Лекція 20

Інсталятор pip

Основні поняття про пакет NetworkX

Установка нових пакетів у мові Python

Python має свій «репозиторій» різних пакетів, який має назву PyPi: Python Package Index

Він розміщений за адресою:

https://pypi.python.org/pypi.

Зараз там розміщено 90741 пакет і їх кількість постійно зростає. В репозиторії можливо безпосередньо скачати дані пакети.

Але в Python цей процес автоматизовано з використанням утиліти інсталятора пакетів pip.

Обновити версію в командному рядку Windows:

```
python -m pip install -U pip setuptools
```

Інсталятор pip

Інсталятор pip поставляється разом з Python

Під Windows ця утиліта запускається з командного рядка: «Пуск-Выполнить-cmd»

Довідку про pip можна одержати, виконавши команду:

pip –h

В результаті одержимо коротку інструкцію по роботі з pip.

Ця інструкція містить пояснення дії основних команд інсталятора pip та перелічує основні опції, що можуть застосовуватися разом з командами, модифікуючи роботу pip

Pip help

Usage:

```
pip <command> [options]
```

Commands:

install	Install packages.
uninstall	Uninstall packages.
freeze	Output installed packages in requirements format.
list	List installed packages.
show	Show information about installed packages.
search	Search PyPI for packages.
wheel	Build wheels from your requirements.
help	Show help for commands.

General Options:

-h, --help Show help.

-v, --verbose Give more output.

Option is additive, and can be used up to 3 times.

-V, --version Show version and exit.

-q, --quiet Give less output.

--log-file <path> Path to a verbose non-appending log, that only logs failures. This log is active by default at /root/.pip/pip.log.

--log <path> Path to a verbose appending log. This log is inactive by default.

--proxy <proxy> Specify a proxy in the form [user:passwd@]proxy.server:port.

Приклад використання pip

Розглянемо роботу pip на прикладі завантаження пакету wget.

Пакет wget містить утиліту, яка дозволяє завантажувати файли безпосередньо з програми на Python, використовуючи Internet.

В командному рядку Windows набираємо:

pip install wget

Ця команда запускає пошук останньої версії пакета wget в репозиторії PyPi.

Якщо такий пакет знайдено, то починається інсталяція пакета.

```
C:\Users\MICHAEL>pip install wget
Collecting wget
Installing collected packages: wget
Successfully installed wget-3.2
```

Команди pip (продовження)

Для видалення пакету wget потрібно набрати в командному рядку:

pip uninstall wget

Спочатку виводиться список файлів пакета

Виконується повторний запит на видалення.

Після підтвердження виконується видалення

```
C:\Users\MICHAEL>pip uninstall wget
Uninstalling wget-3.2:
  c:\program files (x86)\python35-32\lib\site-packages\__pycache__\wget.cpython-35.pyc
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\description.rst
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\installer
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\metadata
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\metadata.json
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\record
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\top_level.txt
  c:\program files (x86)\python35-32\lib\site-packages\wget-3.2.dist-info\wheel
  c:\program files (x86)\python35-32\lib\site-packages\wget.py
Proceed (y/n)? y
  Successfully uninstalled wget-3.2
```

Способи інсталяції з допомогою pip

Приклад 1.

1.Інсталяція найновішої версії

```
pip install 'SomeProject'
```

2.Інсталяція специфічної версії

```
pip install 'SomeProject==1.4'
```

3.Інсталяція більше або дорівнює одній з версій, і менше, ніж інша

```
pip install 'SomeProject>=1,<2'
```

4.Інсталяція версії, яка сумісна з конкретною версію

```
pip install 'SomeProject~=1.4.2'
```

Команди pip (продовження)

Для перегляду списку встановлених пакетів набираємо:

pip list

```
C:\Users\MICHAEL>pip list
algorithms (1.0)
numpy (1.11.1)
pip (8.1.2)
pygame (1.9.2b1)
pyreadline (2.1)
setuptools (25.1.6)
wget (3.2)
wheel (0.29.0)
```

Команди pip (продовження)

Для перегляду інформації про пакет wget набираємо

pip show wget

```
C:\Users\MICHAEL>pip show wget
---
Metadata-Version: 2.0
Name: wget
Version: 3.2
Summary: pure python download utility
Home-page: http://bitbucket.org/techtonik/python-wget/
Author: anatoly techtonik <techtonik@gmail.com>
Author-email: UNKNOWN
Installer: pip
License: Public Domain
Location: c:\program files (x86)\python35-32\lib\site-packages
Requires:
Classifiers:
  Environment :: Console
  License :: Public Domain
  Operating System :: OS Independent
  Programming Language :: Python :: 2
  Programming Language :: Python :: 3
  Topic :: Software Development :: Libraries :: Python Modules
  Topic :: System :: Networking
  Topic :: Utilities
```

