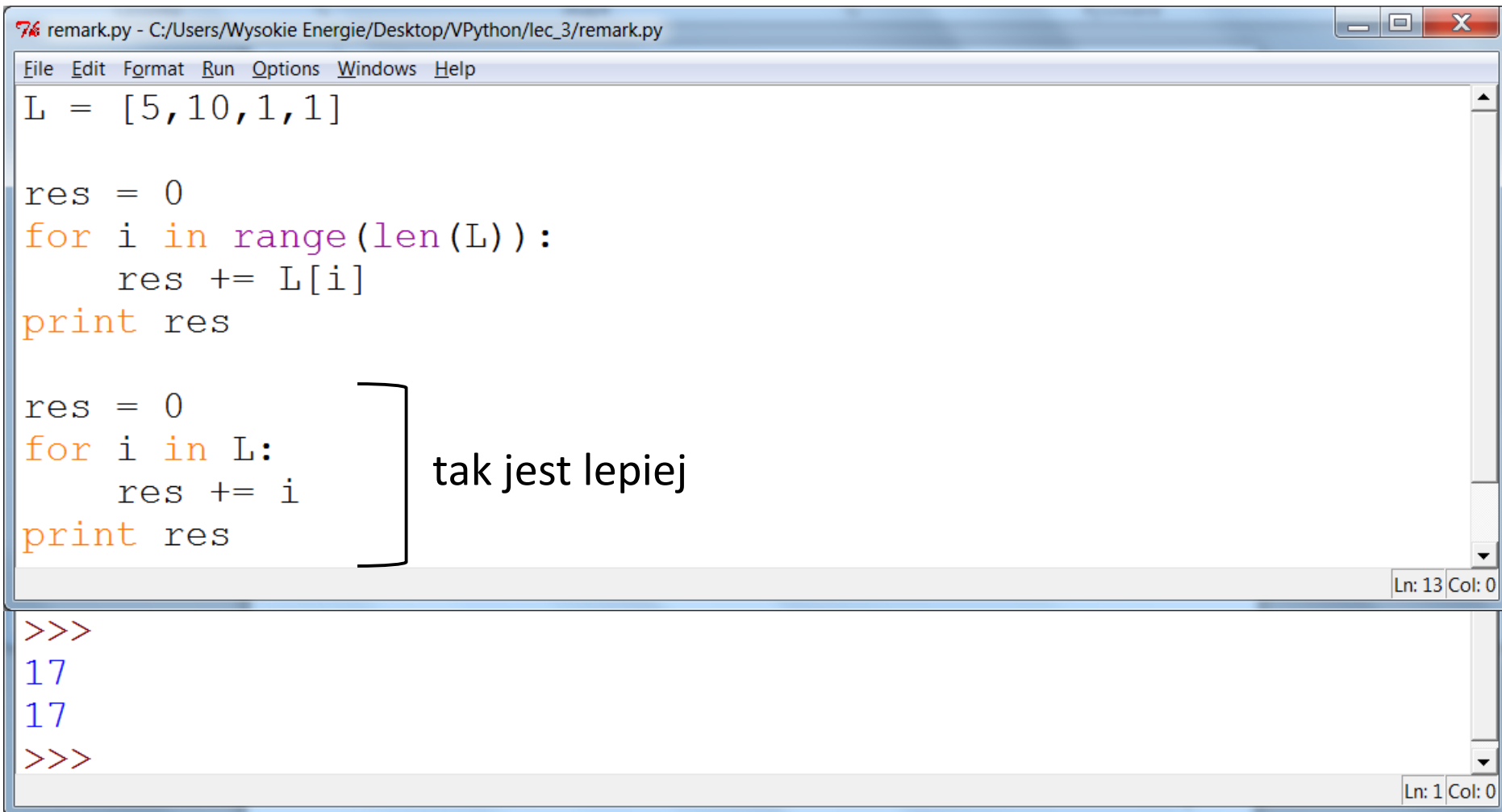


VPython - symulacje fizyczne z grafiką 3D dla każdego

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

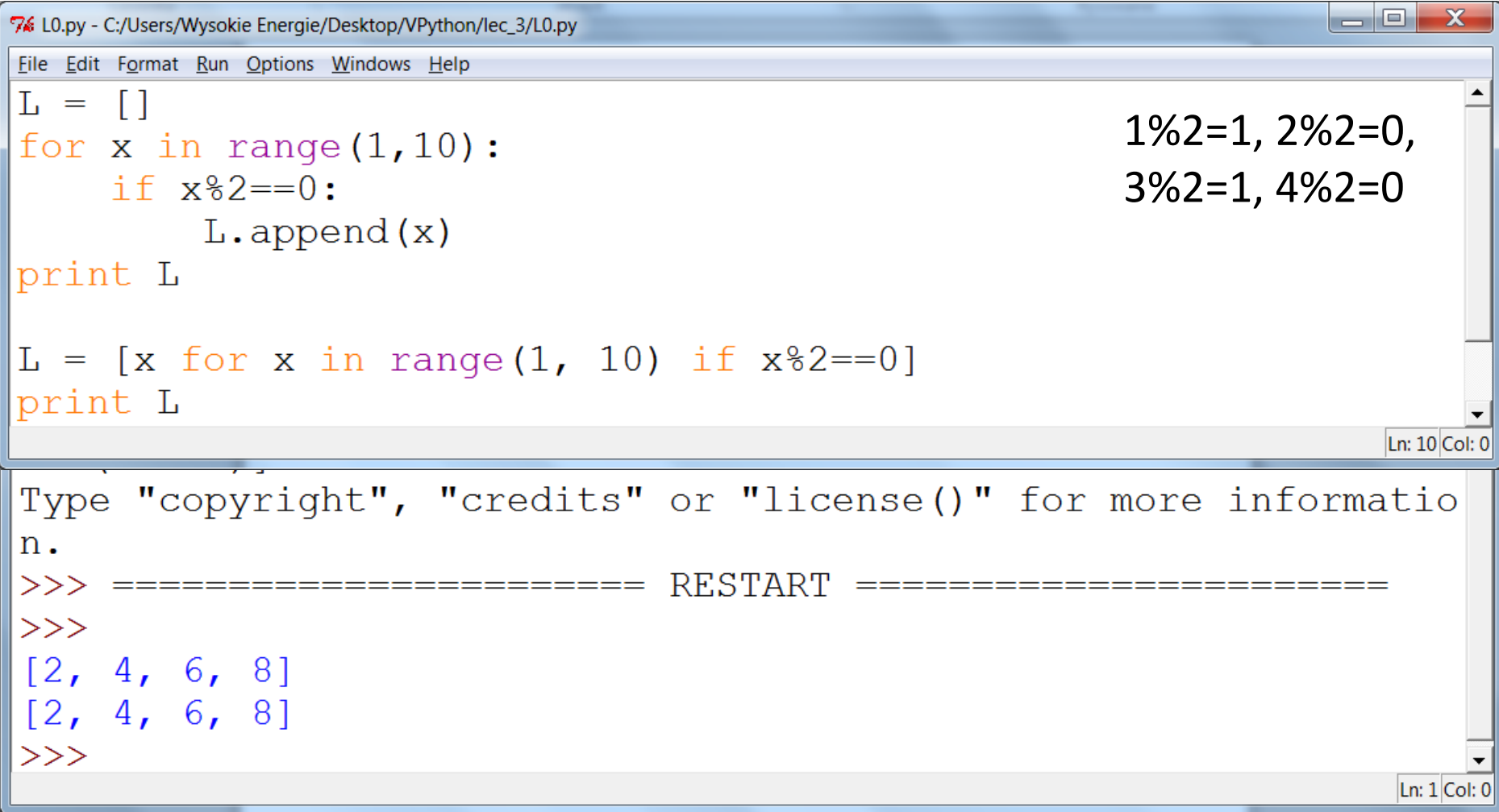
]>>>
17
17
>>>
```

tak jest lepiej

Ln: 13 Col: 0

Ln: 1 Col: 0

List comprehensions



The image shows a screenshot of a Python IDE window titled "L0.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/L0.py". The window contains two parts: a code editor and a console output area.

Code Editor:

