Analiza danych i symulacje w języku Python

dr Przemysław Szufel
Zakład Wspomagania i Analizy Decyzji
SGH

https://szufel.pl/python/

dr Przemysław Szufel

- https://szufel.pl/

- Cloud computing
- Symulacje
- Analiza danych
- Badania operacyjne (optymalizacja)

- Języki programowania, które używam na co dzień: Python, Java, R, SQL

Informacje wstępne,

czyli co warto wiedzieć przed rozpoczęciem zajęć...

Zakres zajęć

- Programowanie w języku Python
  - środowisko programisty
  - składania oraz podstawowe polecenia języka
  - analiza i wizualizacja zbiorów danych
- Analiza i wizualizacja danych
- Teoria symulacji
- Analiza danych z modeli symulacyjnych
- Symulacje systemów sieciowych
- Prowadzenie obliczeń symulacyjnych i przetwarzanie danych w chmurze Amazon AWS w języku Python

Oprogramowanie

- Python Anaconda –wersja Python 2.7 – do pobrania ze strony (http://continuum.io/downloads) zawiera komplet narzędzi oraz zintegrowane środowisko programistyczne Spyder
- Pakiety języka Python (zawarte już w Anaconda)
  - NetworkX
  - Numpy
  - Matplotlib

Pojęcia ze statystyki, które warto sobie przypomnieć

- podstawowe pojęcia statystyczne (średnia, wariancja, odchylenie standardowe, dominanta)
- podstawowe rozkłady teoretyczne zmiennych losowych (rozkład normalny, log-normalny, jednostajny), różnice pomiędzy rozkładem ciągłym a dyskretnym, histogram
- podstawy testów statystycznych (pojęcie hipotezy zerowej zerowej i alternatywnej), przedział ufności dla średniej
- (opcjonalnie) podstawy regresji liniowej oraz narzędzia data mining (np. drzewa decyzyjne)

Jóźwiak J., J. Podgórski (2012) Statystyka od podstaw, PWE, Warszawa, Rozdziały 2, 3.1, 3.2, 3.3, 4.1, 4.2, 6.5, 6.6, 10.1 ,11.1 ,11.2

Podstawy informatyczne, które warto znać (ale nie są konieczne)

- podstawy programowania w dowolnym języku (pętle, instrukcje warunkowa, funkcja/procedura, wyjątek, opcjonalnie – rozumienie podejścia obiektowego)

  http://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
- (opcjonalnie) podstawy języka SQL
- (opcjonalnie) umiejętność posługiwania się konsolą – opcjonalnie – znajomość Linuxa i podstawowych poleceń powłoki bash

  http://wazniak.mimuw.edu.pl/index.php?title=%C5%9Arodowisko_programisty/Wprowadzenie_do_Basha

Przed zajęciami warto przejrzeć ...
... wprowadzenie do języka Python

- Dla początkujących

  http://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial

- Dla osób potrafiących programować w innych językach

  https://docs.python.org/2/tutorial/

Agenda i zasady zaliczeń

Agenda

1. Sprawy organizacyjno-zaliczeniowe
2. Programowanie w języku Python
    1. środowisko programisty
    2. składania języka podstawowe polecenia języka
    3. analiza i wizualizacja zbiorów danych
3. Teoria symulacji
4. Analiza danych z modeli symulacyjnych
5. Symulacje systemów sieciowych
6. Wprowadzenie do symulacji wieloagentowych
7. Prowadzenie obliczeń symulacyjnych i przetwarzanie danych w chmurze Amazon AWS w języku Python

Zasady zaliczeń

- Projekt grupowy (2-4 osób)
- Stworzyć model symulacyjny/machine learningowy w języku Python rozwiązujący problem biznesowy/analityczny
  - dopuszczalne jest opracowanie rozszerzenia jednego z modeli opracowanych w trakcie z zajęć
- Forma przekazania projektu
  - Kod źródłowy modelu
  - Raport z wynikami symulacji pszufe@gmail.com

  - **Termin 31 maja 2017**

Język Python - wprowadzenie

Język Python

„Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms."

Python tutorial

```
print "Hello world"
```

Python - cechy

- Łatwy
- Dynamiczne typowanie - brak deklaracji zmiennych
- Zaawansowane struktury danych dostępne wprost ze składni języka
- Wolne wykonywanie kodu (…ale PyPy, …ale Cython, numba...)

- Zastosowania
  - Szybkie tworzenie prototypowych aplikacji
  - Łączenie kodu różnych aplikacji

Wydajność kodu w języku Python (kod w języku C = 1.0)

	Fortran	Julia	Python	R	Matlab	Octave
	gcc 4.8.1	0.2	2.7.3	3.0.2	R2012a	3.6.4
fib	0.26	0.91	30.37	411.36	1992.00	3211.81
parse_int	5.03	1.60	13.95	59.40	1463.16	7109.85
quicksort	1.11	1.14	31.98	524.29	101.84	1132.04
mandel	0.86	0.85	14.19	106.97	64.58	316.95
pi_sum	0.80	1.00	16.33	15.42	1.29	237.41
rand_mat_stat	0.64	1.66	13.52	10.84	6.61	14.98
rand_mat_mul	0.96	1.01	3.41	3.98	1.10	3.41

Źródło: http://julialang.org/

…ale PyPy, …ale Cython, ale numba

Python cechy

	Brak deklaracji zmiennych	Multiple dispatch	Obiektowy	Funkcyjny
Python	+	+/-	+	+
R	+	+	+/-	+
Matlab/ Octave	+	-	+	-
Java/C#	-	-	+	- / Java8
C/C++	-	-	+	-

Oprogramowanie - instalacja

- Python Anaconda –wersja Python 2.7 – do pobrania ze strony (http://continuum.io/downloads) zawiera komplet narzędzi oraz zintegrowane środowisko programistyczne Spyder
- Pakiety języka Python (zawarte już w Anaconda)
  - NetworkX
  - Numpy
  - Matplotlib

Uwaga! po instalacji Anacondę warto zaktualizować poleceniami (uruchamianymi z konsoli):

```
conda update conda
conda update anaconda
```

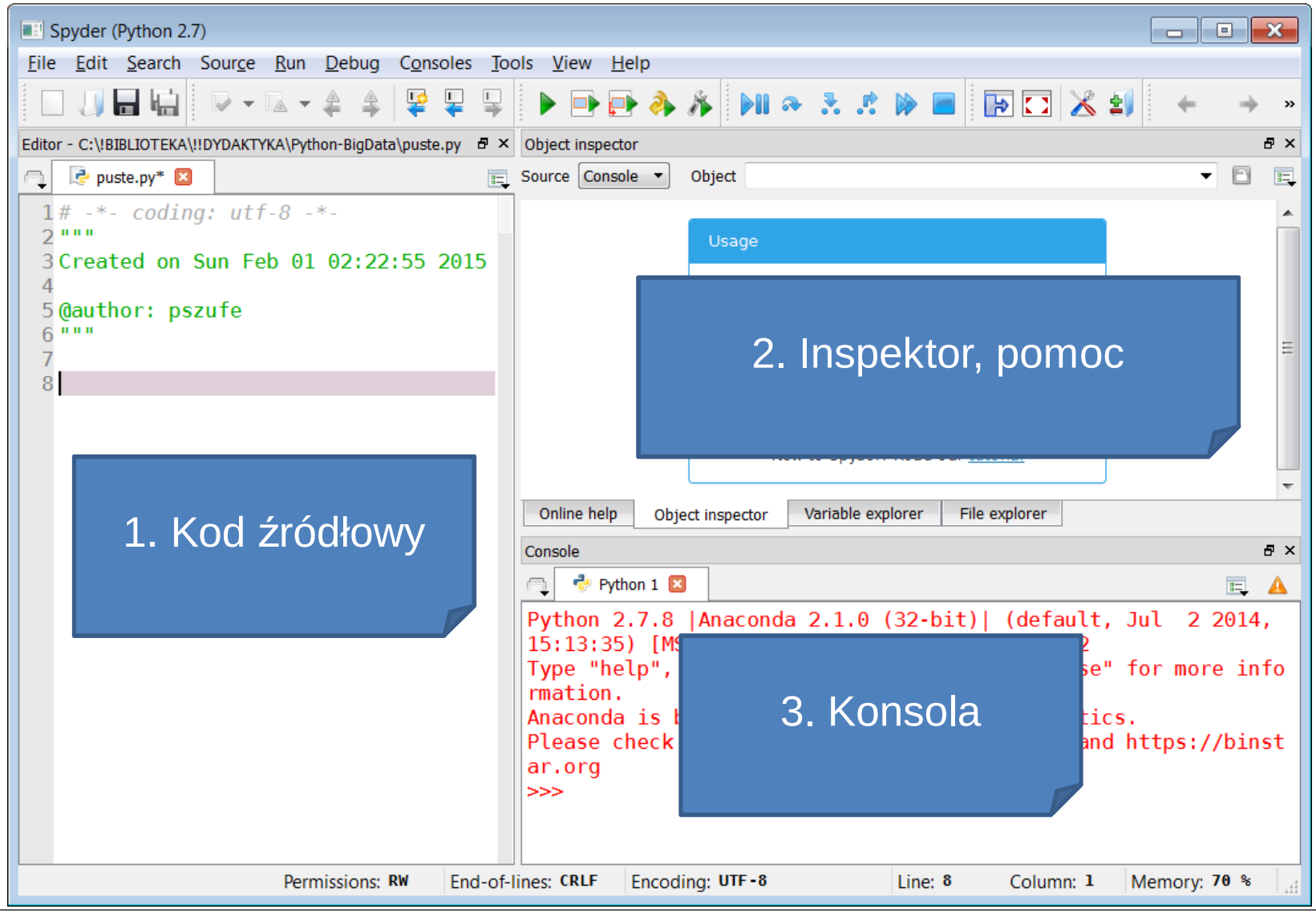
Jak zainstalować Anacondę i Spydera – tutorial video

https://www.youtube.com/watch?v=A6_gh0vrZ-E

Python wersje

- Dialekty języka
  - 2.7.x
    - `print "Hello world", 123`
    - `3/4  = 0`
    - `raise IOError, "file error"`
  - 3.x
    - `print ("Hello world", 123)`
    - `3/4 = 0.75`
    - `raise IOError("file error")`

Środowisko Spyder IDE

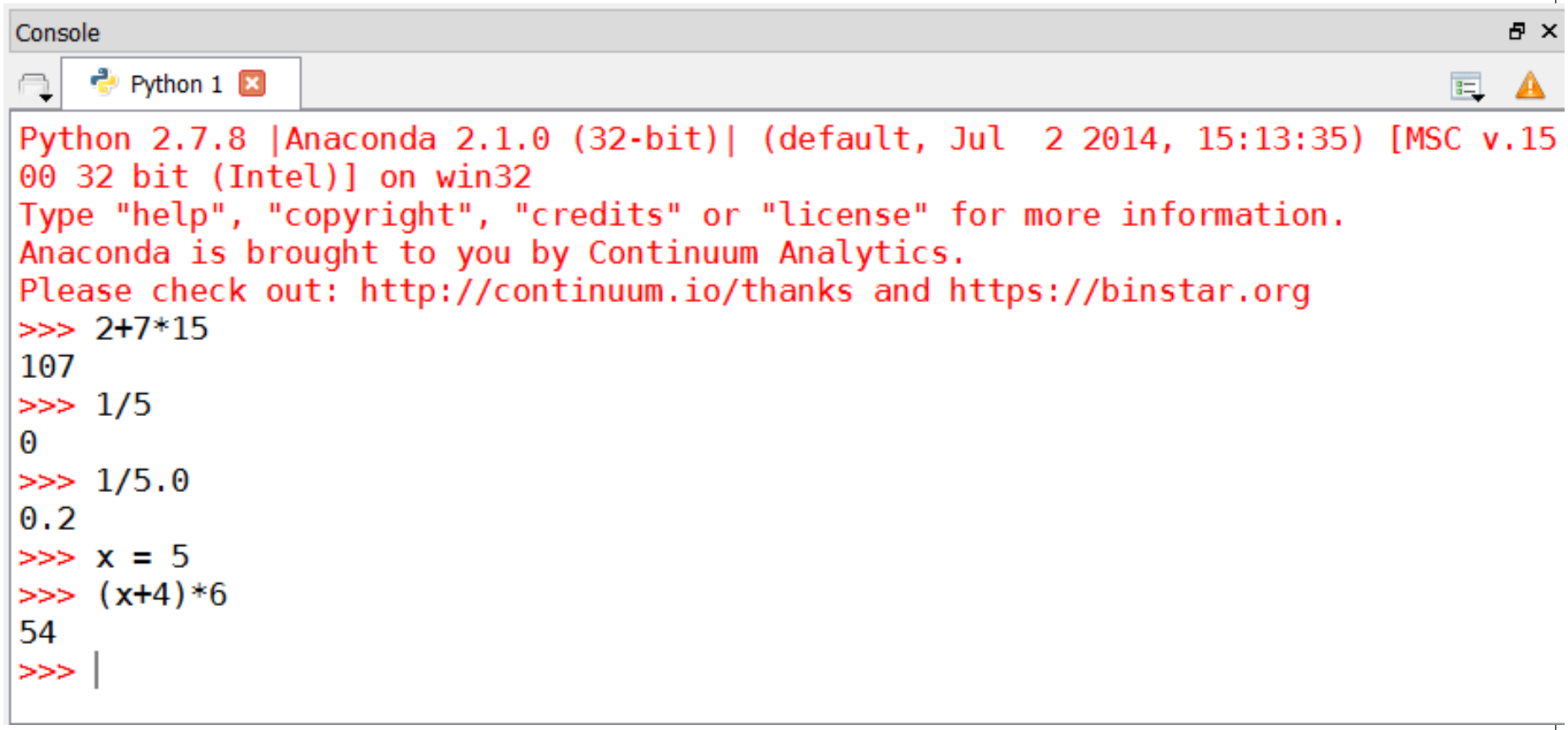


Spyder IDE - przydatne skróty

1. Działają skróty znane z innych edytorów tekstowych na platformie Windows (Ctrl+C, Ctrl+V, Ctrl X, Ctrl+S, Ctrl+O)
2. TAB na zaznaczonym tekście powoduje zwiększenie poziomu wcięcia
3. Ctrl + przycisk myszy - przenosi do definicji wybranego elementu
4. Ctrl + I - pokazuje pomoc dla wybranego elementu

Konsola języka Python

- ... umożliwia interaktywną pracę w trybie "kalkulatora"



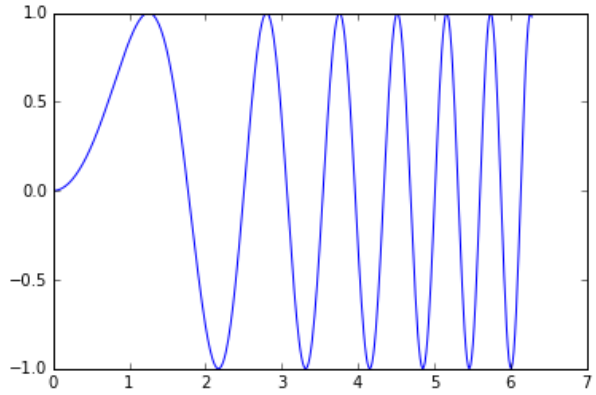
Konsola IPython (interactive Python)

IPython console

Console 1280/A

```
IPython 2.2.0 -- An enhanced Interactive Python.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://binstar.org
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]: import matplotlib.pyplot as plt
   ...: import numpy as np
   ...: x = np.linspace(0, 2 * np.pi, 400)
   ...: y = np.sin(x ** 2)
   ...: plt.plot(x,y)
   ...:
Out[1]: [<matplotlib.lines.Line2D at 0xd2e5c50>]
```

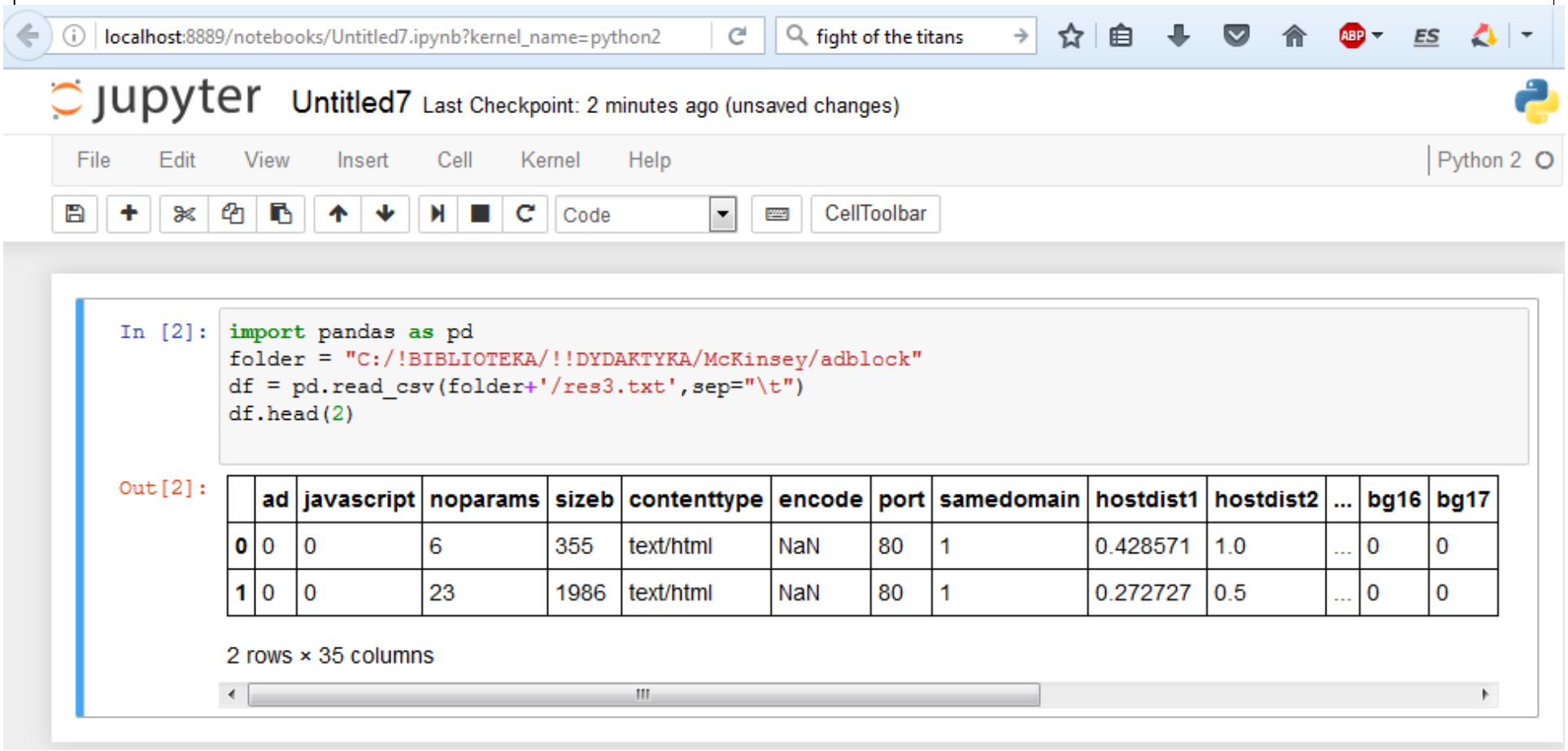


Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 8 | Column: 1 | Memory: 75 %

Jupyter – a web browser based IDE

c:\Anaconda2\Scripts\jupyter notebook

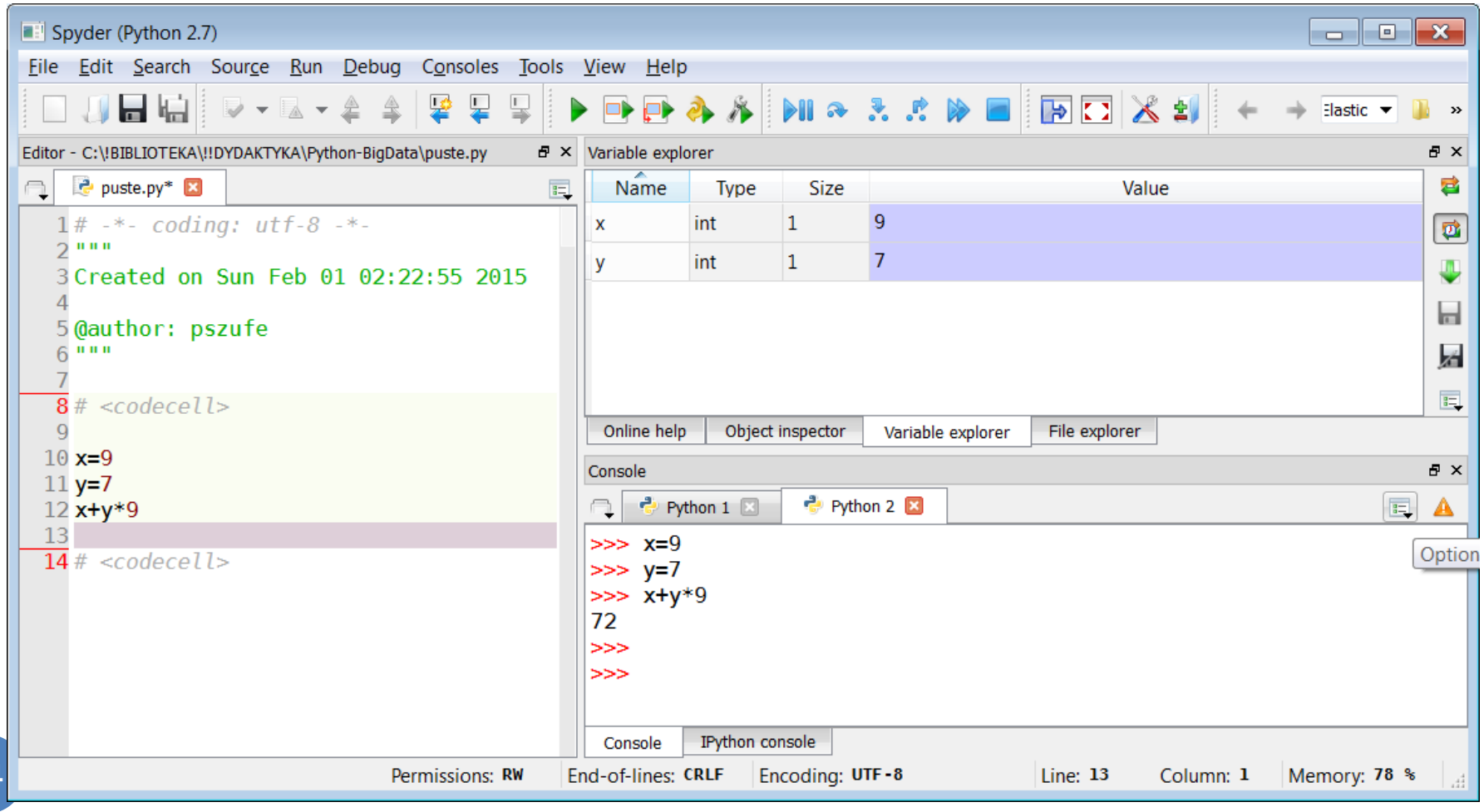
c:\Anaconda3\Scripts\jupyter notebook



	ad	javascript	noparams	sizeb	contenttype	encode	port	samedomain	hostdist1	hostdist2	...	bg16	bg17
0	0	0	6	355	text/html	NaN	80	1	0.428571	1.0	...	0	0
1	0	0	23	1986	text/html	NaN	80	1	0.272727	0.5	...	0	0

Komórki kodu źródłowego w edytorze

`# <codecell>` użyj Ctrl+Enter aby wykonać]



Typy danych w Pythonie

- Liczby całkowite i zmiennoprzecinkowe
  - int, float
- Łańcuchy znaków (tekst)
  - str
- Listy – kolekcje elementów dowolnego typy
  - list
- Krotki – zbiór uporządkowanych elementów, immutable
  - tuple
- Słowniki – tablice typu klucz-wartość
  - dict
- Funkcje, funkcje anonimowe (lambda)
  - function
- Typy elementów tablic numpy.array
  - numpy.float64, numpy.int32 (dostępne też rozmiary na mniejszej liczbie bitów np. numpy.float32)

Przetwarzanie liczb

- Python może być używany w trybie kalkulatora
- Zwróć uwagę na różną obsługę dzielenia w Python 2 oraz Python 3
- Aby wymusić przetwarzanie jako liczby zmiennoprzecinkowe wystarczy dodać kropkę, np.
  - `100/12.` #12. jest typu float

Przetwarzanie tekstu

- Tekstem jest dowolne wyrażenie w cudzysłowie pojedynczym lub podwójnym

  `a = "Text"` albo `b='text'`

- Znak "\" ma znaczenie specjalne (tzw. escape character)
  - \t = tabulacja
  - \n = nowa linia w środowiskach Linux/Unix
  - \r\n = nowa linia w środowiska Windows
  - \\ = znak \ np.: "C:\\Windows"
- Na tekstach można dokonywać m.in. operacje
  - dodawania (łączenia) `a+b`
  - mnożenia (zwielakratniania) `a*4`
  - wyboru podtekstu `a[0:5]`

Listy

- Listy grupują wartości dowolnego typu (tekst, liczby całkowite, liczby rzeczywiste, listy, ...)

```
lista1 = [1, 2, 3, 4]
lista2 = ['a', 'b', 'c']
lista3 = [1, 2, 'a', 'b', 'c', lista1 ]
```

- Na tekstach można dokonywać m.in. operacje
  - dodawania (łączenia) `[1,2,3]+[4,5]`
  - mnożenia (zwielokrotniania) `[1,2]*3`
  - wyboru podlisty `lista3[2:4]`
  - Ostatni element listy `lista[-1]`

Krotki

- Działają tak jak listy
- Ale są immutable (niezmienne)

- lista = [1,2,3,4]
- lista[2] = ["ala", "pies"]

  [1,2, ["ala", "pies"], 4]

- krotka = (1,2,3,4)
- krotka[3] = 8 ← zwróci błąd

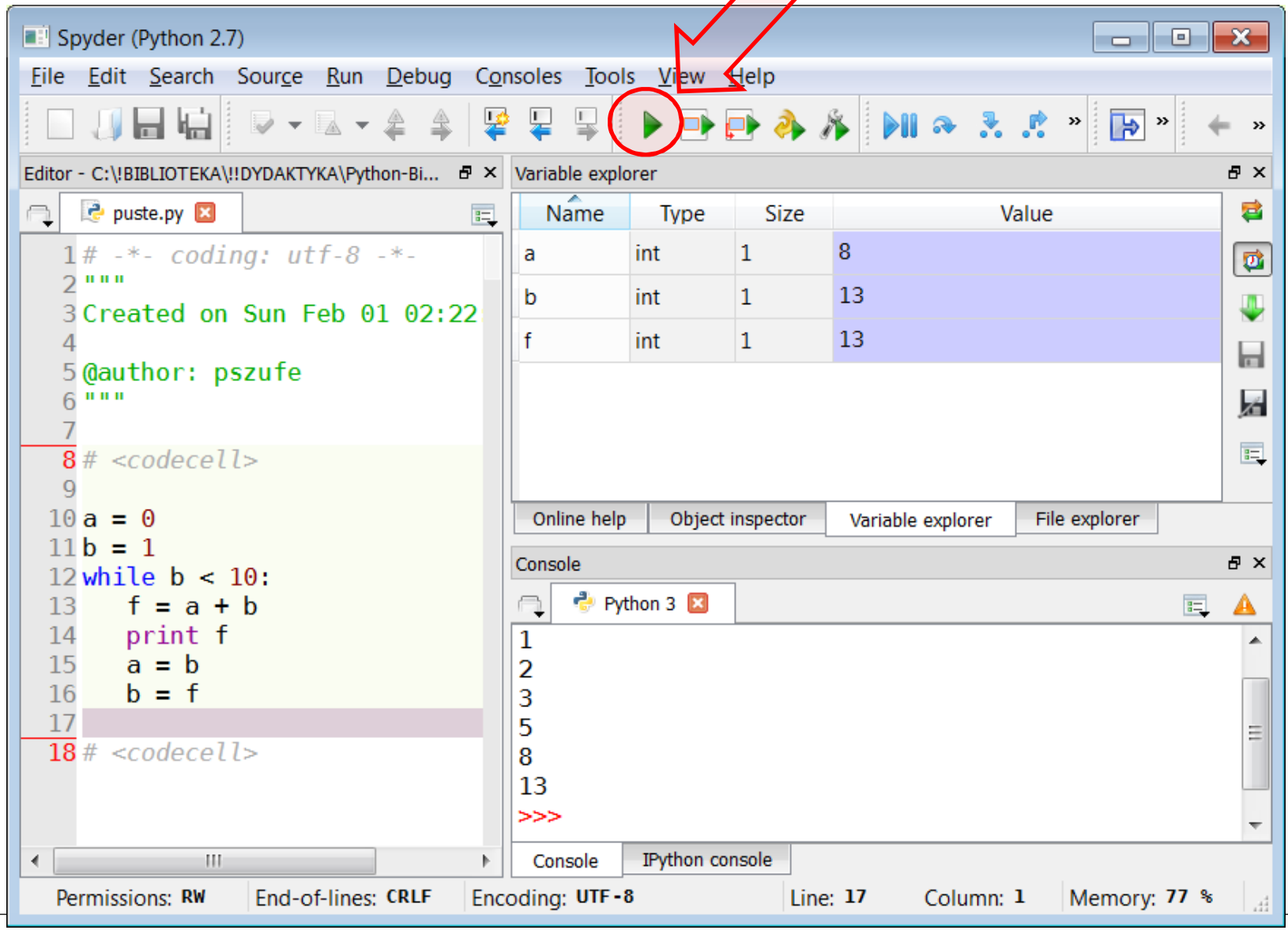
Słowniki

- Słowniki grupują wartości typu klucz-wartość

- slownik1 = { 1:30, 2:12, 3:44, 101:666 }
- slownik1 = { 1:"dom", 2:12, 3:44, "d1":"val" }

- Do istniejącego słownika można dodawać nowe wartości oraz aktualizować istniejące
  - slownik[klucz] = wartość
- Usuwanie elementów ze słownika
  - del slownik[klucz]

Pierwszy program...



