TA 1

Arun Kumar Rajasekaran

Welcome Intro

- Name
- Where are you from?
- Subject interests?
- What reeled you in? Economics or the computational aspect?
- Any previous related experience you would like to share?
- Computing knowledge?
- Hobbies?

Python

https://www.python.org/downloads/

Jupyter

https://jupyter.org/

Google colab

https://colab.research.google.com/?utm_source=scs-index

NetworkX

https://networkx.org/documentation/stable/tutorial.html

iGraph

https://igraph.org/python/tutorial/latest/tutorial.html

Graph tool

https://graph-tool.skewed.de/static/doc/quickstart.html

graspologic

https://microsoft.github.io/graspologic/tutorials/simulations/erdos_renyi.html

NodeXL

https://www.smrfoundation.org/nodexl/features/

Coding standards

- Coding standards are guidelines for code style and documentation.
- They may be formal (IEEE) standards, or company specific standards.
- The aim is that everyone in the organization will be ableto read and work on the code.
- Coding standards cover a wide variety of areas:

– Program design

– Naming conventions

– Formatting conventions

– Documentation

– Use (or not) of language specific features

• Why bother with a coding standard?

– Consistency between developers

– Ease of maintenance and development

– Readability, usability

• Example should make this obvious!

• No standard is perfect for every application.

– If you deviate from the standard for any reason, document it!

Coding style

• There are several examples of coding styles. Often they differ from company to company

• They typically have the following in common:

– Names

• Use full English descriptors

• Use mixed case to make names readable

• Use abbreviations sparingly and consistently

• Avoid long names

• Avid leading/trailing underscores

– Documentation

• Document the purpose of every variable

• Document why something is done, not just what

Coding style

– Accessors

• Use getX(), setX() functions on all class variables.

– Member function documentation

• What & why member function does what it does

• Parameters/return value

• How function modifies object

• Preconditions/postconditions

• Concurrency issues

• Restrictions

– Document why the code does things as well as what it does.

Standards

• Standards rare documented agreements containing technical specifications or other precise criteria to be used consistently as guidelines, rules, or definitions of characteristics, to ensure that materials, products, processes and services are for for their purpose.

• International standards are supposed to contribute to making life simpler, and to increasing reliability and effectiveness of the goods and services we use.

• Standards represent best, or most appropriate, practice:
– They encapsulate historical knowledge often gained through trail and error.
– They preserve and codify organizational knowledge and memory
– They provide a framework for quality assurance.
– Ensure continuity over a project’s lifecycle.

Standards

There are many industry standards governing all aspects of software development:

– Terminology

– Notation

– Requirements gathering

– Design

– Coding

– Documentation

– Human computer interaction

– Verification and validation

– Quality assurance

– Even ethics!

Who writes standards?

– ISO

International Organization for Standardization

– SAA

Standards Australia

– BSI

British Standards Institute

– ANSI

American National Standards Institute

– IEEE

Institute for Electronic and Electrical

Engineers

– And about 80 or so others!

Relevant standards

• ISO 646 – 7-bit ASCII with national variants

• ISO 8859 – several 8-bit ASCII extensions:

– ISO 8859-1: West European languages (Latin-1)

– ISO 8859-2: East European languages (Latin-2)

– ISO 8859-5: Latin/Cyrillic

• ISO 6429 – ASCII control codes

• ISO 2382 – Information technology vocabulary

• ISO 8652 – the Ada programming language

• ISO 9899 – the C programming language

• ISO 9660 – CD-ROM volume and fie structure

• ISO 3166 – codes for the representation of names of counties:

– Defines a 2-letter, 3-letter and numeric code for every country.

– US/USA/840 = United States

– GB/GBR/826 = United Kingdom

• The 2-letter codes are well known as the internet top level domain names.

NetworkX

Feature

- Classes for graphs and digraphs.
- Conversion of graphs to and from several formats.
- Ability to construct random graphs or construct them incrementally.
- Ability to find subgraphs, cliques, k-cores.
- Explore adjacency, degree, diameter, radius, center, betweenness, etc.
- Draw networks in 2D and 3D.

NetworkX is suitable for operation on large real-world graphs: e.g., graphs in excess of 10 million nodes and 100 million edges. Due to its dependence on a pure-Python "dictionary of dictionary" data structure, NetworkX is a reasonably efficient, very scalable, highly portable framework for network and social network analysis.

Original author(s)

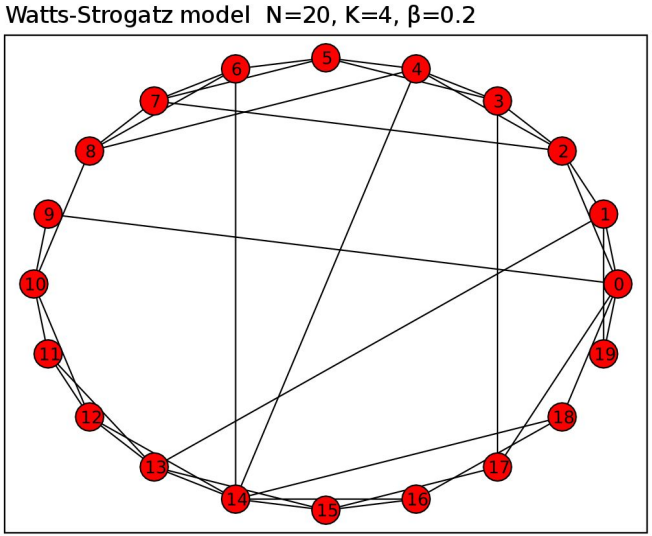
Aric Hagberg

Pieter Swart

Dan Schult

Developer(s)

Many others



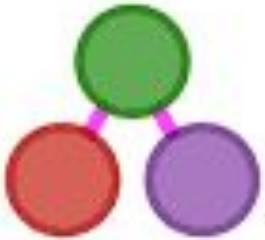
Initial release

11 April 2005; 18 years ago


NetworkX
Network Analysis in Python


igraph


SNAP




NetworKit


graph-tool

Networkx and igraph and graph tool

NetworkX is a pure-python implementation, whereas igraph is implemented in C. Here we select a few representative algorithms which are implemented in all three libraries, and test them on the same graph.

Example N=39,796 vertices and E=301,498 edges

Algorithm	graph-tool (16 threads)	graph-tool (1 thread)	igraph	NetworkX
Single-source shortest path	0.0023 s	0.0022 s	0.0092 s	0.25 s
Global clustering	0.011 s	0.025 s	0.027 s	7.94 s
PageRank	0.0052 s	0.022 s	0.072 s	1.54 s
K-core	0.0033 s	0.0036 s	0.0098 s	0.72 s
Minimum spanning tree	0.0073 s	0.0072 s	0.026 s	0.64 s
Betweenness	102 s (~1.7 mins)	331 s (~5.5 mins)	198 s (vertex) + 439 s (edge) (~ 10.6 mins)	10297 s (vertex) 13913 s (edge) (~6.7 hours)

Network science applications

Social Network Analysis

Transportation Network Analysis

Biological Network Analysis

Recommendation Systems

Internet routing etc, etc.