```
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
```

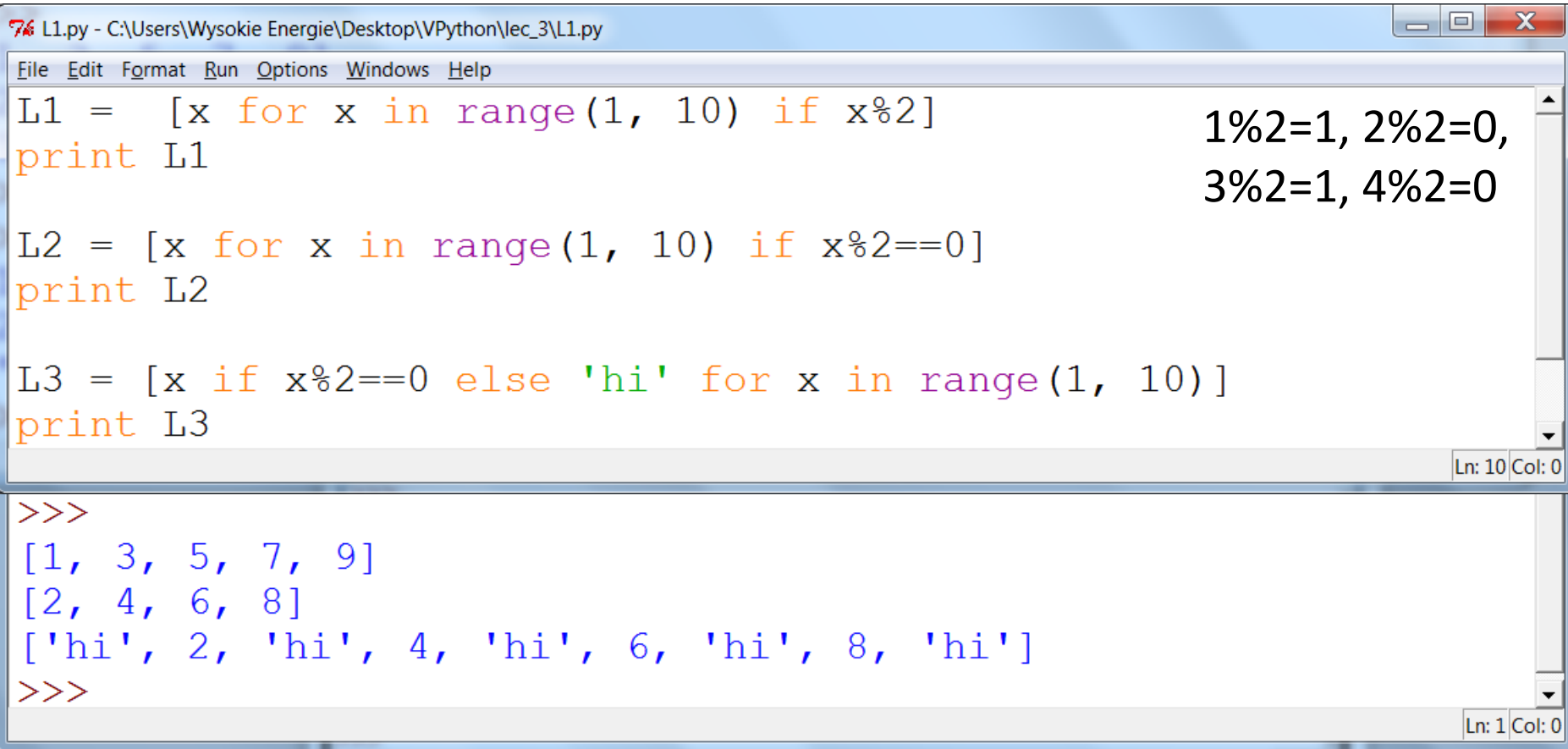
Console Output:

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

Annotations:

- Next to the first code block, the text "1%2=1, 2%2=0, 3%2=1, 4%2=0" is displayed, illustrating the modulo operation results for the first four numbers in the range.
- The status bar at the bottom right of the code editor shows "Ln: 10 Col: 0".
- The status bar at the bottom right of the console shows "Ln: 1 Col: 0".

List comprehensions



The screenshot shows a Python IDE window titled '76 L1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\L1.py'. The code defines three lists: L1 (odd numbers 1-9), L2 (even numbers 2-8), and L3 (a list where even numbers are kept and odd numbers are replaced by 'hi'). The output of the code is shown in the console below the editor.

```
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
```

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']
>>>
```

x if x%2==0 else 'hi' - zwraca x jeśli (x%2==0) jest prawdą,
w przeciwnym razie zwraca 'hi'

List comprehensions

74 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']
```

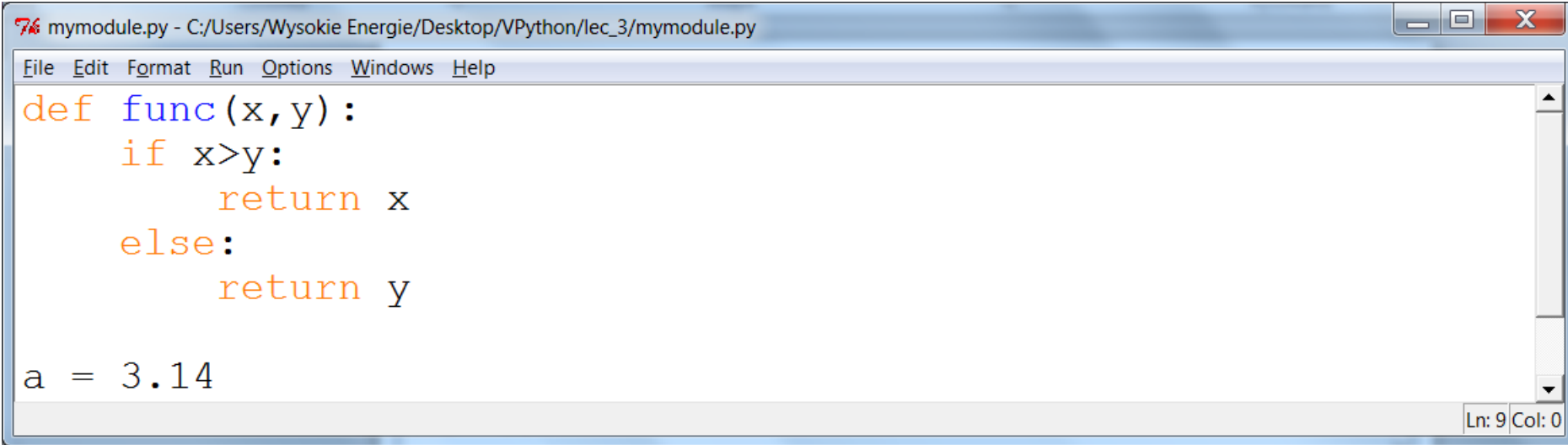
```
[2, 'IS', 'is']
```