Wyrażenie warunkowe if

- Instrukcja **if** pozwala sterować wykonywaniem programu...

```
x = int(raw_input("Podaj liczbe "))
if x > 5:
    print "liczba jest wieksza od 5"
else:
    print "liczba nie jest wieksza od 5"
```

Proszę zwrócić uwagę na wcięcia, które definiują strukturę kodu!
Do wyboru mamy tabulator lub 4 znaki spasji ...

Pętla
for zmienna in lista

- Instrukcja for umożliwia wielokrotne powtarzanie czynności

```
lista1 = ['chf', '$', 'euro']
for element in lista1:
    print element
```

Proszę zwrócić uwagę na wcięcia,
które definiują strukturę kodu!
Do wyboru mamy tabulator lub 4 znaki spasji ...

Wyrażenia listowe / generatory

- [x**2 for x in range(10)]

Z filtrowaniem

- [x**2 for x in range(10) if x % 2 == 0]

To samo działa dla słowników

- { x : x**2 for x in range(10) }

Z filtrowaniem

- { x:x**2 for x in range(10) if x % 2 == 0 }

Generatory - yield

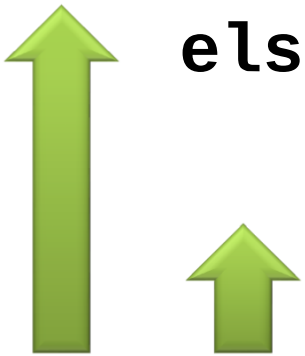
- Elementy listy mogą być generowane wtedy gdy są potrzebne...

```
def list1(count):
    i = 0
    while i < count:
        i = i+1
        yield i
```

Funkcje

```
 if a < b:
        return a
    else:
        return b

def silnia (n):
    if n == 0:
        return 1
    else:
        return n*silnia(n-1)
```



wcięcia pierwsszego i drugiego poziomu

Funkcje 'anonimowe' (lambda)

```
def f1 (x):
    return x*x-x-2

print f1(5)
f2 = f1   #przypisanie funkcji do zmiennej
print f2(5)

f3 = lambda x: x*x-x-2
 #przypisanie anonimowej funkcji do zmiennej
print f3(5)
```

Przykład - liczenie pochodnych

```
def pochodna(f):
    h = 0.000000001
    f1 = lambda x : (f(x+h)-f(x))/h
    return f1

def funk1 (x):
    return x*x+3
funk2 = lambda x : 2*x*x+x-1
print pochodna(funk1)(5)
print pochodna(funk2)(3)
```

Przykład c.d.
miejsce zerowe funkcji kwadratowej
$f(x) = ax^2 + bx + c$

```
def q_solve (f):
    #axx + bx + c
    c = f(0)
    f1 = pochodna(f)
    b = f1(0)
    a = f(1)-b-c
    d_sqrt = sc.sqrt(b*b-4*a*c)
    return ((-b-d_sqrt)/(2*a),(-b+d_sqrt)/(2*a))

funk = 4*x*x - 2*x*x + 2
print q_solve(funk)
print q_solve(lambda x : x*x+2*x+10)
```

Python - moduły

program.py

```
import matem

s = matem.silnia(5)
print s


import matem as m
print m.silnia(5)
```

matem.py

```
def silnia (n):
    if n == 0:
        return 1
    else:
        return n*silnia(n-1)
```

Python - moduły c.d.

program.py

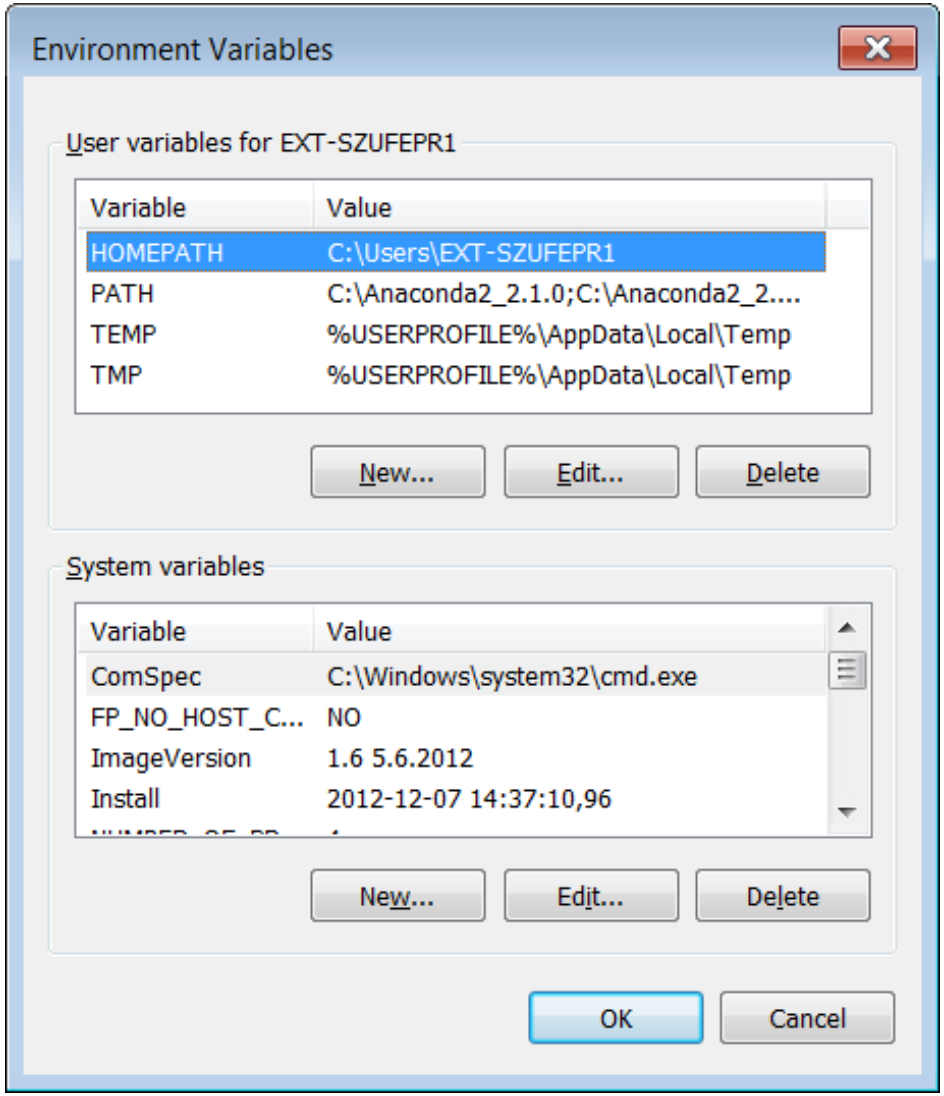
```
from matem import silnia

s = silnia(5)
print s


from matem import *
print silnia(5)
```

Python - moduły, kolejność wyszukiwania

1. Katalog zawierający skrypt (lub katalog aktualny)
2. Zmienna środowiskowa PYTHONPATH
3. Katalog instalacyjny Pythona



Rynek EC2 spot

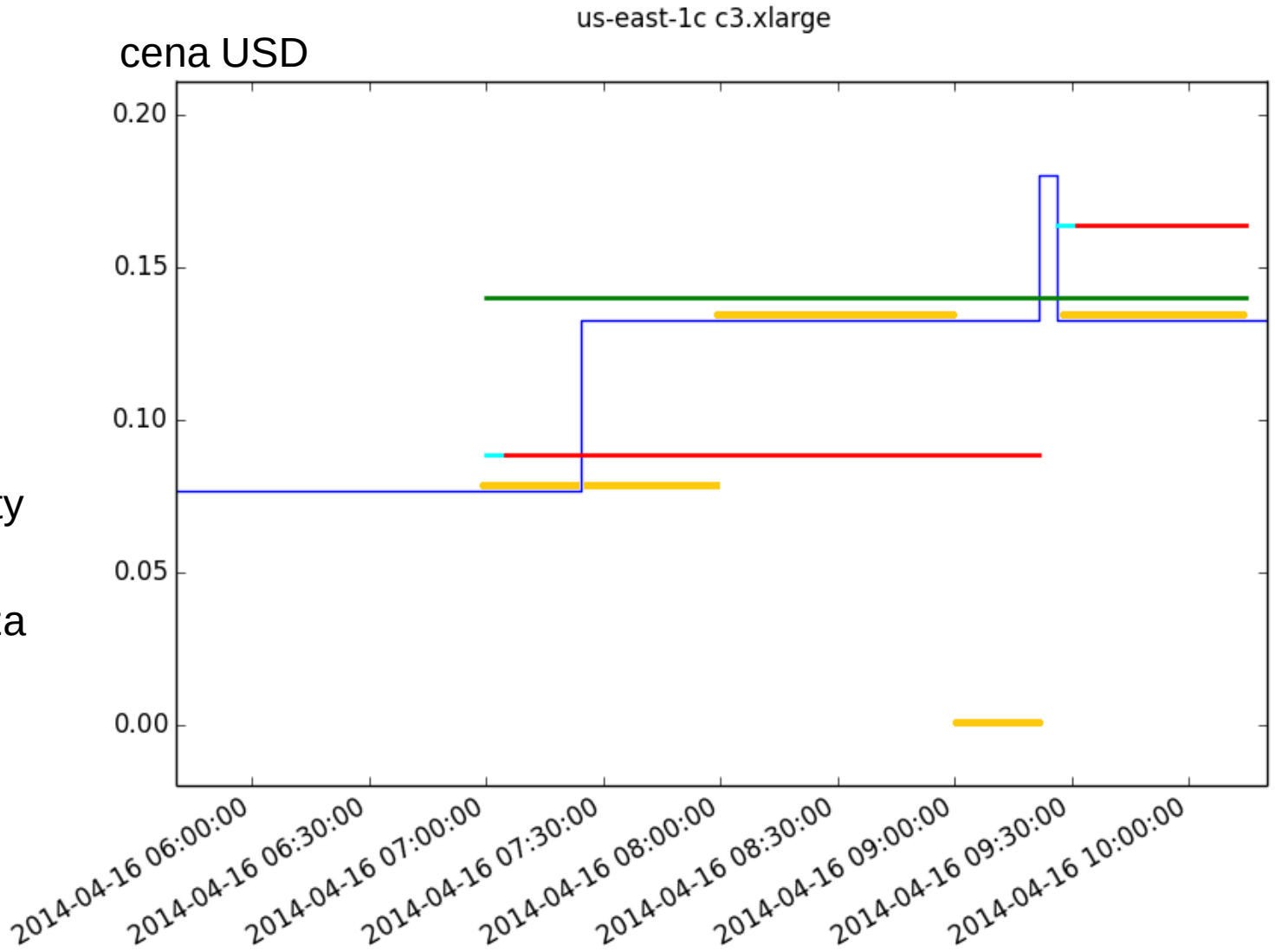
- Rynek aukcyjny
- Ceny niższe o około 80-90% od cen on-demand
- Bardzo duża zmienność (czasami c.spot > c.on_demand)
- Jeżeli cena spot przekroczy bid to instancja jest bezwarunkowo zabijana
- "Darmowy lunch" - jeżeli Amazon zabije instancję to nie płaci się za nieukończone godziny
- Od 2015-01-06 *2-minute termination notice*

ceny spot: c3.8xlarge us-east-1a ostatni tydzień maja

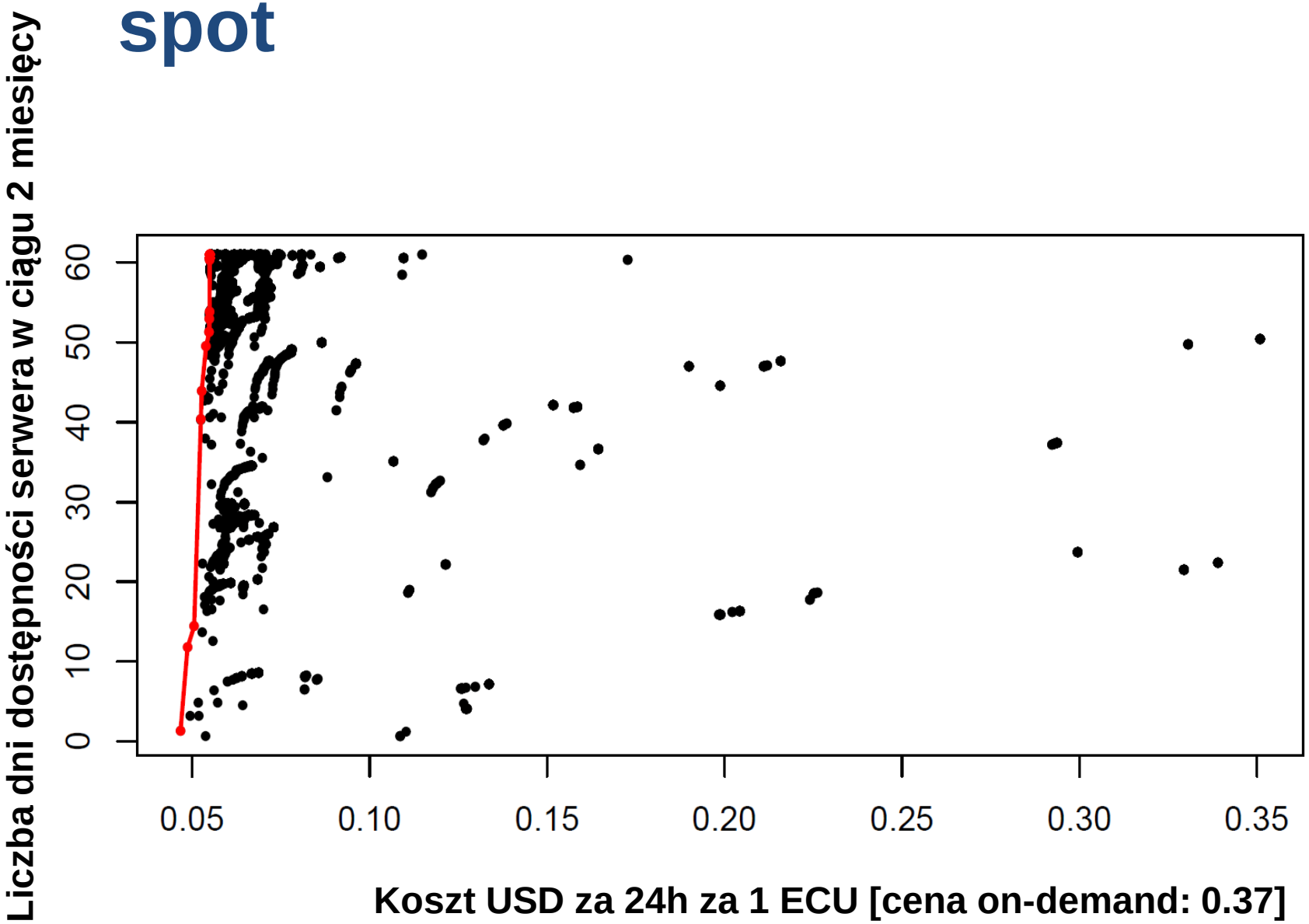
Przykład - bid 0.14$

Legenda

- cena spot
- bid
- cena do zapłaty
- średnia cena za godzinę
- warmup time



Efektywny czas a koszt na rynku spot



Przykład - pobranie cen spot na moc obliczeniową w chmurze Amazona

```
import boto.ec2
conn = boto.ec2.connect_to_region("us-east-1",\
   aws_access_key_id='AKIAJGBLCUFPEHFLTPGQ',\
       aws_secret_access_key=\
       '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')


history = conn.get_spot_price_history(\
   instance_type="x1.32xlarge",\
   product_description="Linux/UNIX",\
   availability_zone="us-east-1a")


for h in history:
    print h.timestamp,h.availability_zone,\
        h.instance_type,h.price
```

Pozyskiwanie danych

- Pliki tekstowe
- CSV
- JSON
- Excel
- Internet
- Relacyjne bazy danych

Pliki tekstowe

```
with open('nazwa_pliku', 'r') as f:
     read_data = f.read()

with open('nazwa_pliku', 'r') as f:
     for linia in f:
          print f
```

Zapisywanie danych o cenach spot do pliku CSV

```
import csv
ofile =  open('ceny_spot.txt', "wb")
ofilewriter = csv.writer(ofile, delimiter='\t')
ofilewriter.writerow(['time','zone','type','price'])
for h in history:
    ofilewriter.writerow([\
        h.timestamp, h.availability_zone,\
        h.instance_type,h.price])
ofile.close()
```

Odczyt danych z plików CSV

```
import csv
csvfile =  open('ceny_spot.txt', "r")
reader = csv.reader(csvfile, delimiter='\t')
for row in reader:
    print row
csvfile.close()
```

Odczyt danych z internetu

```
from pandas import read_csv
import urllib2 as ul #import urllib.request as ul
url = 'http://szufel.pl/pliki/iris_data.csv'
response = ul.urlopen(url)
data = read_csv(response)
response.close()
```

Praca z danymi JSON

```
import json
import urllib2 as ul #import urllib.request as ul

json_data = ""

url = 'http://szufel.pl/pliki/plik.txt'
response = ul.urlopen(url)

for linia in response:
    json_data += linia
# python3: json_data = response.read().decode("utf-8")
slownik = json.loads(json_data)

with open('c:\\temp\\data.txt', 'w') as outfile:
    json.dump(slownik, outfile)
```

Excel zapis

```
from openpyxl import Workbook
wb = Workbook()

ws = wb.active

ws['A1'] = 42


import datetime
ws['A2'] = datetime.datetime.now()

wb.save("sample.xlsx")
```

Praca z plikami Excela

```
import openpyxl

wb = \
   openpyxl.load_workbook\
   ('c:\\temp\\example.xlsx')
sheet_names = wb.get_sheet_names()

sheet = wb.get_sheet_by_name(sheet_names[0])
print sheet.title, type(sheet)
print sheet.max_row, sheet.max_column
print len(sheet.rows)
cell1 = sheet['A1']
cell2 = sheet.cell(row=1, column=2)
print  cell1.value, cell2.value
```

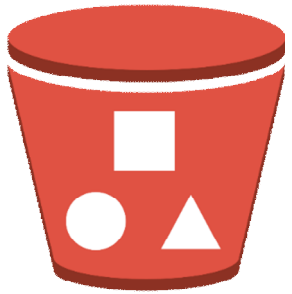
Praca z bazą danych Oracle

wymaga: `conda install cx_oracle`

```
import cx_Oracle
con =
cx_Oracle.connect('python/python@192.168.1.195:1521/DWH')
#haslo: ora12345 user: ora
print con.version
cur = con.cursor()
cur.execute('insert into tabela1 (id, kolumna1)  \
       values (:1, :2)',(5,"tekst") )
cur.execute('select * from tabela1 order by id')
for result in cur:
    print result
con.close()
```


http://www.oracle.com/technetwork/articles/dsl/python-091105.html

Kontener Amazon AWS S3



- Podobny do DropBox
- Key-value storage
- Dowolny typ przechowywanych danych
- Dane nigdy bez wiedzy użytkownika nie opuszczą deklarowanego regionu (np. Frankfurt)
- Przynajmniej trzykrotna replikacja danych
- 99.999999999% durability
- 99.99%

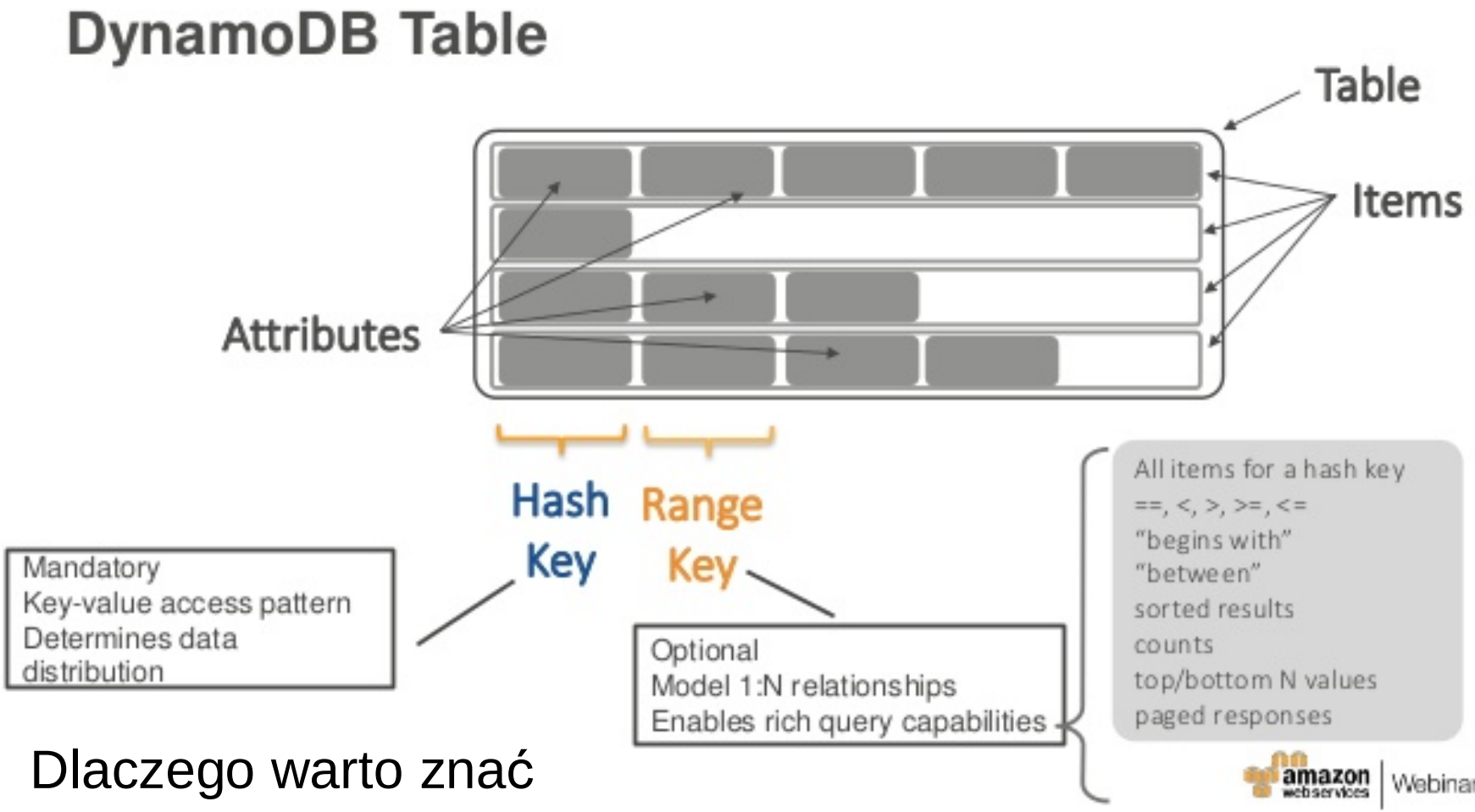
Dlaczego warto znać?

- BigData storage (zamiast HDFS)
- Public datasets https://aws.amazon.com/datasets/

S3 Python - przykład

```
from boto.s3.connection import S3Connection
conn = S3Connection(LOGIN',\
'HASLO')
from boto.s3.key import Key
bucket = conn.get_bucket('szufeldemo')
k = Key(bucket)
k.key = 'somepath/0001.txt'
k.set_contents_from_string(\
     '12345\r\nline 2\r\nline 3')
```

DynamoDB – Highly distributed NoSQL database



Dlaczego warto znać

- IoT oraz dane transakcyjne

Source: aws.amazon.com

DynamoDB z językiem Python

```
from boto.dynamodb2.table import Table
from boto.dynamodb2.layer1 import DynamoDBConnection
from boto.regioninfo import RegionInfo
from datetime import datetime

dynamodb = DynamoDBConnection(  \
   region=RegionInfo(name="us-east-1",endpoint='dynamodb.us-east-1.amazonaws.com'),\
   aws_access_key_id= "LOGIN",\
   aws_secret_access_key="HASLO")

table = Table("demotable",connection=dynamodb)
table.put_item(data={"id":11,\
 "name":"John","familyname":"Smith",\
  "date":datetime.now().strftime('%Y-%m-%d %H:%M:%S')})

item = table.get_item(id=11)
print item["name"]
```

Odczyt i zapis danych do R
[feather – korzysta z Apache arrow dla wymiany danych]

- R

```
library(feather)
write_feather(iris, "C:/temp/iris.dat")
iris_copy <- read_feather("C:/temp/iris.dat")
```

- Python
  - conda install feather-format -c conda-forge

```
import feather
iris = feather.read_dataframe("C:/temp/iris.dat")
feather.write_dataframe(iris, "C:/temp/iris_copy.dat")
```

Scraping WWW

Automatyzacja zbierania danych z internetu

Website scraping....

- Web data extraction
- Analityka, Big Data, śledzenie klientów
- Odwiedzanie szeregu stron WWW w uporządkowany sposób
  - Bezpośrednio przez protokół HTTP (scrapy)
  - Poprzez automatyzację przeglądarki (selenium)

Web scraping ze scrapy

Command line:

- c:\Anaconda2\Scripts\pip.exe install scrapy
- scrapy shell "http://www.dmoz.org/Computers/Programming/Languages/Python/"

Python (lub wewnątrz konsoli scrapy)

- elem = response.xpath("//div[contains(@class, 'title-and-desc')]")[0]
- elem.xpath('a/@href').extract()[0]

- response.xpath("//div[contains(@class, 'title-and-desc')]")[0].xpath("a/@href").extract()[0]

Scrapy – w jupyter notebook

```
import requests
from scrapy.http import TextResponse

r = requests.get('http://www.dmoz.org/Computers/Programming/Languages/Python/')
response = TextResponse(r.url, body=r.text, encoding='utf-8')
```

Scrapy w notebook

```
import scrapy
import scrapy
import scrapy.http
class DmozItem(scrapy.Item):
    link = scrapy.Field()
class DmozSpider(scrapy.Spider):
    name = "dmoz"
    allowed_domains = ["www.dmoz.org","dmoz.org"]
    start_urls =\
     ["http://www.dmoz.org/Computers/Programming/Languages/Python/"]
    data = []
    def parse(self, response):
        print ("Site visited:"+response.url)
        for elem in response.xpath("//div[contains(@class, 'title-and-desc')]"):
            link = elem.xpath("a/@href").extract()[0]
            print ("NEW LINK FOUND:"+link)
            item = DmozItem()
            item['link'] = link
            self.data.append(item)
            yield item
```

Scrapy w notebook

```
pp = response.xpath("//div[contains(@id, 'cat-list-content-main')]")
    for elem in pp.xpath("div[contains(@class,'cat-item')]"):
        href = elem.xpath('a/@href').extract()
        if len(href) > 0:
            url = "http://www.dmoz.org"+href[0]
            print ("WILL GOTO:"+url)
            yield scrapy.http.Request(url,self.parse)
```

Scrapy uruchomienie procesu w notebooku

```
from scrapy.crawler import CrawlerProcess

process = CrawlerProcess({
    'USER_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0;
Windows NT 5.1)'
})
spider = DmozSpider
process.crawl(spider)
process.start()
print(spider.data)

#Uwaga: kazde uruchomienie wymaga restartu
#kernela notebooka....
```

Tworzenie projektu scrapy w konsoli....

1. Utwórz
- scrapy startproject dmoz
- cd dmoz

2. Edytuj plik items.py file – dodaj element DmozItem

3. Utwórz w katalogu spiders plik dmozspider.py
   (następny slajd)

4. Uruchom:
- scrapy crawl -t json -o res.txt dmoz

Scraping – dmozspider.py (wersja dla konsoli)

```
import scrapy
import scrapy.http
from dmoz.items import DmozItem

class DmozSpider(scrapy.Spider):
    name = "dmoz"
    allowed_domains = ["www.dmoz.org","dmoz.org"]

    start_urls =\
 ["http://www.dmoz.org/Computers/Programming/Languages/Python/"]

    def parse(self, response):
        for elem in response.xpath("//div[contains(@class, 'title-and-desc')]"):
            link = elem.xpath("a/@href").extract()[0]
            print "NEW LINK FOUND:"+link
            item = DmozItem()
            item['link'] = link
            yield item
```

Aby scraper odwiedzał strony trzeba mu o tym powiedzieć...

```
for elem in response.xpath("//div[contains(@class, 'cat-item')]"):
    url = "http://www.dmoz.org"+elem.xpath("a/@href").extract()[0]
    print "NEW URL THAT SHOULD BE VISITED FOUND:"+url
    yield scrapy.http.Request(url,self.parse)
```

Selenium – dla bardziej skomplikowanych przypadków...

