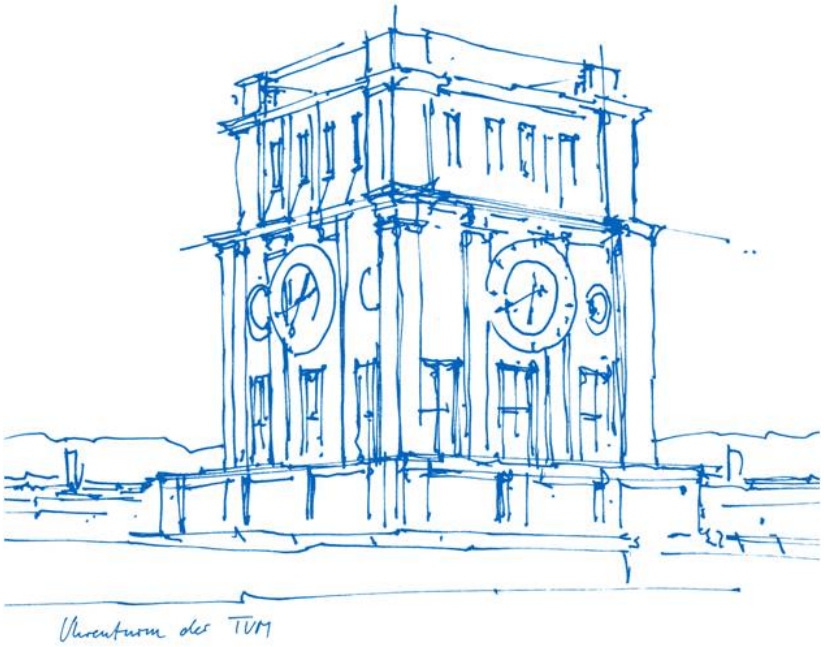
Exercises for Social Gaming and Social Computing (IN2241 + IN0040) – Introduction to Exercise 1



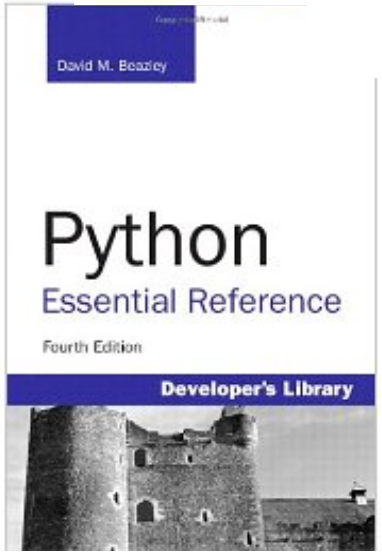
Exercise Content

Sheet Number	Exercise	Working Time
1	• Introduction to Python: basic Python programming language exercises • Graph Drawing using igraph	Monday, May 27 - Monday, June 3, 24:00
2	• Centrality measures	Monday, June 3 - Monday, June 17, 24:00
3	• Recommender Systems as an example for systems using simple forms of social context: Collaborative Filtering	Monday, June 17 - Monday, June 24, 24:00
4	• Inferring social tie strength from activity data in social networking platforms with linear regression	Monday, June 24- Monday, July 01, 24:00
5	• finding social groups with clustering methods on profile date (K-Means) and graph clustering (Girvan Newman method)	Monday, July 01 - Monday, July 08, 24:00
6	• analyzing short-term social context using mobile interaction data (Reality Mining)	Monday, July 08 - Monday, July 15, 24:00

Repitition: Python and IPython Books

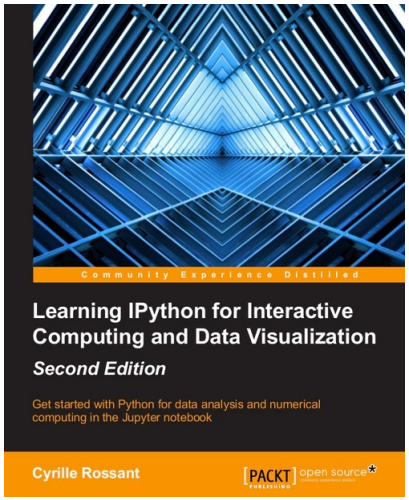
- **Learning Python:**
  Python Essential Reference (2012)
  by David M. Beazley, Safari Books
  (especially **chapter 1: A Tutorial Introduction (25 pages))**

  free eAccess: https://eaccess.ub.tum.de/login



- **Learning IPython / Reference for IPython:**
  Learning IPython for Interactive Computing and Data Visualization (SECOND EDITION) by Cyrille Rossant, 175 pages, Packt Publishing, October 25 2015
  (Especially (free) chapter 1.4. A crash course on Python)

  free access: http://nbviewer.ipython.org/github/ipython-books/minibook-2nd-code/blob/master/chapter1/14-python.ipynb
  (do not try to open this ipynb with Jupyter directly. Instead, download all the ipynb's from the book from Github: https://github.com/ipython-books/minibook-2nd-code → 14-python.ipynb)