```
[5, 'HAPPY', 'happy']
```

```
>>>
```

Ln: 1 Col: 0

Moduły

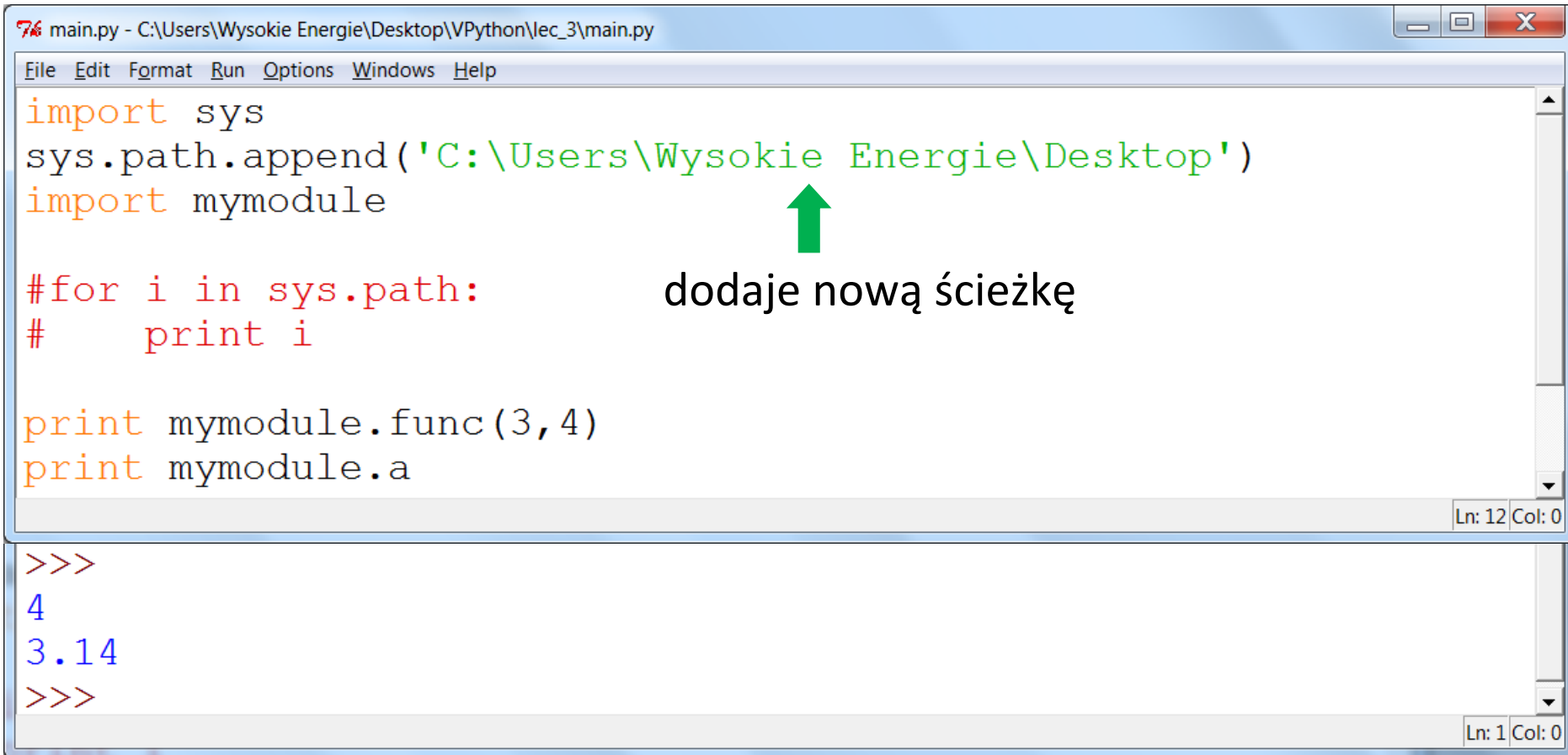
A screenshot of a Python IDE window titled "mymodule.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/mymodule.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main text area contains the following Python code:

```
def func(x, y):  
    if x > y:  
        return x  
    else:  
        return y  
  
a = 3.14
```

The status bar at the bottom right shows "Ln: 9 Col: 0".

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

sys.path



The image shows a screenshot of a Python IDE window titled "main.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\main.py". The window contains a Python script with the following code:

```
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
```

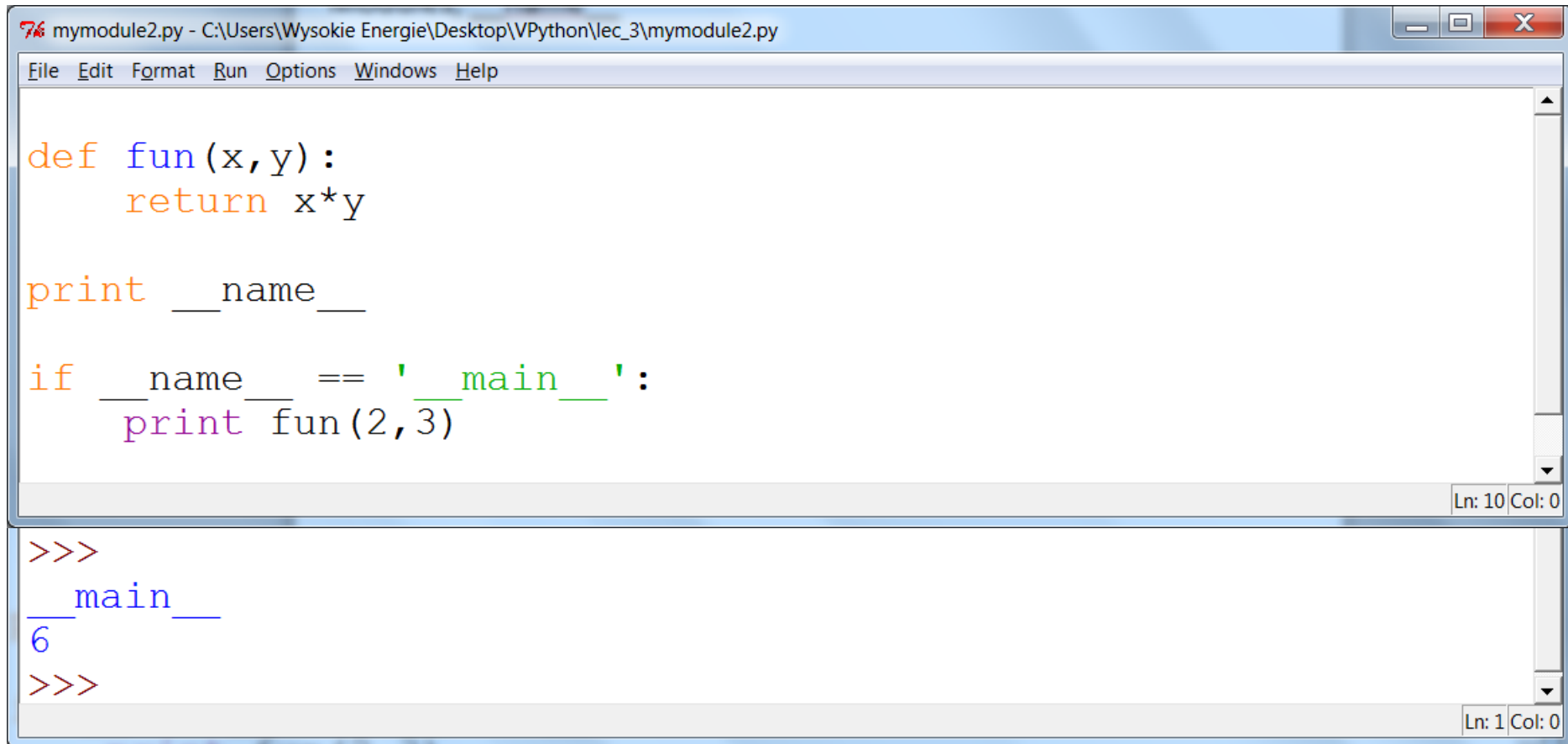
A green arrow points to the string 'C:\Users\Wysokie Energie\Desktop' in the `sys.path.append()` call, with the text "dodaje nową ścieżkę" (adds a new path) written next to it.

Below the script editor is a console window showing the output of the script:

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

The console window status bar shows "Ln: 1 Col: 0".

__name__



The screenshot shows a Python IDE window titled 'mymodule2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\mymodule2.py'. The editor contains the following Python code:

