": 16.3273, "elev": 420.3 }
# or attributes
row = { "id": 0, "city": True, "name": "Brno", "population":
    ↪   377440, "ranking": 2, "latitude": 49.2, "longitude": 16.6,
    ↪   "universities": ["MUNI", "VUT", "MENDELU", "VFU", "JAMU",
    ↪   "UNOB"] }
```

Dictionaries

We can access dictionary values similarly to lists:

```
print coords["lat"]
print row["population"]

# if we don't have numbers as keys, this will not work:
print row[0]

# but this will
dict = {1: "a", 2: "b"}
print dict[1]
# again, this won't
print dict[0]
```

Dictionaries

Some methods:

```
a_dict = {'a': 156, 'b': 89, 'c': 41, 'd': 547}
print a_dict.items() # [('b', 89), ('a', 156), ('d', 547), ('c',
    ↪   41)]
print a_dict.keys() # ['b', 'a', 'd', 'c']
print a_dict.values() # [89, 156, 547, 41]
print len(a_dict) # 4

# get() returns None if key is not defined and not an error
print a_dict.get(4) # None
print a_dict.get('a') # 156
```

Sets

Similar to a dictionary:

```
points = { 9, 7, 9, 10, 3 } # → set([9, 10, 3, 7]) - no duplicities
points.add(15) # set([15, 9, 10, 3, 7])
```

Useful for **union**, **intersect**, **...** operations.

Exercise 3

A dictionary is defined:

```
countriesStats = {
    'Nigeria' : {'area': 923768, 'population': 182202000,
       ↪   'languages': ['English']},
    'South Africa' : {'area': 1219912, 'population': 54490000,
       ↪   'languages': ['Zulu', 'Xhosa', 'Afrikaans', 'English']
       ↪   },
    'Ethiopia' : { 'area': 1127127, 'population': 99391000,
       ↪   'languages': ['Amharic']}
}
```

1. find your favourite African country and add it to the dictionary (after the dict definition)
2. calculate the population density for every country
3. list all languages spoken in those countries with no duplicates (use **sets**)

Homework 1

We have two rivers:

```
riverA = [[3,7], [3,9], [4,11], [6,12]]
riverB = [[12,4], [10,6], [6,7], [3,9], [2,4]]
```

1. Calculate their distances using loops
2. *Hints*
   - use `for i in range(…)`
   - you have to get the value of **two points** in each cycle, **not one**
3. Make a list of points the rivers have in common
4. *Hint:* `point in line`

Bonus homework

2 points

Write a script that will calculate the area of a polygon. Assume it is a **valid simple polygon:** has only one part, edges are straight and the edges are not crossing each other.

How to calculate a polygon area:
`http://www.mathopenref.com/coordpolygonarea.html`

$$area = \left| \frac{(x_1 y_2 - y_1 x_2) + (x_2 y_3 - y_2 x_3) \ldots\ldots + (x_n y_1 - y_n x_1)}{2} \right|$$