Mechanika teoretyczna

Python w obliczeniach naukowych

Janusz Szwabiński

Instytut Fizyki Teoretycznej UWr

11 października 2005

Python

- interpretowany, wysokopoziomowy język programowania
- darmowy
- obiektowy
- dynamiczna semantyka i elegancka składnia
- bogata biblioteka standardowa
- idealny do tworzenia skryptów i tzw. błyskawicznego rozwijania aplikacji

Zasoby w sieci

- Strona główna Pythona
  `http://www.python.org`
- Strona polska Pythona
  `http://www.python.org.pl`
- Narzędzia naukowe dla Pythona
  `http://www.scipy.org/`
- VPython - wizualizacje 3D
  `http://vpython.org/`

- G. van Rossum, *"Python Tutorial"*
  `http://www.python.org/doc/2.3.4/tut/tut.html`
- M. Pilgrim, *"Dive into Python"*
  `http://diveintopython.org/`
- A. Downey, J. Elkner, C. Meyers, *"How to think like a computer scientist"*
  `http://www.greenteapress.com/thinkpython/`
- B. Eckel, *"Thinking in Python"*
  `http://www.mindview.net/Books/TIPython`

- D. Mertz, *"Text Processing in Python"*
  `http://gnosis.cx/TPiP/`
- F. Lundh, *"An introduction to Tkinter"*
  `http://www.pythonware.com/library/tkinter/introduction/`

Pomoc

```
szwabin@voyager:~$ python
Python 2.4.2 (#2, Sep 30 2005, 21:19:01)
[GCC 4.0.2 20050808 (prerelease) (Ubuntu 4.0.1-4ubuntu8)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> help
Type help() for interactive help, or help(object) for help about object.
>>> help()

Welcome to Python 2.4!  This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://www.python.org/doc/tut/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics".  Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help>
```

```
help> modules

Please wait a moment while I gather a list of all available modules...

ADNS                    TgaImagePlugin      dumbdbm              pyid3lib
ArgImagePlugin          TiffImagePlugin     dummy_thread         pyosd (package)
ArrayPrinter            TiffTags            dummy_threading      pyuno
BaseHTTPServer          Tix                 easydoc              quopri
Bastion                 Tkconstants         easydocp             random
BdfFontFile             Tkdnd               email (package)      ranlib
BicycleRepairMan_Idle   Tkinter               encodings (package) re
BmpImagePlugin          UserArray           epydoc (package)     readline
BufrStubImagePlugin     UserDict            errno                reconvert
CDDB                    UserList            eunuchs (package)    regex
CDROM                   UserString          exceptions           regex_syntax
CGIHTTPServer           WalImageFile        fcntl                regsub
CORBA                   WmfImagePlugin      filecmp              reportlab (package)
Canvas                  XVThumbImagePlugin  fileinput            repr
.
.
.
```

```
help> math
Help on module math:

NAME
    math

FILE
    /usr/lib/python2.4/lib-dynload/math.so

MODULE DOCS
    http://www.python.org/doc/current/lib/module-math.html

DESCRIPTION
    This module is always available.  It provides access to the
    mathematical functions defined by the C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured in radians) of x.
.
.
.
```

```
help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

and                 else                import              raise
assert              except              in                  return
break               exec                is                  try
class               finally             lambda              while
continue            for                 not                 yield
def                 from                or
del                 global              pass
elif                if                  print
```

```
help> topics

Here is a list of available topics.  Enter any topic name to get more help.

ASSERTION                DELETION                 LOOPING                  SEQUENCES
ASSIGNMENT               DICTIONARIES             MAPPINGMETHODS           SHIFTING
ATTRIBUTEMETHODS         DICTIONARYLITERALS       MAPPINGS                 SLICINGS
ATTRIBUTES               DYNAMICFEATURES          METHODS                  SPECIALATTRIBUTES
AUGMENTEDASSIGNMENT      ELLIPSIS                 MODULES                  SPECIALIDENTIFIERS
BACKQUOTES               EXCEPTIONS               NAMESPACES               SPECIALMETHODS
BASICMETHODS             EXECUTION                NONE                     STRINGMETHODS
BINARY                   EXPRESSIONS              NUMBERMETHODS            STRINGS
BITWISE                  FILES                    NUMBERS                  SUBSCRIPTS
BOOLEAN                  FLOAT                    OBJECTS                  TRACEBACKS
CALLABLEMETHODS          FORMATTING               OPERATORS                TRUTHVALUE
CALLS                    FRAMEOBJECTS             PACKAGES                 TUPLELITERALS
CLASSES                  FRAMES                   POWER                    TUPLES
CODEOBJECTS              FUNCTIONS                PRECEDENCE               TYPEOBJECTS
COERCIONS                IDENTIFIERS              PRINTING                 TYPES
COMPARISON               IMPORTING                PRIVATENAMES             UNARY
COMPLEX                  INTEGER                  RETURNING                UNICODE
CONDITIONAL              LISTLITERALS             SCOPING
CONVERSIONS              LISTS                    SEQUENCEMETHODS1
DEBUGGING                LITERALS                 SEQUENCEMETHODS2
```

```
help> quit

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
>>>
```