```
def fun(x, y):  
    return x*y  
  
print __name__  
  
if __name__ == '__main__':  
    print fun(2, 3)
```

The status bar at the bottom right of the editor indicates 'Ln: 10 Col: 0'. Below the editor is a console window showing the execution output:

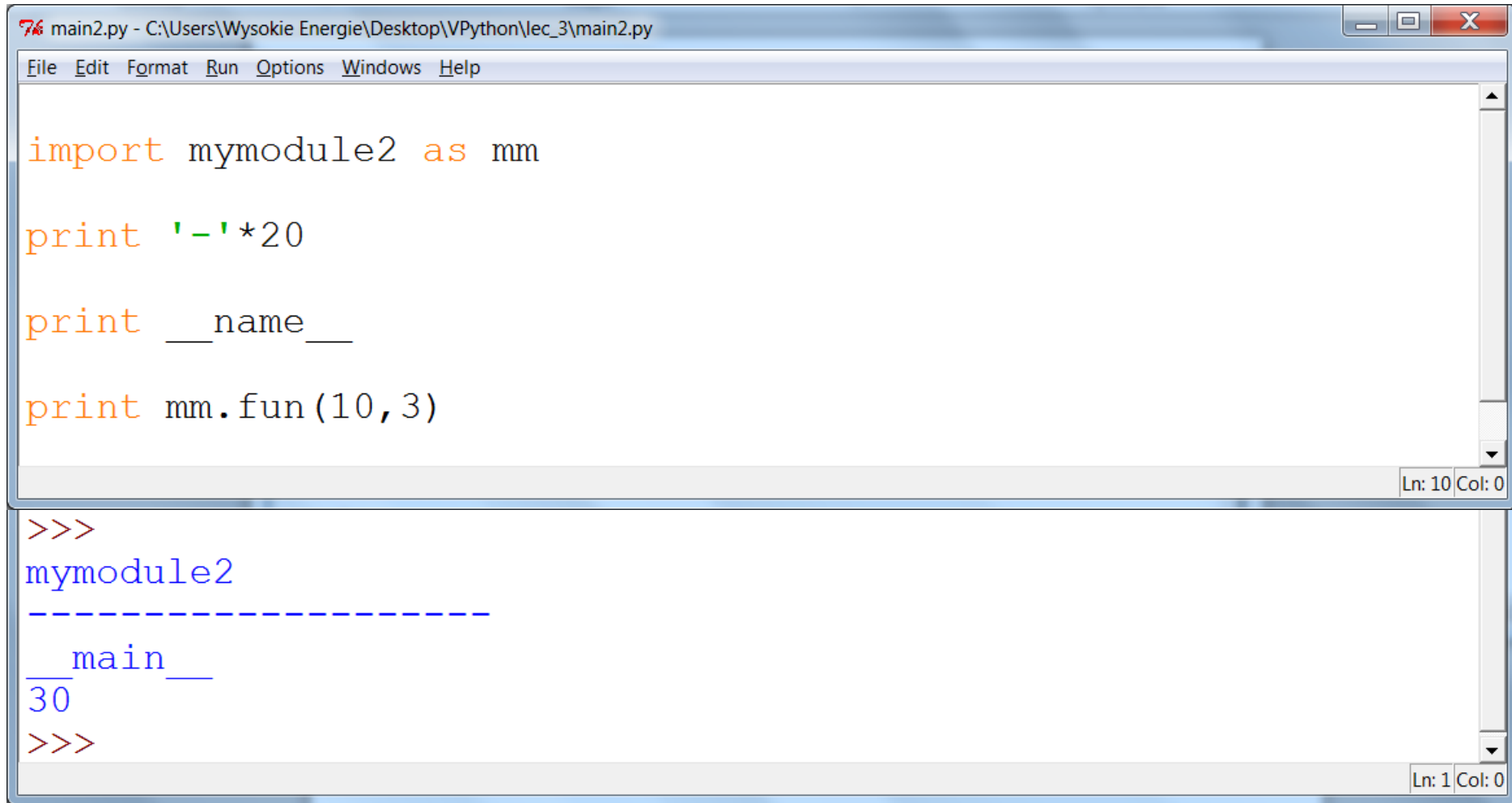
```
>>>  
__main__  
6  
>>>
```

The console status bar at the bottom right indicates 'Ln: 1 Col: 0'.

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

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

__name__



The screenshot shows a Python IDE window titled 'main2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\main2.py'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'. The main text area contains the following Python code:

```
import mymodule2 as mm

print '-'*20

print __name__

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

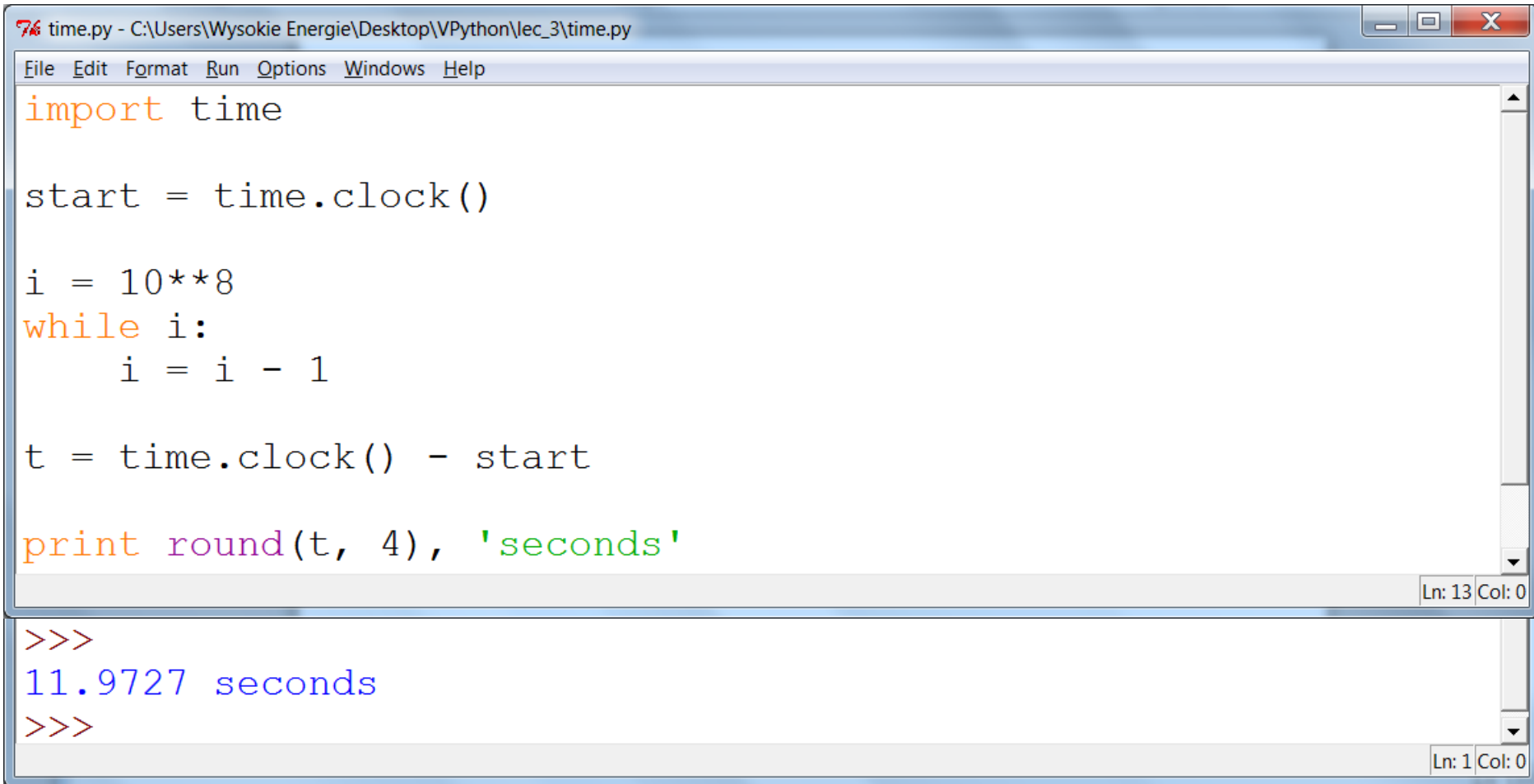
The status bar at the bottom right of the text area shows 'Ln: 10 Col: 0'. Below the text area is a console window showing the output of the script:

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

The status bar at the bottom right of the console window shows 'Ln: 1 Col: 0'.

teraz __name__ z modułu = 'mymodule2'
ale w głównym programie __name__ = '__main__'

