Laboratorium: Python i narzędzia

1 Motywacja

Wokół Pythona zbudowano wiele wygodnych i użytecznych narzędzi, które potrafią ułatwić i wspomóc różne codzienne zadania. Python może z powodzeniem[1] zastępować specjalistyczne narzędzia takie jak:

- Mathematica (obliczenia symboliczne) poprzez sympy lub sage
- Matlab (obliczenia numeryczne) poprzez numpy i scipy
- Narzędzia do analizy danych poprzez pandas
- Bash'a poprzez ipython

2 Narzędzia

2.1 IPython

Wykonaj:

1. Uruchom `ipython`

   Zwróć uwagę na obecność autouzupełniania składni. Możesz wykonywać także polecenia bash'a, np.

   ```
   ls -l
   ```

   Możesz także używać zmiennych:

   ```
   user = 'wojciech'
   res = !ls -l | grep {user}
   res
   ```

   Using the ! symbol runs the following command as a shell command, and the results are all stored in user.

2. Możesz przejrzeć dokumentację
3. Istnieje też shell python'owy:

   ```
   ipython --profile=pysh
   ```

4. Uruchom `ipython qtconsole`. Zwróć uwagę na obecność autopodpowiedzi argumentów funkcji.
5. Wykonaj

   ```
   from matplotlib import pylab as plt
   import numpy as np
   # example data
   x = np.arange(0.1, 4, 0.1)
   y = np.exp(-x)
   ```

[1] Do pewnego stopnia. Specjalistyczne, komercyjne oprogramowanie będzie lepsze dla bardzo skomplikowanych i wymagających zastosowań

```
# example variable error bar values
yerr = 0.1 + 0.1*np.sqrt(x)
plt.errorbar(x, y, yerr=yerr, errorevery=3)
plt.show()
```

Powtórz komendy rysowania z poprzedniego punktu. Gdy rysunek wyświetli się, popraw go wykonując (kolejno!) komendy:

```
plt.title('Ten rysunek moge zmieniac interaktywnie')
plt.plot(x, np.sin(x), 'o')
```

6. Poniższy kod poprzedź magiczną komendą ipythona `%matplotlib inline`:

```
%matplotlib inline
from matplotlib import pylab as plt
import numpy as np
# example data
x = np.arange(0.1, 4, 0.1)
y = np.exp(-x)

# example variable error bar values
yerr = 0.1 + 0.1*np.sqrt(x)
plt.errorbar(x, y, yerr=yerr, errorevery=3)
```

2.2 PIP

Pip jest narzędziem współpracującym z repozytorium pakietów Pythona PyPI

1. Zainstaluj bibliotekę `scikit-image` używając poleceń):

   ```
   pip search scikit
   pip install --user scikit-image
   ```

2. Sprawdź czy biblioteka działa (`ipython qtconsole`):

```
import matplotlib.pylab as plt
from skimage import data, io, filters

image = data.coins()
plt.subplot(1, 2, 1)
io.imshow(image)

edges = filters.sobel(image)
plt.subplot(1, 2, 2)
io.imshow(edges)
```

2.3 IPython notebook

1. Przejdź do katalogu, w którym chcesz utworzyć swój notebook.
2. Uruchom `ipython notebook`.
3. Utwórz notebook: *New→Python*
4. Dodaj w swoim notebooku komórkę. Umieść w niej prosty skrypt pythona i uruchom (*shift + enter*).
5. Dokonaj konwersji notebooka do pliku html: `ipython nbconvert plik.ipynb`.