- http://www.seleniumhq.org/
- Instalacja
  - c:\Anaconda2\scripts\pip.exe install selenium
- Geckodriver.exe
  - https://github.com/mozilla/geckodriver/releases
  - Rozzipuj i skopiuj *.exe do c:\Anaconda2

Selenium – przykładowy kod

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()

driver.get("http://www.python.org")

print driver.title
elem = driver.find_element_by_name("q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
print driver.page_source
driver.close()
```

Obliczenia numeryczne - numpy

Biblioteką numpy - obliczenia na tablicach

```
from numpy import *
a = array([1,2,3])
b = array([4,5,6])
print a+b
print a*b
```

Uwaga! tablice to coś zupełnie innego niż listy:

```
print [1,2,3]+[4,5,6]
```

Elementy listy

```
b = np.arange(12).reshape(3,4)
b[2][3] = 100

# Tak tez mozna
b[2,3] = 100
b[(2,3)] = 100
print "b=\n",b
```

Wymiary list

```
import numpy as np

vector = np.arange(15)
# the same can be achieved by np.array(range(15))

a1 = np.reshape(vector,(3,5))

a2 = np.array( [[1,2,3],[4,5,6]]  )

print "Dimension number of a1", a1.ndim
print "Dimensions ", a1.shape
print "Type a1", a1.dtype.name
print "Memory footprint", a1.itemsize
print "Number of elements in the array", a1.size
```

Listy a macierze

```
# Arrays and matrix operators
print "element by element multiplication", a1*a1
print "Matrix multiplication", np.transpose(a1).dot(a1)


print "Matrix of ones\n", np.ones((3,4),dtype="int64")
print "Matrix of ones\n", np.ones((3,4),dtype="float64")

print "Matrix of zeros\n", np.zeros((3,4),dtype="int64")
print "Matrix of zeros\n", np.zeros((3,4),dtype="float64")
```

Typy elementów list

```
import numpy as np

threes = np.ones((3,4),dtype="int64")*3
fours = np.ones((3,4),dtype="int64")*4

print (threes/fours)

threesfloat = np.ones((3,4),dtype="float64")*3

np.linspace(0,10,10)
```

Prędkość obliczeniowa...

```
a = np.random.random((1000,1000))
b= a*5
del a
del b

timeit.timeit("import numpy as np;a =
np.random.random((1000,10000));print(np.sum(a))"
,number=10)
```

Osie tablic

```
b = np.arange(12).reshape(3,4)
print "b=\n",b

print "b column sum=\n",b.sum(axis=0)
print "b row =\n",b.min(axis=1)

print "b cumulative row sum=\n",b.cumsum(axis=1)
```

Łączenie tablic

```
import numpy as np
x = np.array([1,2,3,4,5])
y = np.array(range(6,11))
Zv = np.vstack((x,y))
Zh = np.hstack((x,y))

x1 = np.array([[1,2],[3,4]])
y1 = np.array([[5,6],[7,8]])
Zv1 = np.vstack((x1,y1))
Zh1 = np.hstack((x1,y1))
```

Wizualizacja danych

Wykresy

- Python zawiera moduł matplotlib oferujący funkcjonalność wzorowaną na środowisku MATLAB
- Funkcje "plot" do generacji wykresów jest dostępna w dwu pakietach (do wyboru)

1. `import pylab`

   albo

2. `import matplotlib.pyplot`

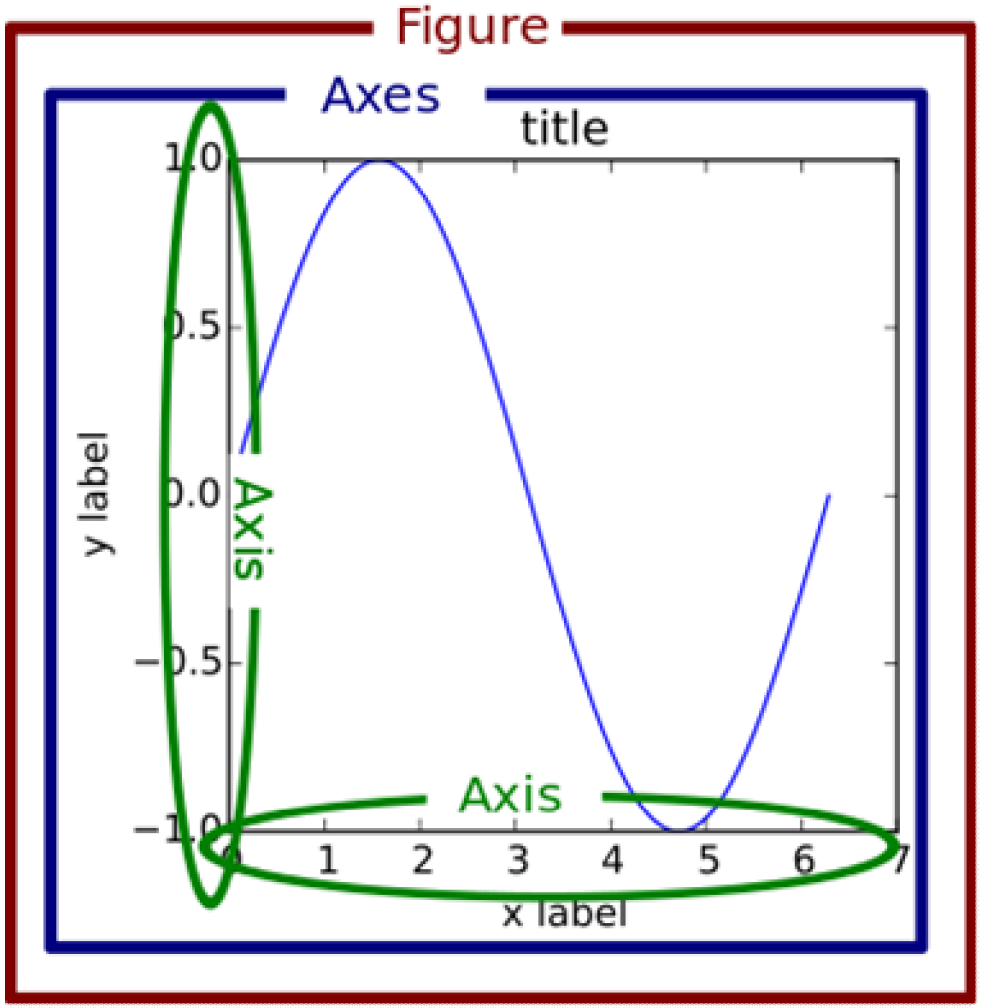
- Wybór pierwszej opcji powoduje zaimportowanie funkcji z pakietu numpy. My będziemy korzystać z

  `import matplotlib.pyplot`

Generacja prostego wykresu

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 2, 100)
plt.plot(x, x, label='liniowa')
plt.plot(x, x**2, label='kwadratowa')
plt.plot(x, x**2*np.sin(x*30), \
    label='sin')
plt.xlabel('os x')
plt.ylabel('os y')
plt.title("Tytul wykresu")
plt.legend()
plt.show()
```

Komponenty wykresu matplotlib



źródło :
http://matplotlib.org/faq/usage_faq.html

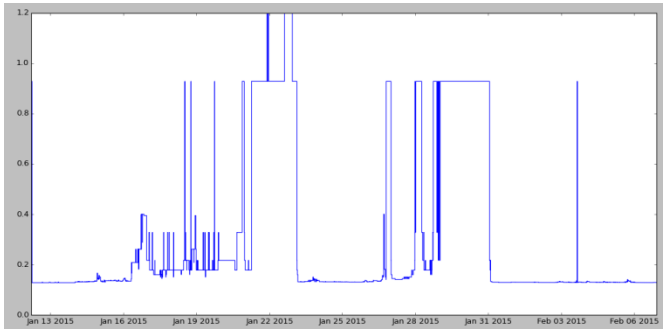
Przykład
dostosowanie osi histogramu

```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)
plt.hist(x, 50, normed=1, facecolor='g')
plt.xlabel('Poziom IQ')
plt.ylabel('Prawdopodobieństwo')
plt.title('Histogram poziomu IQ w populacji')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```

Bardziej praktycznie....
Odczyt danych z pliku + wykres

```
import csv
import matplotlib.pyplot as plt
import datetime
ifile =  open('ceny_spot.txt', "r")
times = []
prices = []
reader = csv.reader(ifile, delimiter='\t')
no = 0
for row in reader:
    no = no+1
    if no > 1:
        prices.append(row[3])
        times.append(datetime.datetime.strptime(row[0],\
            '%Y-%m-%dT%H:%M:%S.%fZ'))
ifile.close()
plt.plot(times,prices)
plt.show()
```

Odczyt danych z pliku + wykres uwzględniamy strukturę zmian cen...




```
import csv
import matplotlib.pyplot as plt
import datetime
ifile =  open('ceny_spot.txt', "r")
times = []
prices = []
reader = csv.reader(ifile, delimiter='\t')
no = 0
for row in reader:
    no = no+1
    if no > 1:
        if (len(times) > 0):
            prices.append(row[3])
            times.append(times[-1])
        prices.append(row[3])
        times.append(datetime.datetime.strptime(row[0],\
            '%Y-%m-%dT%H:%M:%S.%fZ'))
ifile.close()
plt.plot(times,prices)
plt.show()
```

Pobieranie danych z Internetu

```
from pandas import read_csv
import urllib2
url = \
  'http://szufel.pl/python/iris_data.csv'
response = urllib2.urlopen(url)
data = read_csv(response)
response.close()
```

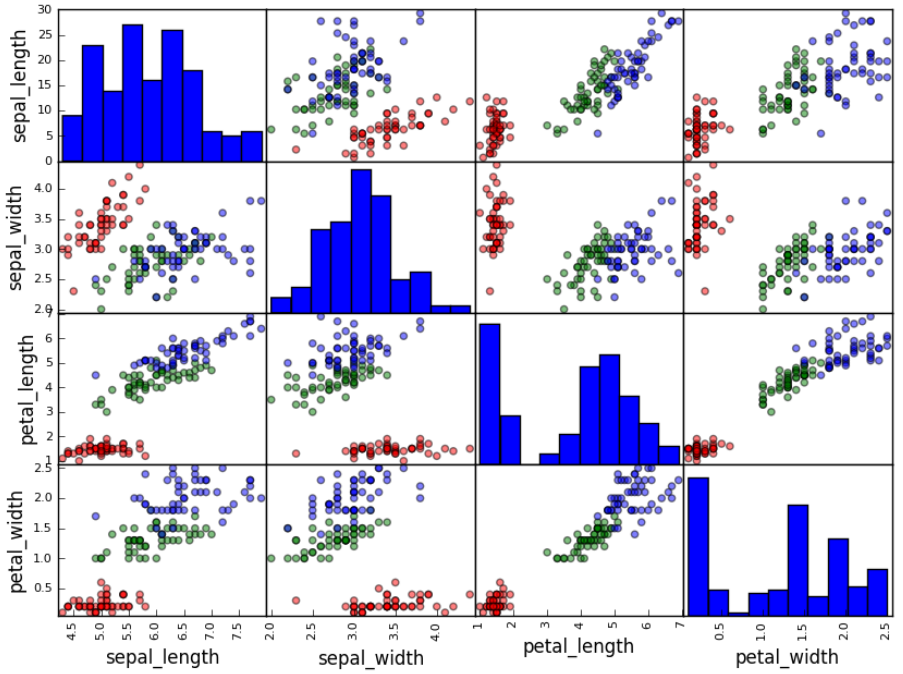
Biblioteka pandas – czytanie ramki danych

```
from pandas import read_csv
import urllib2
url = \
  'http://szufel.pl/pliki/iris_data.csv'
response = urllib2.urlopen(url)
iris_df = read_csv(response)
response.close()

iris_df[1:3]
iris_df['petal_length']
data_df[['petal_length','petal_width']][:5]
data_df['class'].value_counts()
```

Wizualizacja danych wielowymiarowych – scatter plot matrix

```
cmap = {'Iris-setosa': 'red', \
    'Iris-versicolor': 'green', 'Iris-virginica': 'blue'}
from pandas.tools.plotting import scatter_matrix
scatter_matrix(data,\
    c=[cmap[cl] for cl in data['class']] ,marker='o')
```



Machine Learning i Data Mining z językiem Python

- Sieć neuronowa
- Regresja logistyczna
- Drzewo decyzyjne

Scikit-learn – narzędzia data mininig dla Pythona

- http://scikit-learn.org
- Ciągle w wersji beta

```
from sklearn import datasets
import numpy as np
iris = datasets.load_iris()
X = iris.data[:, [2, 3]]
y = iris.target
```


(przykład na podst. Raschka, 2015, Python Machine Learning)

Podział zbioru uczącego i skalowanie

```
from sklearn.cross_validation import \
train_test_split
X_train, X_test, y_train, y_test = \
train_test_split(X, y, test_size=0.3, \
random_state=0)
```

```
from sklearn.preprocessing import \
StandardScaler
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

Sieć neuronowa

```
from sklearn.linear_model import \
Perceptron
ppn = Perceptron(n_iter=50, eta0=0.1,\
random_state=0)
ppn.fit(X_train_std, y_train)


y_pred = ppn.predict(X_test_std)
print 'Bledow klasyfikacji: %d' %
(y_test != y_pred).sum()
```

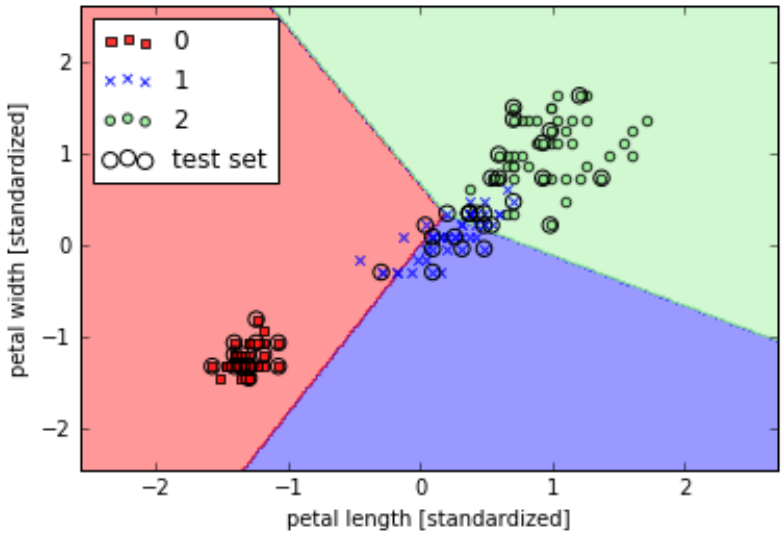
Wizualizacja wyników

(na podst. Raschka, 2015, Python Machine Learning)

```
from matplotlib.colors import ListedColormap
import matplotlib.pyplot as plt
def plot_decision_regions(X, y, classifier, test_idx=None, resolution=0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
    np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    X_test, y_test = X[test_idx, :], y[test_idx]
    for idx, cl in enumerate(np.unique(y)):
        plt.scatter(x=X[y == cl, 0], y=X[y == cl, 1],  \
        alpha=0.8, c=cmap(idx), marker=markers[idx], label=cl)
    if test_idx:
        X_test, y_test = X[test_idx, :], y[test_idx]
        plt.scatter(X_test[:, 0], X_test[:, 1], c='', \
        alpha=1.0, linewidth=1, marker='o', s=55, label='test set')
```

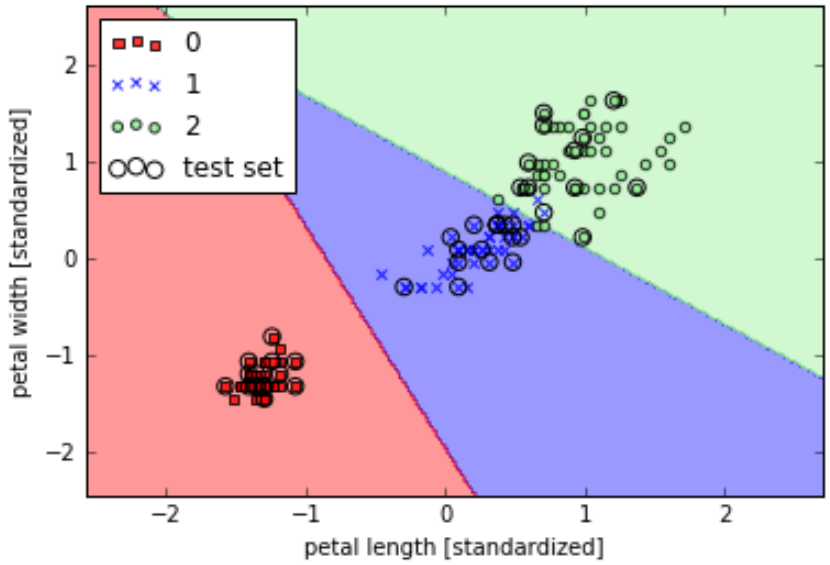
… i sama wizualizacja

```
X_combined_std = np.vstack((X_train_std, X_test_std))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X=X_combined_std, \
  y=y_combined, classifier=ppn, test_idx=range(105,150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```



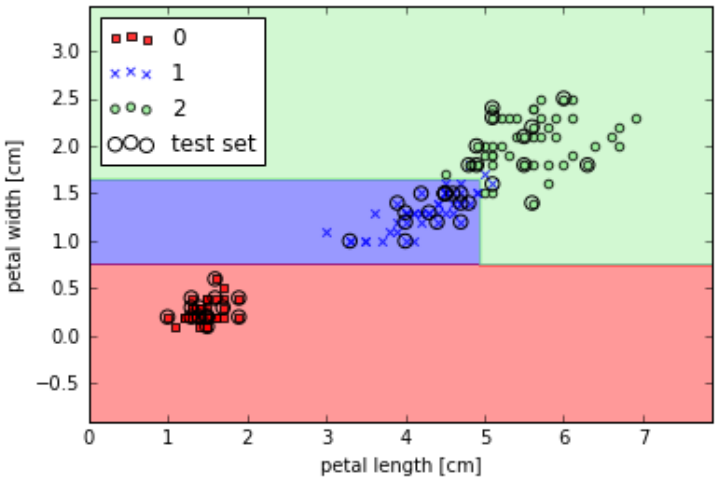
Regresja logistyczna

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(C=1000.0, random_state=0)
lr.fit(X_train_std, y_train)
plot_decision_regions(X_combined_std, y_combined, \
  classifier=lr, test_idx=range(105,150))
plt.xlabel('petal length [standardized]')
plt.ylabel('petal width [standardized]')
plt.legend(loc='upper left')
plt.show()
```



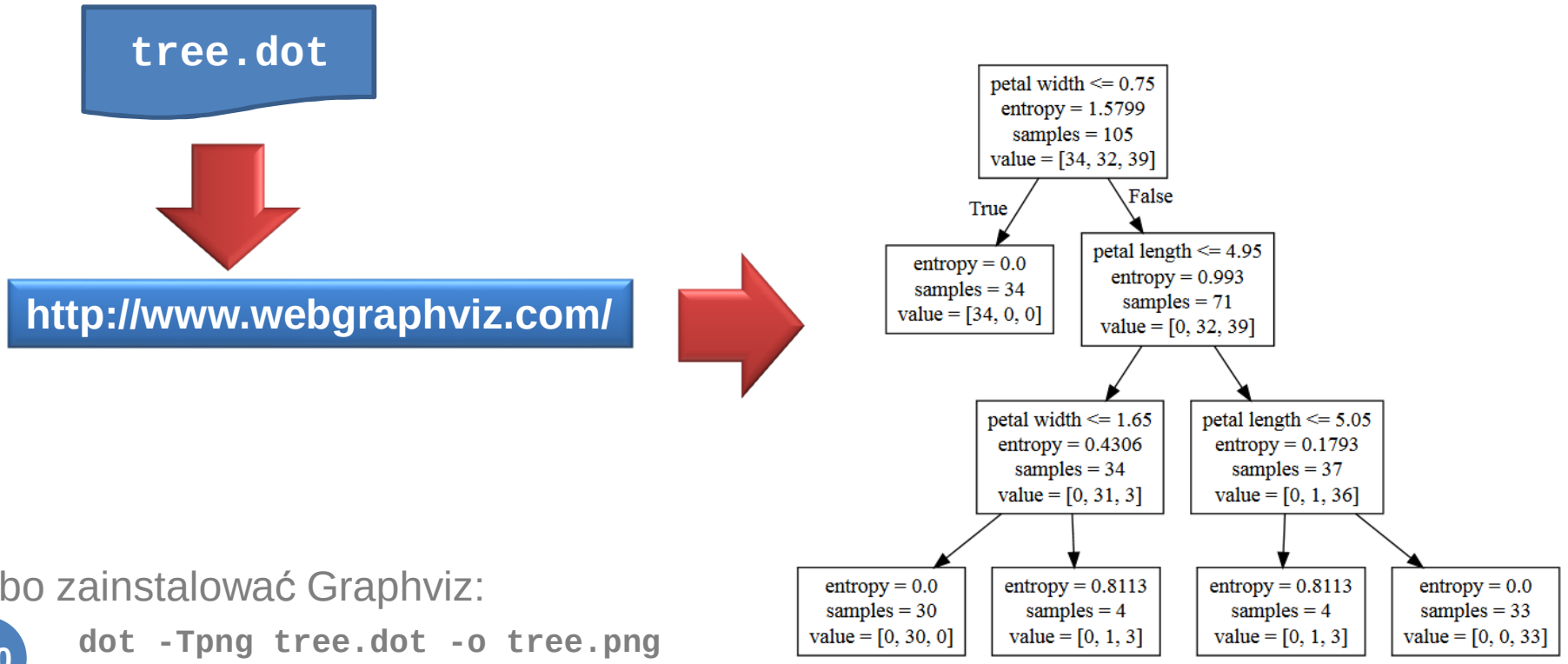
Drzewo decyzyjne

```
from sklearn.tree import DecisionTreeClassifier
tree = \
  DecisionTreeClassifier(criterion='entropy', \
  max_depth=3, random_state=0)
tree.fit(X_train, y_train)
X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined, y_combined, \
  classifier=tree, test_idx=range(105,150))
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.show()
```



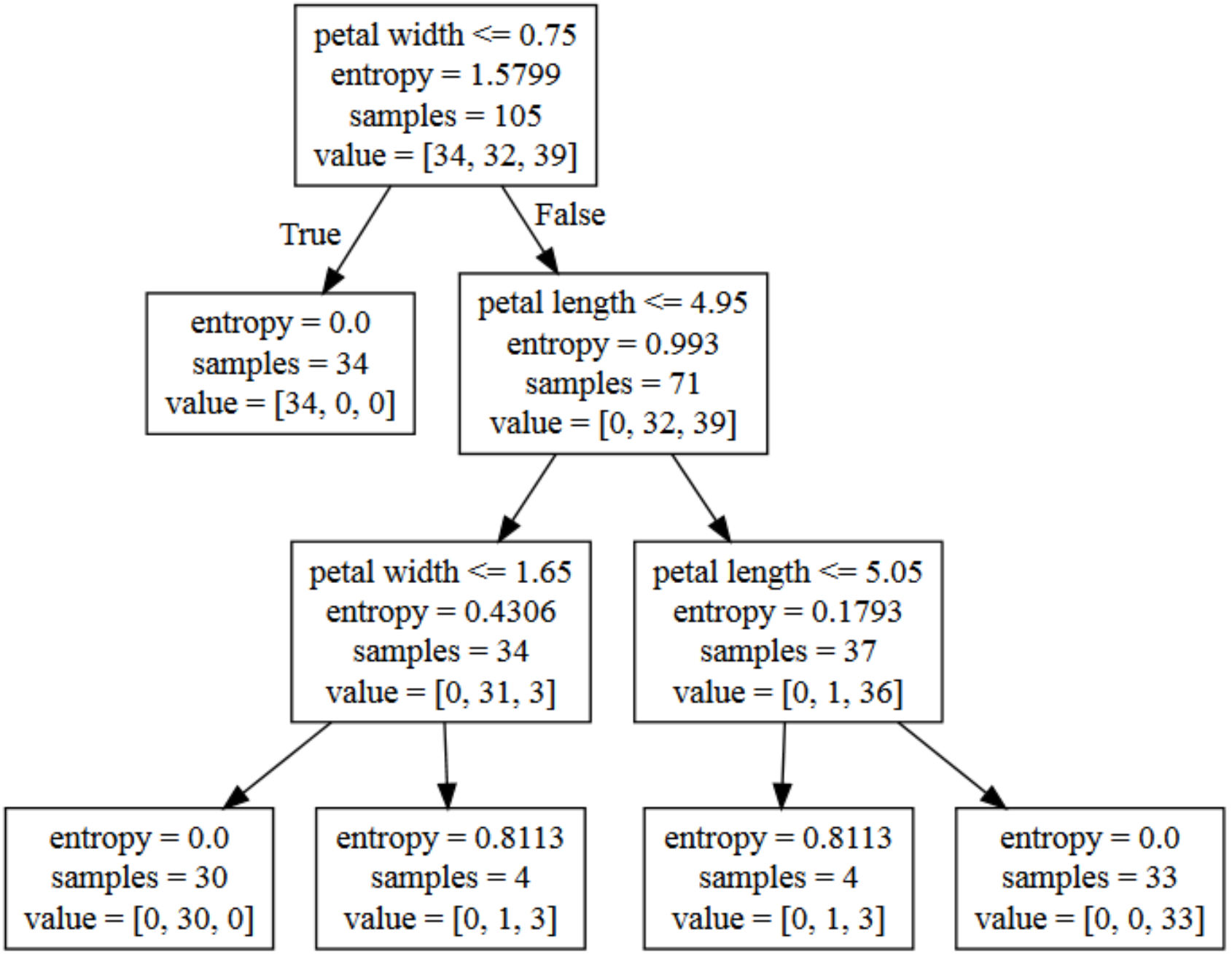
Wizualizacja drzewa niestety wymaga zewnętrznego narzędzia…

```
from sklearn.tree import export_graphviz
export_graphviz(tree,out_file='tree.dot', \
 feature_names=['petal length', 'petal width'])
```



Albo zainstalować Graphviz:

```
dot -Tpng tree.dot -o tree.png
```


petal width <= 0.75
entropy = 1.5799
samples = 105
value = [34, 32, 39]
True
False
entropy = 0.0
samples = 34
value = [34, 0, 0]
petal length <= 4.95
entropy = 0.993
samples = 71
value = [0, 32, 39]
petal width <= 1.65
entropy = 0.4306
samples = 34
value = [0, 31, 3]
petal length <= 5.05
entropy = 0.1793
samples = 37
value = [0, 1, 36]
entropy = 0.0
samples = 30
value = [0, 30, 0]
entropy = 0.8113
samples = 4
value = [0, 1, 3]
entropy = 0.8113
samples = 4
value = [0, 1, 3]
entropy = 0.0
samples = 33
value = [0, 0, 33]

Drzewo decyzyjne – odczyt danych z pliku CSV…

```
import csv
import urllib2
import numpy as np
from sklearn import tree

url = \
    'http://szufel.pl/python/iris_data.csv'
response = urllib2.urlopen(url)
z = np.genfromtxt(response,delimiter=",",\
    dtype=None,names=True)
response.close()

x = np.column_stack([z[name] for name in z.dtype.names[:-1]])
y = z[z.dtype.names[-1]]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x, y)
tree.export_graphviz(clf, out_file="c:/temp/tree.dot")
```

Wprowadzenie do metodyki symulacyjnej

Symulacja

- **Symulacja**

  *Technika **numeryczna**, która polega na przeprowadzaniu **eksperymentów** na **modelu** w celu uzyskania wiedzy na temat wpływu parametrów wejściowych na wybrane mierniki wyjściowe,* por. Law (2007)

- **System** – „…*zestaw elementów wzajemnie powiązanych w sposób bezpośredni lub pośredni…” por. Ackoff (1971)*

- **Eksperyment** - wzbudzanie systemu w kontrolowany sposób w celu obserwacji jego reakcji

- **Model** - opis elementów systemu i relacji pomiędzy nimi

- **Stan systemu** - zbiór zmiennych (parametrów) niezbędnych do opisu systemu w ustalonym okresie

Symulacja - wybrane zastosowania

Biznes

- Modelowanie sieci telekomunikacyjnych
- Badanie wpływu sieci społecznościowych na decyzje klientów

Edukacja i trening

- Pilotaż,
- Gry militarne
- Diagnostyka medyczna

Finanse

- Planowanie awaryjnych scenariuszy
- Ocena ryzyka portfela
- Wycena aktywów

Inżynieria produkcji

- Projektowanie materiałów (samochody, budynki)
- Układ linii produkcyjnych

Przykład zastosowania symulacji dlaczego kawa traci aromat w obecności wody?

The Elusiveness of Coffee Aroma: New Insights from a Non-empirical Approach

LINDSEY J. MUNRO, ALESSANDRO CURIONI, AND WANDA ANDREONI*

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

CHAHAN YERETZIAN AND HERIBERT WATZKE

Nestlé Research Center, 1000 Lausanne 26, Switzerland

Aroma is central to a pleasurable eating/drinking experience but is one of the most labile components of food. Coffee is an outstanding example. Attempts to avoid or control aroma degradation are often frustrated by ignorance of the microscopic mechanisms that are responsible for it. One of the processes most frequently invoked is radical formation, yet the identity of the radicals and their involvement in aroma degradation are poorly understood at the molecular level. Here a step forward in the

Przykład zastosowania symulacji dlaczego kawa traci aromat w obecności wody?