Repitition: Installation using Docker

- **install Docker** as a platform specific software

  e.g. on a Mac:

  ```
  brew cask install docker
  ```

- using the **Dockerfile** (downloaded from our Moodle) you **built a Docker image** (contains an Ubuntu, IPhyton environment (Anaconda), Pythin libs (numpy, scikitlearn, pandas, igraph etc.) :

  ```
  cd /path/to/your/copy/of/dockerfile
  docker build -t dockerfile .
  ```

Repitition: Running using Docker

- you **run the docker container** and **„mount“** a directory in your file system as a working directory, in which you copy the **ipython sheets of the exercises** which you modify and expand and submit :_

  e.g. on a Mac in a command shell you run:

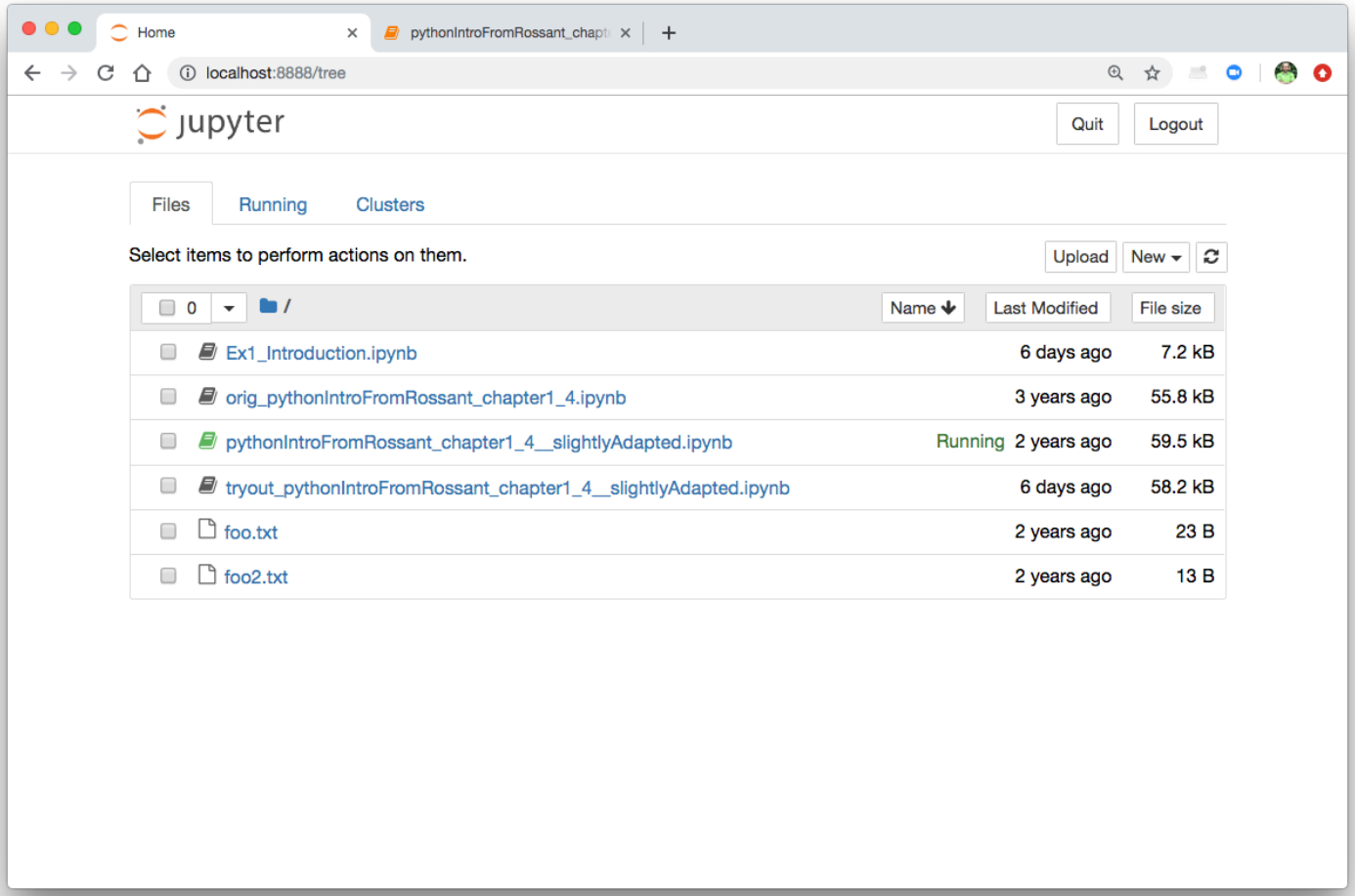
```
docker run -it -p 8888:8888 -v
        your/path/to/working/space/:
              /home/jupyter arbitraryimagename
```