Moduł time



The image shows a screenshot of a Python IDE window titled "time.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\time.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main text area contains the following Python code:

```
import time

start = time.clock()

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

t = time.clock() - start

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

The status bar at the bottom right of the text area indicates "Ln: 13 Col: 0". Below the text area is a console window showing the execution output:

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

The status bar at the bottom right of the console window indicates "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)):  ← mało eleganckie
    print i, L[i]

print 5*'- '

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

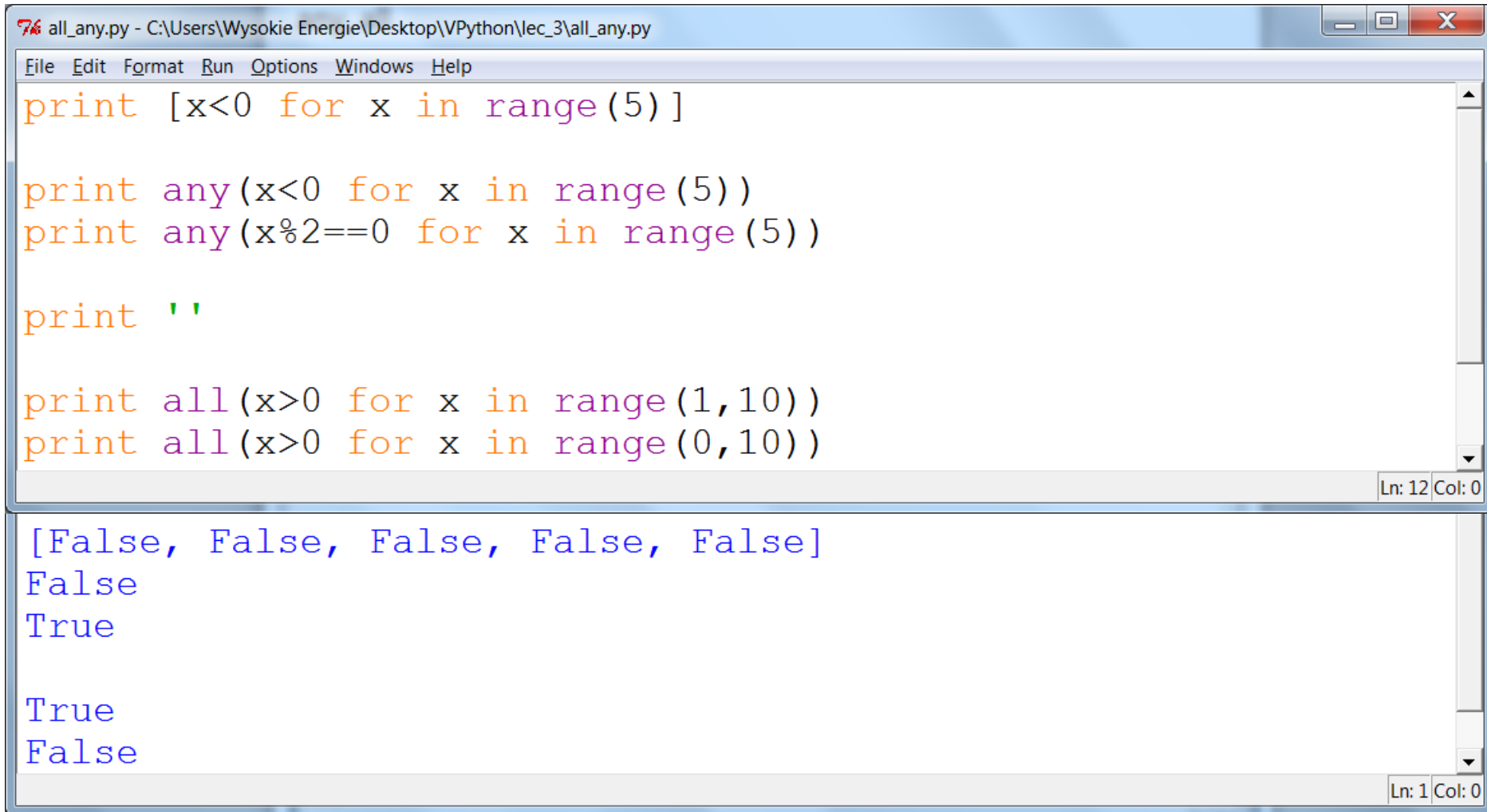
Ln: 11 Col: 0

```
>>>
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



The screenshot shows a Python IDE window titled 'all_any.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\all_any.py'. The window has a menu bar with 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'. The code editor contains the following Python code:

```
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))
```

The output window at the bottom shows the results of the code execution:

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

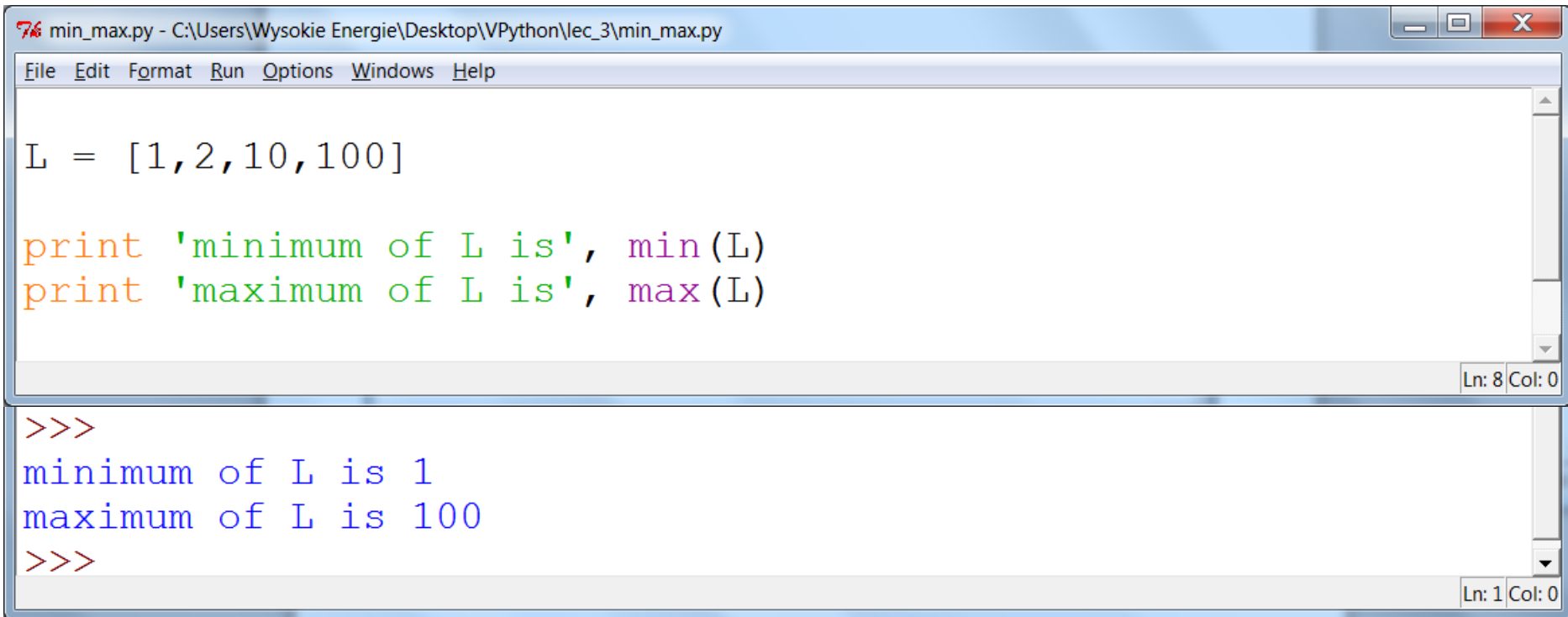
True
False
```

The status bar at the bottom right of the code editor shows 'Ln: 12 Col: 0', and the status bar at the bottom right of the output window shows 'Ln: 1 Col: 0'.

any zwraca True jeśli przynajmniej jeden element jest True

all zwraca True jeśli wszystkie elementy są True

min, max



The image shows a screenshot of a Python IDE window titled "min_max.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\min_max.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main editor area contains the following Python code:

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

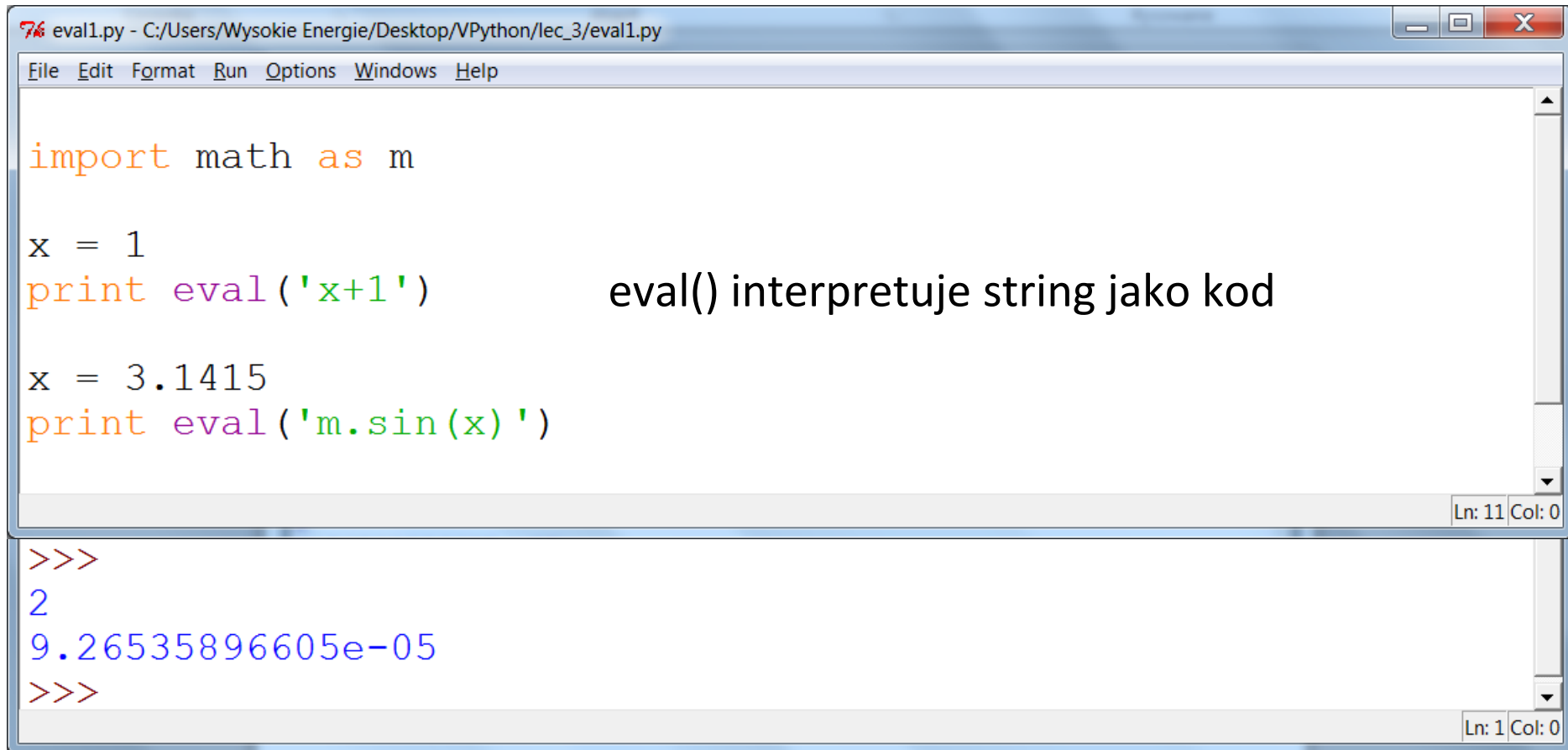
print 'minimum of L is', min(L)
print 'maximum of L is', max(L)
```