Команди pip (продовження)

В репозиторії шукають пакети за допомогою такої команди:

pip search wget

В результаті одержимо все, що пов’язано з цим підрядком.

```
C:\Users\MICHAEL>pip search wget
wgetter (0.6)                             - Another command line download utility
                                            written in python
mwget (3.2.0)                             - support threading download file
wldhx.yadisk-direct (0.0.6)               - Get real direct links usable with
                                            tools like curl or wget for files
                                            stored in Yandex.Disk
wget (3.2)                                - pure python download utility
  INSTALLED: 3.2 (latest)
reget (0.5)                               - HTTP downloader that fix wget issues
cookiestxt (0.3)                          - A version of MozillaCookieJar that
                                            works with curl/wget
yaurtww (0.0.1)                           - Yet Another Urban Terror Wget
                                            Wrapper.
datedown (0.2)                            - Small library to download files with
                                            date and time based filenames or
                                            folder structures. In parallel using
                                            wget.
wgety (1.2.2)                             - wgety is a Python library for non-
                                            interactive download of files from
                                            the Web.
Products.ClockServer (0.2-Zope2.9dev)     - The Zope ClockServer product provides
                                            a mechanism for users to call Zope
                                            object methods without the use of an
                                            external clock source (e.g.
                                            cron/wget).
wpull (2.0.1)                             - Wget-compatible web downloader and
                                            crawler.
wgetdb (0.1.4)                            - Download and store webpages in a
                                            sqlite database
wgetplus (0.0.0)                          - Extended web downloader

C:\Users\MICHAEL>
```

Команди pip (продовження)

Отже, з рисунку видно, що останньою версією пакета wget є версія wget<3.2> і саме ця версія завантажена та інстальована.

Тепер ми можемо скористатися функціональними можливостями пакету wget.
Для цього нам потрібно виконати наприклад такий код:

```
>>>url = 'http://amodm.pp.ua/files/programming/presentations/pp1.pdf'
>>> filename = wget.download(url)
>>> filename
'pp1.pdf'
```

Після виконання цього коду файл презентації pp1.pdf буде знаходитися в папці вашого проекту

Пакет networkx та winplottlib

NetworkX – це пакет Python для створення та маніпуляції графами та мережами.

Завантаження networkx:

pip install networkx

MatPlotLib – це пакет для створення графів та графіків на Python

Існують різні класи графів для ненаправлених і направлених мереж.

Завантаження matplotlib:

pip install matplotlib

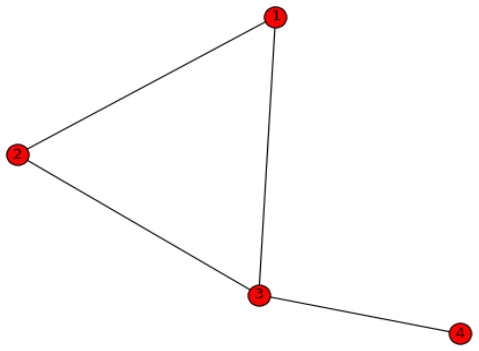
Імпортування пакетів networkx та pylab

```
>>> import networkx as nx
>>> import pylab as plt
```

Створюють об’єкт базового класу Graph таким чином:

Приклад 2.

```
>>> g  = nx.Graph()
>>> gd = nx.DiGraph()
>>> mg = nx.MultiGraph()
>>> mgd = nx.MultiDiGraph()
```



Граф g можна створити кількома способами.

NetworkX містить велику кількість функцій-генераторів графів і функцій для читання і запису графів в різних форматах.

Створення, додавання та видалення вершин

Для додавання однієї вершини до графа використовуємо метод `add_node()`.

Приклад 3.

Метод додає вершину з номером 1.

```
>>> g.add_node(1)
```

Метод додає вершину з номером `k`.

```
>>> g.add_node(k)
```

Метод список вершин з номерами 2 і 3:

```
>>> g.add_nodes_from([2 ,3])
```

Контейнер вершин.

Створюємо вершини з номерами 0 до 10 включно

```
>>> h = nx.path_graph(10)
>>> g.add_nodes_from(h)
```

Видалити вершини 2

```
>>> g.remove_node(2)
```

Що може бути вершиною у графі

Вершиною може бути об'єктом будь-який будь-якого типу, такі як рядки, числа, файли, функції і багато іншого. Це забезпечує гнучкість для проектів.

Приклад 4.

```
import networkx as nx
import math
g=nx.Graph()
h = nx.path_graph(10)
g.add_nodes_from(h)
g.add_node(math.cos)
fh=open('tmp.txt','w')
g.add_node(fh)
print(g.nodes())
```

Ex1

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, <_io.TextIOWrapper name='tmp.txt' mode='w' encoding='cp1251'>, <built-in function cos>]

Створення, додавання та видалення ребер

Приклад 5.