Zaawansowany kalkulator

```
>>> 2+2
4
>>> x=20
>>> y=2*3
>>> x*y
120
>>> _*0.1
12.0
>>> a,b=3,56
>>> a,b
(3, 56)
>>> a
3
>>> b
56
>>> 7/3
2
>>> 7.0/3
2.3333333333333335
>>> x=y=z=0
>>> x,y,z
(0, 0, 0)
```

- operatory arytmetyczne

```
+ - * / **
```

- komentarze

```
>>> 7/3 # dzielenie calkowite zwraca liczbe zaokraglona w dol
2
```

- liczby zespolone

```
>>> complex(2,3)
(2+3j)
>>> 1j*1J
(-1+0j)
>>> a = 1.5 + 0.5j
>>> a.real
1.5
>>> a.imag
0.5
>>> abs(a)
1.5811388300841898
>>> 3*a
(4.5+1.5j)
```

Pierwszy program w Pythonie

- plik tekstowy o dowolnej nazwie, np. `rkwad.py`

```
# rkwad.py
# Program obliczajacy pierwiastki
# rzeczywiste trojmianu kwadratowego

import math

a,b,c = input ("Podaj wspolczynniki (a,b,c): ")
pd = math.sqrt(b**2 - 4*a*c)
x1 = (-b+pd)/(2*a)
x2 = (-b-pd)/(2*a)
print "\nPierwiatki wynosza: x1=",x1," x2=",x2
```

- wywołanie w konsoli

```
szwabin@voyager:~/UniWroclaw/Talks/PythonIntro$ python rkwad.py
Podaj wspolczynniki (a,b,c): 1,4,2
Pierwiatki wynosza: x1= -0.585786437627  x2= -3.41421356237
szwabin@voyager:~/UniWroclaw/Talks/PythonIntro$ python rkwad.py
Podaj wspolczynniki (a,b,c): 1,1,1
Traceback (most recent call last):
  File "rkwad.py", line 8, in ?
    pd = math.sqrt(b**2 - 4*a*c)
ValueError: math domain error
```

Moduły

- pliki zawierające instrukcje Pythona
- definicje z modułu mogą być importowane
- do „przeglądania” modułów służy funkcja `dir()`

```
>>> import math
>>> dir(math)
['__doc__', '__file__', '__name__', 'acos', 'asin',
'atan', 'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e',
'exp', 'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp',
'log', 'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh']
```

- kilka sposobów importowania

```
>>> import math
>>> a=math.sqrt(2)

>>> import math as m
>>> a=m.sqrt(2)

>>> from math import *
>>> a=sqrt(2)

>>> from math import sqrt as squareRoot
>>> a=squareRoot(2)
```

- możliwość tworzenia własnych modułów
- Global Module Index
  http://www.python.org/doc/2.4.2/modindex.html
- Python Library Reference
  http://www.python.org/doc/2.4.2/lib/lib.html