2.4 Biblioteka sympy

Sympy pozwala na obliczenia symboliczne. Szczególnie dobrze sprawdza się w połączeniu z `ipython notebook`

In [4]:
```
# Trochę magii
from sympy.interactive import printing
printing.init_printing(use_latex=True)

from __future__ import division
import sympy as sym
from sympy import *

# Definicje zmiennych
x, y, z = symbols("x y z")
k = Symbol("k", integer=True)
f = Function('f')
```

In [5]:
```
eq = ((x+y)**2 * (x+1))
eq
```

Out[5]:

$$(x+1)\,(x+y)^2$$

In [6]:
```
expand(eq)
```

Out[6]:

$$x^3 + 2x^2y + x^2 + xy^2 + 2xy + y^2$$

In [7]:
```
a = 1/x + (x*y - 1)/x + 1
a
```

Out[7]:

$$1 + \frac{1}{x}\,(xy - 1) + \frac{1}{x}$$

In [8]:
```
simplify(a)
```

Out[8]:

$$y + 1$$

In [9]:
```
eq = Eq(x - 3)
eq
```

Out[9]:

$$x - 3 = 0$$

In [10]:
```
solve(eq, x)
```

Out[10]:

$$[3]$$

In [11]:
```
eq = Eq(x**3 + 3*x**2 - 13*x - 15)
 eq
```

Out[11]:

$$x^3 + 3x^2 - 13x - 15 = 0$$

In [ ]:
```
solve(eq, x)
```

In [12]:
```
eq = Eq(x**3 + 3*x**2 - 13*x + 15)
solve(eq, x)
```

Out[12]:

$$\left[-1 - \frac{16}{3\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{\sqrt{5937}}{9}+15}} - \left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{\sqrt{5937}}{9}+15},\quad -1 - \left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{\sqrt{5937}}{9}+15} - \frac{16}{3\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{\sqrt{5937}}{9}+15}},\quad -\sqrt[3]{\frac{\sqrt{5937}}{9}+15} - \frac{16}{3\sqrt[3]{\frac{\sqrt{5937}}{9}+15}} - 1\right]$$

In [13]:
```
eq = Eq(x**3 + 3*x**2 - 13*x + z)
solve(eq, x)
```

Out[13]:

$$\left[-\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}} - 1 - \frac{16}{3\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}}},\quad -\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}} - 1 - \frac{16}{3\left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}}},\quad -\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}} - 1 - \frac{16}{3\sqrt[3]{\frac{z}{2}+\sqrt{\frac{1}{4}(z+15)^2 - \frac{4096}{27}}+\frac{15}{2}}}\right]$$

In [14]:
```
s = Sum(6*k**2 - 1, (k, 1, 10))
s
```

Out[14]:

$$\sum_{k=1}^{10}\left(6k^2 - 1\right)$$

In [15]:
```
lim = Limit(1/x, x, 0)
lim
```

Out[15]:

$$\lim_{x \to 0^+} \frac{1}{x}$$

In [16]:
```
lim.doit()
```

Out[16]:

$$\infty$$

In [17]:
```
lim = 2*Limit((1+1/x)**x, x, +oo)
lim
```

Out[17]:

$$2\lim_{x \to \infty}\left(1 + \frac{1}{x}\right)^x$$

In [18]:
```
lim.doit()
```

Out[18]:

$$2e$$

In [19]:
```
q = ((3*x**2))
q
```

Out[19]:

$$3x^2$$

In [20]:
```
q.diff(x)
```

Out[20]:

$$6x$$

```
In [21]: eqn = Eq(Derivative(f(x),x,x) + 9*f(x), 1)
         eqn
```

Out[21]:

$$9f(x) + \frac{d^2}{dx^2} f(x) = 1$$

```
In [22]: dsolve(eqn, f(x))
```

Out[22]:

$$f(x) = C_1 \sin(3x) + C_2 \cos(3x) + \frac{1}{9}$$

2.5 Biblioteka numpy

Numpy jest biblioteką do obliczeń numerycznych. Szczególnie dobrze nadaje się do operacji na macierzach i wektorach.