Додати одне ребро між вершинами 0 та 3:

```
>>> g.add_edge(0,3)
```

Додавання ребра між вершинами 3 і 4 розпакуванням кортежа:

```
>>> e=(2,3)
>>> g.add_edge(*e)
```

Додавання списку кортежей:

```
>>> g.add_edges_from([(1 ,2) ,(1 ,3)])
```

Додавання контейнера ребер:

```
>>> g.add_edges_from(h.edges())
```

Видалення ребра

```
>>> g.remove_edge(1,2)
```

Доступ до вершин

```
>>> import networkx as nx
>>> g = nx.Graph()
```

Додаємо ребра з вершинами

```
>>> g.add_edges_from([(1 ,2) ,(1 ,3)])
```

Додаємо вершину

```
>>> g.add_node('a')
```

Визначити число вершин

```
>>> g.number_of_nodes()
4
```

Другий спосіб визначення числа вершин

```
>>> g.order()
4
```

Доступ до ребер та вивід параметрів графа

Визначаємо кількість ребер

```
>>> g.number_of_edges()
```

Другий спосіб визначення кількості ребер

```
g.size()
2
```

Вивід списку вершин

```
>>> g.nodes()
[1, 2, 3, 'a']
```

Вивід списку ребер

```
>>> g.edges()
```

[(1, 2), (1, 3)]

```
>>> g.neighbors(1)
[2, 3]
```

```
>>> g.degree(1)
```

2

Зважені ребра та іменовані вершини

Довільний граф NetworkX поводиться як словник Python з вершинами в якості первинних ключів:

```
>>> g.add_node(1, time='5pm')
>>> g.node[1]['time']
'5pm'
>>> g.node[1] # Python dictionary
{'time': '5pm'}
```

Спеціальний атрибут ребра "**weight** " завжди повинен бути числовим і одержує значення, яке використовується алгоритмами, що вимагають зважених ребер.

```
>>> g.add_edge(1, 2, weight=4.0 )
>>> g[1][2]['weight'] = 5.0
>>> g[1][2]
{'weight': 5.0}
```

Ітерування по вершинах і ребрах

Багато додатків вимагають перебору вузлів або ребер.

```
>>> g.add_edge(1,2)
>>> for node in g.nodes():
...     print (node, g.degree(node))
...
1 1
2 1

>>> g.add_edge(1,3,weight=2.5)
>>> g[1][2]['weight'] = 1.5
>>> for n1,n2,attr in g.edges(data=True):
...     print(n1,n2,attr['weight'])
...
1 2 1.5
1 3 2.5
```

NetworkX має класи для графів, які дозволяють розміщувати кілька ребер між будь-якою парою вузлів: MultiGraph і MultiDiGraph.

Це потужний механізм для деяких додатків, але багато алгоритмів неправильно працюють на таких графіках: найкоротший шлях є одним із прикладів. В такому випадку необхідно перетворити мультиграф в стандартний граф таким чином, щоб для нього можна було застосувати такі алгоритми.

```
>>> mg = nx.MultiGraph()
>>> mg.add_weighted_edges_from([(1,2,.5), (1,2,.75),(2,3,.5)])
>>> mg.degree()
{1: 2, 2: 3, 3: 1}
```

Ex2

Оператори на графах

`subgraph(G, nbunch)` - створює підграф G на групі зв’язаних аершин

`union(G1,G2)` - об'єднання графів

`disjoint_union (G1, G2)` - об'єднання графів за умови, що всі вершини різні

`cartesian_product (G1, G2)` - повертає декартовий добуток графів

`compose(G1,G2)` - об'єднати графи, що ідентифікуючи вершини, які спільні для обох графів

`complement(G)` – доповнення графа

`create_empty_copy (G)` - повертає порожню копію того ж самого класу графа

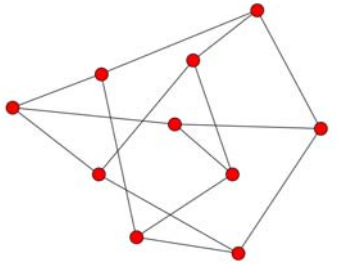
`convert_to_undirected (G)` - повертає неорієнтовний граф G

`convert_to_directed (G)` - повертає орієнтований G

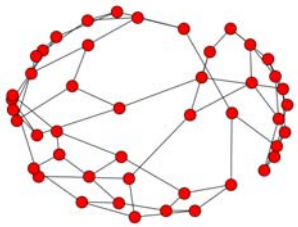
Генератори графів

Невеликі відомі графи

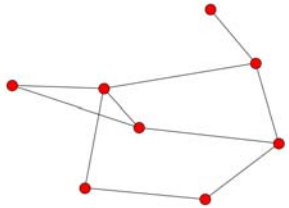
```
>>> petersen=nx.petersen_graph()
```