Łańcuchy znaków

```
>>> s1 = "To jest lancuch znakow"
>>> s2 = 'To jest inny lancuch znakow'
>>> s3 = """ To jest lancuch znakow, ktory
      rozciaga sie na wiele wierszy"""
>>> s4 = "\"Cytowanie\""
>>> print s4
"Cytowanie"
>>> "Witaj" + " swiecie"
'Witaj swiecie'
>>> "Witaj"*3
'WitajWitajWitaj'
>>> greet = "Witaj"
>>> print greet[0]
W
>>> greet[2:4]
'ta'
>>> greet[2:]
'taj'
>>> len(greet)
5
>>> print greet[-1],greet[-2]
j a
>>> str(10)
'10'
>>> eval('3+4')
7
```

- łańcuchów znaków w Pythonie nie można modyfikować!

```
>>> s="Ala ma kota"
>>> s[0]="O"
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assigment
```

- biblioteka standardowa `string`:

`capitalize(s)`	`rfind(s,substring)`
`upper(s)`	`replace(s,old,new)`
`lower(s)`	`strip(s)`
`ljust(s)`	`rstrip(s)`
`center(s)`	`lstrip(s)`
`rjust(s)`	`split(s,char)`
`count(substring,s)`	`join(stringList)`
`find(s,substring)`	

Złożone typy danych - listy

```
>>> lista=["a","b","przyklad"]
>>> lista[0]
'a'
>>> lista[2]
'przyklad'
>>> lista[-1]
'przyklad'
>>> lista.append('nowy')
>>> lista
['a', 'b', 'przyklad', 'nowy']
>>> lista.insert(2,"trzeci")
>>> lista
['a', 'b', 'trzeci', 'przyklad', 'nowy']
>>> lista.extend(['jeden','dwa'])
>>> lista
['a', 'b', 'trzeci', 'przyklad', 'nowy', 'jeden', 'dwa']
>>> lista.index('trzeci')
2
>>> 'c' in lista
False
>>> lista.remove('nowy')
>>> lista
['a', 'b', 'trzeci', 'przyklad', 'jeden', 'dwa']
>>> lista.pop()
'dwa'
>>> lista
['a', 'b', 'trzeci', 'przyklad', 'jeden']
```

```
>>> lista+['inny','element']
['a', 'b', 'trzeci', 'przyklad', 'jeden', 'inny', 'element']
>>> li=[1,2]*3
[1, 2, 1, 2, 1, 2]
>>> li.sort()
>>> li
[1, 1, 1, 2, 2, 2]
>>> li = [1,2,3,4]
>>> li2 = [elem**2 for elem in li]
>>> li2
[1, 4, 9, 16]
```

Złożone typy danych - krotki (ang. *tuples*)

```
>>> t = ("a","b","przyklad")
>>> t[0]
'a'
>>> t[2]
'przyklad'
>>> t[-1]
'przyklad'
>>> 'a' in t
True
```

Uwaga!

Krotki nie mogą być modyfikowane!

```
>>> t[2]=3
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment
```

Złożone typy danych - słowniki

```
>>> d = {"server":"defiant","user":"szwabin"}
>>> d
{'user': 'szwabin', 'server': 'defiant'}
>>> d['server']
'defiant'
>>> d['server']='voyager'
>>> d
{'user': 'szwabin', 'server': 'voyager'}
>>> d['uid']=501
>>> d
{'uid': 501, 'user': 'szwabin', 'server': 'voyager'}
>>> del d['uid']
>>> d
{'user': 'szwabin', 'server': 'voyager'}
>>> d.clear()
>>> d
{}
>>> d = {"server":"defiant","user":"szwabin"}
>>> d
{'user': 'szwabin', 'server': 'defiant'}
>>> d.keys()
['user', 'server']
>>> d.values()
['szwabin', 'defiant']
>>> d.items()
[('user', 'szwabin'), ('server', 'defiant')]
```

```
>>> d.clear()
>>> d
{}
>>> d = {"server":"defiant","user":"szwabin"}
>>> d
{'user': 'szwabin', 'server': 'defiant'}
>>> d.keys()
['user', 'server']
>>> d.values()
['szwabin', 'defiant']
>>> d.items()
[('user', 'szwabin'), ('server', 'defiant')]
```

