Ćwiczenia I
Wprowadzenie do Pythona

Jakub Tworzydło

Katedra Teorii Materii Skondensowanej
Instytut Fizyki Teoretycznej
telefon: (022)5532-314, pokój 344

3/10/2012 Hoża, Warszawa

Plan

Plan

Plan

Plan

Plan

Zasady zaliczania przedmiotu

- 80% punktów z ćwiczeń
- 20% punktów z testu
- test jest obowiązkowy
- obowiązuje jedna prezentacja w semestrze (5 min) podumowująca wynik poprzednich ćwiczeń

Plan

PYTHON

named after the BBC show "Monty Python's Flying Circus"
... has nothing to do with reptiles

About the origin of Python, Guido Van Rossum wrote in 1996:

Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately.

Cechy Pythona

- easy to learn, very clear syntax, very short but readable code (+ educational)
- interpreter and a scripting language
- real language: simple oo programing, high-level data structures
- powerful (Batteries Included philosophy)
  The Python Standard Library (is huge), NumPy, SciPy. PyPlot
- works on all platforms, easy to install, open source
- efficiency
  - state of art libraries
  - profiling
- easily extensible (e.g some part in C, C++ or FORTRAN)
- open source and very well documented (on Web)

Wyznawcy Pythona

Bruce Eckel (author of "Thinking in Java", "Thinking in C++")
... Python is the only language that focuses on making things easier for the programmer ... nothing has made me more productive ...

Peter Norvig (Lisp author, Director of Search Quality at Google)
... Python is integral part of Google, a requirement for software engineers

... an idea, which can’t be tried in one afternoon is the dead idea ..

Dokumentacja i literatura

- http://www.python.org
  The major Python Web site. It contains code, documentation, and pointers to Python-related pages.
- http://docs.python.org
  Fast access to Python documentation.
- http://aspn.activestate.com/ASPN/Python/Cookbook
  The Python Cookbook collection of code examples, larger modules, and useful scripts
- http://docs.python.org/tutorial/
  Python Tutorial (by Guido van Rossum)
- http://wiki.python.org/moin/
  http://wiki.python.org/moin/PolishLanguage
  Python Wiki
- http://www.swaroopch.com/notes/Python
  http:
  //wiki.lo5.bielsko.pl/index.php/Ukąś_Pythona

Plan

Interaktywny kalkulator

INSTALACJA
pakiety dostępne we wszystkich dystrybucjach Linuxa
także użytkownicy Windows: 70% Python downloads

```
$ python -V
```

LICZBY

```
$ python

>>> 2+2      # this is a comment
>>> 2/3      # integer division
>>> 2/3.     # floating point division

>>> 10**100           # googol
```

Zmienne

```
>>> h = 20.1      # assigment
>>> w = 3.2
>>> w*h
>>> w = h = 12    # simultaneous assigment
>>> w
>>> h
>>> w, h = 5, 6. # multiple assigment
>>> t = w, h      # composed t
>>> t             # t is a TUPLE
>>> t[0]
>>> t[1]
```

Liczby zespolone

```
>>> 1j              # complex unit
>>> (1+2j)*3
>>> (1+2j)/(1+1j)
>>> z = 1.5+.5j
>>> abs(z)
>>> z.real
>>> z.imag
```

Moduł “math”

```
>>> import math  # some good stuff to play with
                 # simplest use of a module

>>> math.factorial(70)
>>> math.factorial(70)/math.factorial(69)

>>> ( math.e**(1j*math.pi) ).imag
>>> ( math.e**(1j*math.pi) ).real

>>> help(math)
>>> help(math.tanh)
```

Moduł “cmath”

```
>>> z = (1+2j)*3
>>> math.cos(z)

>>> import cmath     # import a module
>>> cmath.cos(z)
>>  dir(cmath)       # lists available methods

>>> cmath.polar(z)  # returns polar coordinates
                    # radius and angle
>>> rfi = cmath.polar(z)
>>> rfi[0]          # illustrates use of a TUPLE
>>> rfi[1]

>>> r, fi = cmath.polar(z) # ... is also fine
```

Stałe znakowe

```
>>> word = "Help" + "A"    # gluing together strings
>>> word

>>> "<" + word*5 + ">"

>>> word[0:2]              # SLICING
>>> "<" + word[0:2]*5 + ">"

>>> word[2:4]
>>> word[:2] + word[3:]
>>> len(word)
```