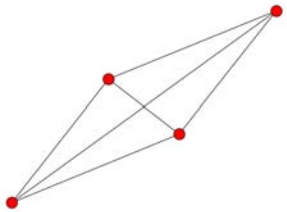
```
>>> tutte=nx.tutte_graph()
```



```
>>> maze=nx.sedgewick_maze_graph()
```



```
>>> tet=nx.tetrahedral_graph()
```

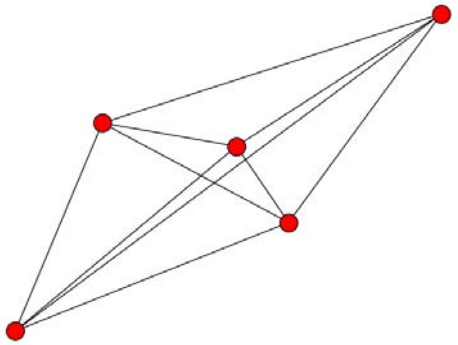


Ex3

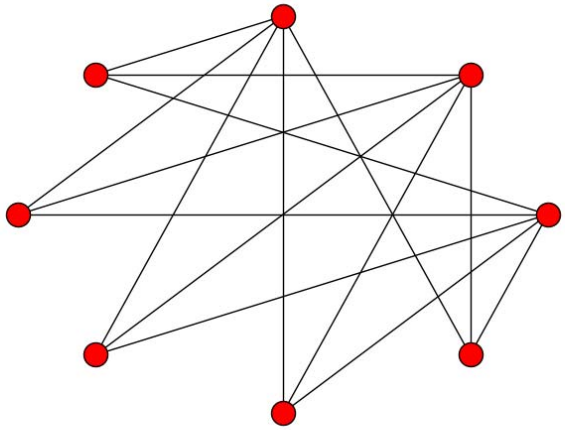
Класичні графи

```
>>> K_5=nx.complete_graph(5)
```

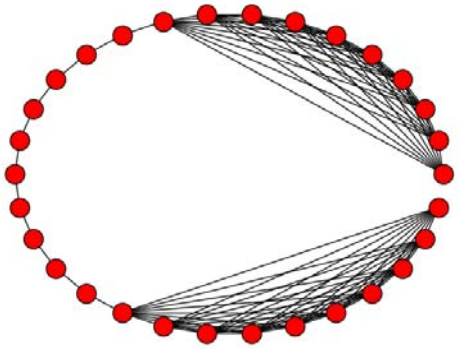
Ex4



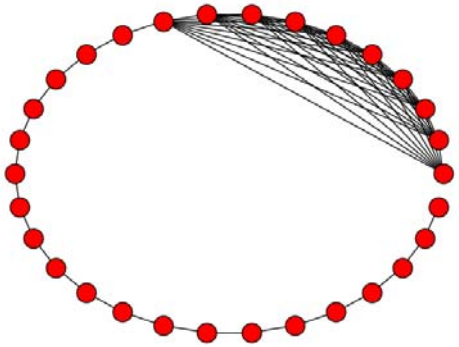
```
>>> K_3_5=nx.complete_bipartite_graph(3,5)
```



```
>>> barbell=nx.barbell_graph(10,10)
```

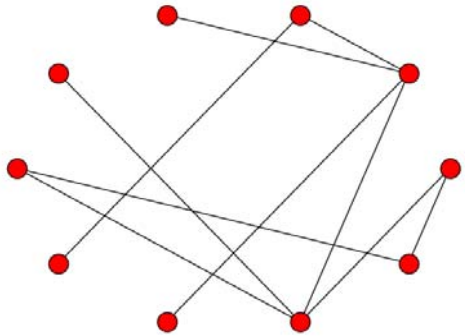


```
>>> lollipop=nx.lollipop_graph(10,20)
```

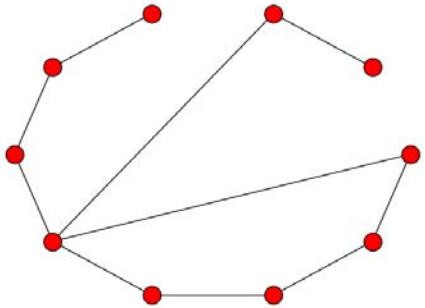


Графи з випадковими параметрами

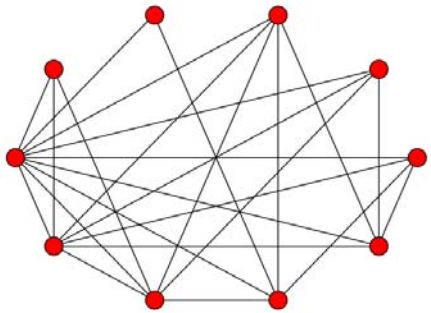
```
>>> er=nx.erdos_renyi_graph(10,0.15)
```
Ex5