Pętla while

```
>>> a,b =0,1
>>> while b<10:
...     print b,
...     a,b = b,a+b
...
1 1 2 3 5 8
```

- operatory porównania: < > == <= >=
- każda sekwencja o długości niezerowej oznacza prawdę

Pętla if

```
>>> x = int(raw_input("Podaj liczbe: "))
>>> if x < 0:
...     print 'liczba ujemna''
... elif x == 0:
...     print 'zero'
... else:
...     print 'liczba dodatnia''
...
```

Pętla for

- iteracja po elementach sekwencji:

```
>>> a = ['jeden', 'dwa', 'trzy']
>>> for x in a:
...     print x, len(x)
...
jeden 5
dwa 3
trzy 4
```

- iteracja od liczby do liczby z użyciem funkcji `range()`

```
>>> for i in range(10):
...     print i,
...
0 1 2 3 4 5 6 7 8 9
```

Definiowanie własnych funkcji

```
>>> def fib(n):
...     "Ciag Fibonacciego"
...     a, b = 0, 1
...     while b < n:
...         print b,
...         a, b = b, a+b
...
>>> fib(2000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

```
>>> def fib2(n):
...     "Wartosci ciagu Fibonacciego"
...     wynik = []
...     a, b = 0, 1
...     while b < n:
...         wynik.append(b)
...         a, b = b, a+b
...     return wynik
...
>>> f100 = fib2(100)
>>> f100
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Odczyt z i zapis do pliku

```
szwabin@voyager:~$ cat lud.dat
1850.0   1.402
1950.0   2.486
1960.0   3.014
1970.0   3.683
1985.0   4.842
2000.0   6.127
2025.0   8.177
```

```
>>> import string
>>> x=[]; y=[]
>>> f=open('lud.dat','r')
>>> lines=f.readlines()
>>> for line in lines:
...     xval,yval=line.split()
...     x.append(float(xval))
...     y.append(float(yval))
...
>>> f.close()
>>> x
[1850.0, 1950.0, 1960.0, 1970.0, 1985.0, 2000.0, 2025.0]
>>> y
[1.4019999999999999, 2.4860000000000002, 3.0139999999999998,
3.6829999999999998, 4.8419999999999996, 6.1269999999999998,
8.1769999999999996]
```

```
>>> x=[1.0,2.0,3.0,4.0,5.0,6.0]
>>> y=[i**2 for i in x]
>>> f=open('kwadrat.dat','w')
>>> for i in range(len(x)):
...         f.write('%g %g\n' % (x[i],y[i]))
...
>>> f.close()
```

```
szwabin@voyager:~$ cat kwadrat.dat
1 1
2 4
3 9
4 16
5 25
6 36
```

SciPy

- biblioteka numeryczna dla Pythona
- strona główna
  `http://www.scipy.org`
- wymagania
  - gcc/g77
  - Python z obsługą biblioteki zlib
  - Numerical Python
  - biblioteki Atlas i FFTW (wydajność!)
  - WxPython, PyGist, Gnuplot
  - F2PY

Pierwsze kroki

- sposoby importowania

```
>>> from scipy import *
```

```
>>> import scipy
```

- pomoc

```
>>> scipy.info(scipy.optimize)
 Optimization Tools
=====================

 A collection of general-purpose optimization routines.

   fmin        --  Nelder-Mead Simplex algorithm
                     (uses only function calls)
   fmin_powell --  Powell's (modified) level set method (uses only
                     function calls)
   fmin_cg     --  Non-linear (Polak-Rubiere) conjugate gradient algorithm
                     (can use function and gradient).
.
.
.
```

- kod źródłowy używanych funkcji

```
>>> scipy.source(scipy.integrate.odeint)
>>> scipy.source(scipy.integrate.odeint)
In file: /usr/lib/python2.4/site-packages/scipy/integrate/odepack.py

def odeint(func, y0, t, args=(), Dfun=None, col_deriv=0, full_output=0,
           ml=None, mu=None, rtol=None, atol=None, tcrit=None, h0=0.0,
           hmax=0.0, hmin=0.0, ixpr=0, mxstep=0, mxhnil=0, mxordn=12,
           mxords=5, printmessg=0):

    """Integrate a system of ordinary differential equations.

    Description:

      Solve a system of ordinary differential equations Using lsoda from the
      FORTRAN library odepack.

      Solves the initial value problem for stiff or non-stiff systems
      of first order ode-s:
           dy/dt = func(y,t0,...) where y can be a vector.
.
.
.
```