- Rozwiązanie
  - modelowanie na poziomie kwantowym zachowania cząstek wpływających na aromat kawy
  - model numeryczny – obliczenia na superkomputerach

- Zamawiający
  - Nestle
- Dostawca
  - IBM

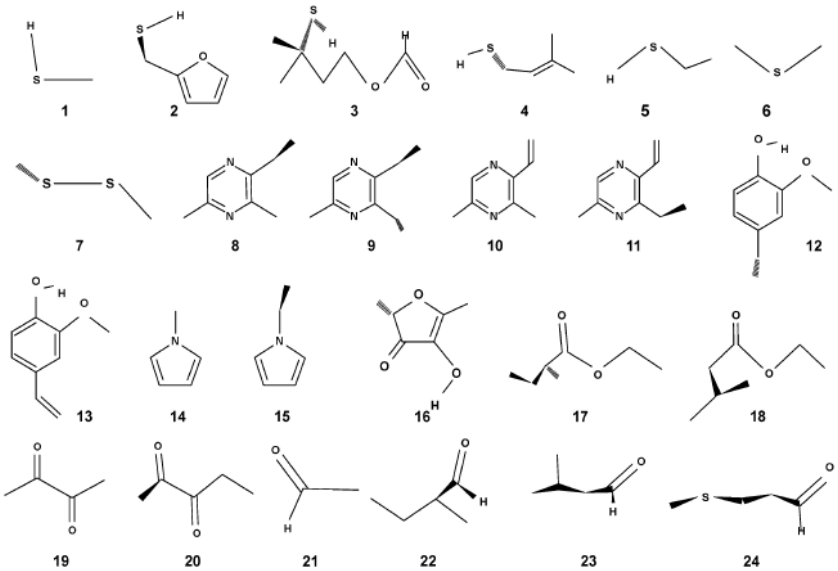
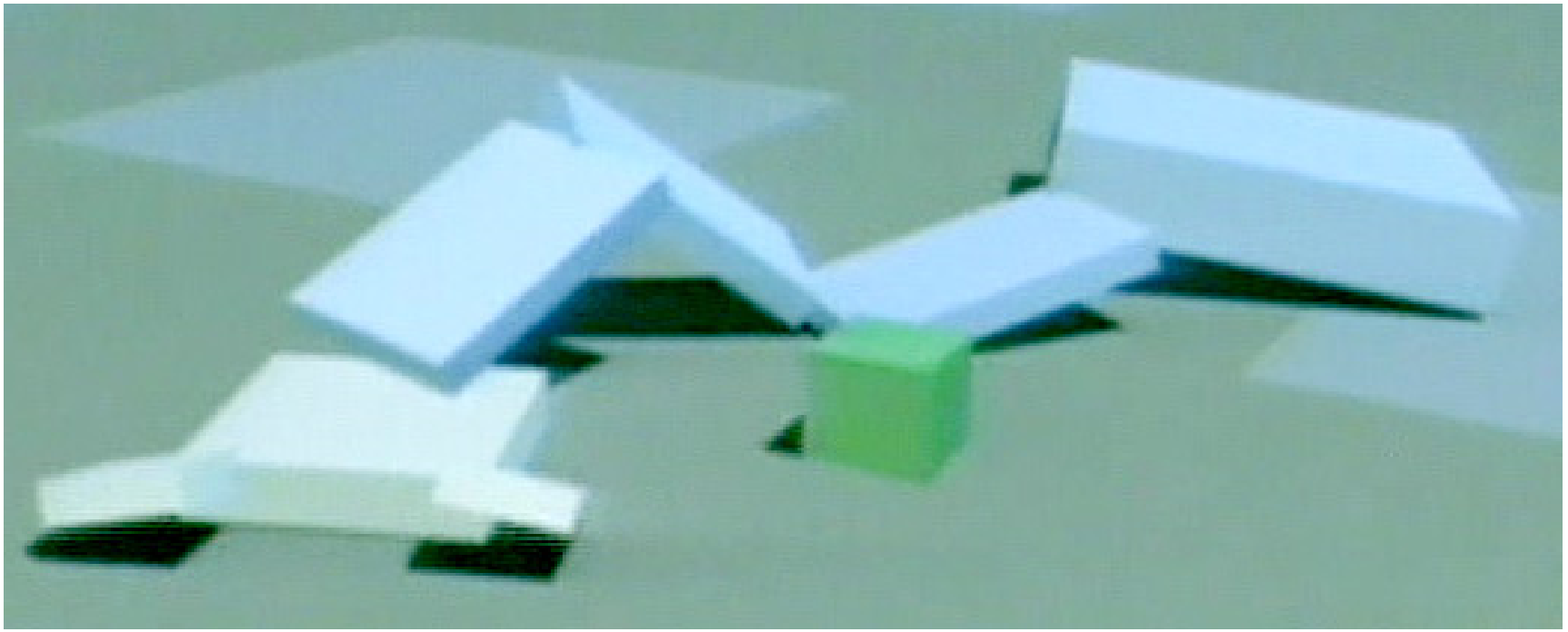


Figure 1. Aroma compounds: (**1**) methanethiol; (**2**) 2-furfurylthiol (FFT); (**3**) 3-mercapto-3-methylbutyl formate; (**4**) 3-methyl-2-butenethiol; (**5**) ethanethiol; (**6**) dimethyl sulfide; (**7**) dimethyl disulfide; (**8**) 2-ethyl-3,5-dimethylpyrazine; (**9**) 2,3-diethyl-5-methylpyrazine; (**10**) 2-vinyl-3,5-dimethylpyrazine; (**11**) 2-vinyl-3-ethyl-5-methylpyrazine; (**12**) 4-ethyl-2-methoxyphenol; (**13**) 4-vinyl-2-methoxyphenol; (**14**) *N*-methylpyrrole; (**15**) *N*-ethylpyrrole; (**16**) 4-hydroxy-2,5-dimethyl-3(2*H*)-furanone (Furaneol); (**17**) ethyl 2-methylbutyrate; (**18**) ethyl 3-methylbutyrate; (**19**) 2,3-butanedione; (**20**) 2,3-pentanedione; (**21**) ethanal; (**22**) 2-methylbutanal; (**23**) 3-methylbutanal; (**24**) 3-(methylthio)propanal (methional).

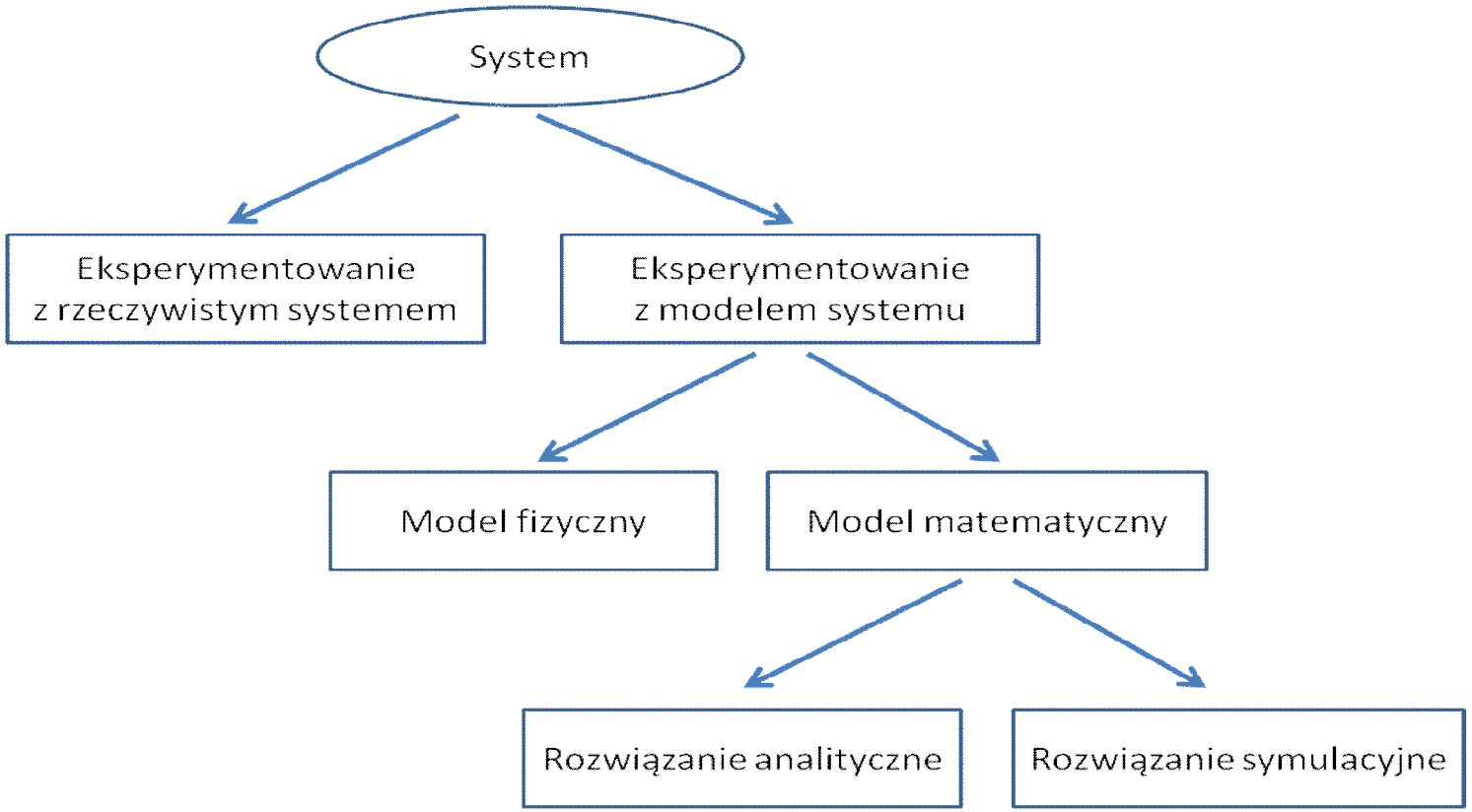
Przykład modelu symulacyjnego – zrozumienie procesu ewolucji

- Karl Sims - Evolved Virtual Creatures, Evolution Simulation, 1994



Kiedy stosować symulację?

por. Law (2007)

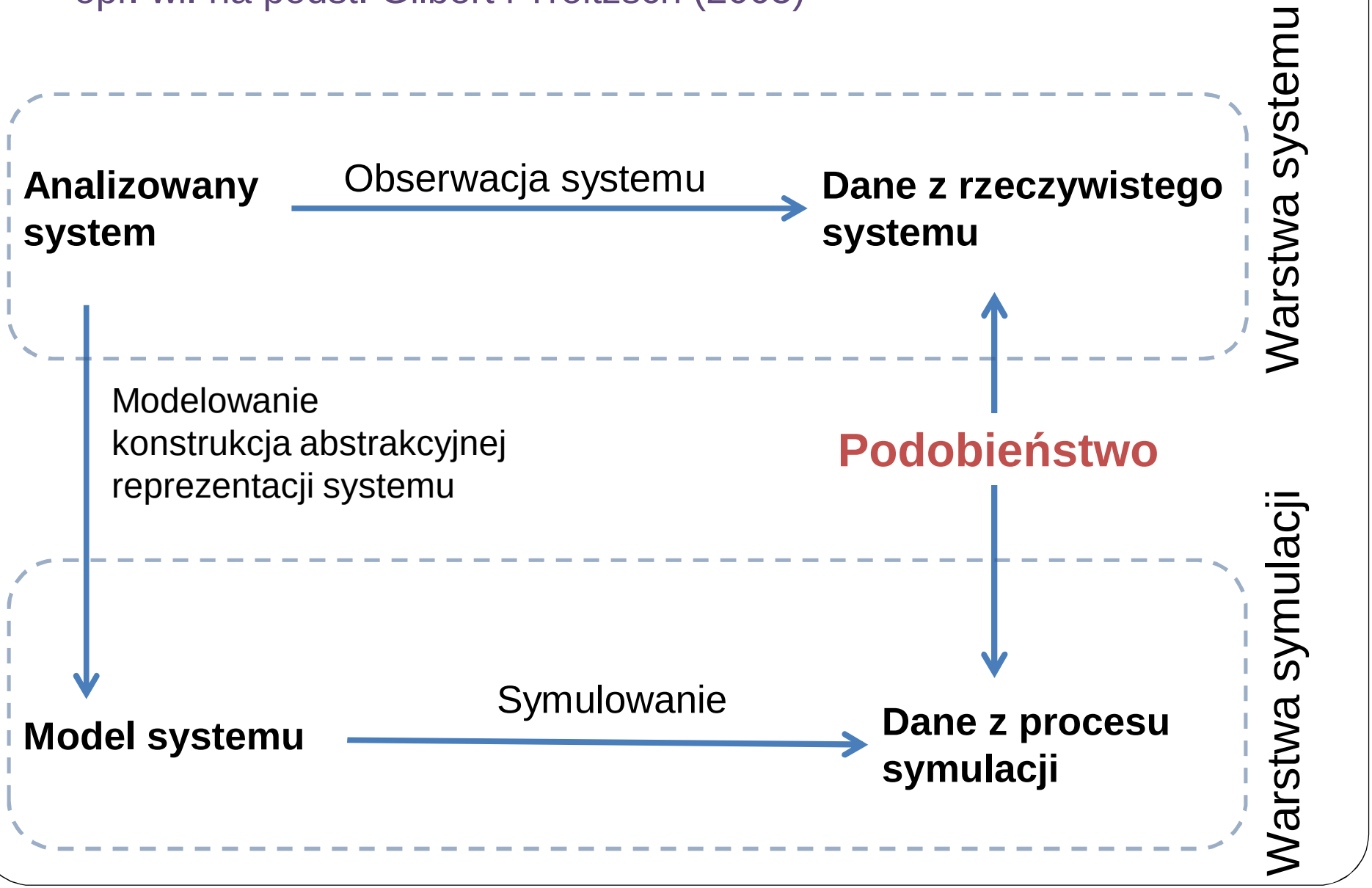


system = zestaw elementów wzajemnie powiązanych w sposób bezpośredni lub pośredni

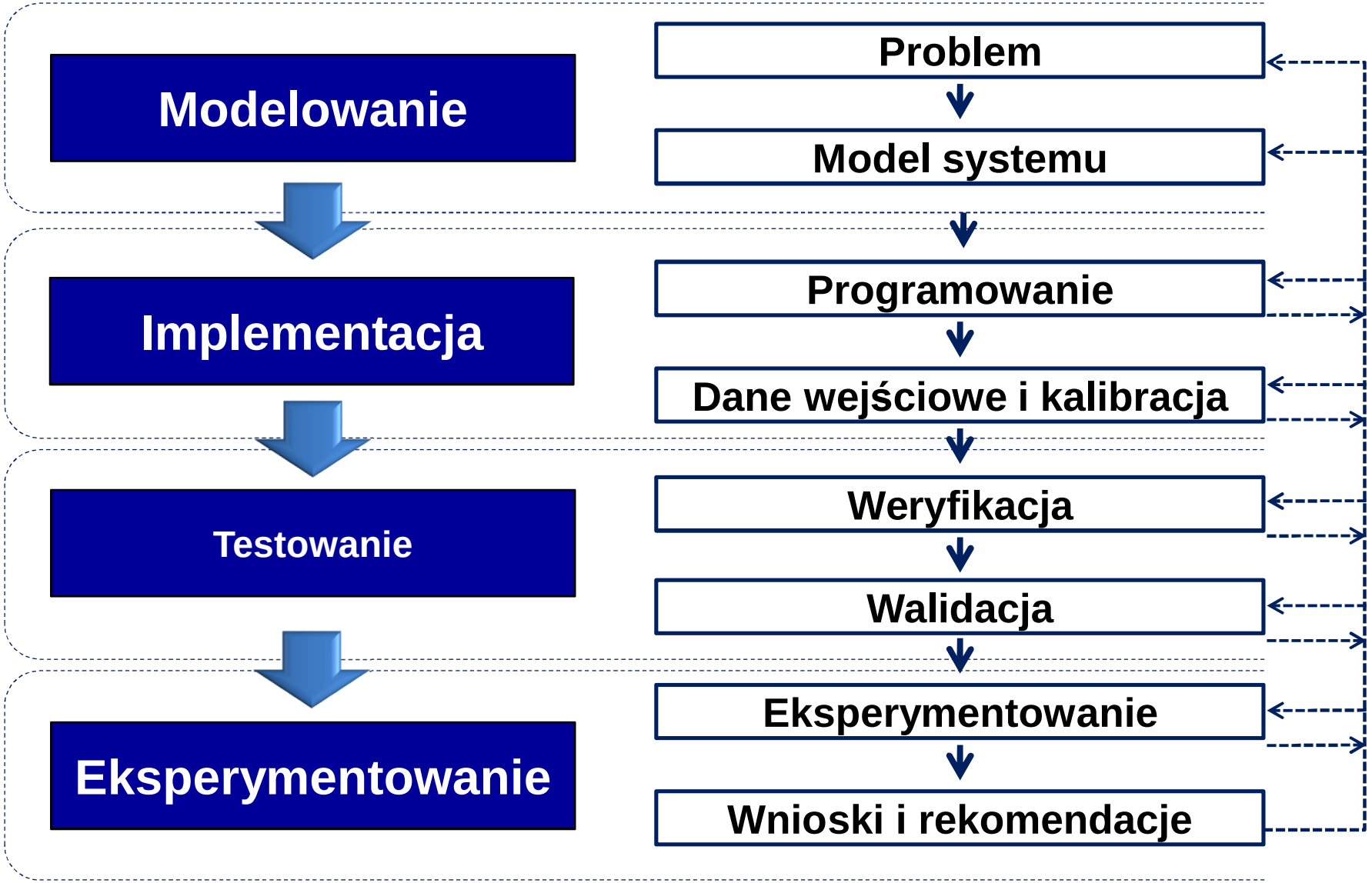
[Ackoff , 1971]

Modelowanie systemu

opr. wł. na podst. Gilbert i Troitzsch (2005)



Etapy procesu symulacji



Źródło: opr. wł. na podstawie: Bennett (1995),Schroeder (1993), Law (2007)

Budowa modelu systemu - trzeba określić...

- ... Procedury przyrostu czasu:
  - Stałe – przegląd działań,
  - Zmienne - przyrosty odpowiadają kolejnym zdarzeniom
- ... Warunki początkowe i stabilizacja modelu
  - Minimalizacja okresu rozruchu,
  - Minimalizacja obciążenia wyników.
- ... Długość eksperymentu:
  - Do osiągnięcia zbieżności do częstości empirycznych – zm. wejściowe
  - Do osiągnięcia zbieżności statystycznej – mierniki wyjściowe,
  - Do uzyskania jednoznacznych wyników w testach statystycznych,
  - Arbitralny czas wynikający ze specyfiki modelowanego systemu

Dane wejściowe i kalibracja

- Dane wejściowe
  - Dane empiryczne umieszczane bezpośrednio w modelu
  - Eksperci
  - Dopasowywanie rozkładów zmiennych losowych (testy statystyczne KS, AD, Chi-kwadrat)
- Kalibracja
  - Poszukiwanie zestawów parametrów prowadzących do zachowania modelu podobnego do rzeczywistego systemu
  - Analiza wrażliwości

Weryfikacja i walidacja modeli symulacyjnych

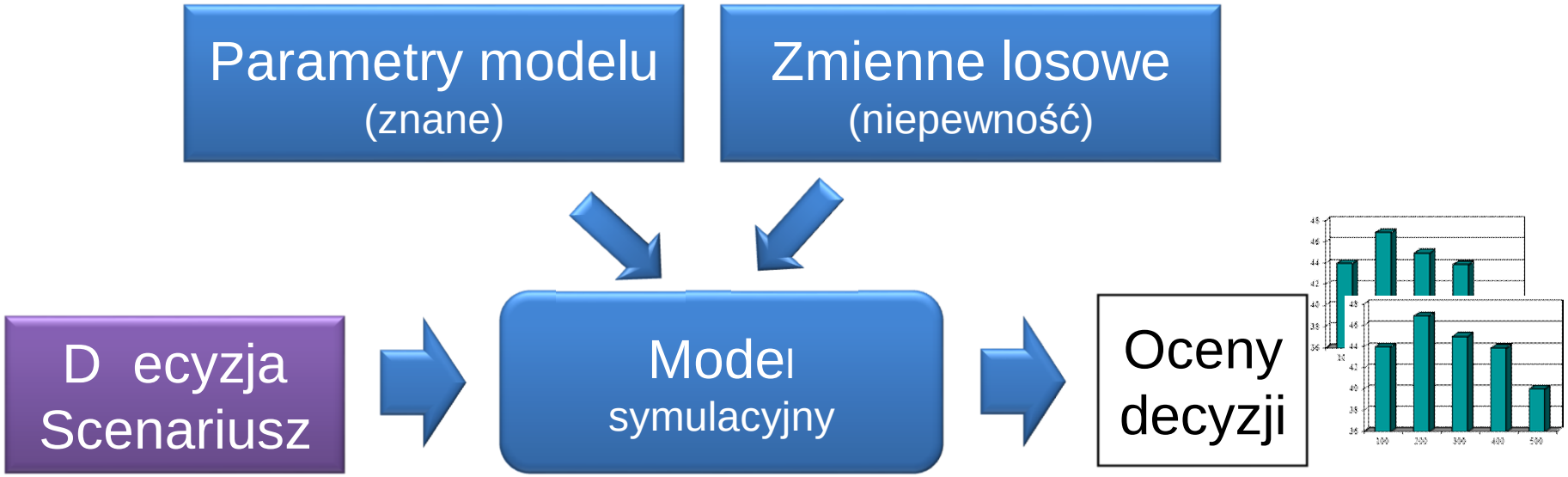
- Weryfikacja – dowodzenie, że model zachowuje się zgodnie z założeniami
  - Metody inżynierii oprogramowania
    - testy jednostkowe (unit testing)
    - Debuggowanie - Śledzenie obiektów w systemie
  - Dokumentacja – diagramy UML i inne, pseudo-kod
  - Animacja
- Walidacja – dowodzenie, że model poprawnie opisuje system rzeczywisty
  - Reprezentatywność danych wyjściowych
  - Porównanie modelu ze scenariuszami historycznymi
  - Porównanie z innymi modelami, w tym warunki brzegowe
  - **Analiza wrażliwości** modelu symulacyjnego

Inne techniki weryfikacji i walidacji modelów symulacyjnych

- Animacja – obserwacja zachowania się modelu w ruchu
- Określenie reprezentatywności danych wejściowych (testy statystyczne, eksperci)
- Walidacja zdarzeniowa – konfrontacja zdarzeń modelowych z realnie działającym systemem (m.in. testy warunków ekstremalnych)
- Śledzenie obiektów w systemie
- Porównanie z innymi modelami

Symulacyjne wspomaganie decyzji

- Eksperymentalna ocena wpływ zjawisk losowych na sytuację decyzyjną
  - Model rzeczywistego systemu
  - Ocena reakcji systemu na zmiany zasad działania lub zmiany struktury systemu
  - Eksperyment powtarzamy **wielokrotnie** w celu oszacowania **rozkładu zmiennej losowej** opisującej skutki decyzji



Meta-modelowanie - analiza wyników symulacji

- Konstrukcja modeli opisujących wyniki eksperymentów złożonych modeli symulacyjnych
  - Modele ekonometryczne
  - Modele eksploracji danych (klasyfikacyjne, reguły asocjacyjne, analiza skupień)
- Symulacja-optymalizacja – poszukiwanie parametrów modelu symulacyjnego maksymalizujących wartości oczekiwane zmiennych wynikowych
  - algorytmy genetyczne i symulowane wyżarzanie
  - Stochastic kriging
  - Cross entropy method

Wybrane rozkłady zmiennych losowych w języku Python

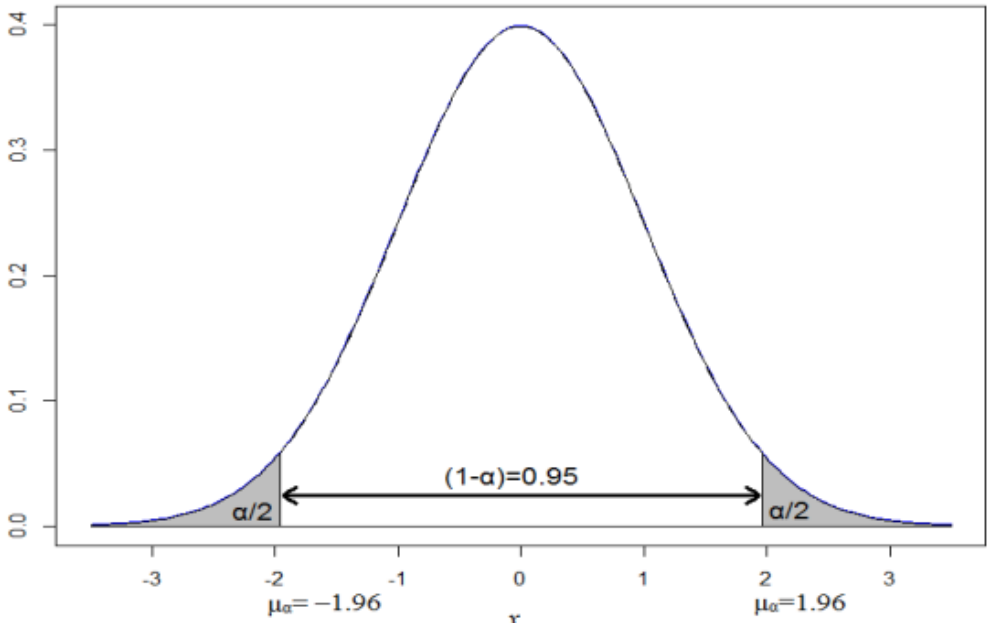
Rozkłady dostępne w Pythonie - numpy

http://docs.scipy.org/doc/numpy/reference/routines.random.html

- beta(a, b[, size])
- binomial(n, p[, size])
- chisquare(df[, size])
- dirichlet(alpha[, size])
- exponential([scale, size])
- f(dfnum, dfden[, size])
- gamma(shape[, scale, size])
- geometric(p[, size])
- gumbel([loc, scale, size])
- hypergeometric(ngood, nbad, nsample[, size])
- laplace([loc, scale, size])
- logistic([loc, scale, size])
- lognormal([mean, sigma, size])
- logseries(p[, size])
- multinomial(n, pvals[, size])
- multivariate_normal(mean, cov[, size])
- negative_binomial(n, p[, size])
- noncentral_chisquare(df, nonc[, size])
- noncentral_f(dfnum, dfden, nonc[, size])
- normal([loc, scale, size])
- pareto(a[, size])
- poisson([lam, size])
- power(a[, size])
- rayleigh([scale, size])
- standard_cauchy([size])
- standard_exponential([size])
- standard_gamma(shape[, size])
- standard_normal([size])
- standard_t(df[, size])
- triangular(left, mode, right[, size])
- uniform([low, high, size])
- vonmises(mu, kappa[, size])
- wald(mean, scale[, size])
- weibull(a[, size])
- zipf(a[, size])

Rozkład normalny
`normal([loc, scale, size])`

- Ciągły rozkład prawdopodobieństwa
- Charakteryzowany przez parametry
  - Położenie (średnia) μ
  - Skala (odchylenie standardowe) σ
- Zapisujemy $N(\mu,\sigma)$



źródło ilustracji: wikipedia.org

Rozkład logarytmiczno-normalny
`lognormal([mean, sigma, size])`

- Ciągły rozkład prawdopodobieństwa
- Charakteryzowany przez parametry
  - Położenia μ
  - Rozproszenia σ
- Wartości cechy o rozkładzie LN(μ,σ) po logarytmizacji mają rozkład N(μ,σ)

- Wartość oczekiwana

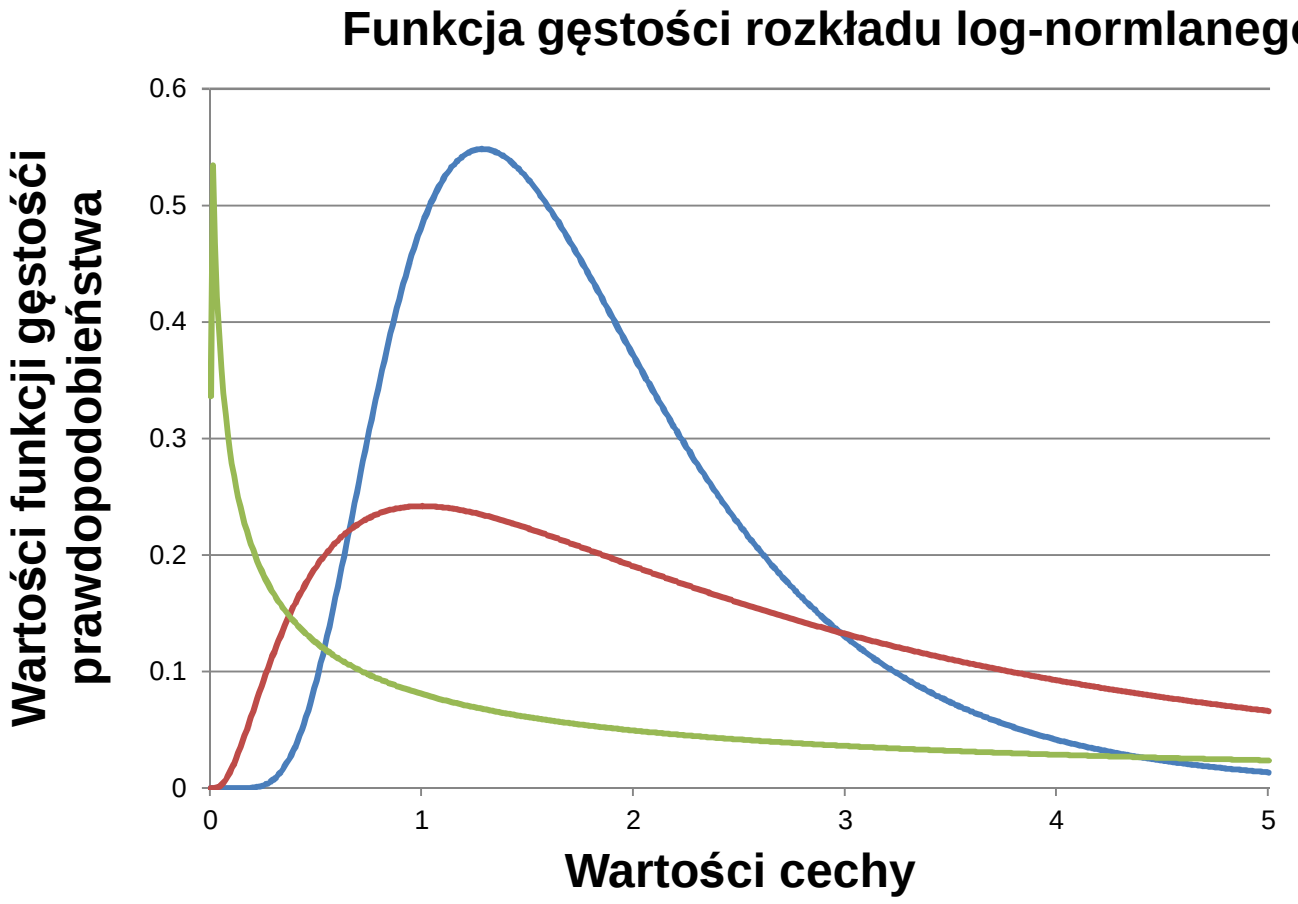
  $E(X)=\exp(\mu+0{,}5\cdot\sigma^2)$
- Wariancja