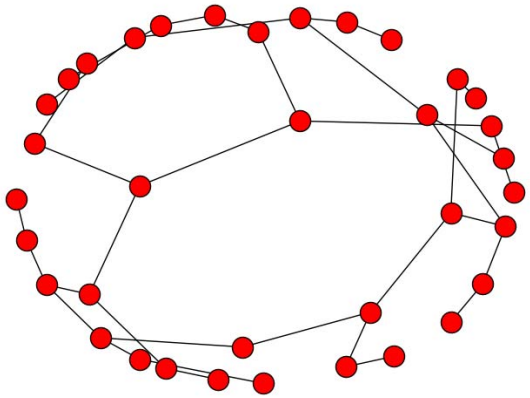
```
>>> ws=nx.watts_strogatz_graph(10,3,0.1)
```



```
>>> ba=nx.barabasi_albert_graph(10,5)
```



```
>>> red=nx.random_lobster(20,0.9,0.9)
```



Створення зображення графів

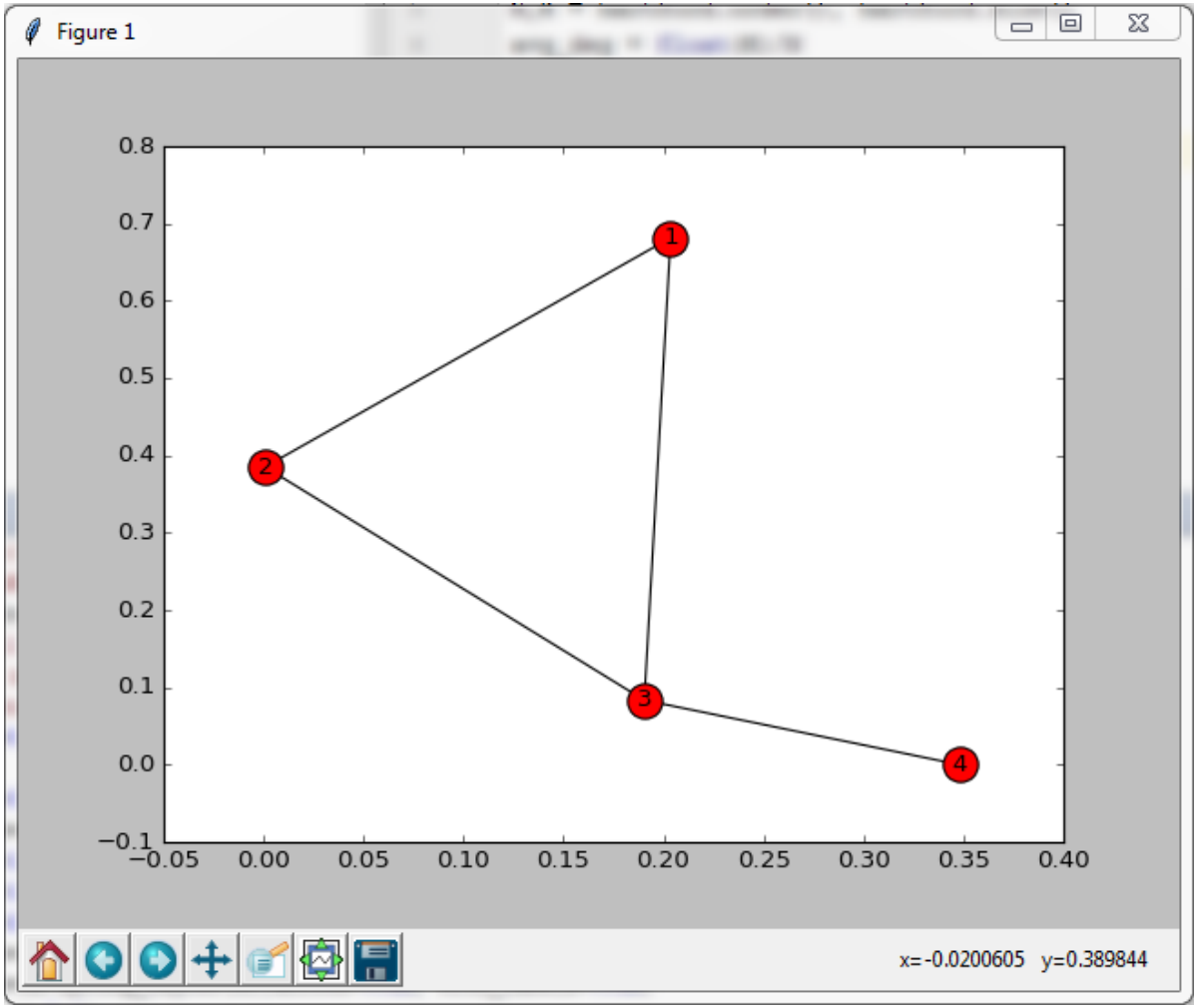
NetworkX не має можливостей малювання графів, але він забезпечує можливості застосування бібліотеки Matplotlib.

```
#імпорт інтерфейсу Matplotlib
>>> import pylab as plt
>>> g = nx.erdos_renyi_graph(10,0.15)
>>> nx.draw(g)
>>> nx.draw_random(g)
>>> nx.draw_circular(g)
>>> nx.draw_spectral(g)
>>> plt.savefig('graph.png')
```