Moduły dostępne w pakiecie

cluster	algorytmy klasteryzacji danych
cow	pakiet do programowania równoległego
fftpack	domyślny pakiet FFT
fftw	pakiet do FFT oparty o bibliotekę FFTW
ga	algorytmy genetyczne
gplt	pakiet do wykresów w Gnuplocie
integrate	całkowanie numeryczne
interpolate	interpolacja
io	funkcje wejścia/wyjścia
linalg	algebra liniowa
optimize	zagadnienia optymalizacji
plt	pakiet do wykresów w WxPython
signal	przetwarzanie sygnałów
special	funkcje specjalne
stats	funkcje statystyczne
xplt	pakiet do wykresów w Gist

Macierze

- tworzenie

```
>>> A1=scipy.mat('[1 3 5; 2 5 1; 2 3 8]')
>>> A2=scipy.mat('[5 3 5; 2 3 1; 4 3 1]')
>>> A1
 Matrix([[1, 3, 5],
        [2, 5, 1],
        [2, 3, 8]])
```

- dodawanie

```
>>> A1+A2
Matrix([[ 6,  6, 10],
        [ 4,  8,  2],
        [ 6,  6,  9]])
```

- mnożenie przez skalar

```
>>> 2*A1
Matrix([[ 2,  6, 10],
        [ 4, 10,  2],
        [ 4,  6, 16]])
```

- mnożenie macierzy

```
>>> A1*A2
Matrix([[31, 27, 13],
        [24, 24, 16],
        [48, 39, 21]])
```

- macierz odwrotna

```
>>> A = scipy.mat('[1 3 5; 2 5 1; 2 3 8]')
>>> B = A.I
>>> A*B
Matrix([[  1.00000000e+00,   5.55111512e-17,   4.85722573e-17],
        [ -1.38777878e-16,   1.00000000e+00,   7.63278329e-17],
        [ -2.22044605e-16,   5.55111512e-17,   1.00000000e+00]])
>>> C=scipy.linalg.inv(A)
>>> A*C
Matrix([[  1.00000000e-00,  -6.93889390e-17,   1.04083409e-16],
        [ -2.77555756e-16,   1.00000000e+00,   1.31838984e-16],
        [ -3.33066907e-16,   0.00000000e+00,   1.00000000e+00]])
>>> scipy.round(A*C)
array([[ 1., -0.,  0.],
       [-0.,  1.,  0.],
       [-0.,  0.,  1.]])
```

- wyznacznik

```
>>> A = scipy.mat('[1 3 5; 2 5 1; 2 3 8]')
>>> scipy.linalg.det(A)
-25.000000000000004
```

- norma

```
>>> A = scipy.mat('[1 5 2; 2 4 1; 3 6 2]')
>>> scipy.linalg.norm(A,2)
9.918391609146008
>>> scipy.linalg.norm(A,1)
15
>>> scipy.linalg.norm(A,scipy.inf)
11
```

$$\|\mathbf{A}\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|$$

$$\|\mathbf{A}\|_2 = \sqrt{\lambda_{max}}$$

$$\|\mathbf{A}\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|$$

$$\|\mathbf{A}\|_E = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2}$$

- wartości i wektory własne

```
>>> A = scipy.mat('[1 5 2; 2 4 1; 3 6 2]')
>>> la,v = scipy.linalg.eig(A)
>>> l1,l2,l3 = la
>>> print l1,l2,l3
(7.95791620491+0j) (-1.25766470568+0j) (0.299748500767+0j)
>>> print v[:,0]
[-0.5297175  -0.44941741  -0.71932146]
>>> print v[:,1]
[-0.90730751  0.28662547  0.30763439]
>>> print v[:,2]
[ 0.28380519  -0.39012063  0.87593408]
```