  $D^2(X)=(\exp(\sigma^2)-1)\cdot \exp(2\cdot\mu+\sigma^2)$

- Występowanie
  - kursy akcji giełdowych (ważne, o ile procent zmienia się cena, a nie, o ile złotych)
  - ceny nieruchomości [zł/m^2]
  - Przybliża rozkład cech, gdzie istotne są ilorazy wartości, a nie różnice pomiędzy nimi

Rozkład logarytmiczno-normalny

`lognormal([mean, sigma, size])`



$\mu = 0.5 \quad \sigma = 0.5$

$\mu = 1 \quad \sigma = 1$

$\mu = 3 \quad \sigma = 3$

Rozkład Poissona

`poisson([lam, size])`

- Dyskretny rozkład prawdopodobieństwa
- Zakładamy, że pojedyncze wydarzenia występują ze znaną średnią częstotliwością, λ, $\lambda>0$, i w sposób niezależny od siebie
- Rozkład Poissona wyraża prawdopodobieństwo wystąpienia określonej liczby wydarzeń, k=0,1,2,…, w ustalonym czasie

- Wartość oczekiwana

  $E(k)=\lambda$
- Wariancja

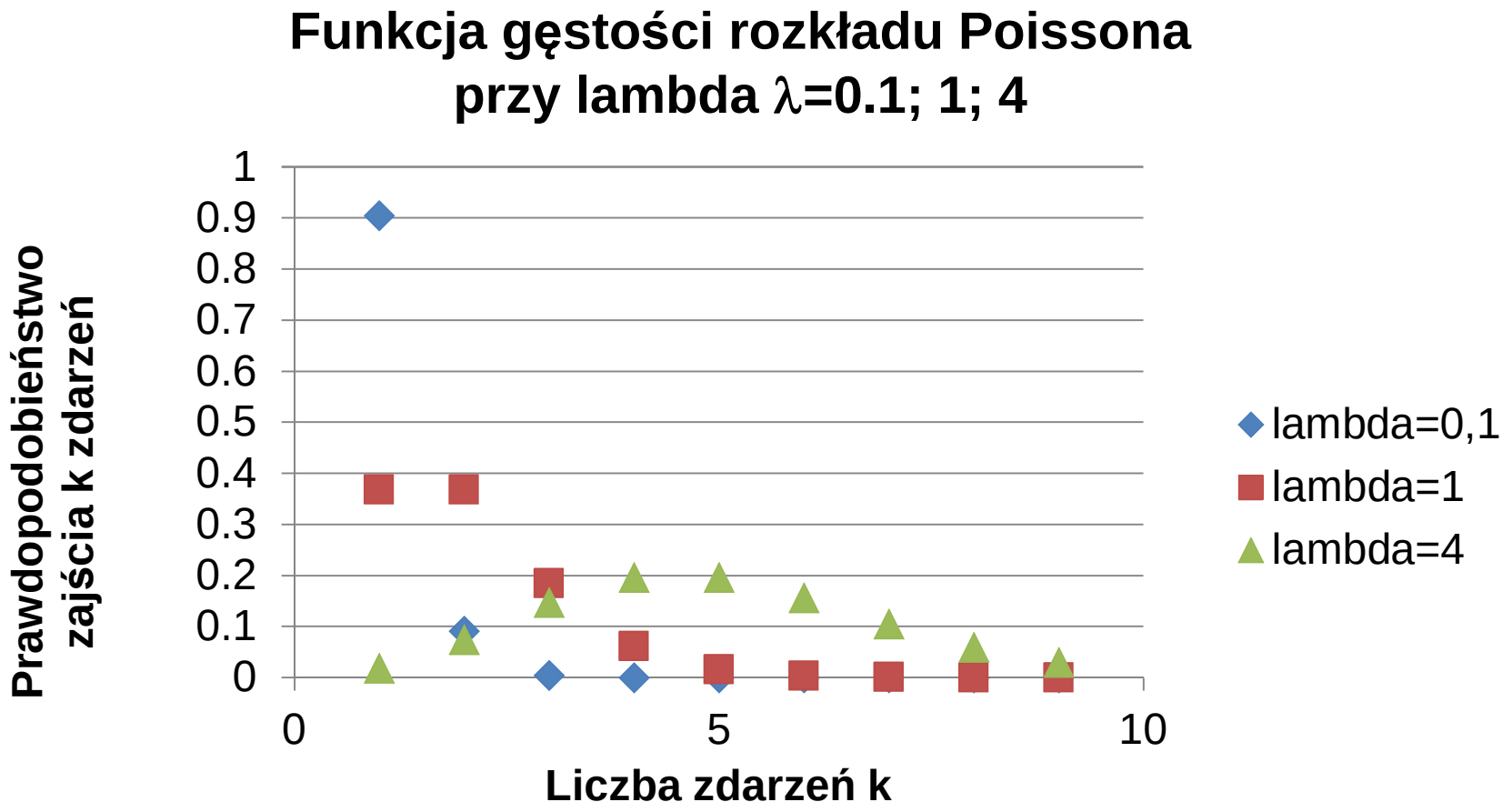
  $D^2(k)=\lambda$
- Dominanta

  $\lambda \in \mathbb{Z}: \lambda - 1$ oraz λ

  $\lambda \notin \mathbb{Z}$: üλü
- Występowanie
  - Liczba połączeń przychodzących do centrali telefonicznej na sekundę
  - Liczba uderzeń pioruna w powiecie grajewskim w sierpniu
  - Liczba wypadków drogowych w Warszawie w ciągu dnia

Rozkład Poissona

`poisson([lam, size])`



Rozkłady dostępne w Pythonie - numpy

http://docs.scipy.org/doc/numpy/reference/routines.random.html

- beta(a, b[, size])
- binomial(n, p[, size])
- chisquare(df[, size])
- dirichlet(alpha[, size])
- exponential([scale, size])
- f(dfnum, dfden[, size])
- gamma(shape[, scale, size])
- geometric(p[, size])
- gumbel([loc, scale, size])
- hypergeometric(ngood, nbad, nsample[, size])
- laplace([loc, scale, size])
- logistic([loc, scale, size])
- lognormal([mean, sigma, size])
- logseries(p[, size])
- multinomial(n, pvals[, size])
- multivariate_normal(mean, cov[, size])
- negative_binomial(n, p[, size])
- noncentral_chisquare(df, nonc[, size])
- noncentral_f(dfnum, dfden, nonc[, size])
- normal([loc, scale, size])
- pareto(a[, size])
- poisson([lam, size])
- power(a[, size])
- rayleigh([scale, size])
- standard_cauchy([size])
- standard_exponential([size])
- standard_gamma(shape[, size])
- standard_normal([size])
- standard_t(df[, size])
- triangular(left, mode, right[, size])
- uniform([low, high, size])
- vonmises(mu, kappa[, size])
- wald(mean, scale[, size])
- weibull(a[, size])
- zipf(a[, size])

Generacja liczb losowych w Pythonie

- Zaimportuj bilbiotekę
- `import numpy as np`
- Zainicjalizuj generator liczb losowych - ustal stałą losowania
- `np.random.seed(0)`
- Wybierz rozkład xxxxx i losuj
- `np.random.xxxxx`

Studium przypadku

prosta analiza symulacyjna - PlusMinus

"Firma PlusMinus importuje do Polski z Chin kurtki narciarskie. Pomiędzy złożeniem zamówienia a przypłynięciem kontenera z Chin upływa 6 miesięcy. 12 pracowników firmy PlusMinus niezależnie oszacowało popyt na kurtki w nadchodzącym sezonie na [14000, 13000, 14000, 14000, 15500, 10500, 16000, 8000, 5000, 11000, 8000, 15000] sztuk.

Kurtki są kupowane w Chinach w cenie 80zł/szt., a następnie sprzedawane do hurtowni w Polsce po 100 zł/szt. Jeżeli kurtki nie zostaną sprzedane w sezonie to ich cena hurtowa spada do 40zł/szt. Całkowity koszt transportu kontenera z Chin wynosi 100'000zł."

Model symulacyjny - implementacja

```
import numpy as np
popyt = np.array([14000, 13000, 14000, 14000, 15500,\
10500, 16000, 8000, 5000, 11000, 8000, 15000])
cena_ch, cena_pl, cena_wyprz = 80, 100, 40
koszt_imp = 100000
srednia = np.average(popyt)
odch_std = np.std(popyt, ddof=1)
np.random.seed(0)
popyt_symul = np.random.normal(srednia,odch_std,10000)
zakupy = 12000
zyski = [min(popyt,zakupy)*cena_pl+ \
max(zakupy-popyt,0)*cena_wyprz - zakupy*cena_ch - \
koszt_imp for popyt in popyt_symul]
```

Model symulacyjny - decyzje

```
zyski = {}
scenariusze = list(range(10000,15000,1000))
for zakupy in scenariusze:
    zyski[zakupy] = \
    [min(popyt,zakupy)*cena_pl+ \
        max(zakupy-popyt,0)*cena_wyprz - \
        zakupy*cena_ch - koszt_imp \
        for popyt in popyt_symul]
```

Model symulacyjny - wizualizacja wyników

```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 100, 15
plt.cla()


plt.hist([zyski[zakupy] for zakupy in scenariusze], \
    50, normed=1, \
    label=["zakupy="+str(zakupy) for zakupy in scenariusze])
plt.xlabel('Zysk')
plt.ylabel('Prawdopodobienstwo')
plt.title('Rozklad poziomu zyskow')
plt.grid(True)
plt.legend(loc=2)
plt.show()
```

Proste studium przypadku - Wilton Toy Company

Studium przypadku – Wilton Toy Company

(treść na podst. Grindley, 1980)

- Firma Wilton Toy Company specjalizuje się w produkcji zabawek. Zarząd firmy planuje rozszerzyć linię produktów.
- Rozważane są dwie opcje
  - Pistolety (najbardziej prawdopodobna jest sprzedaż 30’000szt. po cenie $4 przy kosztach budowy linii produkcyjnej $110’000, spodziewane koszty stałe $4’000/rok, a koszty zmienne $3/szt)
  - Motocykle (najbardziej prawdopodobna jest sprzedaż 6’000szt., po cenie $11,50, przy kosztach budowy linii produkcyjnej $116’000, spodziewane koszty stałe $5’000/rok, a koszty zmienne $6/szt)
- Jako kryterium oceny przedsięwzięć w Wilton przyjęto wskaźnik ROI (Return on Investment)
  - ROI = Zysk netto / koszty inwestycji
  - Pistolety 23.6%
  - Motocykle 24.1%

Wilton c.d.

(treść na podst. Grindley, 1980)

- Firma Wilton zatrudniła konsultanta celem dokładniejszego zbadania niepewności dotyczącej założeń modelu

	Sprzedaż Sztuki	Prawdopod., że wielkość nie zostanie przekroczona
Pistolety	**24 000**	**5%**
	28 000	**30%**
	30 000	**50%**
	31 000	**75%**
	33 000	**95%**
	36 000	**100%**

Wilton Toy Company c.d.

Pistolety

Sprzedaż Sztuki	Prawdopod., że wielkość nie zostanie przekroczona
24 000	5%
28 000	30%
30 000	50%
31 000	75%
33 000	95%
36 000	100%

Koszt zmienny	Prawdopod., że wielkość nie zostanie przekroczona
$ 2,94	5%
$ 2,96	25%
$ 2,98	40%
$ 3,00	50%
$ 3,02	70%
$ 3,04	90%
$ 3,07	100%

Nakłady na Inwestycje	Prawdopod., że wielkość nie zostanie przekroczona
$ 106 000	5%
$ 108 000	15%
$ 109 000	30%
$ 110 000	50%
$ 112 000	80%
$ 115 000	100%

Motocykle

2 500	5%
3 500	30%
6 000	50%
7 500	80%
9 000	95%
11 000	100%

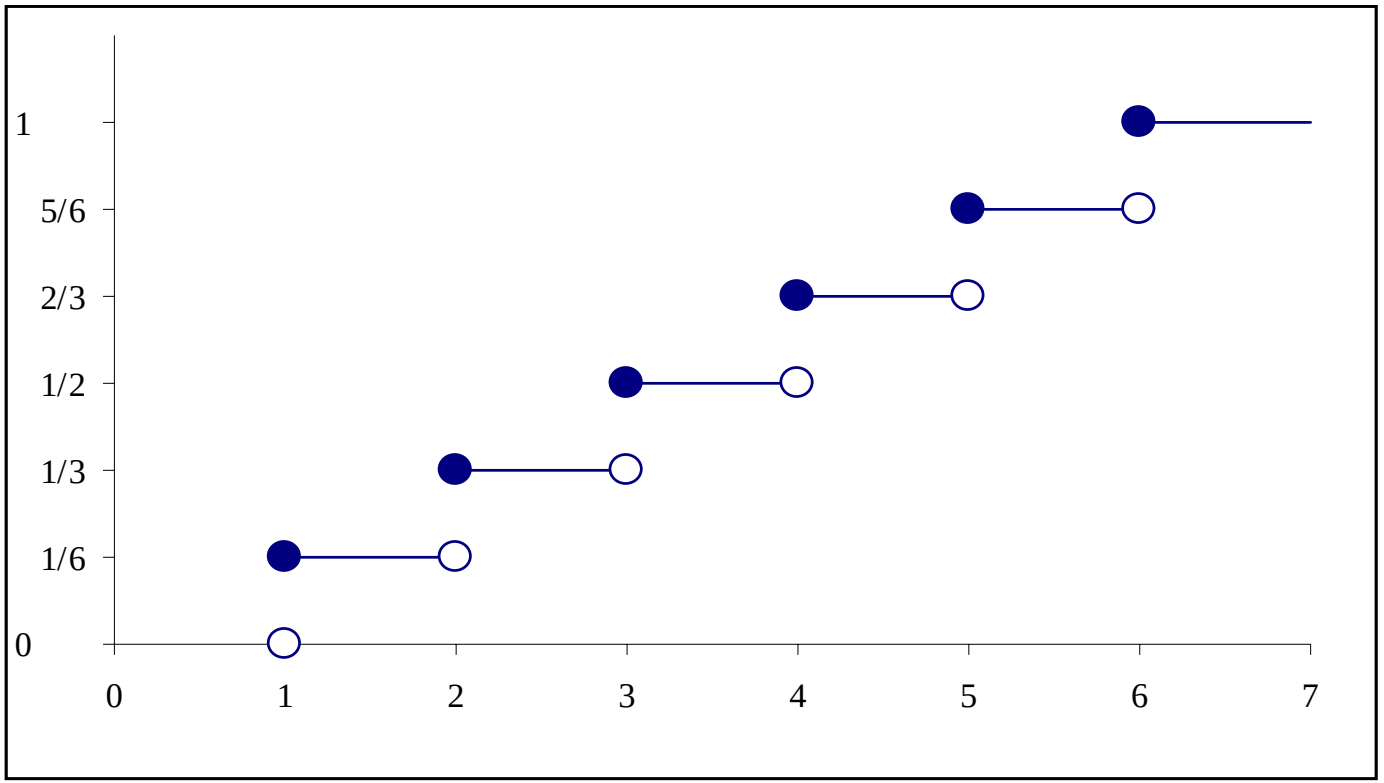
$ 5,85	5%
$ 5,90	20%
$ 5,95	40%
$ 6,00	50%
$ 6,05	75%
$ 6,10	90%
$ 6,15	100%

$ 113 000	5%
$ 115 000	25%
$ 116 000	50%
$ 118 000	70%
$ 120 000	90%
$ 123 000	100%

Jak generować liczby losowe
Metoda odwracania dystrybuanty

- **Odwracanie dystrybuanty F(x) = P(X ≤ x)**
  - **przyporządkuj wartości liczb losowych do przedziałów dystrybuanty**
  - **wylosuj liczbę losową i znajdź ją w przedziale dystrybuanty**
  - **wyznacz wartość zmiennej losowej odpowiadającą wartości dystrybuanty**

$F(x) =$



Wilton Toy Company - dane wejsciowe

```
sprz_pr=[[0.05,0.25,0.20,0.25,0.20,0.05],\
[0.05,0.25,0.20,0.30,0.15,0.05]]
sprz=[[24000,28000,30000,31000,33000,36000],\
[2500,3500,6000,7500,9000,11000]]
koszt_pr=[[0.05,0.20,0.15,0.10,0.20,0.20,0.10],\
[0.05,0.15,0.20,0.10,0.25,0.15,0.10]]
koszt=[[2.94,2.96,2.98,3.00,3.02,3.04,3.07],\
[5.85,5.90,5.95,6.00,6.05,6.10,6.15]]
inwest_pr=[[0.05,0.10,0.15,0.20,0.30,0.20],\
[0.05,0.20,0.25,0.20,0.20,0.10]]
inwest=[[106000,108000,109000,110000,112000,115000],\
[113000,115000,116000,118000,120000,123000]]
```

Metoda odwracania dystrybuanty dla rozkładu dyskretnego w Pythonie

```
def losuj_wagi(values, probs, size):
    bins = np.add.accumulate(probs)
    return np.array(values)[\
       np.digitize(\
       rd.random_sample(size), bins)]
```

Porównywanie i ocena wyników symulacji

Porównywanie wyników

- Testy statystyczne
  - Średnie (istotność różnicy)
  - Przedziały ufności
- Badanie dominacji stochastycznej
- Inne…

Analiza wyników symulacji

podstawowe miary statystyczne

- Liczebność próby a (liczba symulacji) n `len(a)`
- Wartość minimalna i maksymalna `(np.min(a), np.max(a))`
- Średnia z próby u= `np.average(a)`
- Odchylenie standardowe s = `np.std(a,ddof=1)`
- Wariancja s^2
- Skośność (miara asymetrii)

$$b_1 = \frac{m_3}{s^3} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^3}{\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2\right]^{3/2}},$$

- Kurtoza (miara spłaszczenia)

$$g_2 = \frac{m_4}{m_2^2} - 3 = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2\right)^2} - 3$$

- **Kod Python - moduł stats**

```
import scipy.stats
n, min_max, srednia, wariancja, skosnosc, kurtoza =  \
        stats.describe(s)
```

Analiza wyników symulacji
porównywanie średnich

- Testy statystyczne (dla dużej próby = „wystarczającej” liczby symulacji)
  - Test na istotność różnicy pomiędzy średnimi, żaden z parametrów nie jest znany, rozkład nie jest znany

    H^0 $m_1 = m_2$

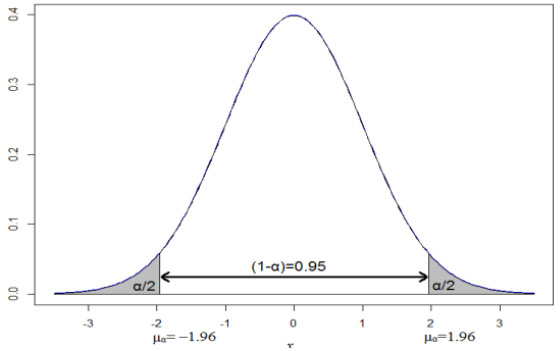
    H^1 $m_1 \neq m_2$

$$U = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

```
scipy.stats.ttest_ind(a, b, equal_var=False)
```

- Estymacja przedziałowa średniej (dla dużej próby)

$$P\left(\overline{X} - u_{1-\frac{\alpha}{2}}\frac{S}{\sqrt{n}} < m < \overline{X} + u_{1-\frac{\alpha}{2}}\frac{S}{\sqrt{n}}\right) = 1 - \alpha$$

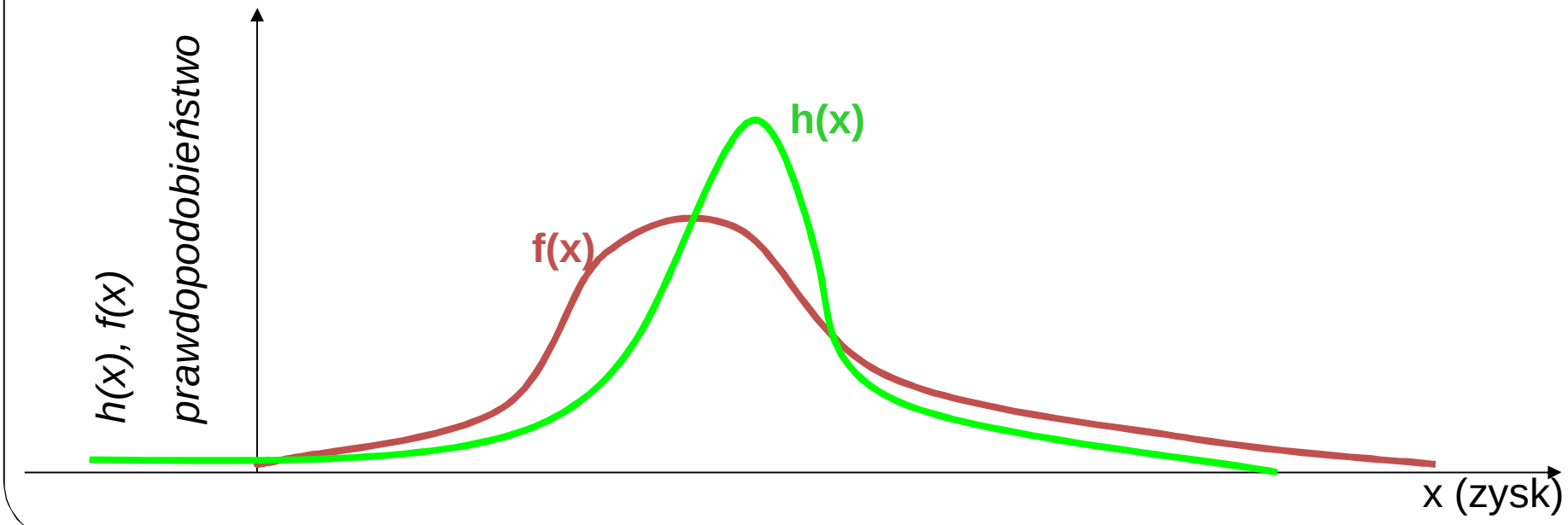



```
import numpy as np
x = np.mean(a),
z = scipy.stats.sem(a) * sp.stats.t._ppf(1-alfa/2., n-1)
przedział = (x-z, x+z)
```

Dominacja stochastyczna

- Porównujemy dwa rozkłady prawdopodobieństwa cechy ciągłej x: F(x) oraz H(x)

  np. rozkład zysku przy inwestycjach F: f(x) oraz H: h(x)

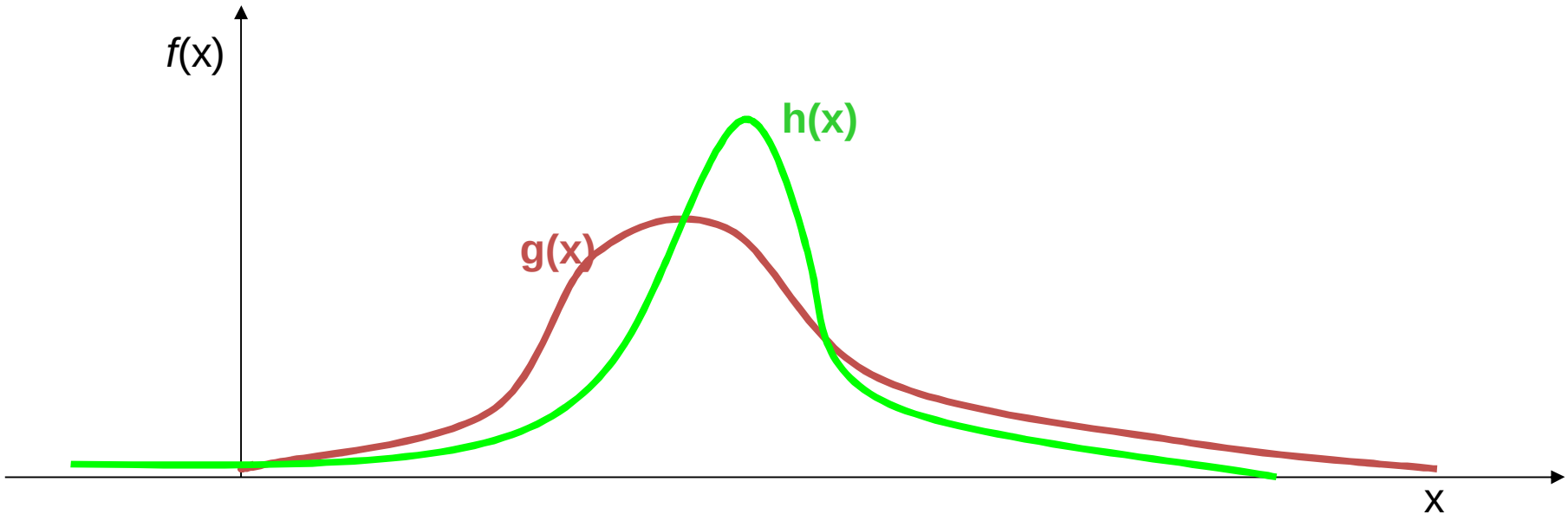


Analiza wyników symulacji

Dominacja stochastyczna

- Porównujemy dwa rozkłady prawdopodobieństwa cechy ciągłej x: F(x) oraz G(x)

  np. rozkład zysku przy inwestycjach H: h(x) oraz G: g(x)

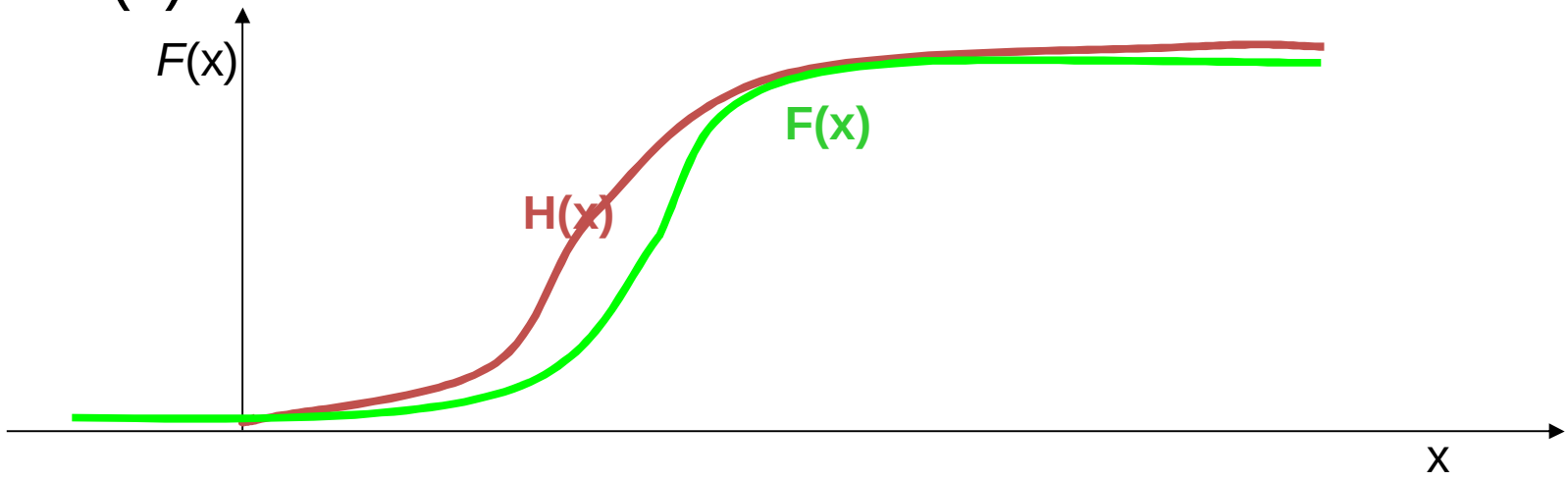


Analiza wyników symulacji

- Dominacja stochastyczna I rzędu:

  zmienna losowa F dominuje stochastycznie zmienną losową H, jeżeli jej wartości jej dystrybuanty są nie większe i w przynajmniej jednym punkcie mniejsze a dokładnie:

$$F(x) \succ H(x) \Leftrightarrow \forall_x H(x) \geq F(x) \wedge \exists_x H(x) > F(x)$$