Приклад створення графа

```
>>> import networkx as nx
>>> import pylab as plt
>>> g = nx.Graph()

>>> g.add_edge(1,2,weight=0.1)
>>> g.add_edge(2,3,weight=1.5)
>>> g.add_edge(1,3,weight=1.0)
>>> g.add_edge(3,4,weight=2.2)

>>> nx.draw_networkx(g,
        pos=nx.spring_layout(g),
        arrows=True, with_labels=True)

>>> print(nx.shortest_path(g,2,4))  Ex6
[2, 3, 4]
```

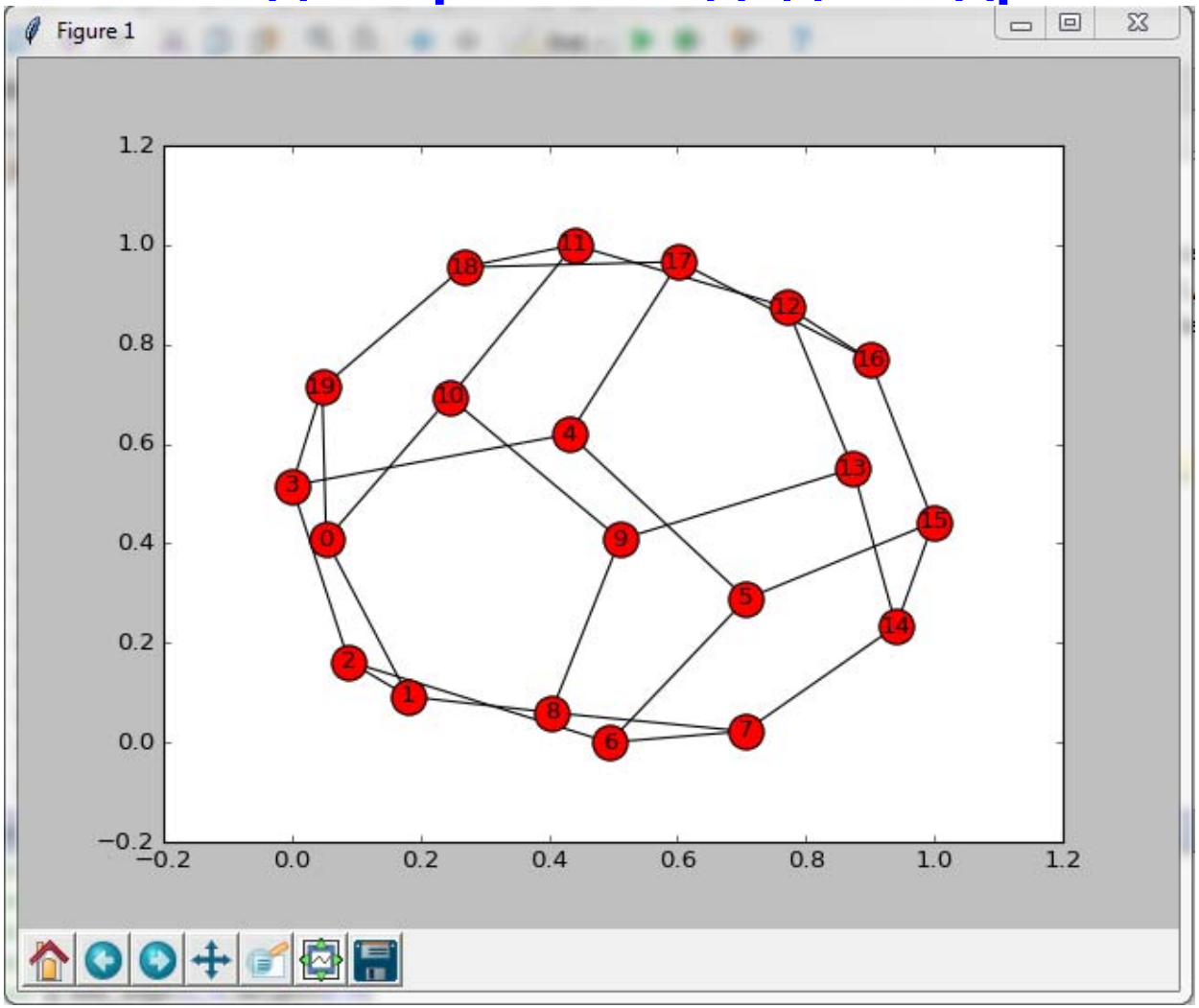
Вивід зображення графа



Приклад формування стандартного графа

```
>>> D = nx.Graph()

>>>D=nx.dodecahedral_graph()

>>> nx.draw_networkx(D,
        pos=nx.spring_layout(D),
        arrows=True, with_labels=True)
```

Вивід зображення додекаедра



Пакет myplotlib

matplotlib.pyplot
являє собою набір функцій командного стилю, які роблять роботу Matplotlib схожою на MATLAB.