Układy równań liniowych

```
>>> A = scipy.mat('[1 3 5; 2 5 1; 2 3 8]')
>>> b = scipy.mat('[10;8;3]')
>>> A.I*b
Matrix([[-9.28],
        [ 5.16],
        [ 0.76]])
>>> scipy.linalg.solve(A,b) # lepsza metoda
array([[-9.28],
       [ 5.16],
       [ 0.76]])
```

Równania nieliniowe

```
>>> def fun(x):
...     return x**2 - 2
...
>>> scipy.optimize.fsolve(fun,1)
1.4142135623730947
>>> scipy.round(_**2-2)
-0.0
```

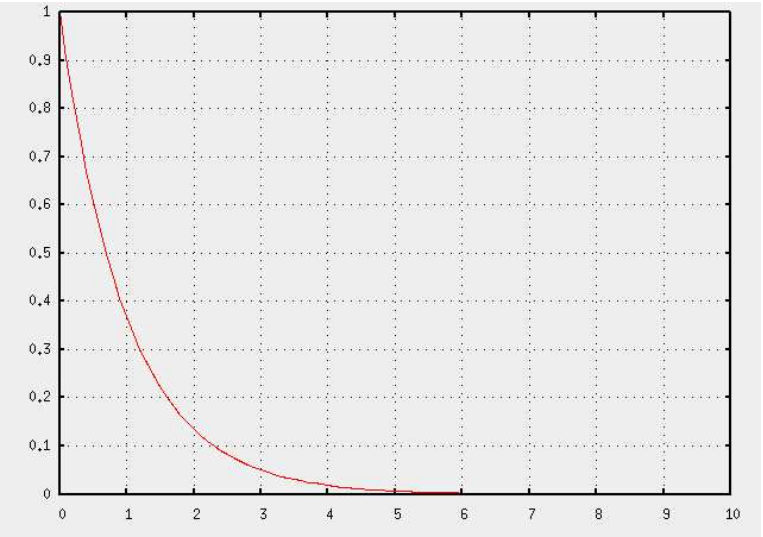
Całkowanie

```
>>> def fun(x):
...     return scipy.exp(-x*x)
...
>>> wynik = scipy.integrate.quad(fun,-scipy.Inf,scipy.Inf)[0]
>>> dokladny_wynik = scipy.sqrt(scipy.pi)
>>> wynik-dokladny_wynik
-2.2204460492503131e-16
>>> scipy.round(_)
-0.0
```

Równania różniczkowe zwyczajne

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -N(t), \quad N(0) = 1$$

```
>>> def der(N,t):
...     return -N
...
>>> N0=1;
>>> t=scipy.arange(0,10.0,0.1);
>>> N=scipy.integrate.odeint(der,N0,t);
>>> scipy.gplt.plot(t,N)
```



SciPy w przykładach

Rozwiąż układ równań

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{pmatrix} x = \begin{pmatrix} 2 \\ 10 \\ 44 \\ 190 \end{pmatrix}$$

Oblicz $\|b - \mathbf{A}\tilde{x}\|_\infty$, gdzie $\tilde{x}$ jest rozwiązaniem numerycznym

```
>>> import scipy
>>> A = scipy.mat('[1 2 3 4; 1 4 9 16; 1 8 27 64; 1 16 81 256]')
>>> b = scipy.mat('[2; 10; 44; 190]')
>>> res = scipy.linalg.solve(A,b)
>>> res
array([[-1.],
       [ 1.],
       [-1.],
       [ 1.]])
>>> norma = scipy.linalg.norm(b-A*res,scipy.inf)
>>> norma
0.0
```

Oblicz całkę

$$\frac{1}{2\pi}\int_0^{2\pi} \exp\left\{2^{-1/2}\sin(x)\right\}\mathrm{d}x$$

```
>>> import scipy
>>> def fun(x):
...     return scipy.exp(scipy.sin(x)/scipy.sqrt(2))/(2*scipy.pi)
...
>>> calka = scipy.integrate.quad(fun,0,2*scipy.pi)
>>> calka
(1.1289609294541902, 7.4290708898647612e-09)
```