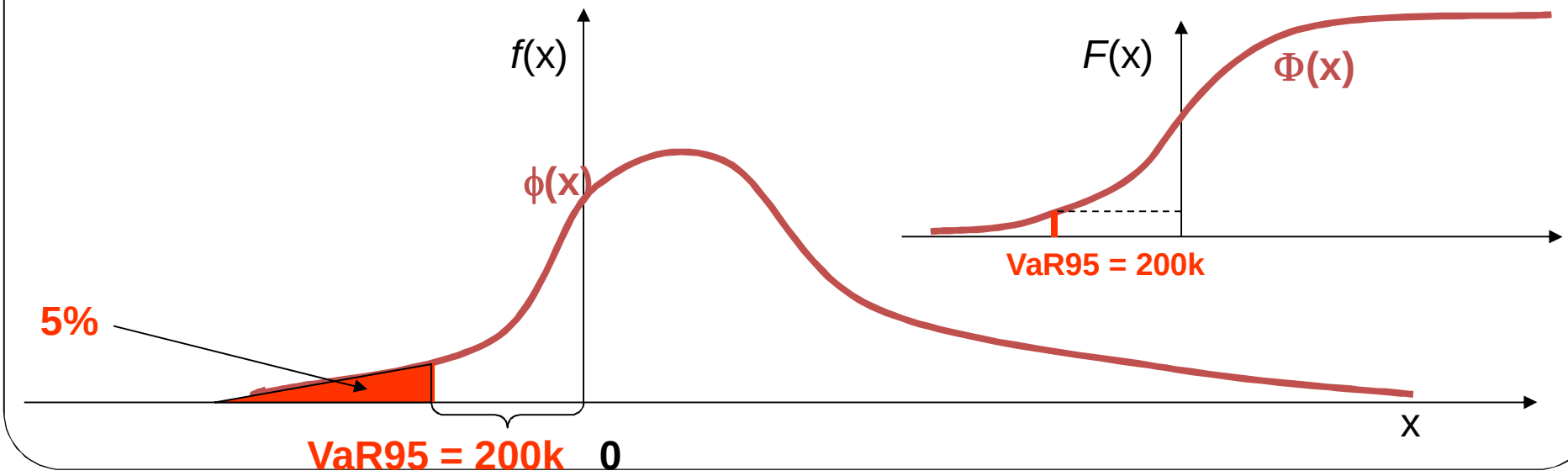


VaR / EaR / EPSaR / CFaR / CCFaR / LaR

- Jeśli rozważana zmienna cechuje się rozkładem o dystrybuancie Φ, to VaRα podaje poziom strat względem W_0 przy poziomie ufności α, wtedy szansa przekroczenia VaRα wynosi 1- α

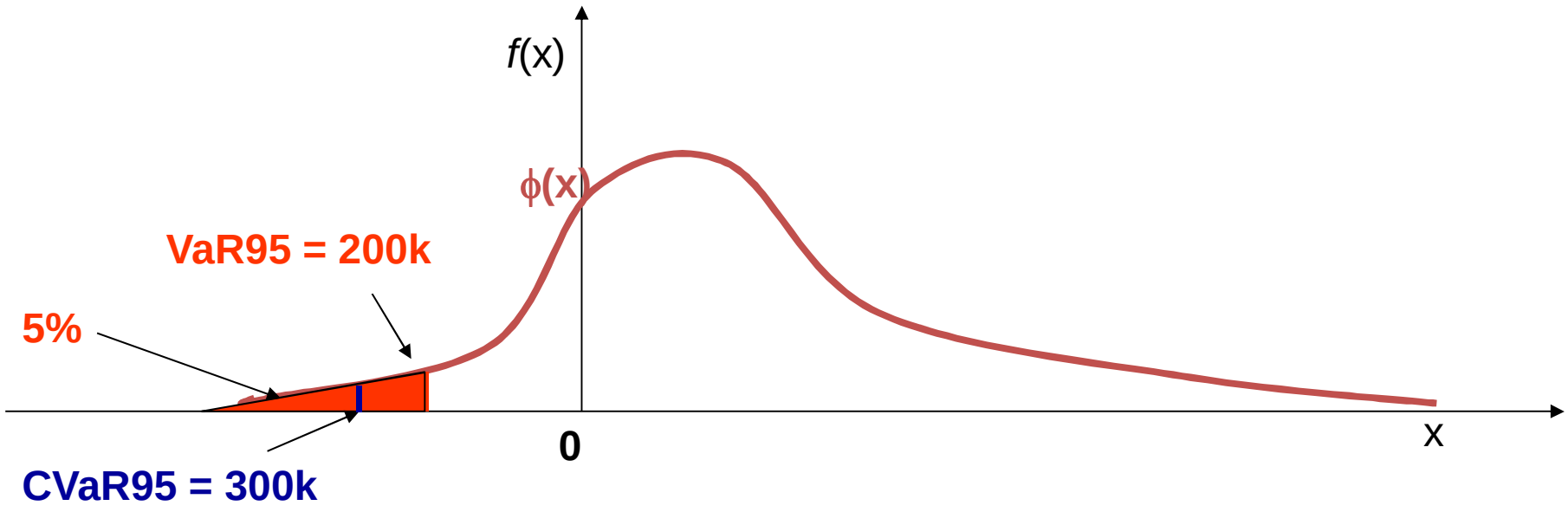
$$P(W \leq W_0 - VaR\alpha) = 1-\alpha$$

- Np. Jeśli $W_0=0$, to VaR95 = $-\Phi^{-1}(0,05)$ – jest jedynie 5% szans, że realne straty będą większe niż VaR95, VaR99 = $-\Phi^{-1}(0,01)$



CVaR / ES

- Jeśli rozważana zmienna cechuje się rozkładem o dystrybuancie Φ, to CVaRα podaje przeciętny poziom najgorszego 1- α odsetka strat
- Np. $\text{CVaR}\alpha = -{}^{100}/_{100-\alpha} \cdot \int_{x<-\text{Var}\alpha} x \cdot \phi(x)dx$
  – przeciętny poziom najgorszych 5% wyników wynosi CVaR95



Przykład - badanie dominacji stochastycznej

```
import statsmodels.api as sm
import numpy as np
import matplotlib.pyplot as plt
plt.cla()
for zakupy in scenariusze:
    x = np.linspace(min(zyski[zakupy]),max(zyski[zakupy]),100)
    ecdf = sm.distributions.ECDF(zyski[zakupy])
    plt.plot(x,ecdf(x), label= "zakupy="+str(zakupy))
x1,x2,y1,y2 = plt.axis()
plt.axis((x1,x2,y1,1.1))
plt.xlabel('Zysk')
plt.ylabel('Skumulowane prawdopodobienstwo')
plt.title('Dystrybuanty poziomu zyskow')
plt.grid(True)
plt.legend(loc=2)
plt.show()
```

Studium przypadku
prosty model analizy ubezpieczeniowej

Prosty przykład ubezpieczeniowy

Firma ubezpieczeniowa dokonuje rekalkulacje składki dla ubezpieczenia komunikacyjnego.

Historyczne dane o liczbie szkód przedstawia tabela:

```
l_szkod = {          # liczba szkod : liczba polis
  0 : 3437,
  1 : 522,
  2 : 40,
  3 : 2,
  4 : 0,
  5 : 0
}
```

Historia szkód…

- https://szufel.pl/pliki/szkody.txt

```
1;4365
2;27088
3;585
4;1104
5;54581
6;7084
7;819
8;12192
9;1649
10;12609
11;6440
12;1546
13;3656
```

Dobór parametrów modelu

```
import csv
import urllib2 as ul  #import urllib.request as ul
import codecs
import matplotlib.pyplot as plt
import scipy as sc
from scipy.stats.stats import kstest

srednia_l_szkod = sum( [x*y for x,y in l_szkod.items()] )/sum(l_szkod.values())

szkody = []
response = ul.urlopen('http://szufel.pl/pliki/szkody.txt')
data = csv.reader (codecs.iterdecode(response, 'utf-8') ,delimiter=";")
for row in data:
        szkody.append(int(row[1]))
plt.hist(szkody,bins=20)
plt.show()

szkody_ln = sc.log(szkody)
plt.hist(szkody_ln,bins=20)
plt.show()
print("Test KS",kstest(szkody_ln, sc.stats.norm.cdf, args=(sc.mean(szkody_ln),sc.std(szkody_ln))))
#H0 – rozkład normalny – p-value=0.99 brak podstaw do odrzucenia H0

sr_szkoda_ln = sc.mean(szkody_ln)
std_szkoda_ln = sc.std(szkody_ln)
```

Model ubezpieczeniowy

```
import numpy.random as rd
def symuluj_ubezpieczenia(l_klientow,nadwyzka,skladka,srednia_l_szkod,sr_szkoda_ln,std_szkoda_ln):
    daty_umow = rd.randint(0,365, l_klientow)
    kal_l_wplat = np.zeros(365+365+30, dtype="int")
    for dataUmowy in daty_umow:
        kal_l_wplat[dataUmowy] += 1
    l_szkod_k = rd.poisson(srednia_l_szkod,l_klientow)
    kal_l_wyplat = np.zeros(365+365+30, dtype="int") #365 to zapas
    for k in range(l_klientow):
        for s in range(l_szkod_k[k]):
            #dla kazdej szkody ustal date wyplaty
            data_wyp = daty_umow[k]+rd.randint(0,365)+rd.randint(15,30)
            kal_l_wyplat[data_wyp] += 1
    for dzien in range(len(kal_l_wyplat)):
        nadwyzka += kal_l_wplat[dzien]*skladka
        l_wyplat = kal_l_wyplat[dzien]
        odszkodowania = 0
        if l_wyplat>0:
            odszkodowania=np.sum(rd.lognormal(sr_szkoda_ln,std_szkoda_ln,l_wyplat))
        if (nadwyzka<odszkodowania):
           return (nadwyzka-odszkodowania, dzien)
        nadwyzka -= odszkodowania
    return (nadwyzka, dzien)
```

Przeszukiwanie przestrzeni parametrów modelu

```
def run_symulacja(blok):
    l_szkod = {            # liczba szkod : liczba polis
      0 : 3437,
      1 : 522,
      2 : 40,
      3 : 2,
      4 : 0,
      5 : 0
    }
    srednia_l_szkod = sum( [x*y for x,y in l_szkod.items()] )*1./sum(l_szkod.values())
    sr_szkoda_ln = 7.9953648143576634
    std_szkoda_ln = 0.9644771368064744
    with open("C:\\temp\\wynik"+str(blok)+".txt","w") as f:
        cs = csv.writer(f,delimiter="\t",quotechar=None)
        for skladka in range(500+blok*100,500+(blok+1)*100,25):
            rd.seed(0)
            wyniki=[symuluj_ubezpieczenia(10000,10000,\
                skladka,srednia_l_szkod,sr_szkoda_ln,std_szkoda_ln) \
                    for i in range(100)]
            srednia = np.mean([x[0] for x in wyniki if x[0] >= 0])
            liczba_ruin = np.sum([1 for x in wyniki if x[0] < 0])
            sredni_dzien_ruiny = np.mean([x[1] for x in wyniki if x[0] < 0])
            cs.writerow([skladka,srednia,liczba_ruin,sredni_dzien_ruiny])
```

Studium przypadku
- modelowanie chmury

Bardziej zaawansowany model...

optymalizacja kosztów portalu internetowego w chmurze

- Popularny polski portal internetowy buziewalbumie spodziewa się wzmożonego ruchu w okresie świątecznym. Portal korzysta obecnie korzysta z 300 serwerów Amazon EC2 typu c3.4xlarge w regionie us-east-1, które zostały zarezerwowane w cenie 0.42$/h. Każdy serwer c3.4xlarge w danej minucie może obsłużyć jednocześnie 100 użytkowników portalu.
- Portal zarabia na śledzeniu swoich użytkowników i sprzedawaniu wszystkim chętnym danych przechowywanych i przetwarzanych przez użytkowników. Rząd Państwa Miłującego Pokój (PMP) zaoferował firmie 0.00021$ za każdą minutę szczegółowych danych o aktywności użytkowników w okresie świątecznym żądając przy tym wyłączności.

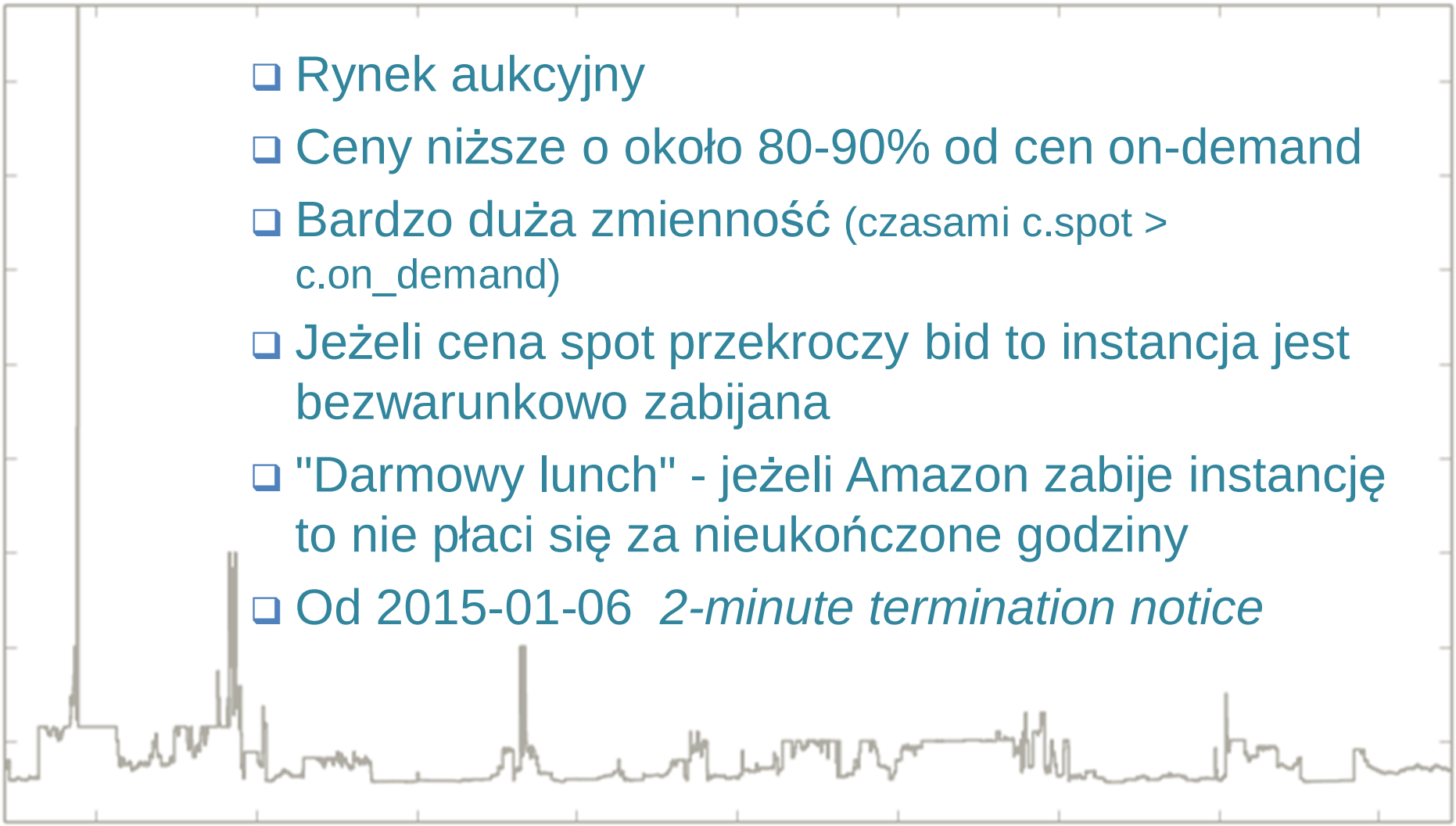
Zaawansowany model c.d.

- Firma buziewalbumie szacuje, że w okresie świątecznym w każdej minucie z portalem będzie próbowało się łączyć średnio 40'000 osób z odchyleniem standardowym 5'000. Firma rozważa:
  - wynajem serwerów on-demand w cenie 0.84$
  - zakup serwerów spot

- *Uwaga!* Przyjęte uproszczenia: W analizie pomijamy dobowe wahania liczby użytkowników oraz zakładamy rozważamy czas w odstępach minutowych

Więcej o rynku Amazon EC2 spot

- Rynek aukcyjny
- Ceny niższe o około 80-90% od cen on-demand
- Bardzo duża zmienność (czasami c.spot > c.on_demand)
- Jeżeli cena spot przekroczy bid to instancja jest bezwarunkowo zabijana
- "Darmowy lunch" - jeżeli Amazon zabije instancję to nie płaci się za nieukończone godziny
- Od 2015-01-06 *2-minute termination notice*

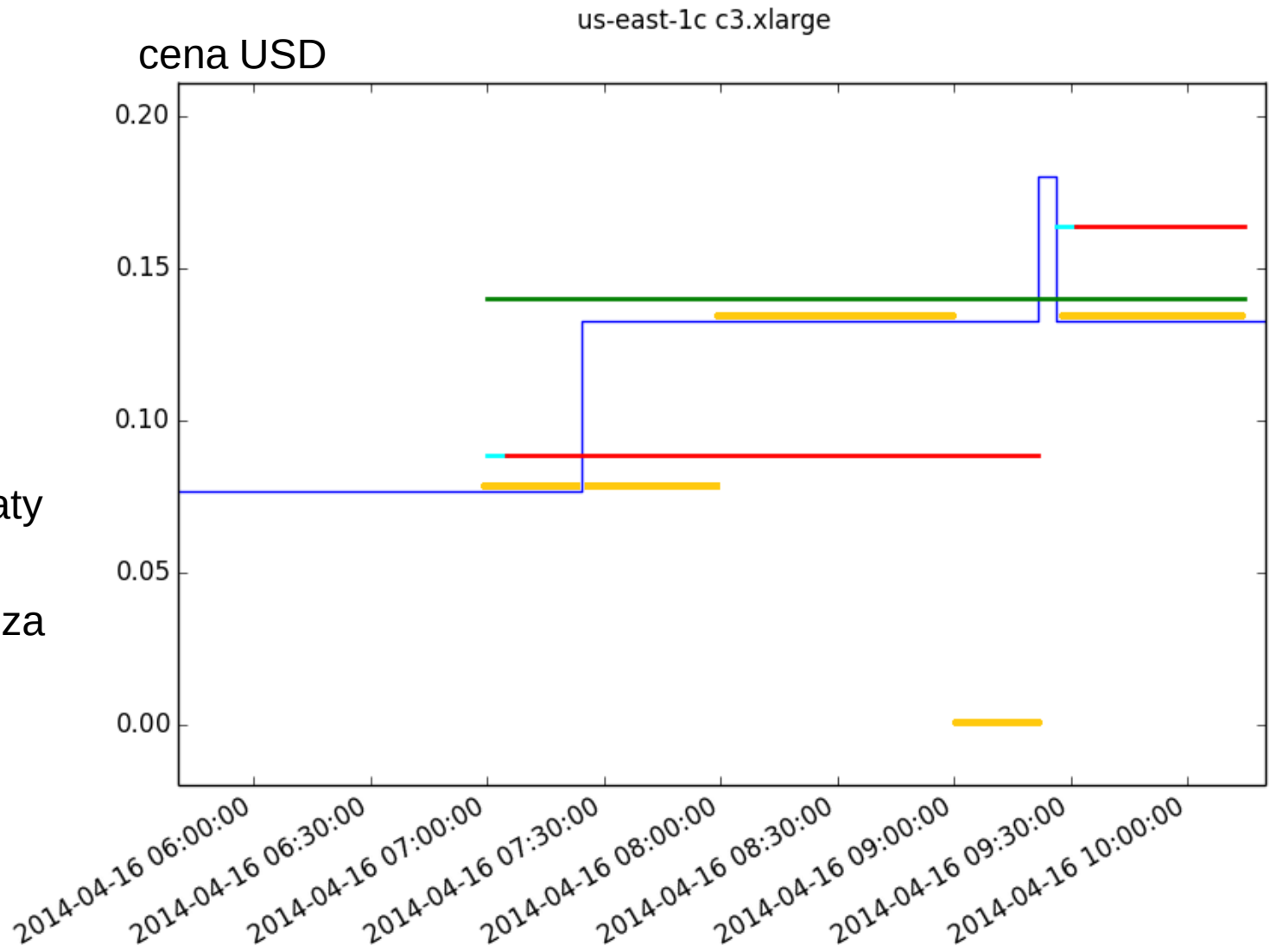


ceny spot: c3.8xlarge us-east-1a ostatni tydzień maja

Przykład - bid 0.14$

Legenda

- cena spot
- bid
- cena do zapłaty
- średnia cena za godzinę
- warmup time



Symulator kosztów spot...

```
def estimate_cost_d (self, bid_price, zone_machine, start_datetime, end_datetime=None,
warmup_time_s=0,request_time_s=None,request_sim_runs=None,single_sim_time_s=1,use_full_last_hour=T
rue,stop_on_terminate = False):
        """ Calculates computing costs for a given bid at given time zone and starting at specific
point of time.


            Keyword arguments:

            bid_price -- the bid

            zone_machine -- a tuple of (zone, machine)

            start_datetime -- the datetime when the bid was placed

            end_datetime -- maximum time (the instance will be terminated at this time even if
            requested simulations are not finished)

            warmup_time_s -- warmup time in seconds for a newly started instance (e.g. server
            boot-up time)

            request_time_s -- computation time in seconds that was request by a user

            request_sim_runs -- number of simulation runs requested by a user

            single_sim_time_s -- time to run a single simulation in seconds

            use_full_last_hour -- use fully the last hour to run additional simulations than
            requested by the user

            stop_on_terminate -- stop adding time as soon as the instance is terminated due to
            spot price increase

        """
```

Symulator kosztów

```
import datetime
import ec2_spot_pricing_simulator as ecs
sim = ecs.Ec2Simulator("ceny_spot_04072014.txt",\
      "2014-06-01","2014-06-15")
start = datetime.datetime.strptime(\
      "2014-06-03","%Y-%m-%d")
end = datetime.datetime.strptime(\
      "2014-06-05","%Y-%m-%d")
sim.estimate_cost_d(0.5,(\
      "us-east-1a","c3.large"),\
      start,end,single_sim_time_s=3600)
```

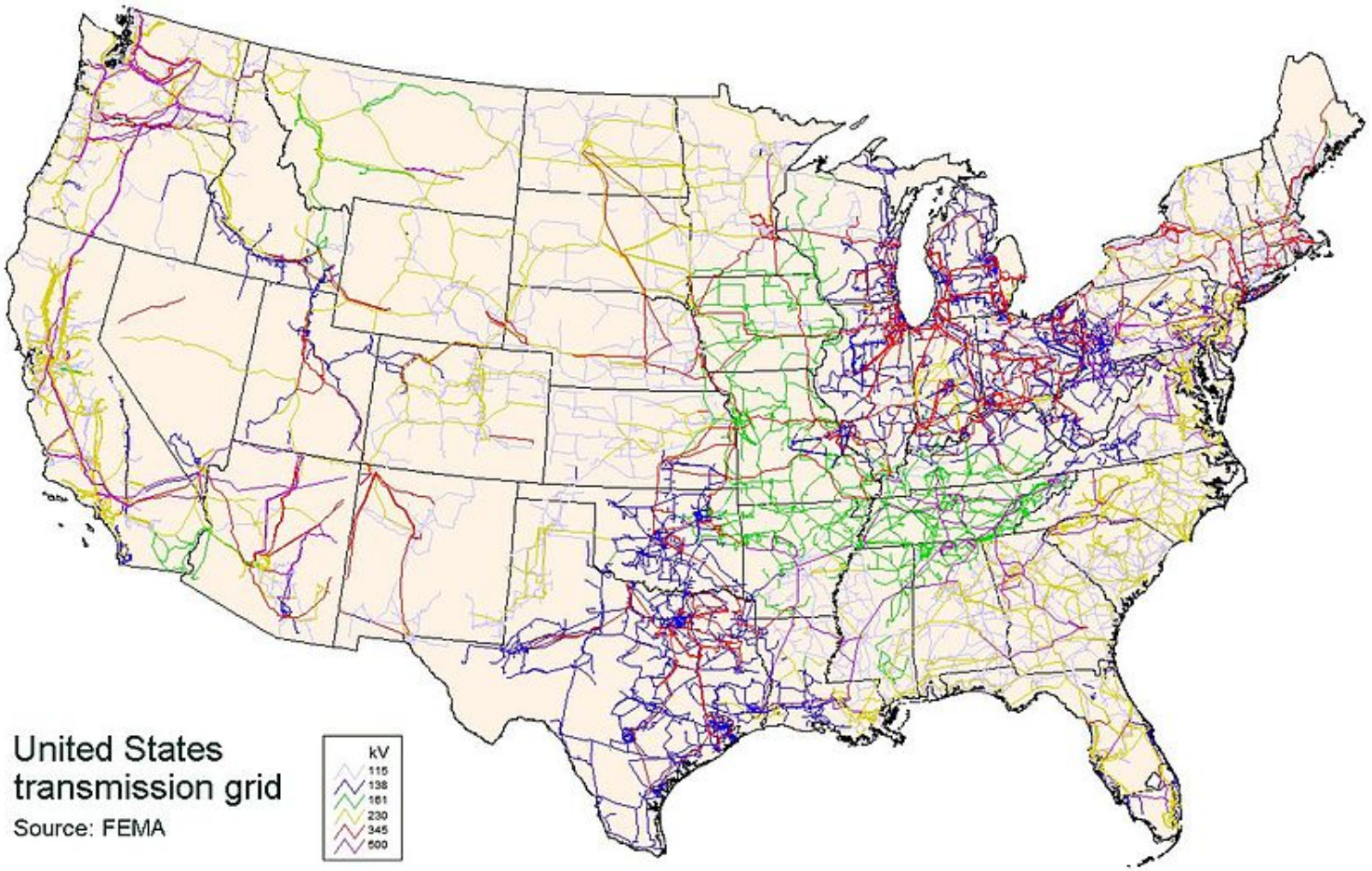
Optymalizacja kosztów

- Decyzje
  - ile serwerów on-demand?
  - ile serwerów spot?
  - jaki bid (jaka oferta na rynku spot?)