Кожна функція pyplot робить деякі зміни у графіку: наприклад,

створює графік,

створює область побудови графіків на рисунку,

відображає певні рядки в області побудови,

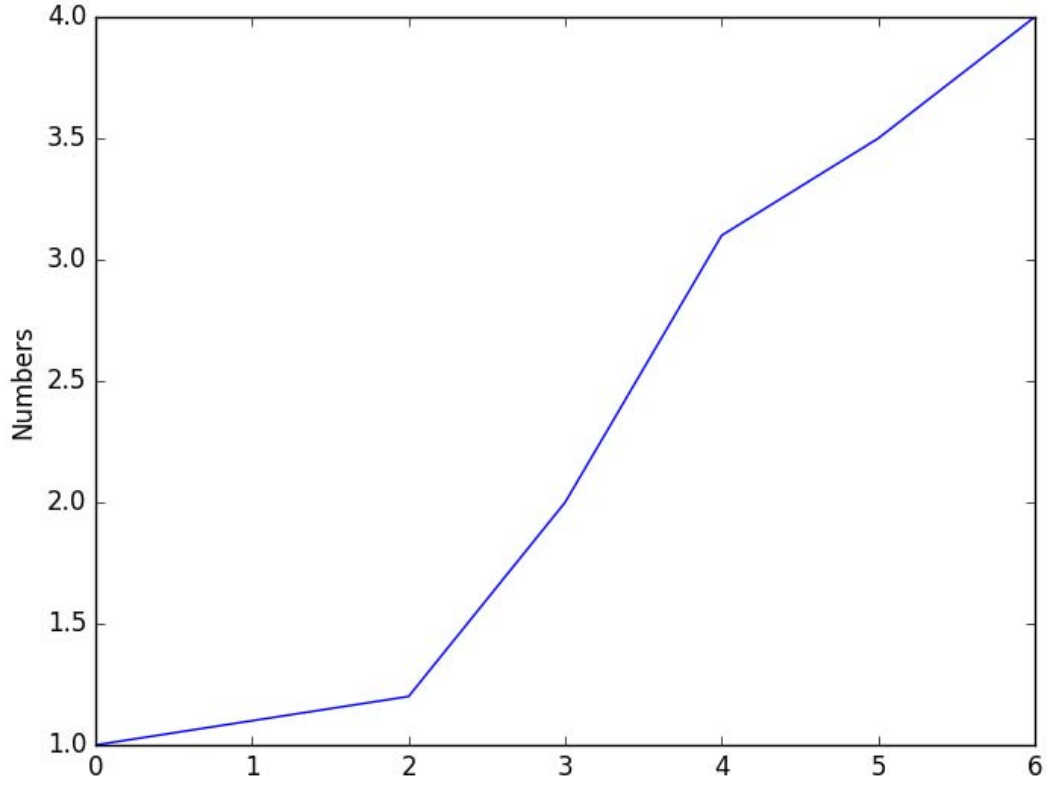
відображає різні позначки і т.д.

Приклад

```
import matplotlib.pyplot as plt

plt.ylabel('Numbers')
<matplotlib.text.Text object at 0x05678430>

plt.ylabel('Numbers')
<matplotlib.text.Text object at 0x05678430>

plt.plot([1,1.1,1.2,2,3.1,3.5,4])
[<matplotlib.lines.Line2D object at
0x0569B910>]

plt.show()
```

Ex7

Приклад відображення графіка



Модифікація вигляду графіка

Для кожних пари аргументів x, y, є необов'язковий третій аргумент, який є форматом рядка, що вказує на колір і тип лінії графіка.