Ustal punkt, w którym przecinają się krzywe $y = 3x$ oraz $y = e^x$

```
>>> import scipy
>>> def fun(x):
...     return scipy.exp(x)-3*x
...
>>> pp = scipy.optimize.fsolve(fun,1)
>>> pp
0.61906128673594507
>>> fun(pp)
0.0
```

Znajdź pierwiastki rzeczywiste równania

$$9x^4 + 18x^3 + 38x^2 - 57x + 14 = 0$$

```
>>> import scipy
>>> wiel = scipy.poly1d([9,18,38,-57,14])
>>> print wiel
   4       3       2
9 x + 18 x + 38 x - 57 x + 14
>>> for x in wiel.r:  #wiel.r zawiera wszystkie pierwiastki
...     if x.imag==0: print x
...
(0.666666666667+0j)
(0.333333333333+0j)
```

Jest dana macierz hermitowska

$$\mathbf{A} = \begin{pmatrix} 1 & 1-i & 1-2i & 1-3i \\ 1+i & 2 & 2-i & 2-2i \\ 1+2i & 2+i & 3 & 3-i \\ 1+3i & 2+2i & 3+i & 4 \end{pmatrix}$$

Znajdź wartości i wektory własne macierzy **A**

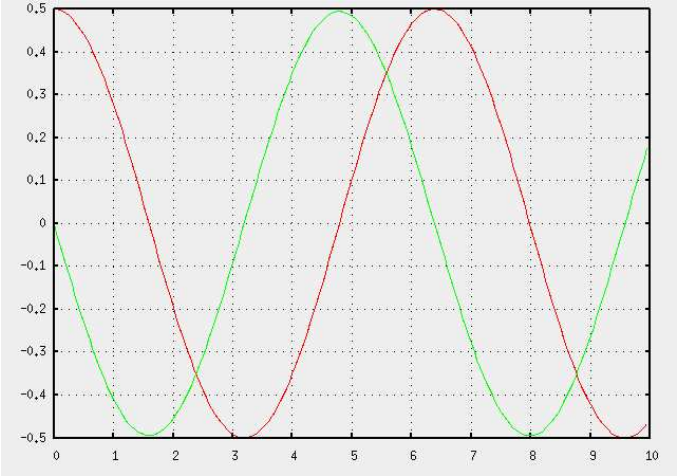
```
>>> import scipy
>>> A = scipy.mat('[1 1-1*j 1-2*j 1-3*j; 1+1*j 2 2-1*j 2-2*j;
      1+2*j 2+1*j 3 3-1*j; 1+3*j 2+2*j 3+1*j 4]')
>>> la,v=scipy.linalg.eig(A)
>>> for l in la: print l
...
(10.3993569323 -1.29665879217e-17j)
(-1.22156553583+1.36699815578e-16j)
(0.518667170807+1.4608912657e-16j)
(0.303541432686 -4.07914215526e-17j)
```

Rozwiąż równanie opisujące ruch wahadła matematycznego

$$\ddot{\phi} + \frac{g}{R}\sin\phi = 0$$

Wyniki przedstaw na wykresie.

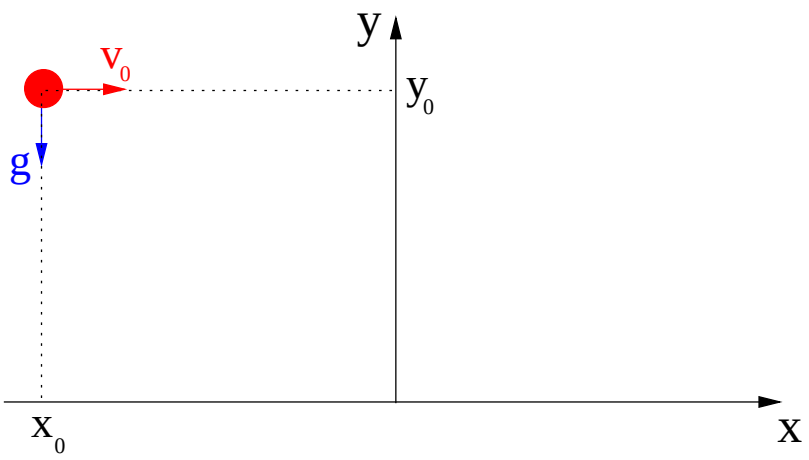
```
>>> import scipy,math
>>> phi0=0.5
>>> v0=0.0
>>> init=[phi0,v0]
>>> def der(phi,t):
...     return [phi[1],-math.sin(phi[0])]
...
>>> t=scipy.arange(0,10.0,0.05)
>>> phi=scipy.integrate.odeint(der,init,t)
>>> scipy.gplt.plot(t,phi)
```