Skrypty

... commands can be collected in a file and run as a program

The Scientific Hello World script (hello.py) illustrates
— how to initialize a variable from the command line
— how to print a combination of numbers and plain text

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

import sys, math          # load system and math modules

r = float(sys.argv[1]) # extract the 1st command-line
                       # argument
s = math.sin(r)

print "Hello, World! sin(" + str(r) + ")=" + str(s)
```

Edytor

?? Does your preferred editor highlights syntax

??? Do you want to use IDLE

```
$ idle &
File -> New Window
2+2
Run -> Module  (or F5)
```

Formatowanie

```
r = 1.2; s = math.sin(r)
# format control via the printf-like syntax:
print "\t sin( %g ) = %12.5e \n" % (r,s)

More examples are listed below (try out 3).
    %g : a float in %f or %e notation
    %e : a float in scientific notation
%11.3f : a float in scientific notation
         with 3 decimals
         in a field of width 11 chars
   %5d : an integer in a field of width 5 chars
```

Komenda “while”

```
# Fibonacci series:
# the sum of two elements defines the next
a, b = 0, 1           # multiple assignment
while b < 10 :        # ! remember colon :
    print b           # ! indentation
    a, b = b, a+b

print "Done!"

-------------------------------------------------

while b < 1000:  # Same as before
   print b,      # ... but no newline
```

Listy oraz komenda “for”

```
# This is a list of strings:
a = ["cat", "window", "defenestrate"]
for x in a:               # iterates over list
   print x, len(x)    # string length

-------------------------------------------------------
b = ["Mary", "had", "a", "little", "lamb"]

print b[3]        # accessing list elements
print b[3][1:2]

for i in xrange(len(b)):    # iterates over indexes
    print i, b[i], len(b[i])

for x in b:       # iteration over list elements
    print b.index(x), x, len(x)
```

Funkcje

```
#Defining a function
def fib(n):
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
      print a,
      a, b = b, a+b

fib(2000) # Now call the function we have defined
```

Plan

Moduł “SciPy”

```
>>> from scipy import constants # import a sub-module
>>> help(constants)

>>> constants.c # speed of light
>>> constants.h # Plank's constant
>>> constants.N_A # Avogadro's number
>>> constants.physical_constants["electron mass"]
>>> constants.physical_constants

-------------------------------------------------------
>>> from scipy import special
>>> special.jn(2,0.345)    # Bessel function Jn_2(x)

>>> help(special)   # SciPy knows everything
```

Całkowanie

```
# SciPy has interfaces to the classical
#                QUADPACK Fortran library
from scipy import integrate

def myfunc(x) :
      return math.sin(x)

res, err = integrate.quad(myfunc, 0, math.pi)
print res, err
-------------------------------------------------
# parameters can be included
def yourfunc(x, a, b):
       return a + b*math.sin(x)
p, q=0, 1
res, err = integrate.quad(yourfunc, 0, math.pi,\
                          args=(p,q), epsabs=1.0e-9)
```

Tablice i macierze w NumPy

```
import numpy as np # everything, shorter to write
n = 5
A = np.zeros((n,n))
x = np.zeros(n)

for i in xrange(n):
    x[i] = i/2.0            # a given vector
    for j in xrange(n):     # ... and a given matrix
        A[i,j] = 2.0 + (i+1.0)/(j+i+1.0)

print x
print A
```

Algebra liniowa z NumPy

```
b = np.dot(A, x) # matrix-vector product b = A*x

# solve linear system A*y=b
y = np.linalg.solve(A, b)

# you should find y same as x
#           ... within numerical precision
```

Spróbować własnych sił

Zadanie 1:
Zaznajomić się z elektroniczną dokumentacją Pythona.
Napisać skrypt, który znajduje liczbę przypadkową z jendorodnego rozkładu losowego z przedziału [-1,1[.
Wydrukować tę liczbę z dokładnością 4 cyfr znaczących.
Standardowy moduł Pytona do generowania liczb przypadkowych nazywa sie random.

Zadanie 2:
Można rozszerzyć napisany skrypt tak, aby losował n liczb, gdzie n podawane jest z linii polecenia,
a następnie obliczał średnią i odchylenie standardowe.
Eleganckie rozwiązanie otrzymujemy stosując tablice i metody NumPy: np.sum, np.random