- what you get is an **output** similar to:

```
To access the notebook, open this file in a browser
      file:///home/jupyter/.local/share/jupyter/runtime/
            nbserver-5-open.html
Or copy and paste one of these URLs:
      http://(97cb65496918 or 127.0.0.1):8888/
            ?token=da81d515702f701r55t54e17f759
                  8d9137f22d067c074rff
```

Repitition: Running using Docker

- using the URL from the output in a browser on your system you get something like this:



Problem 1.1: Fibonacci Series

Problem 1.1: Fibonacci Series

Write a Python method called fibonacci that takes an integer number N as argument and returns the Fibonacci sequence of length N.

Note: A Fibonacci sequence is a sequence of numbers where each number is the sum of the two previous ones. For example, the Fibonacci sequence of length 5 is: 0, 1, 1, 2, 3 (or 1, 1, 2, 3, 5)

In [3]:
```
def fibonacci(N):
    # TODO: Implement the fibonacci function
```

- Simple recursion:

$$F(0) = 0; \quad F(1) = 1; \quad F(n) = F(n-1) + F(n-2)$$

- call `fibonacci(6)` should return `0,1,1,2,3,5,8`

Problem 1.2: Loops and User Input

Problem 1.2: Loops and User Input

In this section, your task is to program a small "Guess the number" game. A number between 1 and 100 should be randomly selected and the player should have 5 tries to guess that number correctly. Each time the player guesses, a hint should be given (higher/lower). At the end, the correct guessed or not guessed number as well as the needed tries to get there should be printed out in the console.

In [ ]:
```
print('Welcome to \'Guess My Number\'!\n')
print('I\'m thinking of a number between 1 and 100.')
print('Try to guess it in as few attempts as possible.\n')

# Initialize values
the_number = random.randint(1, 100)

# TODO: Implement the game
```

```
        Welcome to 'Guess My Number'!

I'm thinking of a number between 1 and 100.
Try to guess it in as few attempts as possible.

Take a guess: 50
Lower...
Take a guess: 25
Lower...
Take a guess: 12
Lower...
Take a guess: 6
('You guessed it! The number was', 6)
('And it only took you', 4, 'tries!\n')


Press the enter key to exit.
```

Your code

- Thinks of a number between 1 and 100
- Takes user input
- Tests if the number is too high or low
- Gives feedback (e.g., „higher“, „lower“)
- User has 5 guesses

Problem 1.3: Graph Visualization and Analysis with IGraph Library

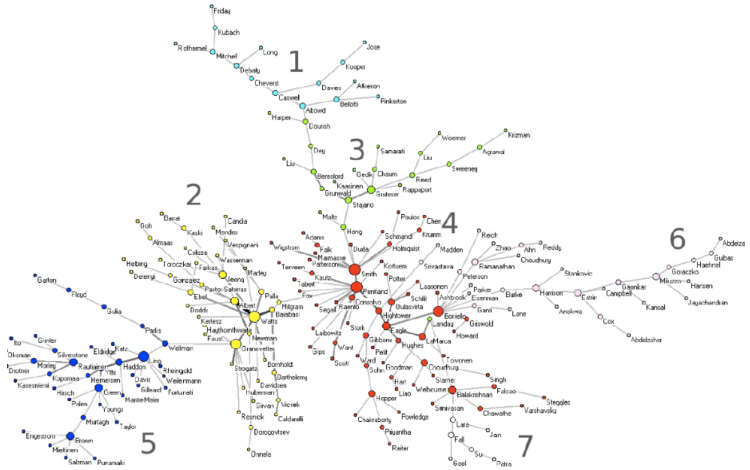
Purposes of social network visualizations:

- Get an **overview** about a particular social network
- **Perceive structures** (clusters etc.) in the SN via human visual system
- **Show dynamic changes** in a network

Properties of good visualizations

- **optimal dispersion** (nodes should spread well)
- **crossing** minimalization (ideally planar graph)
- **clusters** correctly displayed
- dynamic visualization: **preserve** the **mental map**
- **....**

example: co-citation network in the field „mobile social networking“([6b])