VPython

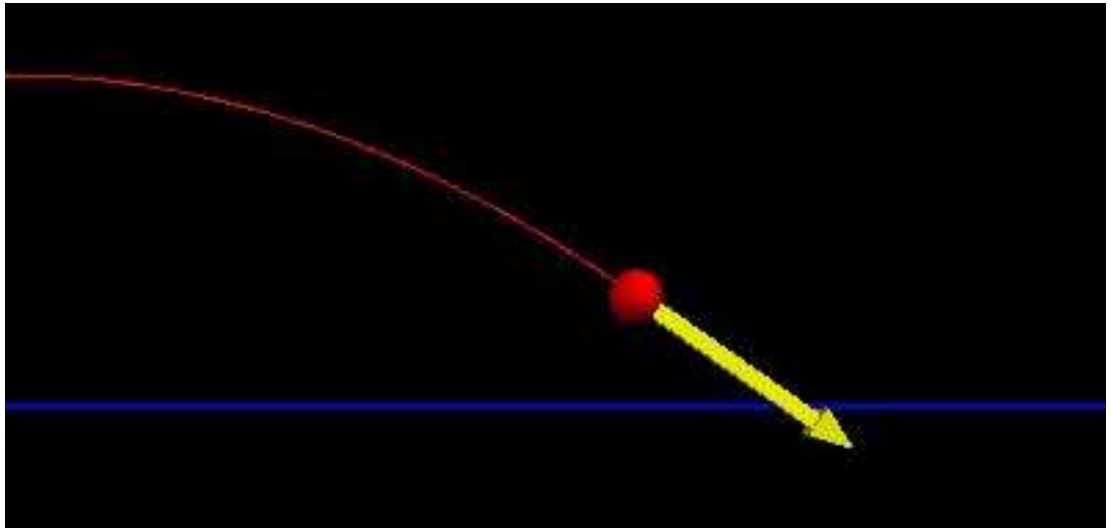
- `http://vpython.org`
- biblioteka do tworzenia grafiki 3D w Pythonie
- zaawansowane animacje niewielkim nakładem sił

Rzut poziomy



$$
\begin{aligned}
x &= x_0 + V_0 * t \\
y &= y_0 - \frac{gt^2}{2}
\end{aligned}
$$

Cel

- animacja lotu piłki
- narysowanie toru piłki
- wizualizacja wektora prędkości piłki

```
#!/usr/bin/env python

from visual import *

x0,y0 = -10,6
yz0=0
v0=4.0
g=0.981
scene = display(title='Rzut poziomy',width=400,height=400)
scene.autoscale = 0
ziemia = box(pos=(x0,yz0,0),size=(40,0.1,0),color=color.blue)
pilka = sphere(pos=(x0,y0,0),radius=0.5,color=color.red)
pilka.velocity = vector(v0,0,0)
wektor = arrow(pos=pilka.pos, axis=pilka.velocity, color=color.yellow,
                              shaftwidth=0.3)
pilka.trail = curve(color=pilka.color)
t=0
while 1:
    rate(100)
    pilka.pos = [x0+v0*t,y0-g*t*t/2.0,0]
    if pilka.y < ziemia.y:
        break
    pilka.trail.append(pos=pilka.pos)
    pilka.velocity=vector(v0,-g*t,0)
    wektor.pos=pilka.pos
    wektor.axis=pilka.velocity
    t=t+.01
```