VPython - symulacje fizyczne z grafiką 3D dla każdego

wykład 3

Dr hab. Adam Bzdak

uwaga

remark.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/remark.py

File Edit Format Run Options Windows Help

```
L = [5,10,1,1]

res = 0
for i in range(len(L)):
    res += L[i]
print res

res = 0
for i in L:
    res += i
print res
```

tak jest lepiej

Ln: 13 Col: 0

```
>>>
17
17
>>>
```

Ln: 1 Col: 0

List comprehensions

L0.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/L0.py

File Edit Format Run Options Windows Help

```
L = []
for x in range(1,10):
    if x%2==0:
        L.append(x)
print L

L = [x for x in range(1, 10) if x%2==0]
print L
```

Ln: 10 Col: 0

1%2=1, 2%2=0,
3%2=1, 4%2=0

```
Type "copyright", "credits" or "license()" for more informatio
n.
>>> ========================= RESTART =========================
>>>
[2, 4, 6, 8]
[2, 4, 6, 8]
>>>
```

Ln: 1 Col: 0

List comprehensions

L1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\L1.py

File Edit Format Run Options Windows Help

```
L1 =  [x for x in range(1, 10) if x%2]
print L1

L2 = [x for x in range(1, 10) if x%2==0]
print L2

L3 = [x if x%2==0 else 'hi' for x in range(1, 10)]
print L3
```

Ln: 10 Col: 0

1%2=1, 2%2=0,
3%2=1, 4%2=0

```
>>>
[1, 3, 5, 7, 9]
[2, 4, 6, 8]
['hi', 2, 'hi', 4, 'hi', 6, 'hi', 8, 'hi']
>>>
```

Ln: 1 Col: 0

x **if** x%2==0 **else** ‘hi’ - zwraca x jeśli (x%2==0) jest prawdą,
w przciwnym razie zwraca ‘hi’

List comprehensions

L2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\L2.py

File Edit Format Run Options Windows Help

```
words = 'Bob is happy'
words = words.split()
print words, '\n'

L = [[len(w), w.upper(), w.lower()] for w in words]

for i in L:
    print i
```

Ln: 10 Col: 0

```
>>>
['Bob', 'is', 'happy']

[3, 'BOB', 'bob']
[2, 'IS', 'is']
[5, 'HAPPY', 'happy']
>>>
```

Ln: 1 Col: 0

Moduły

mymodule.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/mymodule.py

File Edit Format Run Options Windows Help

```
def func(x,y):
    if x>y:
        return x
    else:
        return y

a = 3.14
```

Ln: 9 Col: 0

Zapisujemy *mymodule.py* w jakimś egzotycznym miejscu, np. na pulpicie

sys.path

main.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\main.py

File Edit Format Run Options Windows Help

```
import sys
sys.path.append('C:\Users\Wysokie Energie\Desktop')
import mymodule

#for i in sys.path:
#     print i

print mymodule.func(3,4)
print mymodule.a
```

dodaje nową ścieżkę

Ln: 12 Col: 0

```
>>>
4
3.14
>>>
```

Ln: 1 Col: 0

__name__

mymodule2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\mymodule2.py

File Edit Format Run Options Windows Help

```
def fun(x,y):
    return x*y

print __name__

if __name__ == '__main__':
    print fun(2,3)
```

Ln: 10 Col: 0

```
>>>
__main__
6
>>>
```

Ln: 1 Col: 0

Jeśli to jest nasz główny program to **__name__ = '__main__'**

Co jeśli importujemy ten moduł (*mymodule2.py*)?

__name__

main2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\main2.py

File Edit Format Run Options Windows Help

```
import mymodule2 as mm

print '-'*20

print __name__

print mm.fun(10,3)
```

Ln: 10 Col: 0

```
>>>
mymodule2
--------------------
__main__
30
>>>
```

Ln: 1 Col: 0

teraz __name__ z modułu = ‘mymodule2’
ale w głównym programie __name__ = ‘__main__’

Moduł time

time.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\time.py

File Edit Format Run Options Windows Help

```
import time

start = time.clock()

i = 10**8
while i:
    i = i - 1

t = time.clock() - start

print round(t, 4), 'seconds'
```

Ln: 13 Col: 0

```
>>>
11.9727 seconds
>>>
```

Ln: 1 Col: 0

Enumerate

enumerate.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\enumerate.py

File Edit Format Run Options Windows Help

```
L = ['a', 'b', 'c', 'd']

for i in range(len(L)):
    print i, L[i]

print 5*'-'

for i, ele in enumerate(L):
    print i, ele
```

mało eleganckie

lepiej

Ln: 11 Col: 0

```
>>>
0 a
1 b
2 c
3 d
-----
0 a
1 b
2 c
3 d
>>>
```

proszę sprawdzić np. enumerate(L, 5)

Ln: 1 Col: 0

any, all

all_any.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\all_any.py

File Edit Format Run Options Windows Help

```
print [x<0 for x in range(5)]

print any(x<0 for x in range(5))
print any(x%2==0 for x in range(5))

print ''

print all(x>0 for x in range(1,10))
print all(x>0 for x in range(0,10))
```

Ln: 12 Col: 0

```
[False, False, False, False, False]
False
True

True
False
```