Analiza sieci w języku Python

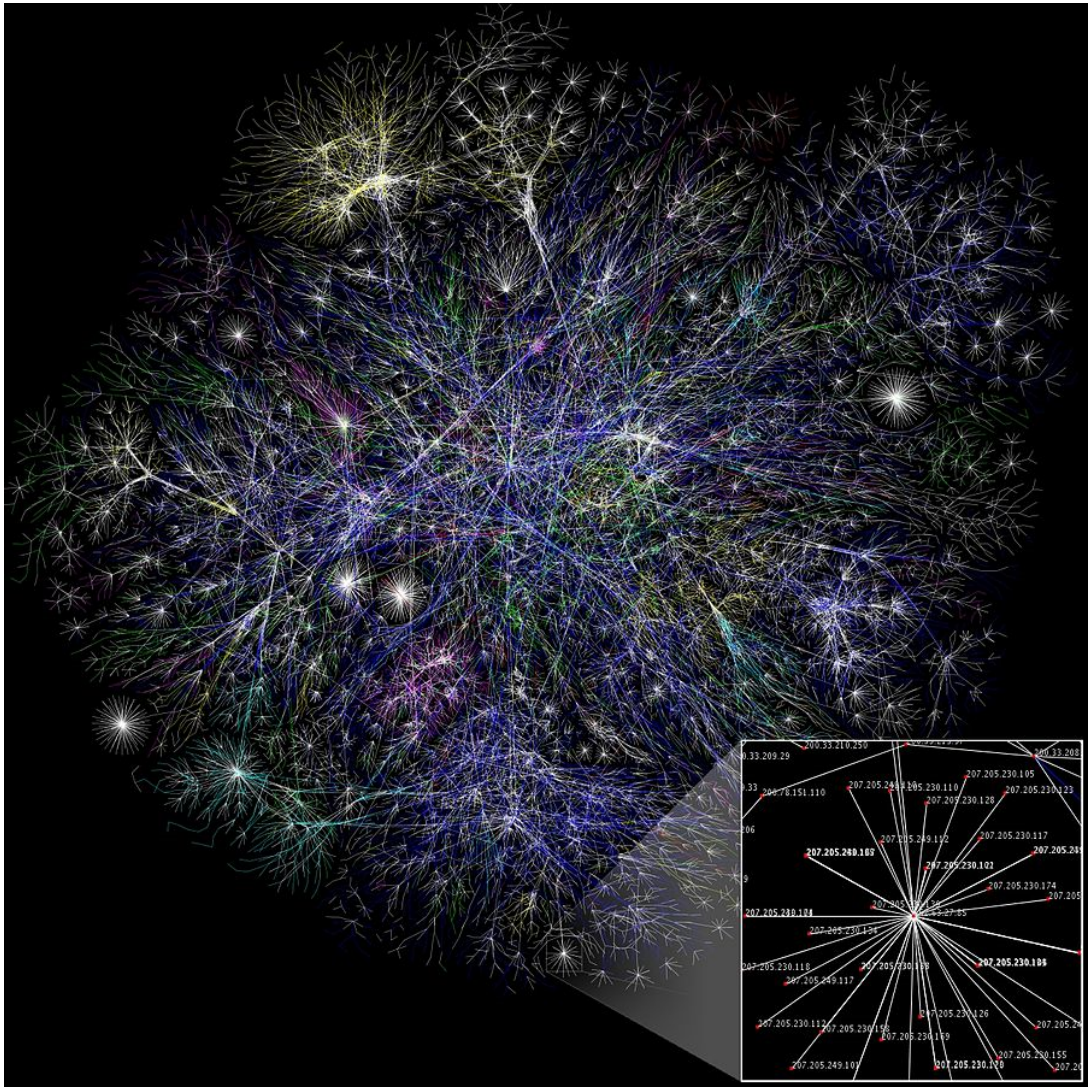
Sieci społeczne

Sieci elektryczne



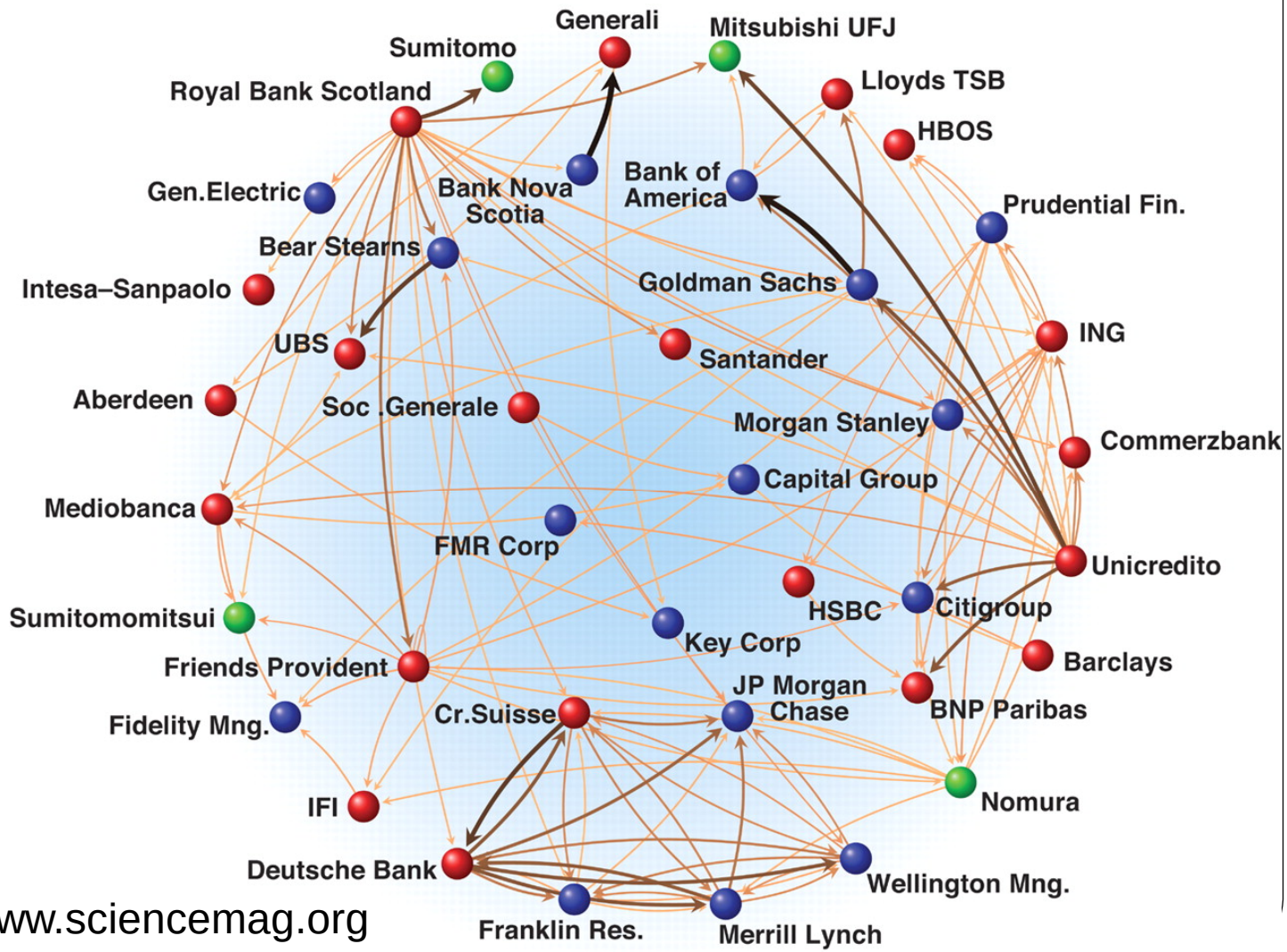
http://commons.wikimedia.org/wiki/File:UnitedStatesPowerGrid.jpg

Internet



http://en.wikipedia.org/wiki/File:Internet_map_1024.jpg

Sieci finansowe



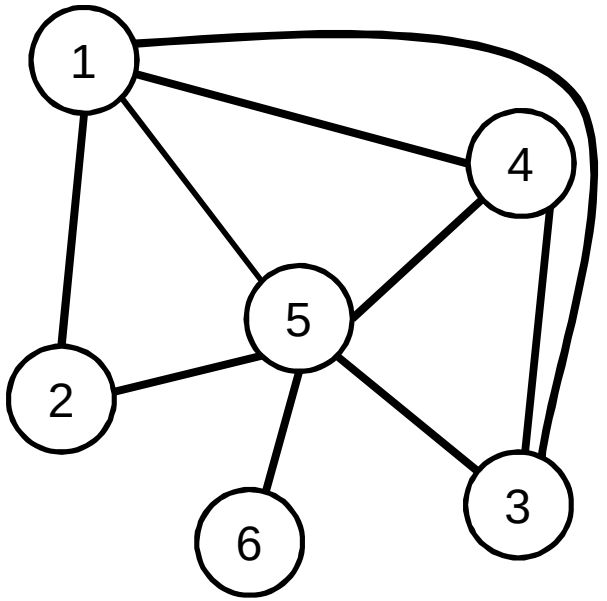
źródło: http://www.sciencemag.org

Obszary analizy sieciowej

- Dynamics **on** networks
  - Analiza złożonych systemów w których występują zależności sieciowe
  - Np.
    - Rynki ekonomiczne
- Dynamics **of** networks
  - Analiza procesu tworzenia się sieci
  - Np.
    - Powstawanie firm i instytucji finansowych

Sieć = Graf

- Wierzchołki (węzły), vertices(nodes)
  - atrybuty (cechy)
- Krawędzie (połączenia) edges (links)
  - kierunek
  - wagi



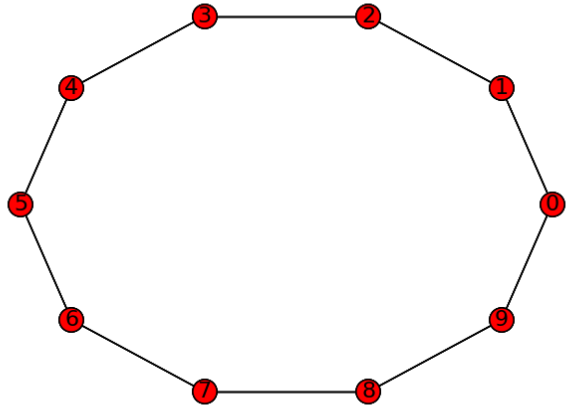
Python:

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

Graf

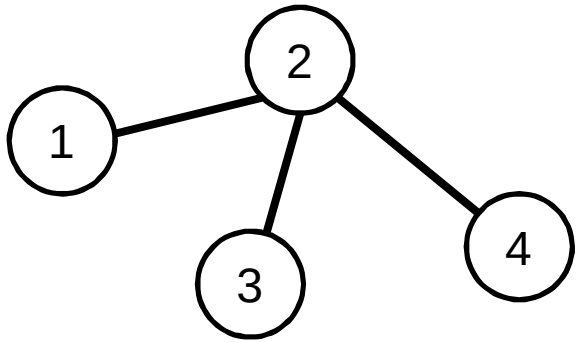
- *Graf (graph)* - zbiór wierzchołków, które mogą być połączone krawędziami
- *Graf skierowany (directed)* – graf z krawędziami posiadającymi kierunek (początek i koniec)
- *Droga (path)* – uporządkowany podzbiór sąsiadujących krawędzi
- *Graf spójny (connected)* – dla każdego wierzchołka istnieje droga do każdego innego wierzchołka

Tworzenie i wizualizacja sieci w języku Python




```
import networkx as nx
g = nx.Graph()
for n in range(10):
    g.add_node(n)
for n in range(10):
    g.add_edge(n,n+1 if n+1 < 10 else 0)
pos = nx.shell_layout(g)
nx.draw(g, pos)
nx.draw_networkx_labels(g,pos,\
    {n : str(n) for n in g.nodes_iter()},\
    font_size=16)
```

Sposób zapisu grafu



- Macierz sąsiedztwa (adjacency matrix)

```
nx.adjacency_matrix(g)
[[ 0.  1.  0.  0.]
 [ 1.  0.  1.  1.]
 [ 0.  1.  0.  0.]
 [ 0.  1.  0.  0.]]
```

- Listy sąsiedztwa (adjacency list)

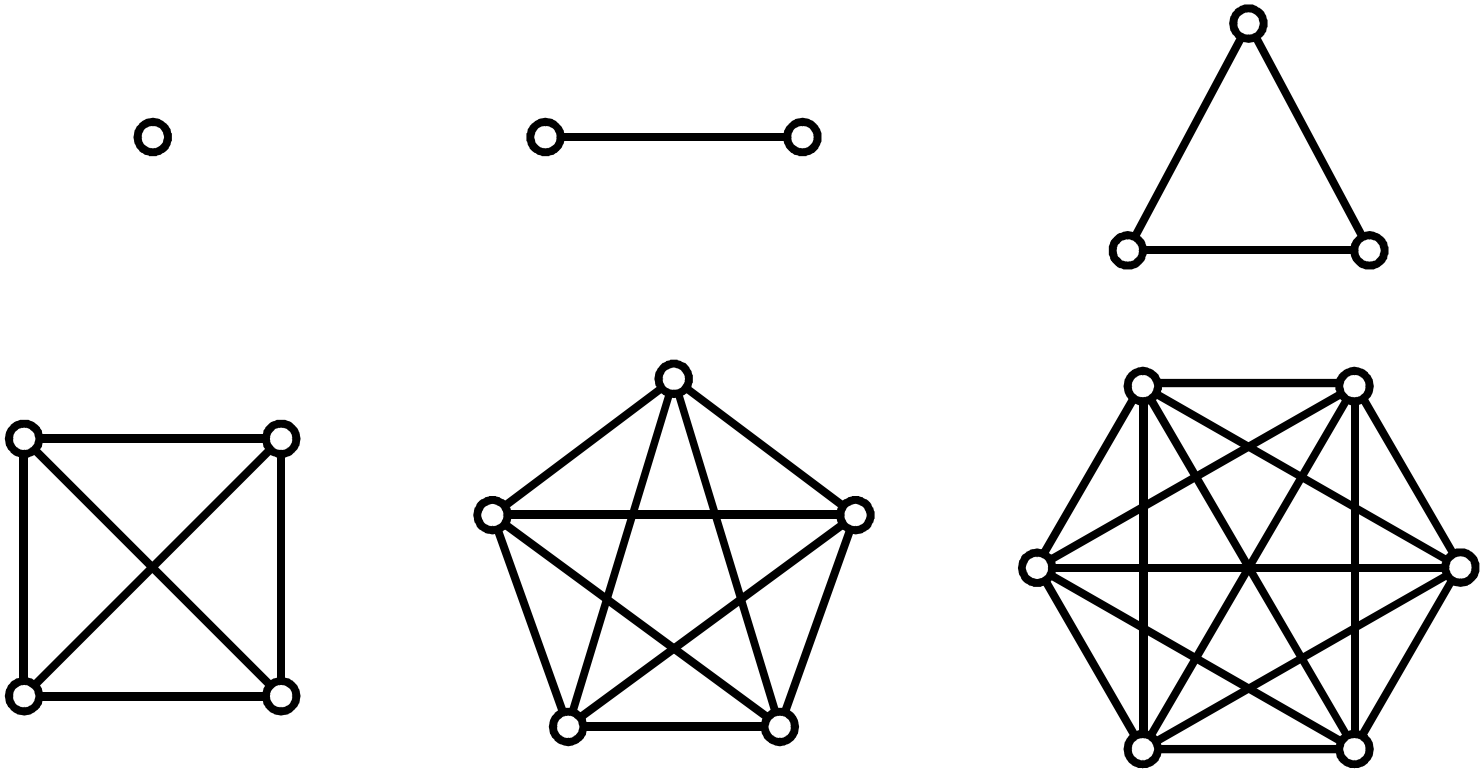
```
g.adjacency_list()
[[2], [1, 3, 4], [2],
[2]]
```

Wybrane typy grafów

- Graf pełny (complete) - wszystkie wierzchołki są połączone
- Graf regularny (regular) stopnia n – z każdego wierzchołka wychodzi n łuków
- Graf dwudzielny (bipartite graph, n-partite graph) – zbiór wierzchołków można podzielić na dwa podzbiory takie że ich wierzchołki nie są połączone
- Pełny graf dwudzielny
- Grafy planarne - można przedstawić na płaszczyźnie tak, że nie przecinają się łuki

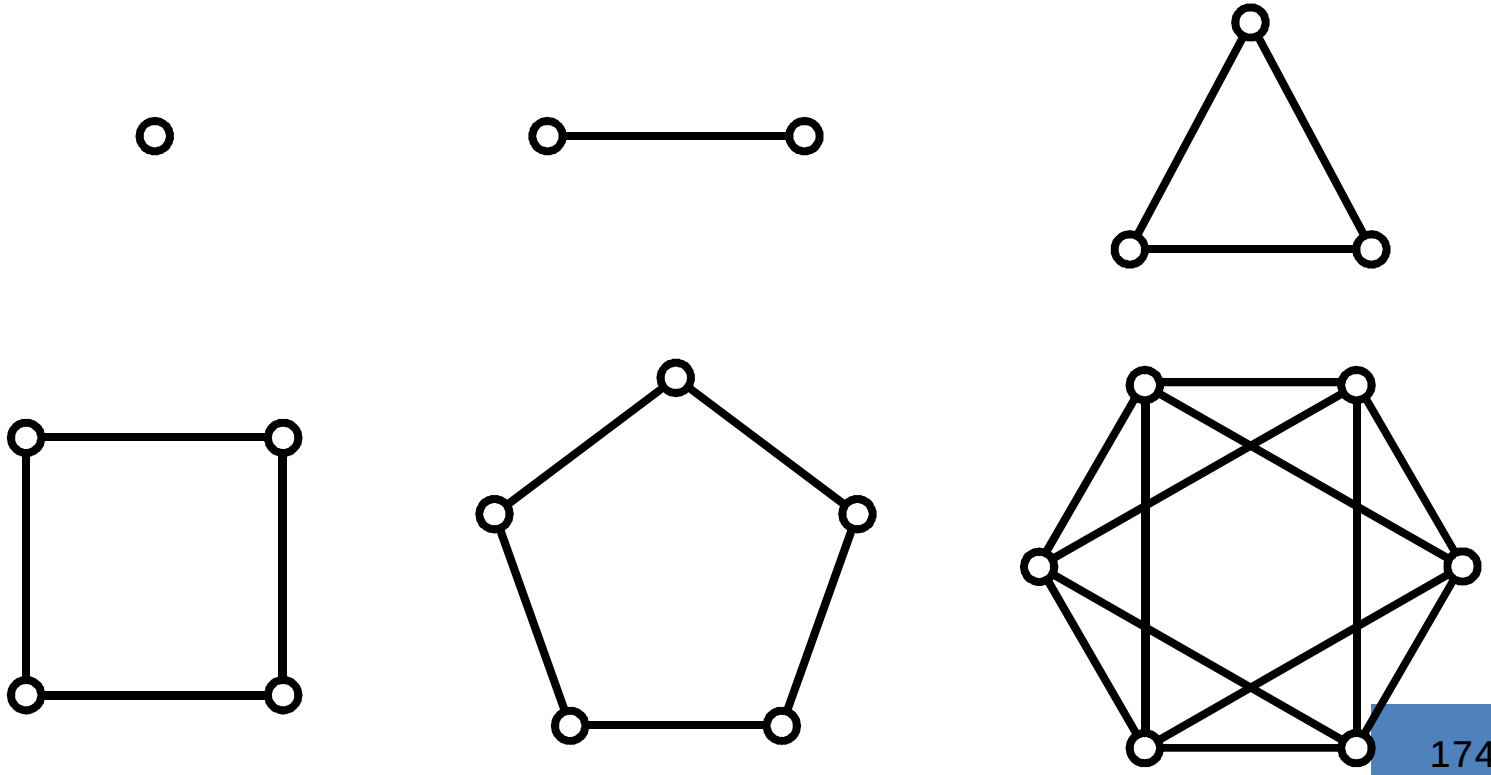
Graf pełny (ang. complete graph)

- Wszystkie wierzchołki są ze sobą połączone…



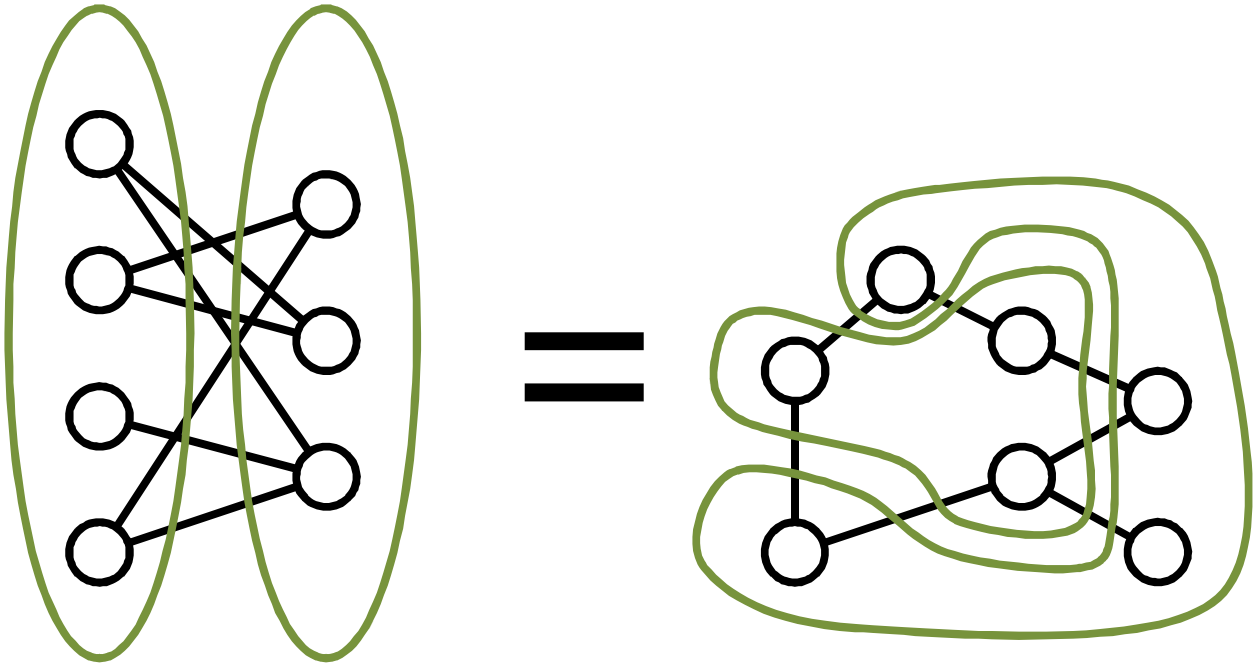
Graf regularny (ang. Regular graph)

- Wszystkie wierzchołki są tego samego k-tego stopnia



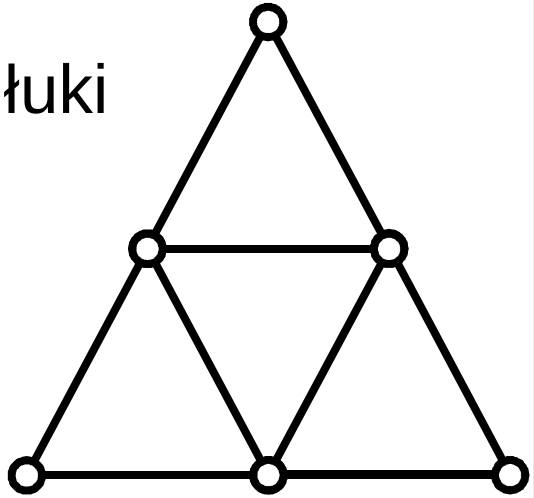
Graf dwudzielny (bipartite graph, n-partite graph)

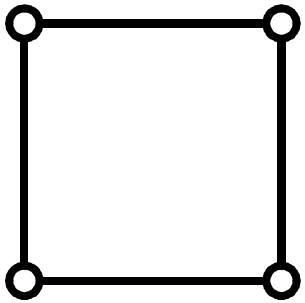
- Graf dwudzielny (bipartite graph, n-partite graph) – zbiór wierzchołków można podzielić na dwa podzbiory takie że ich wierzchołki nie są połączone

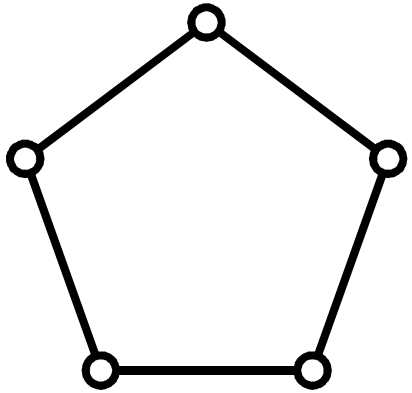


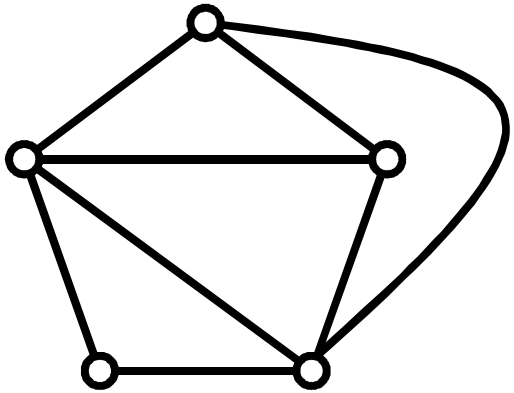
Graf planarny

- można przedstawić na płaszczyźnie w taki sposób, że nie przecinają się łuki

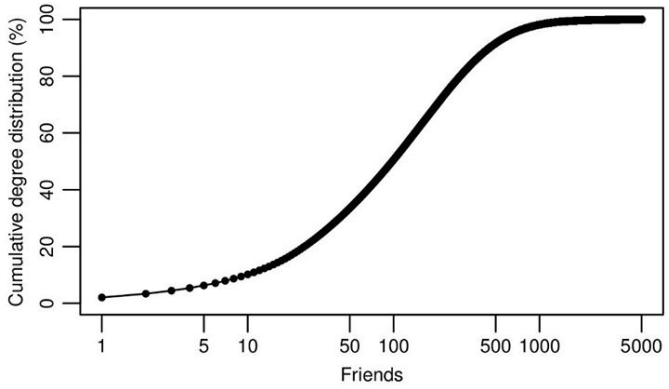


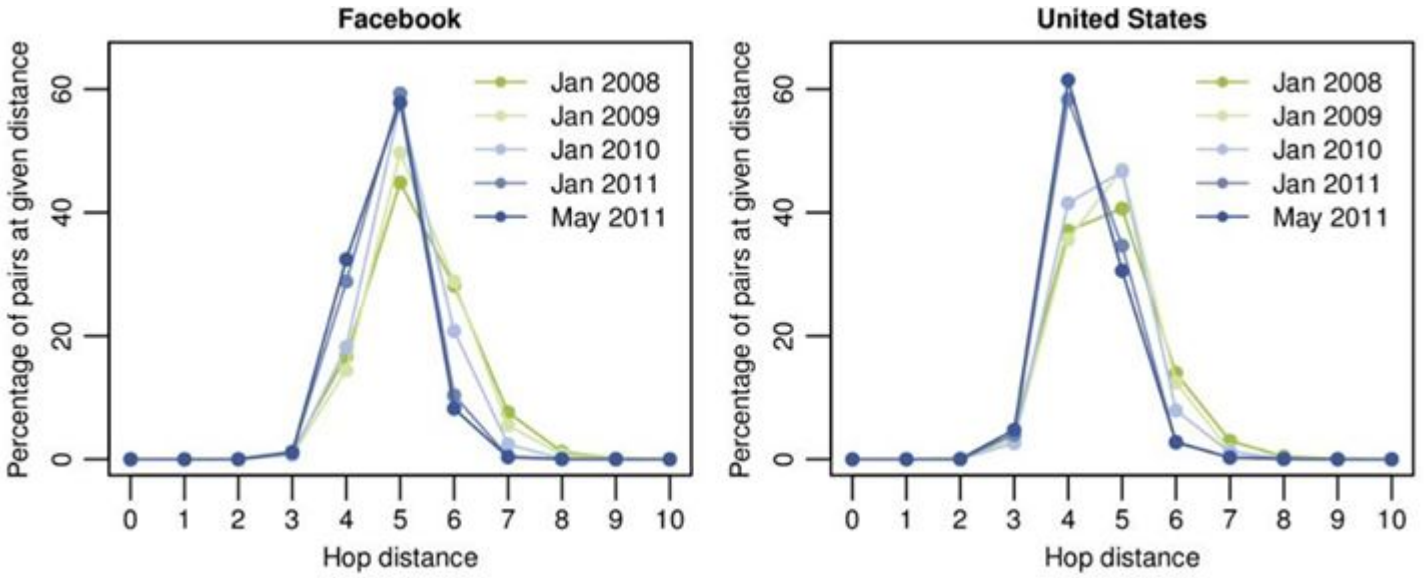






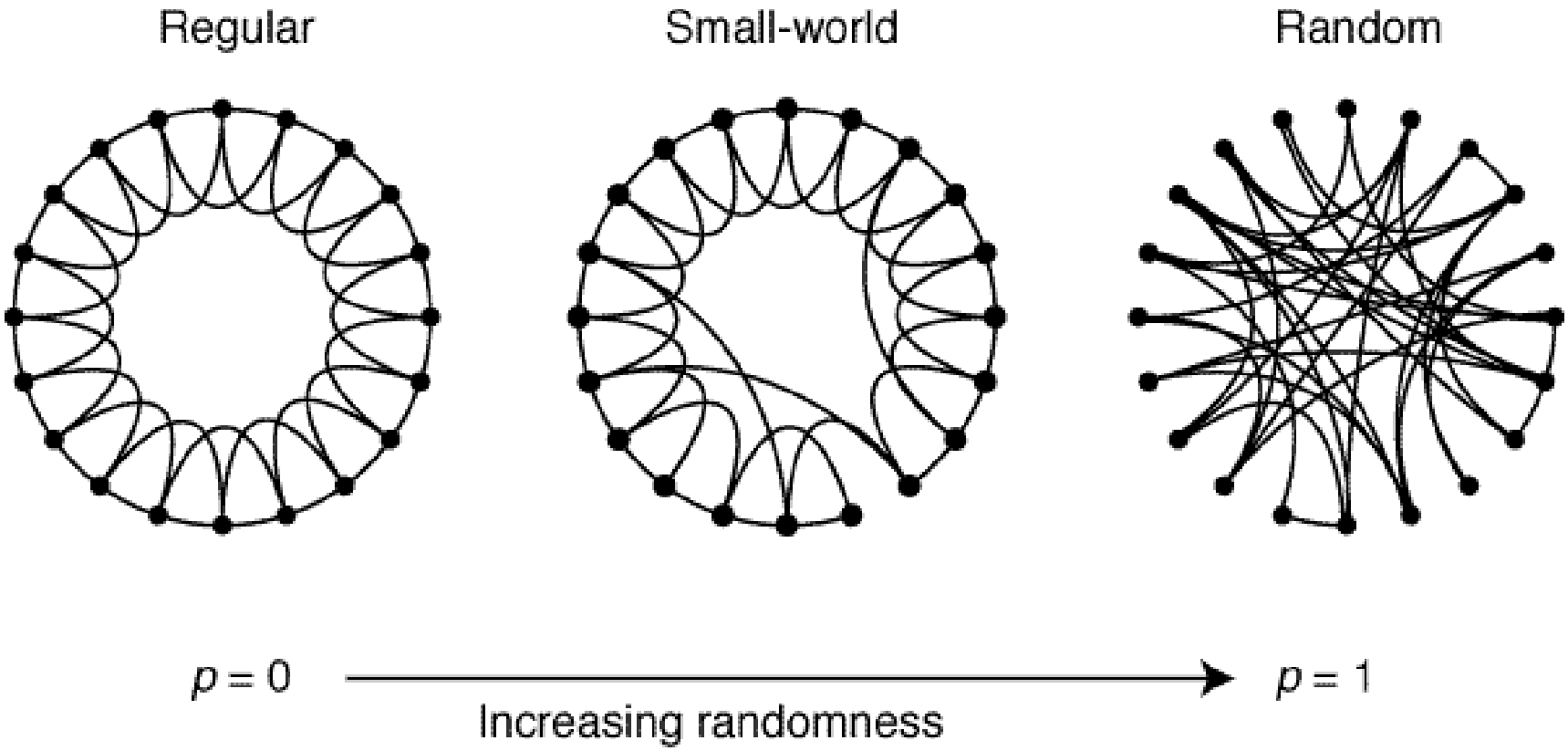
Sieci małego świata ...





Źródło: https://web.facebook.com/note.php?note_id=10150388519243859&_rdr

Typy sieci ...