Букви і символи формату рядка співпадають з MATLAB і визначають стиль лінії та її колір.

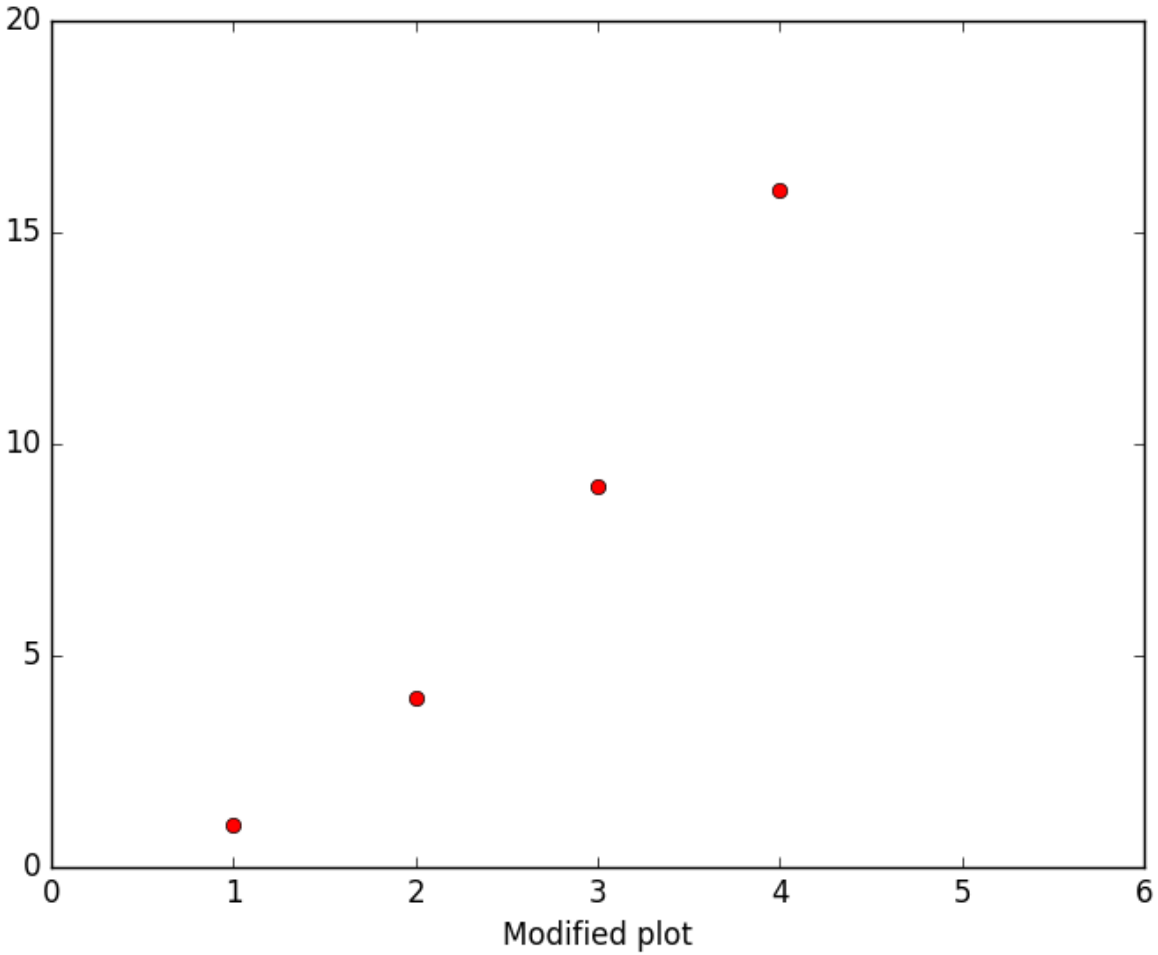
За замовчуванням. Рядок формату **'b-',** який вказує на суцільну синю лінію.

Наприклад, щоб побудувати графік червоними колами, треба записати:

Приклад

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4], [1,4,9,16], 'ro')
plt.axis([0, 6, 0, 20])
plt.xlabel("Modified plot")
plt.show()
```

Приклад модифікованого графіка



Використання масивів numpy для задавання даних

Matplotlib використовує не тільки списки для задавання параметрів графіка. Найчастіше використовують numpy масиви. Всі послідовності перетворюються в numpy масиви всередині. У наведеному нижче прикладі показано кілька ліній з різними стилями форматування в одній команді з використанням масивів.

```
import numpy as np
import matplotlib.pyplot as plt
# Встановимо рівномірно час через 200ms
t = np.arange(0., 5., 0.2)
# червоний пунктир, голубі квадрати і зелені трикутники
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```

Ex9

Кілька графіків на одному рисунку