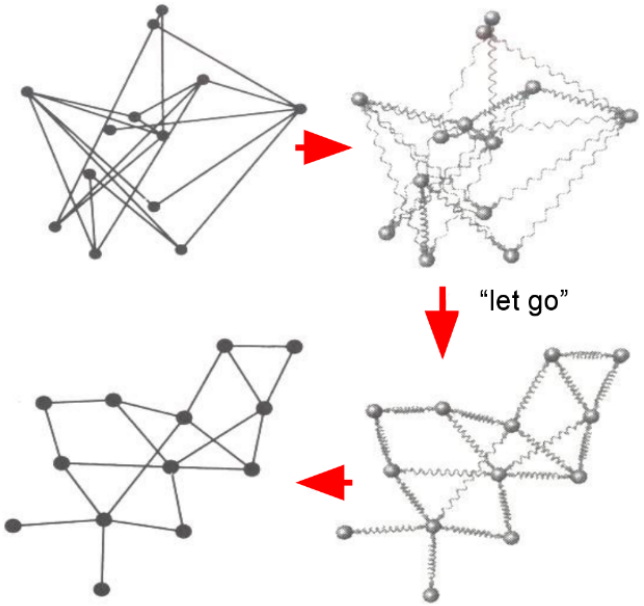


Algorithms help with visualization

An example of an visualization algorithm is **Fruchtermann & Reingold**

Problem 1.3: Graph Visualization and Analysis with IGraph Library

Fruchtermann & Reingold approach uses an analogy to classical mechanics for the graph modelling



$$\mathbf{F}_{repell}(\mathbf{v}_i, \mathbf{v}_j) = -\frac{l^2}{\| \mathbf{v}_i - \mathbf{v}_j \|^2} \frac{(\mathbf{v}_i - \mathbf{v}_j)}{\| \mathbf{v}_i - \mathbf{v}_j \|}$$

$$\mathbf{F}_{spring}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\| \mathbf{v}_i - \mathbf{v}_j \|^2}{l} \frac{(\mathbf{v}_i - \mathbf{v}_j)}{\| \mathbf{v}_i - \mathbf{v}_j \|}$$

$$\mathbf{F}_{repell} - \mathbf{F}_{spring} = 0 \quad \text{if} \quad \| \mathbf{v}_i - \mathbf{v}_j \| = l$$

- By introducing **forces between the nodes**, we receive edges with equal length and few crossings
- Mechanical analogon: springs between the nodes

Further reading:

- http://igraph.org/r/doc/layout_with_fr.html
- Graph visualization slides on Moodle (just FYI, relevant for exam: just these exercise intro slides

Problem 1.3: Graph Visualization and Analysis with IGraph Library

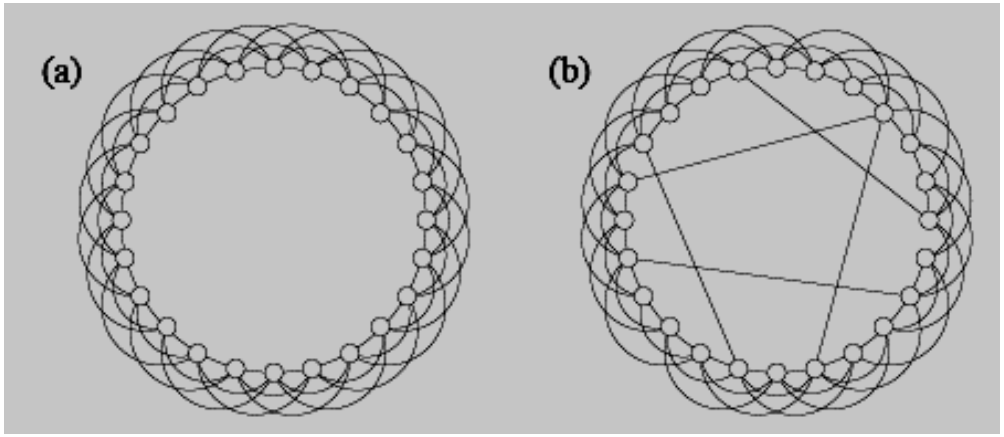
Simulated networks should fulfill two properties to be realistic:

- High **clustering coefficient**
- Small **average path length**

Watts & Strogatz 1998 **small world model**

- **L** nodes in regular **D**-dim. lattice + periodic boundary cond.; **D=1: Ring**
- Each node connected to neighbors in lattice at distance of most **k**
- „Rewiring“ of edges with probability **p**

before rewiring | after rewiring



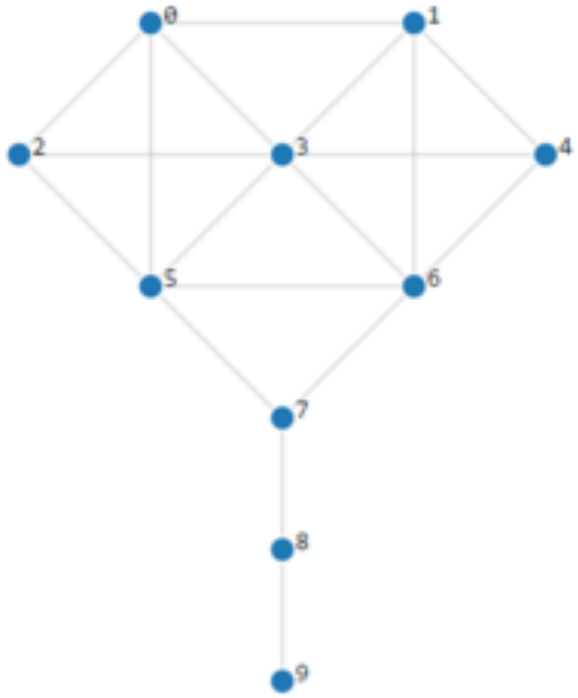
D = 1; L = 24; k = 3; p = 0 | D = 1; L = 24; k = 3; p > 0

Problem 1.3: Graph Visualization and Analysis with IGraph Library

Task 1: Graph Objects

When using igraph, there is the option of initializing Graph objects by hand or by passing an edgelist. Nevertheless, the library also includes famous graphs that can readily be instantiated and used. **Create a Graph object that represents the *Krackhardt Kite graph* which is a simple connected, unweighted and undirected social network. Then plot the graph.**

In [ ]:
```
# TODO: Create the Krackhardt Kite graph
```



Problem 1.3: Graph Visualization and Analysis with IGraph Library

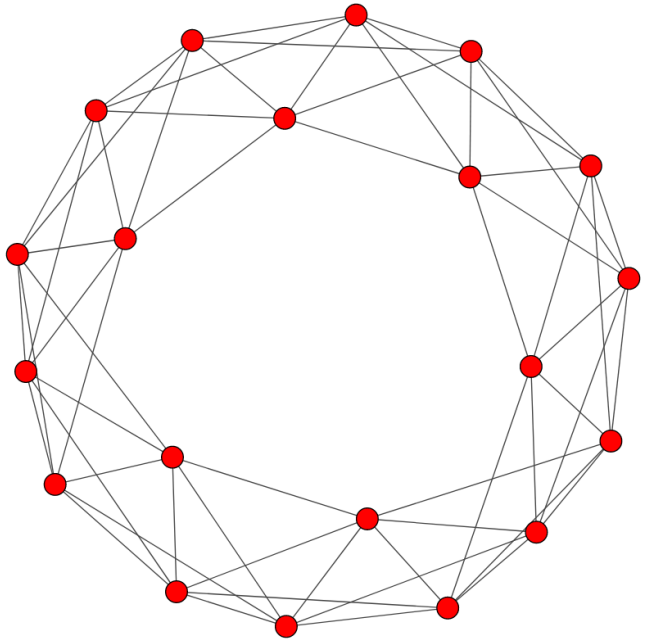
Task 2: Small World Graph

In the lecture, you learned about the small world graph, also known as *Watts-Strogatz graph*. **Create such a graph with 20 nodes and the following parameters: dimension 1, nodes connected to neighbors of distance of most 3, and rewiring probability 0**

Afterwards, vary the parameters (increase/decrease the dimension, rewiring probability, ...) and run your code again. Also set the layout to Fruchtermann-Reingold and see what happens. **What changes do you observe?** Don't write more than 5 sentences.

In [ ]:
```
# TODO: Create a Watts-Strogatz graph
```

TODO: Write your observations here!



Submitting Your Solution

- work by expanding the .ipynb iPython notebook for exercise 1 that you downloaded from Moodle.
- save your expanded .ipynb iPython notebook in your working directory.
  Submit your .ipynb iPython notebook via Moodle (nothing else)
- remember: working in groups is not permitted.
  Each student must submit her own ipynb notebook!
- we check for plagiarism. Each detected case will have the consequence of 5.0 for the whole exercise grade.
- Deadline:

Citations

(1) [Beazley 2013) David Beazley: Python Essential Reference, Safari Books 2013, E-Book available via www.ub.tum.de

(2) [Rossant 2015] Learning IPython for Interactive Computing and Data Visualization (SECOND EDITION) by Cyrille Rossant, 175 pages Packt Publishing, October 2015