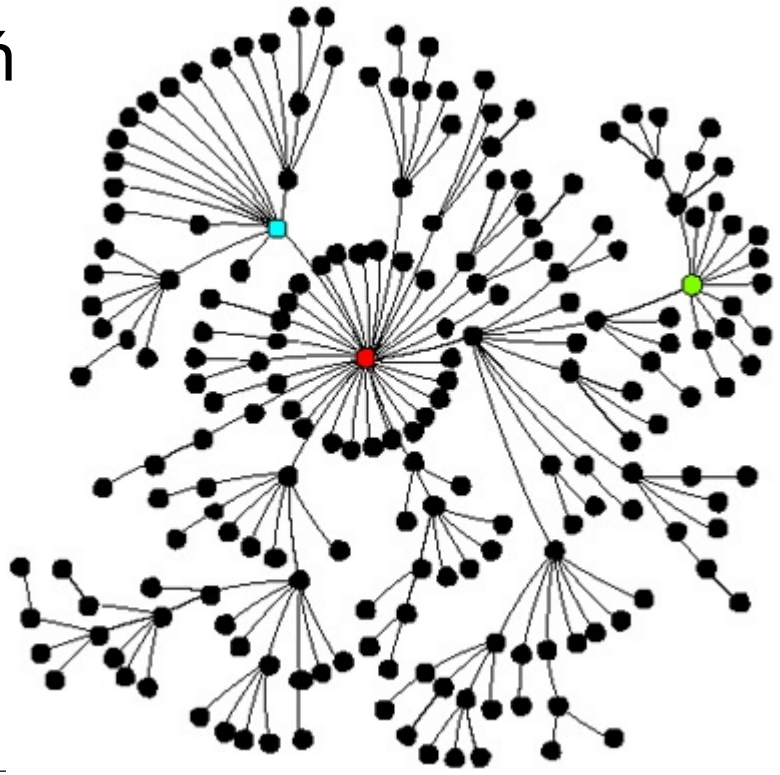


źródło: Duncan J. Watts and Steven H. Strogatz (1998) Collective dynamics of 'small-world' networks, Nature 393, 440-442

Sieci bezskalowe (scale-free networks)

- Wiązanie preferencyjne (ang. preferential attachment)
- Rozkład węzłów - prawdopodobieństwo, że węzeł posiada k połączeń (ang. power-law distribution)

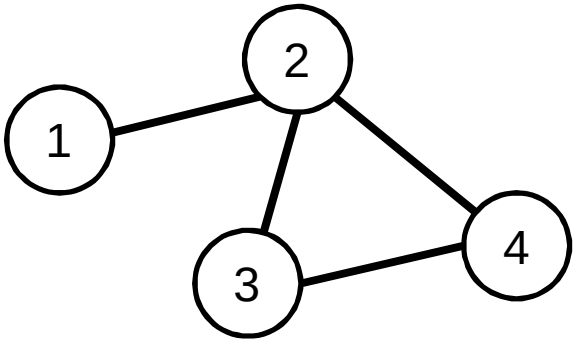
$$P(k) \sim k^{-\gamma}$$



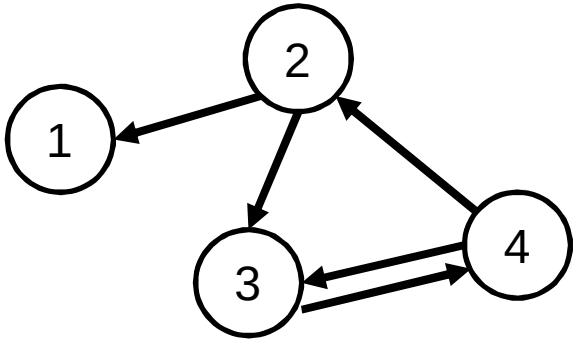
Generacja grafów w modelach symulacyjnych

- Sieci losowe
  - połączenia pomiędzy wierzchołkami są generowane w sposób losowy
  - `nx.erdos_renyi_graph(n, p[, seed, directed])`
  - `nx.fast_gnp_random_graph(n, p[, seed, directed])`
- Sieci regularne
  - z każdego wierzchołka wychodzi dokładnie n łuków
  - `nx.random_regular_graph(d, n[, seed])`
- Sieci małego świata (small world networks)
  - Niewielka odległość pomiędzy dwoma sąsiadującymi węzłami
  - `nx.watts_strogatz_graph(n, k, p[, seed])`
- Sieć bezskalowa (scale free network)
  - Generacja algorytmem preferential attachment `nx.barabasi_albert_graph(n, m, seed=None)`

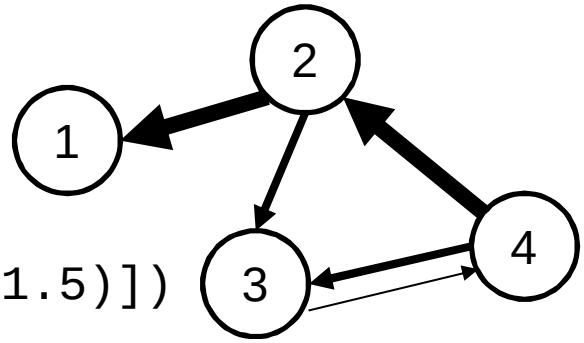
Grafy skierowane, nieskierowane i ważone...




```
g = nx.Graph()
```




```
g = nx.DiGraph()
```




```
g.add_weighted_edges_from([(2,1,1.5)])
```

- Długość ścieżki w grafie

```
nx.shortest_path_length(g,a,b)
```

Przykład - marketing szeptany

- Firma farmaceutyczna FarHaz ma zamiar wprowadzić na rynek nowy suplement diety o nazwie FarSup7 zapewniający piękny odcień skóry. Nie ma żadnych badań naukowych potwierdzających wpływ specyfiku na zdrowie ani jego skuteczność dlatego firma może polegać wyłącznie na marketingu szeptanym. Na ten sam rynek właśnie ma zamiar wejść groźny konkurent - firma BioSio oferujący środek PlaceBio posiadający dokładnie ten sam skład...

Przykład - marketing szeptany c.d.

- Osoby kupujące suplementy diety mogą mieć jedną z trzech opinii na temat produktów FarSupt7 i PlaceBio
  0. Nie slyszałem
  1. żaden z produktów nie jest skuteczny
  2. lepszy jest produkt FarSup7
  3. lepszy jest produkt PlaceBio
- Potencjalni klienci cały czas rozmawiają ze sobą na temat produktów na portalu internetowym "MojaCera" i w ten sposób na wzajem wpływają na swoją opinię o ich skuteczności.
- Firma FarHaz ustaliła, że BioSio ma zamiar umieścić na forum 3 swoich pracowników, którzy będą udawać zadowolonych użytkowników produktu PlaceBio.
- Ile osób powinna wprowadzić firma FarHaz jeżeli wiadomo, że forum liczy 100 użytkowników, a każdy użytkownik ma średnio 3 kontaktów w portalu.

Marketing szeptany - inicjalizacja sieci

```
import matplotlib.pyplot as plt
import networkx as nx
import numpy as np
size, p, f1, f2 =100, 0.1, 10, 10
np.random.seed(0)
g = nx.erdos_renyi_graph(size, p)
pos = nx.spring_layout(g)
for n in g.nodes_iter():
    g.node[n]['state'] = 0
f1_count = 0
for n in np.random.choice(list(g.nodes_iter()),f1+f2,False):
    if f1_count < f1:
        g.node[n]['state'] = 1
        f1_count  += 1
    else:
        g.node[n]['state'] = 2
plt.cla()

nx.draw(g, pos,node_color=[g.node[n]['state'] for n in g.nodes_iter()])
nx.draw_networkx_labels(g,pos,\
    {n : str(n) for n in g.nodes_iter()}, font_size=16)
```

Dynamika modelu

```
nodes_to_say = [n for n in g.nodes_iter() \
    if g.node[n]['state'] > 0]
speaker = np.random.choice()
if g.neighbors(speaker) != []:
    listener = np.random.choice(g.neighbors(speaker))
    g.node[listener]['state'] = \
        g.node[speaker]['state']
```

Opakowanie wizualizacji...

http://pycx.sourceforge.net

```
import matplotlib; matplotlib.use('qt4agg')
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
size,p,f1,f2=100,0.1,10,10
np.random.seed(0)
g = None
pos = None
def init():
    global pos, g
    #tutaj kod inicjulizaujacy
def draw():
    plt.cla()
    #tutaj wizualizacja stanu sieci
def step():
    #tutaj dynamika sieci
def firma1(newValue=f1):
    global f1;f1 = newValue;return f1
def firma2(newValue=f2):
    global f2;f2 = newValue;return f2
import pycxsimulator
pycxsimulator.GUI(title='Informacja w sieci',interval=0, \
    parameterSetters = [firma1,firma2]).start(func=[init,draw,step])
```

Modele symulacyjne w języku Python w chmurze

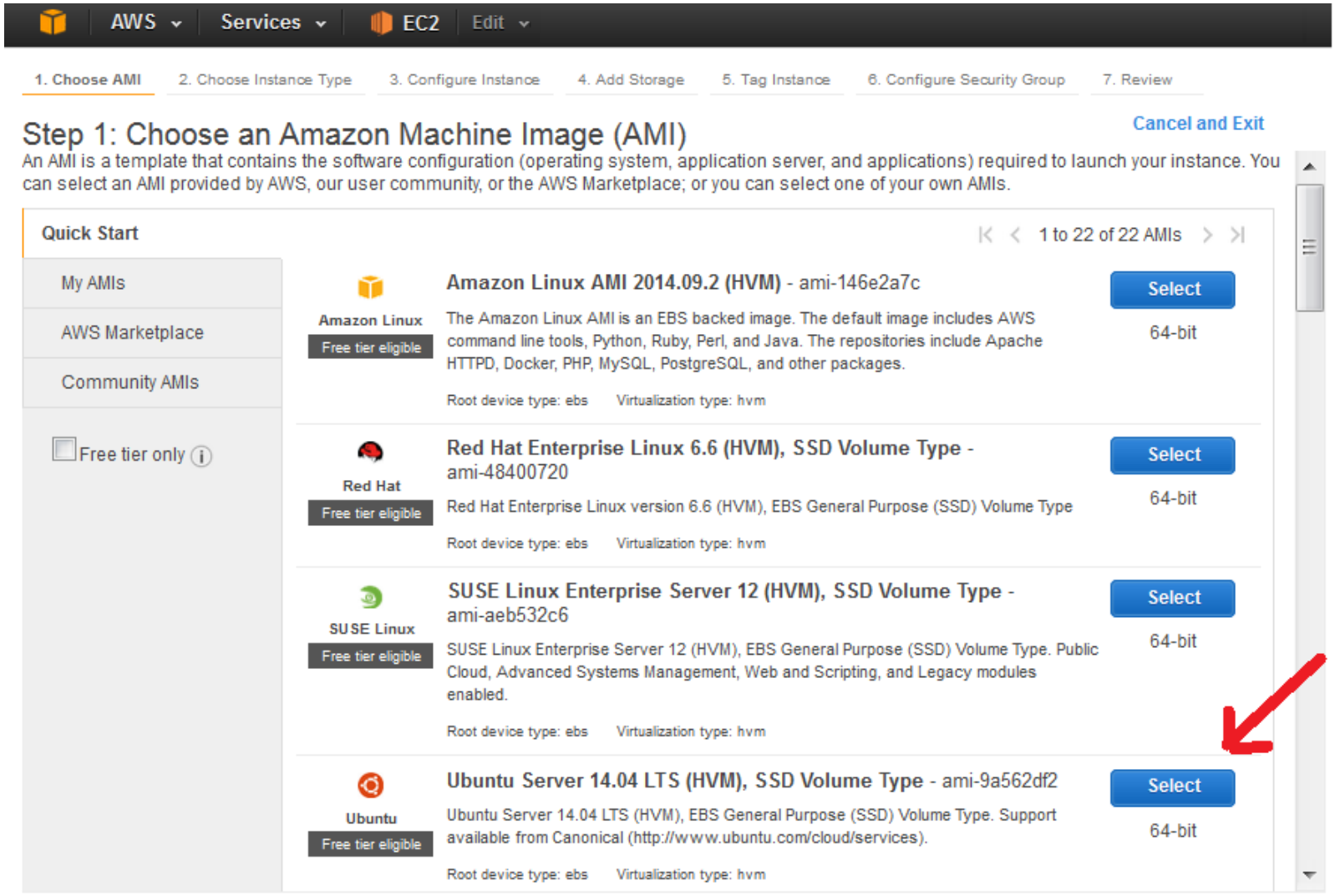
Potrzebne oprogramowanie:

- putty
- WinScp lub FileZilla
- przeglądarka internetowa

Scenariusze wdrożeniowe dla obliczeń symulacyjnych w chmurze

- Wynajęcie jednej kilku "dużych komputerów" - zależnie od typu problemu
  - "Obliczeniożerny" - C4.8xlarge
    - 36 rdzeni Intel Xeon
    - 60GB RAM
  - "Pamięciożerny" r3.8xlarge
    - 32 rdzeni Xeon (nieco słabszych do C4)
    - 244 GB RAM
- Stworzenie klastra obliczeniowego w chmurze
  - Połączenie i wspólne zarządzanie wieloma maszynami c4.* lub r3.*
  - Darmowe oprogramowanie StarCluster

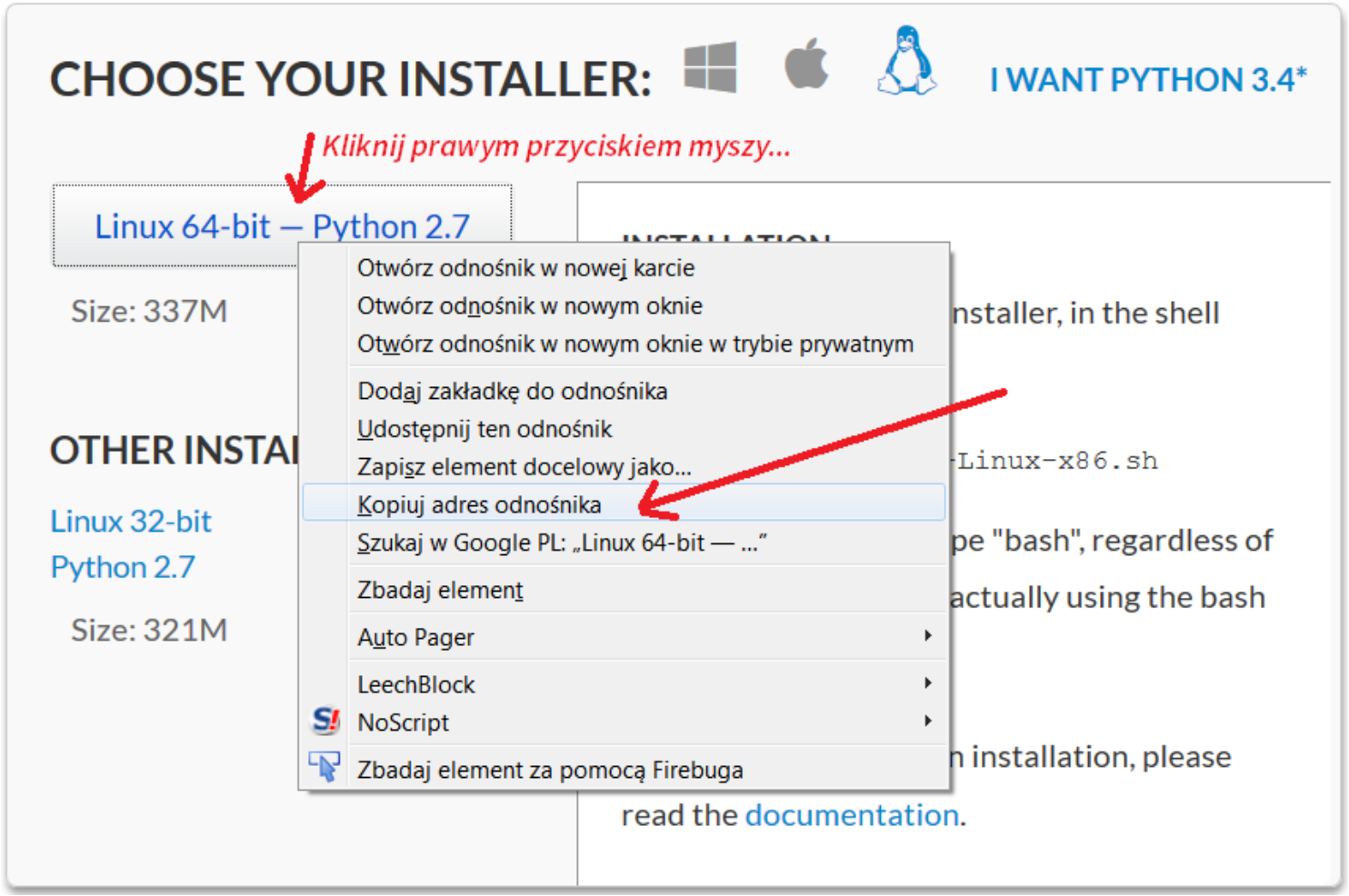
Tworzenie instancji w chmurze Amazon AWS



Prowadzenie obliczeń i przetwarzania danych na instancji EC2

- Instancja ma wiele rdzeni (do 36 w przypadku c4.8xlarge) stąd należy utworzyć wiele równoległych procesów obliczeniowych
  - Skrypt powłoki (bash)
  - Moduł multiprocessing w Pythonie
  - Systemy zarządzania kolejką zdań
    - Apache Spark
    - Starcluster
    - AWS SQS + Autoscaling

Konfiguracja środowiska Anaconda-Python
na przykładzie systemu operacyjnego Ubuntu



Instalacja

Narzędzia dla Linuxa...

```
sudo apt-get update
sudo apt-get install -y awscli htop mc
```

Python-anaconda

`wget` https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda3-2.4.1-Linux-x86_64.sh

```
bash Anaconda3-2.4.1-Linux-x86_64.sh -b
```

wciśnij kilka razy ENTER czytając instrukcje na ekranie - przy pytaniu o licencję wpisz "yes" i ENTER, w pozostałych przypadkach zaakceptuj opcje wskazywane przez instalator

Zrównoleglanie obliczeń w bash

- Utworzenie skryptu w języku powłoki bash

uruchom.sh

```
#!/bin/bash

for i in `seq 0 3`;
do
    nohup anaconda3/bin/python symuluj_ubezp.py $i >
                                log$i.csv 2>error$i.csv &
done
```

- Dopasowanie kodu

symuluj.py

```
if __name__ == '__main__':
    jobid = int(sys.argv[1])
    rd.seed()
    runjob(jobid)
```

```
bash uruchom.sh
```

Zrównoleglanie bezpośrednio z poziomu Pythona

- Przetwarzanie wieloprocesowe w kodzie w języku Python

symuluj.py

```
from multiprocessing import Pool

if __name__ == '__main__':
    PROCESSES = 3
    pool = Pool(processes=PROCESSES)
    print("Created a pool of",PROCESSES," processes")
    pool.map(run_symulacja, range(20))
    pool.terminate()
```

Prowadzenie obliczeń symulacyjnych

- Krok pierwszy to zawsze wyodrębnienie kawałka kodu repprezentującego porcję symulacji

symuluj_zapasy.py

```
def runjob(jobid):
    lp = 0
    with open('simresults.'+str(jobid)+'.csv', 'w') as csvfile:
        writer = csv.writer(csvfile, delimiter='\t', lineterminator="\n")
        for s in range(10+jobid*50,10+(jobid+1)*50,5):
            print(str(jobid),str(s))
            sys.stdout.flush()
            for S in range(10,210,5):
                for repeat in range(1000):
                    lp += 1
                    profit = simulateOneRun(300,s,S)
                    writer.writerow([lp,repeat,s,S,profit])
                csvfile.flush()
```

Równoległe uruchamianie obliczeń symulacyjnych z poziomu języka bash

- Krok1: dopasowanie kodu w Pythonie
  - pobieranie argumentu z wiersza poleceń
  - wykorzystanie argumentu do parametryzacji symulacji

symuluj.py

```
if __name__ == '__main__':
    jobid = int(sys.argv[1])
    runjob(jobid)
```

Równoległe uruchamianie obliczeń symulacyjnych z poziomu języka bash

- Krok2: Skorzystaj z gotowego skryptu

start_symulacji.sh

```
#!/bin/bash

nproc=`nproc`
#zwroc uwage na typ cudzyslowu

logfile="wyniki_$(date '+%Y-%m-%d_%H%M%S').log.txt"

start=$1
end=$((start+5000-1))
# kod ponizej powinien byc w jednej linii
nohup seq $start $end | xargs --max-args=1 --max-procs=$nproc
        python symulacje.py &>> $logfile &
```

- Krok3: Uruchom symulacje

```
bash start_symulacji.sh 0
```

Zrównoleglanie bezpośrednio z poziomu Pythona

- Przetwarzanie wieloprocesowe w kodzie w języku Python

symuluj_zapasy.py

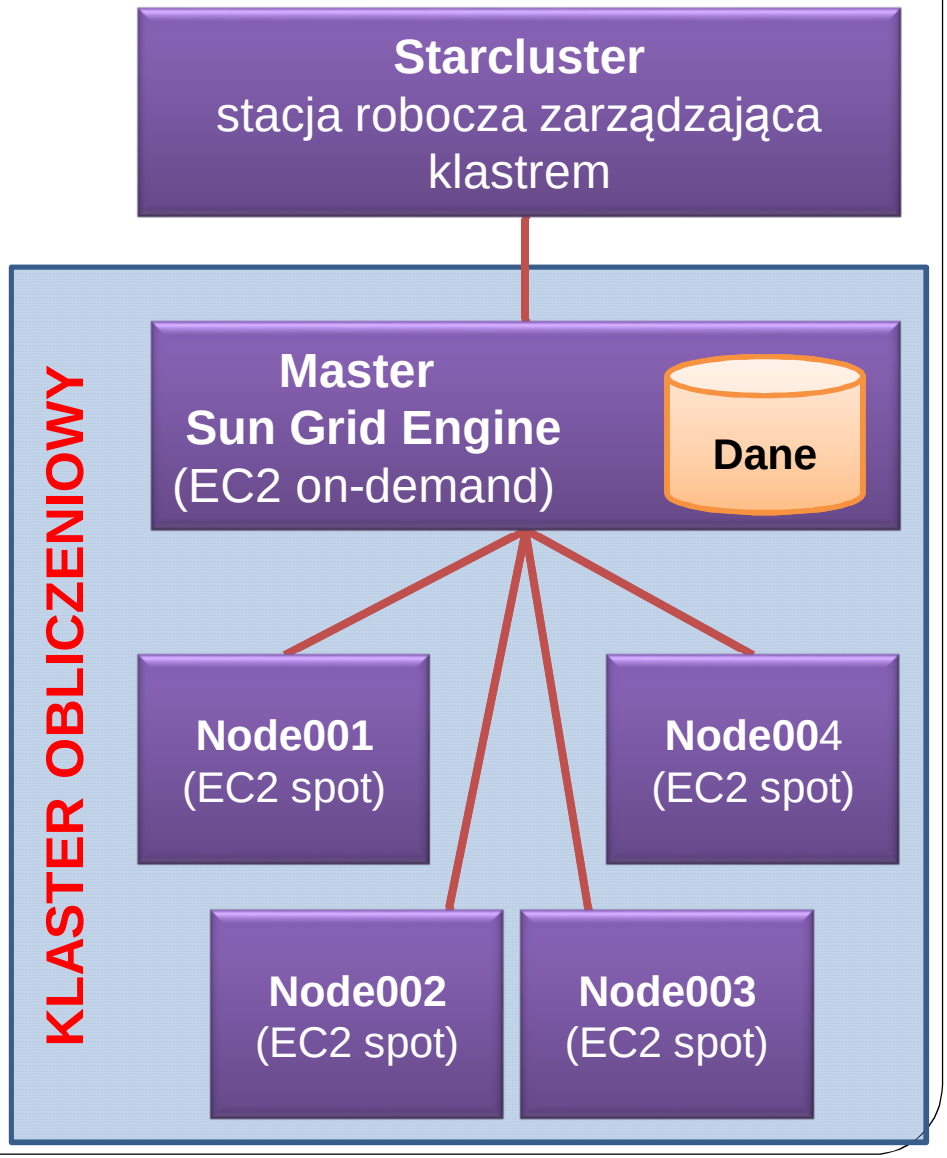
```
from multiprocessing import Pool

if __name__ == '__main__':
    PROCESSES = 4
    pool = Pool(processes=PROCESSES)
    print("Created a pool of",PROCESSES," processes")
    pool.map(runjob, range(40))
    pool.terminate()
```

Ultra technicznie – zbuduj sobie superkomputer w chmurze

Superkomputery - scenariusz wdrożeniowy aplikacji StarCluster na chmurze Amazon EC2

- Klaster umożliwia zarządzanie i monitorowanie zadaniami obliczeniowymi



Dla dociekliwych – instalacja

1. Uruchom instancję Workstation (np. t2.micro)
2. Wyeksportuj credentials.csv i wgraj do katalogu domowego np. /home/ubuntu
3. Pobierz kod instalatora (np. do /home/ubuntu)

   wget https://szufel.pl/sc_setup.sh

   bash sc_setup.sh
4. Gotowe!

```
starcluster -b 0.50 start klaster
```

Bardziej szczegółowy przewodnik znajdziesz poniżej https://szufel.pl/euro2016.html

Jak korzystać z klastra?

1. Uruchomienie klastra obliczeniowego
   ```
   starcluster -b 0.50 start klaster
   ```
2. Umieszczenie danych na klastrze
   ```
   starcluster put klaster /katalog /remote/path/
   ```
3. Połączenie się z węzłem master
   ```
   starcluster sshmaster klaster
   ```
4. Uruchomienie obliczeń
   ```
   qsub -V -t 1-4000 -cwd -N "symulacje" licz.sh
   ```
5. Pobranie pliku (np. katalogu z wynikami symulacji)
   ```
   starcluster get /remote/path/ /home/ubuntu/
   ```
6. Wyłączenie klastra
   ```
   starcluster terminate klaster
   ```

Open Grid Scheduler zarządzanie zadaniami

[Polecenia są wydawane na węźle master]

- Wysłanie zadań do węzłów klastra

  ```
  qsub -cwd -o
  qsub -cwd -o log.txt -e log_e.txt job.sh
  ```

  opcja `-t` pozwala na wysyłanie grup prac (array jobs)

- Usuwanie zadań

  ```
  qdel nazwa_zadania
  ```

Open Grid Scheduler monitorowanie statusu

[Polecenia są wydawane na węźle master]

- `qstat` - Informacje o kolejkacj prac
  - `-f` Show full listing of all queues
  - `-j` Shows detailed information on pending/running job
- `qhost` - informacje o węzłach klastra
  - `-j` Shows detailed information on pending/running job
  - `-q` Shows detailed information on queues at each host

Typowy scenariusz obliczeń symulacyjnych/przetwarzania danych na klastrze obliczeniowym

1. Uruchomienie klastra obliczeniowego
   - `starcluster start klaster1`
2. Umieszczenie danych na klastrze
   - `starcluster put /katalog /remote/path/`
3. Połączenie się z węzłem master
   - `starcluster sshmaster klaster1`
4. Uruchomienie obliczeń
   - *[przykładowy skrypt na kolejnym slajdzie]*
5. Koniec obliczeń i rozłączenie się z węzłem master
6. Pobranie pliku (np. katalogu z wynikami symulacji)
   - `starcluster get /remote/path/ /home/ubuntu/`
7. Wyłączenie klastra obliczeniowego
   - `starcluster terminate mycluster`

Skrypty - obliczenia w Pythonie na klastrze

uruchom_obliczenia.sh

```
#!/bin/bash

mkdir ./logs
qsub -V -t 1-4000 -cwd -N "symulacje" licz.sh
```

licz.sh

```
#!/bin/bash

#dostowowanie numerowania tak aby zaczelo sie od zera
i=$(($SGE_TASK_ID-1))
python oblicz.py $i
```

Podsumowanie

Zasady zaliczeń

- Projekt grupowy (2-4 osób)
- Stworzyć model symulacyjny w języku Python rozwiązujący problem biznesowy/analityczny
  - dopuszczalne jest rozszerzenie modeli z zajęć
- Forma przekazania projektu
  - Kod źródłowy modelu
  - Raport z wynikami symulacji

  5-7 stron

  Termin: 19 grudnia 2016

Przykładowy układ raportu

1. Strona tytułowa

2. Podsumowanie: (tzw. *executive summary*) Wskazanie głównych wyników (również liczbowych) i wniosków z raportu – propozycji rozwiązania problemu wraz z krótkim uzasadnieniem. W podsumowaniu powinno używać się łatwego słownictwa nie zawierającego żargonu technicznego.

3. Opis organizacji: Podsumowanie rodzaju działalności organizacji, rodzaj produktów/usług oferowanych klientom, wielkość organizacji, rodzaj rynku (monopol, doskonała konkurencja itp.), główni konkurenci.

Układ raportu

4. Opis problemu, który ma być analizowany metodą symulacyjną :

- jaka decyzja jest rozważana?
- jaki jest zbiór decyzji dopuszczalnych (ograniczenie budżetowe, dostępne zasoby, dostępne opcje)?
- jak są oceniane skutki decyzji (zysk, ryzyko)?
- w jaki sposób zostanie dokonany wybór optymalnej decyzji
- jakie przyjęto założenia/uproszczenia i jakie są ograniczenia przyjętej analizy?

5. Wyniki analizy : Liczby wraz z ich interpretacją. Czy rozwiązanie jest akceptowalne? Czy może być wdrożone w praktyce? Czy wynik jest zgodny z intuicją? .

Układ raportu

6. Analiza wrażliwości : zmiana parametrów a rozwiązanie optymalne (RO)

- czy zmiana parametru (np. wzrost ceny produktu o 5% / wzrost wynagrodzenia godzinowego) spowoduje zmianę R.O., czy ciągle ono pozostanie takie samo?
- o ile musi zmienić się parametr (np. cena produktu / wynagrodzenie pracownika) żeby R.O. uległo zmianie?
- Opcjonalnie meta-model tłumaczący zachowanie modelu symulacyjnego

7. Wnioski i zalecenia : Jak wyniki analizy symulacyjnej mogą usprawnić działanie organizacji.

Literatura

- Python
  - http://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial
  - https://docs.python.org/2/tutorial/
- Teoria symulacji
  - *Simulation Modeling and Analysis*, Fourth Edition A. M. Law, McGraw-Hill, 2007
  - Simulation for the Social Scientist, Gilbert i Troitzsch, Springer, 2005