Ln: 1 Col: 0

any zwraca True jeśli przynajmniej jeden element jest True

all zwraca True jeśli wszystkie elementy są True

min, max

min_max.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\min_max.py

File Edit Format Run Options Windows Help

```
L = [1,2,10,100]

print 'minimum of L is', min(L)
print 'maximum of L is', max(L)
```

Ln: 8 Col: 0

```
>>>
minimum of L is 1
maximum of L is 100
>>>
```

Ln: 1 Col: 0

eval()

eval1.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/eval1.py

File Edit Format Run Options Windows Help

```
import math as m

x = 1
print eval('x+1')

x = 3.1415
print eval('m.sin(x)')
```

Ln: 11 Col: 0

eval() interpretuje string jako kod

```
>>>
2
9.26535896605e-05
>>>
```

Ln: 1 Col: 0

eval() może być niebezpieczne:

https://stackoverflow.com/questions/9383740/what-does-pythons-eval-do

eval()

eval2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\eval2.py

File Edit Format Run Options Windows Help

```
import math as m

f = raw_input('Type a function: ')
print f, '->', type(f)

x = 2*m.pi
print eval(f)
```

m.pi = pi

Ln: 11 Col: 0

```
>>>
Type a function: m.cos(x)
m.cos(x) -> <type 'str'>
1.0
>>>
```

Ln: 1 Col: 0

and, or

and_or.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\and_or.py

File Edit Format Run Options Windows Help

```
for i in range(1000):
    if (i<=4 or i==100 or i==500):
        print i

for i in range(1, 100):
    if (i>0 and i<10):
        print i,
```

przecinek jest do druku poziomego

print(i, end=" ") w Python 3

Ln: 10 Col: 0

```
>>>
0
1
2
3
4
100
500
1 2 3 4 5 6 7 8 9
```

Ln: 1 Col: 0

False

false.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/false.py

File Edit Format Run Options Windows Help

```
L = [None, False, 0, '', [], 'abc', '0', 0.1]

for ele in L:
    if ele:
        print ele
```

Ln: 8 Col: 0

```
>>>
abc
0
0.1
>>>
```

Ln: 1 Col: 1

zip

zip.py - C:\Users\Wysokie Energie\Desktop\VPython_17_pl\lec_3\zip.py (2.7.12)

File Edit Format Run Options Window Help

```
L1 = [1,2,3]
L2 = [10,20,30]
L3 = [100,200,300]

Z = zip(L1,L2,L3)
print Z

for i,j,k in Z:
    print i,j,k
```

Ln: 13 Col: 0

print(list(Z)) w Python 3

```
[(1, 10, 100), (2, 20, 200), (3, 30, 300)]
1 10 100
2 20 200
3 30 300
>>>
```

Ln: 1 Col: 0

(a,b,c) - krotka (tuple)

Na kolejnych 4 stronach, dyskutuje funkcje, które nie są uważane zbyt "Pythonic“

map

map1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\map1.py

File Edit Format Run Options Windows Help

```
def f(x):
    return x**2 + 0.1

L = [1, 2, 3]

print map(f, L)
```

list(map(f, L)) w Python 3

Ln: 8 Col: 0

```
>>>
[1.1, 4.1, 9.1]
>>>
```

Ln: 1 Col: 0

map działa f(x) na każdy element L

to samo z [f(x) for x in L]

map

map2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\map2.py

File Edit Format Run Options Windows Help

```
def func(x,y):
    return x*y

L = [1, 2, 3]
G = [10, 20, 30]

print map(func, L, G)
```

Ln: 10 Col: 0

```
>>>
[10, 40, 90]
>>>
```

Ln: 1 Col: 0

map działa func(x,y) na pierwsze elementy L i G, potem na drugie itd.

lambda (inline functions)

lambda.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\lambda.py

File Edit Format Run Options Windows Help

```
f1 = lambda x: x**2

f2 = lambda x,y: x*y

print f1(10)
print f2(5,5)

# print map((lambda x, y: x**y), [2, 5], [3, 2])
```

Ln: 10 Col: 0

mało czytelne
kwestia gustu

```
>>>
100
25
>>>
```

Ln: 1 Col: 0

[Python-Dev] Let's just *k

https://mail.python.org/pipermail/python-dev/2006-February/060415.html

[Python-Dev] Let's just *keep* lambda

Guido van Rossum guido at python.org
Sun Feb 5 18:43:28 CET 2006

- Previous message: [Python-Dev] math.areclose ...?
- Next message: [Python-Dev] Let's just *keep* lambda
- **Messages sorted by:** [date] [thread] [subject] [author]

```
After so many attempts to come up with an alternative for lambda,
perhaps we should admit defeat. I've not had the time to follow the
most recent rounds, but I propose that we keep lambda, so as to stop
wasting everybody's talent and time on an impossible quest.

--
--Guido van Rossum (home page: http://www.python.org/~guido/)
```

filter

filter.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\filter.py

File Edit Format Run Options Windows Help

```
f1 = lambda x: x%3==0    # True or False
L = range(10)

print filter(f1, L)

f2 = lambda x: 'a' in x
G = ['dog', 'cat']

print filter(f2, G)
```

filter(f1, L) zwraca listę z tych elementów L dla których funkcja f1 zwraca True

list(filter(f2, G)) w Python 3

Ln: 11 Col: 0

```
>>>
[0, 3, 6, 9]
['cat']
>>>
```

Ln: 1 Col: 0

To samo z: [x for x in L if x%3==0]

[x for x in G if 'a' in x]

i taka metoda jest zalecana