Zwróć uwagę:

- numpy działa w oparciu nie o wbudowany typ `list`, ale o specjalny typ macierzowy `array`.
- `array` umożliwia bardzo wygodne "krojenie" macierzy

Krótki tutorial:

```
In [1]: import numpy as np

        x = [1, 2, 3, 4, 5, 6 ]
        v = np.array(x)
        v
```

```
Out[1]: array([1, 2, 3, 4, 5, 6])
```

```
In [2]: M = np.array([[1,2], [3,4]])
        M
```

```
Out[2]: array([[1, 2],
               [3, 4]])
```

```
In [3]: type(v), type(M)
```

```
Out[3]: (numpy.ndarray, numpy.ndarray)
```

```
In [4]: v.shape
```

```
Out[4]: (6,)
```

```
In [5]: M.shape
```

```
Out[5]: (2, 2)
```

```
In [6]: M.size
```

```
Out[6]: 4
```

```
In [7]: M.dtype
```

```
Out[7]: dtype('int64')
```

```
In [8]: np.random.rand(5,3)
```

```
Out[8]: array([[ 0.8419523 ,  0.57688591,  0.04213716],
               [ 0.95775267,  0.82243871,  0.9712648 ],
               [ 0.63947369,  0.62505725,  0.30284852],
               [ 0.70642883,  0.25074107,  0.91719955],
               [ 0.42341419,  0.32112286,  0.61188051]])
```

```
In [9]: np.zeros([5,3])
```

```
Out[9]: array([[ 0.,  0.,  0.],
               [ 0.,  0.,  0.],
               [ 0.,  0.,  0.],
               [ 0.,  0.,  0.],
               [ 0.,  0.,  0.]])
```

```
In [10]: M = np.array([[1,2,3], [4,5,6]])
         N = np.array([[1,1,1], [2,2,2]])
         print(N)
         print(M)
```

```
[[1 1 1]
 [2 2 2]]
[[1 2 3]
 [4 5 6]]
```

```
In [11]: M + N
```

```
Out[11]: array([[2, 3, 4],
                [6, 7, 8]])
```

```
In [12]: 5*M
```

```
Out[12]: array([[ 5, 10, 15],
                [20, 25, 30]])
```

```
In [13]: # Mnozenie kolejnych komorek macierzy
         M*N
```

```
Out[13]: array([[ 1,  2,  3],
                [ 8, 10, 12]])
```

```
In [14]: # Mnożenie macierzy
         M.dot(N)
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-14-f4eb9d722fc3> in <module>()
          1 # Mnożenie macierzy
    ----> 2 M.dot(N)


        ValueError: shapes (2,3) and (2,3) not aligned: 3 (dim 1) != 2 (dim 0)
```

```
In [15]: # Transpozycja
         M.dot(N.T)
```

```
Out[15]: array([[ 6, 12],
                [15, 30]])
```

```
In [16]: # Generowanie rownych odstepow
         v=np.linspace(0,10,21)
         v
```

```
Out[16]: array([  0. ,   0.5,   1. ,   1.5,   2. ,   2.5,   3. ,   3.5,   4. ,
                  4.5,   5. ,   5.5,   6. ,   6.5,   7. ,   7.5,   8. ,   8.5,
                  9. ,   9.5,  10. ])
```

```
In [17]: np.sin(v)

Out[17]: array([ 0.        ,  0.47942554,  0.84147098,  0.99749499,  0.90929743,
                 0.59847214,  0.14112001, -0.35078323, -0.7568025 , -0.97753012,
                -0.95892427, -0.70554033, -0.2794155 ,  0.21511999,  0.6569866 ,
                 0.93799998,  0.98935825,  0.79848711,  0.41211849, -0.07515112,
                -0.54402111])

In [18]: from matplotlib import pyplot as plt
         plt.plot(v, np.sin(v))
         plt.show()

In [19]: M = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]])
         M

Out[19]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12],
                [13, 14, 15, 16]])

In [20]: # Krojenie macierzy
         # Wszystkie wiersze razy 2ga (trzecia) kolumna
         M[:,2]

Out[20]: array([ 3,  7, 11, 15])

In [22]: M[2,1:3]

Out[22]: array([10, 11])

In [23]: M[1:3,2:4]

Out[23]: array([[ 7,  8],
                [11, 12]])

In [24]: M[::2,:]

Out[24]: array([[ 1,  2,  3,  4],
                [ 9, 10, 11, 12]])

In [25]: M

Out[25]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12],
                [13, 14, 15, 16]])

In [26]: M[::2,::2]=0
         M

Out[26]: array([[ 0,  2,  0,  4],
                [ 5,  6,  7,  8],
                [ 0, 10,  0, 12],
                [13, 14, 15, 16]])
```

2.6 PyCharm

Zapoznaj się ze środowiskiem PyCharm. Zwróć uwagę, że ma on wbudowaną konsolę IPython.

3 Zadania

3.1 IPython

1. Wypisz wszystkie pliki z bieżącego katalogu, których wielkość jest z zakresu od 4000B do 10000B (Dobierz zakres tak, by pokazał część plików, a część nie). Oczekiwany (przykładowy) wynik:

```
[('4096', 'Dropbox'),
('4096', 'eclipse'),
('4096', 'fontconfig'),
('7340', 'launchy.db')]
```

2. Następnie zapisz cały program w pliku `list-files.ipy` w pierwszej linii dodając:

   ```
   #/usr/bin/env ipython
   ```

   Uruchom skrypt:

   ```
   ipython list-files.ipy
   ```

3.2 Biblioteka sympy

1. Dany jest wielomian: $x^3 + 3x^2 - 10x + 24$. Wyrysuj wykres tego wielomianu (pylab, numpy, od -5 do 5 z krokiem 0.1) i określ punkty przecięcia z osią OX (pierwiastki wielomianu). Oblicz te pierwiastki, wykorzystując sympy.

   Wskazówka: użyj arange do wygenerowania listy x, wygeneruj liste y używając comprehensions ([funkcja for .. in ..]). Do wyciągnięcia wartości równania sympy w punkcie: $eq.evalf(subs = \{x : -3\})$

2. rozwiąż układ równań:

$$x^2 + 3y = 10$$
$$4x - y^2 = -2$$

   Zacznij od:

```
from sympy.interactive import printing
printing.init_printing(use_latex=True)
import sympy as sym

# Definicje zmiennych
x, y = sym.symbols("x y")
```

   Pokazaliśmy, że funkcja solve potrafi przyjąć jako argument równanie. Jak dać jej na wejściu dwa równania? Uwaga: kilka sekund się to liczy.

3. Ile rozwiązań ma ten układ równań? Rozwiąż to równanie używając `sympy`.

   Oczekiwane rozwiązanie

$$\left[\left\{x : -\frac{1}{2} + \frac{1}{4}\left(-\frac{1}{2}\sqrt{-\frac{928}{9\sqrt[3]{\frac{1072}{27} + \frac{16}{3}\sqrt{4873}}}((+\frac{8}{3} + 2\sqrt[3]{\frac{1072}{27} + \frac{16}{3}\sqrt{4873}})) + \dots\right), \dots\right\}, \dots\right]$$

4. Wypisz rozwiązania także w postaci numerycznej. Np. tak:

```
for s in solutions:
    for k, v in s.items():
        print(k, v.evalf())
```

5. Wyznacz pochodną funkcji

$$\sin(log_2(x)) * \cos(x^2)/x$$

6. Przekształć notatnik do strony html oraz pdf'a używając `ipython nbconvert` (więcej informacji: `ipython --help`).

7. Dla chętnych: Przejrzyj przykłady możliwości `ipython notebook`: http://nbviewer.ipython.org/. Z tej strony możesz także ściągnąć i uruchomić lokalnie wybrane notatniki (niepozorny przycisk u góry po prawej stronie).

3.3 Biblioteka numpy

1. Utwórz macierz 2D:

```
[ 1,  3,  1,  2]
[ 1,  2,  5,  8]
[ 3,  1,  2,  9]
[ 5,  4,  2,  1]
```

2. Wytnij z utworzonej macierzy pierwszy i ostatni wiersz oraz ostatnią kolumnę.

3. Utwórz macierz 2D:

```
[ 2,  3,  1]
[ 5,  1,  3]
```

4. Dokonaj transpozycji powyższej macierzy.

5. Oblicz iloczyn macierzy z punktu 2 i 4.

6. Stwórz wykres funkcji *sin* w przedziale od $-\pi$ do π z krokiem co: π, 10, 100.

7. Dla chętnych tutorial: tutorial.