The status bar at the bottom right of the editor shows "Ln: 8 Col: 0". Below the editor is a console window showing the output of the script:

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

The status bar at the bottom right of the console shows "Ln: 1 Col: 0".

eval()



The screenshot shows a Python IDE window titled 'eval1.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/eval1.py'. The code in the editor is as follows:

```
import math as m

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

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

To the right of the code, the text "eval() interpretuje string jako kod" is displayed. Below the editor is a console window showing the execution output:

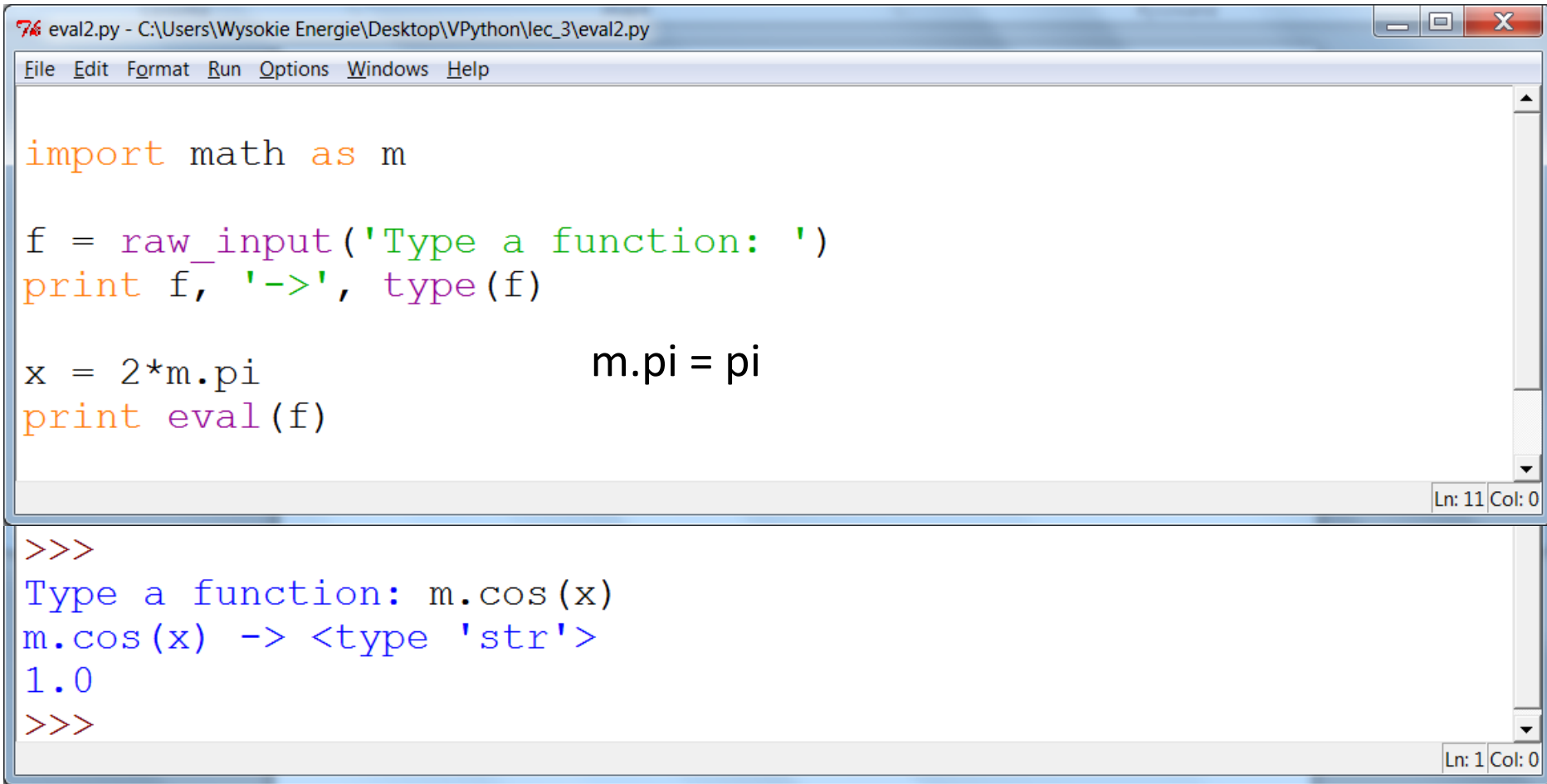
```
>>>
2
9.26535896605e-05
>>>
```

The console status bar shows 'Ln: 1 Col: 0'.

eval() może być niebezpieczne:

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

eval()



The image shows a screenshot of a Python IDE window titled "eval2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\eval2.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main editor area contains the following Python code:

```
import math as m

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

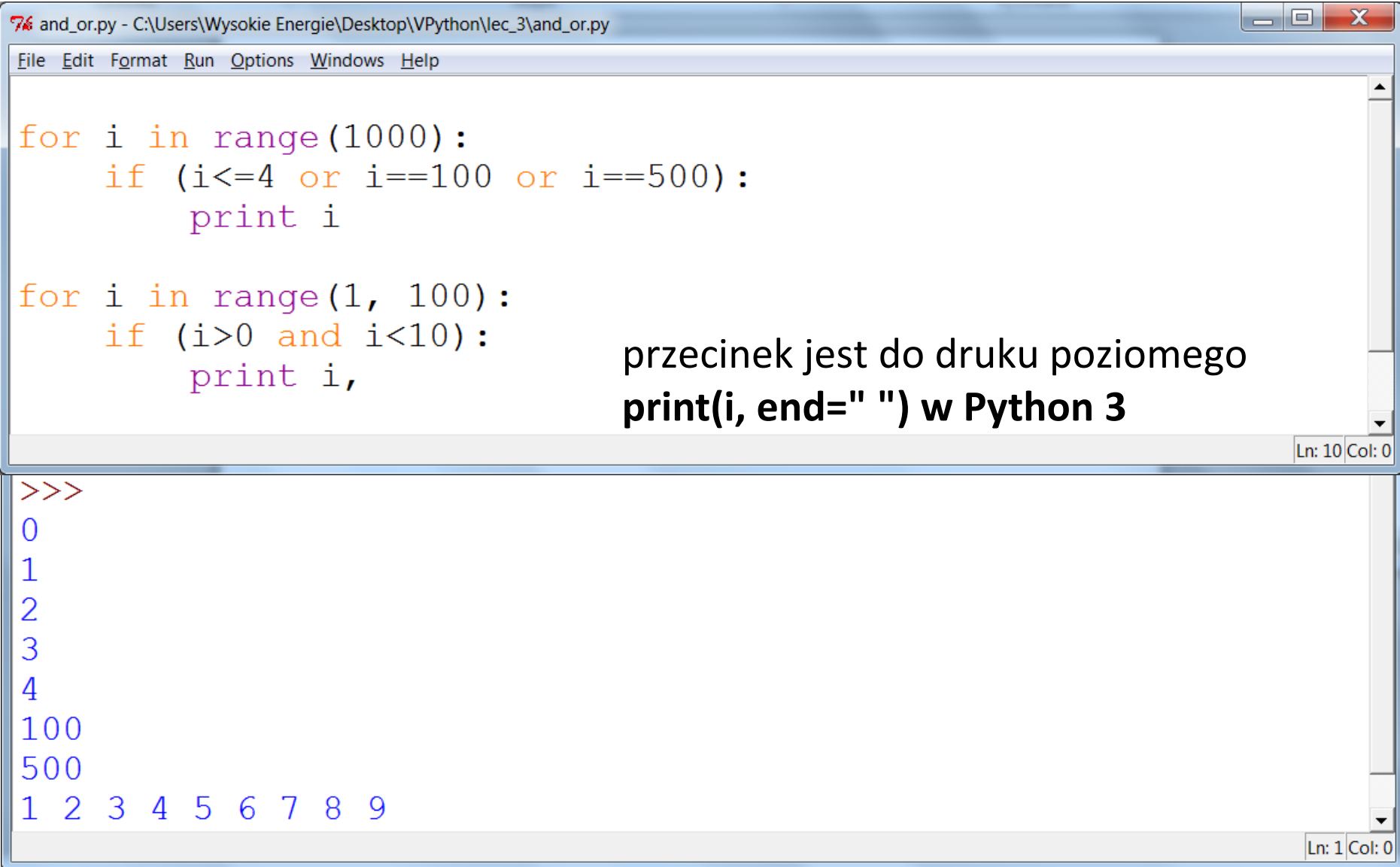
x = 2*m.pi
print eval(f)
```

Below the code editor is a console window showing the execution output:

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

The status bar at the bottom right of the console window indicates "Ln: 1 Col: 0".

and, or



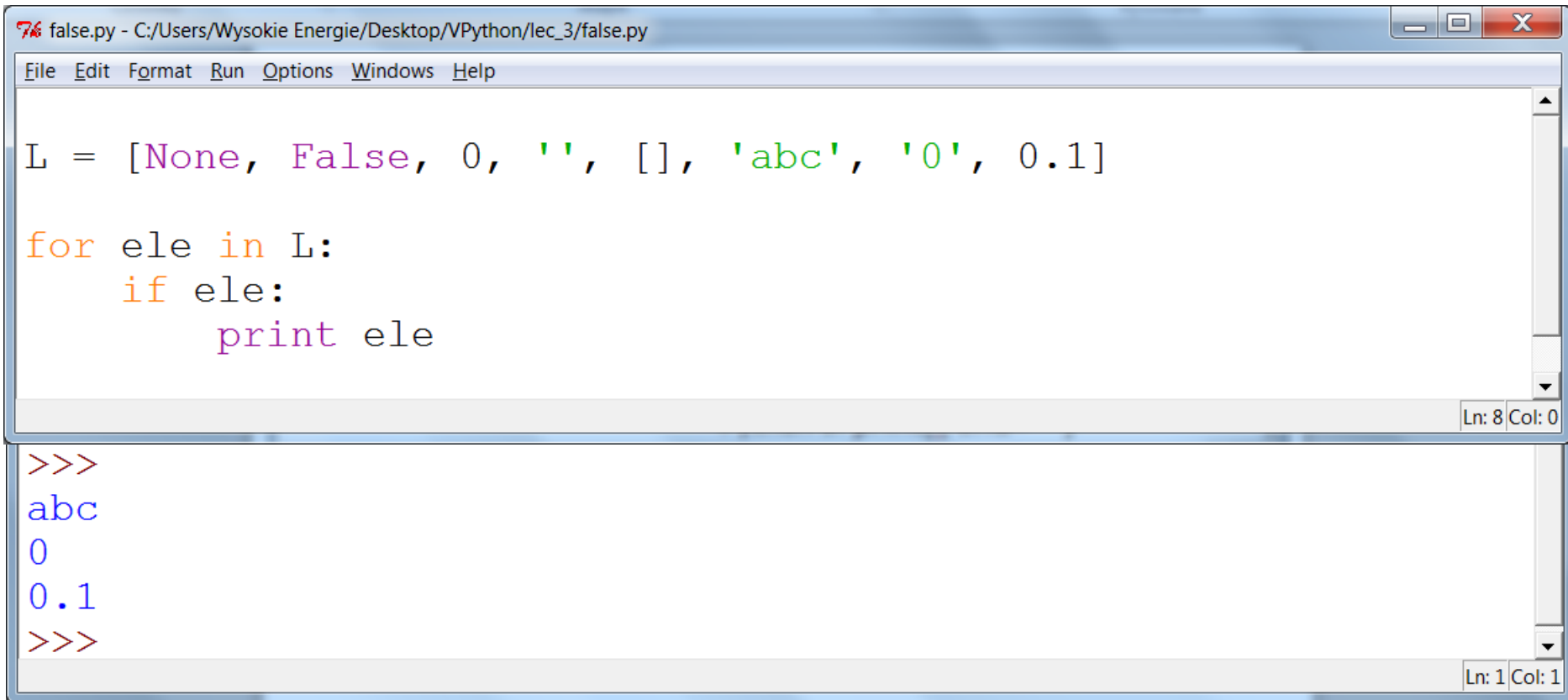
The image shows a screenshot of a Python IDE window titled "and_or.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\and_or.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main text area contains the following Python code:

```
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,
```

To the right of the code, there is a text annotation in black font: "przecinek jest do druku poziomego print(i, end=" ") w Python 3".

Below the code editor is a console window showing the output of the program. It starts with three red prompt characters ">>>". The output consists of the numbers 0, 1, 2, 3, 4, 100, and 500, each on a new line. The final line of output is "1 2 3 4 5 6 7 8 9", where the numbers are separated by spaces. The console window has a status bar at the bottom right showing "Ln: 1 Col: 0".

False



The image shows a screenshot of a Python IDE window titled "false.py - C:/Users/Wysokie Energie/Desktop/VPython/lec_3/false.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main text area contains the following Python code:

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

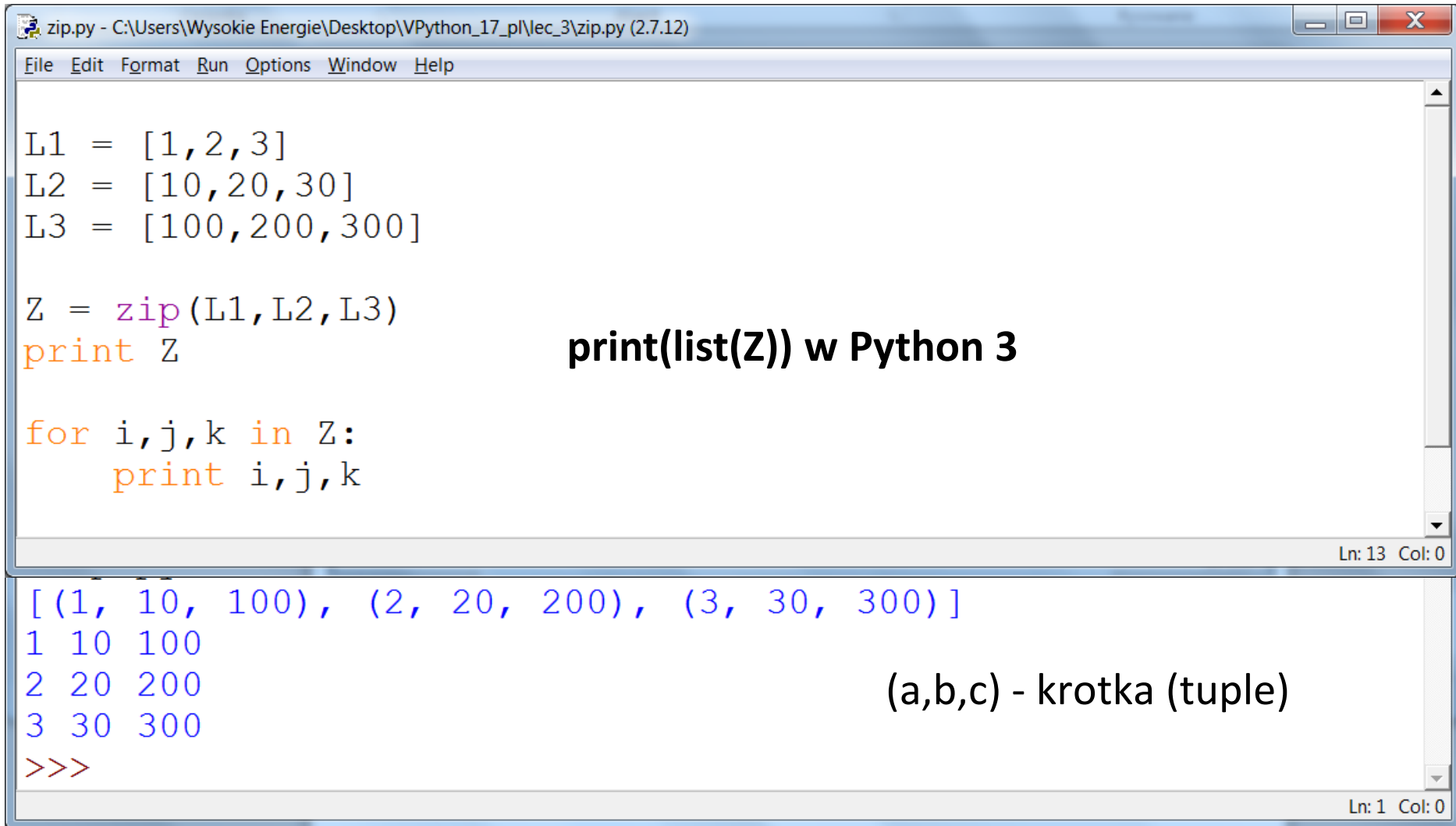
for ele in L:
    if ele:
        print ele
```

The status bar at the bottom right of the text area shows "Ln: 8 Col: 0". Below the text area is a console window showing the output of the script:

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

The status bar at the bottom right of the console window shows "Ln: 1 Col: 1".

zip



The image shows a screenshot of a Python IDE window titled "zip.py - C:\Users\Wysokie Energie\Desktop\VPython_17_pl\lec_3\zip.py (2.7.12)". The window contains Python code that demonstrates the use of the `zip` function. The code defines three lists: `L1 = [1, 2, 3]`, `L2 = [10, 20, 30]`, and `L3 = [100, 200, 300]`. It then creates a zip object `Z = zip(L1, L2, L3)` and prints it using `print Z`. A `for` loop iterates over `Z`, printing each element: `for i, j, k in Z: print i, j, k`. The output of the program is shown in a separate window, displaying the list of tuples `[(1, 10, 100), (2, 20, 200), (3, 30, 300)]` and the individual elements of each tuple on separate lines: `1 10 100`, `2 20 200`, and `3 30 300`. The prompt `>>>` is also visible. To the right of the code, the text `print(list(Z)) w Python 3` is displayed. To the right of the output, the text `(a,b,c) - krotka (tuple)` is displayed. The status bar at the bottom right of the IDE window shows "Ln: 13 Col: 0" and "Ln: 1 Col: 0".

```
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`

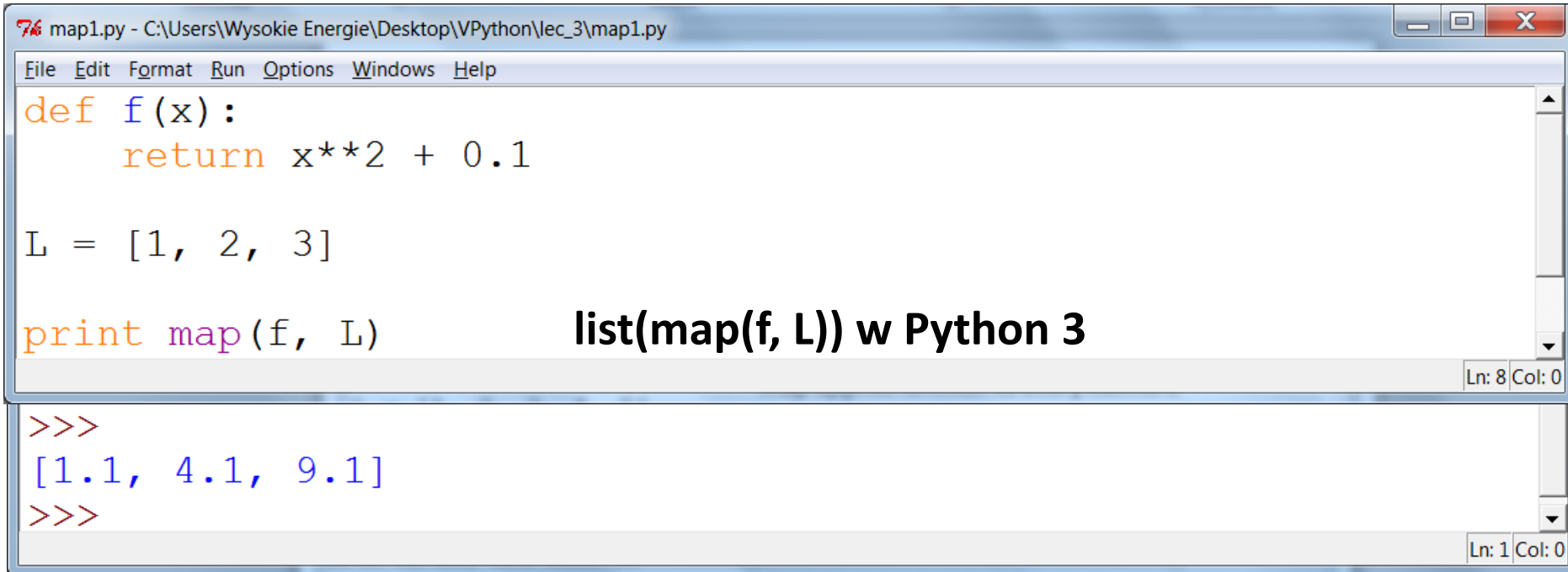
```
[(1, 10, 100), (2, 20, 200), (3, 30, 300)]
1 10 100
2 20 200
3 30 300
>>>

Ln: 1 Col: 0
```

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

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

map



The screenshot shows a Python IDE window titled "map1.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\map1.py". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The script content is as follows:

```
def f(x):  
    return x**2 + 0.1  
  
L = [1, 2, 3]  
  
print map(f, L)
```

Below the script, the execution output is shown:

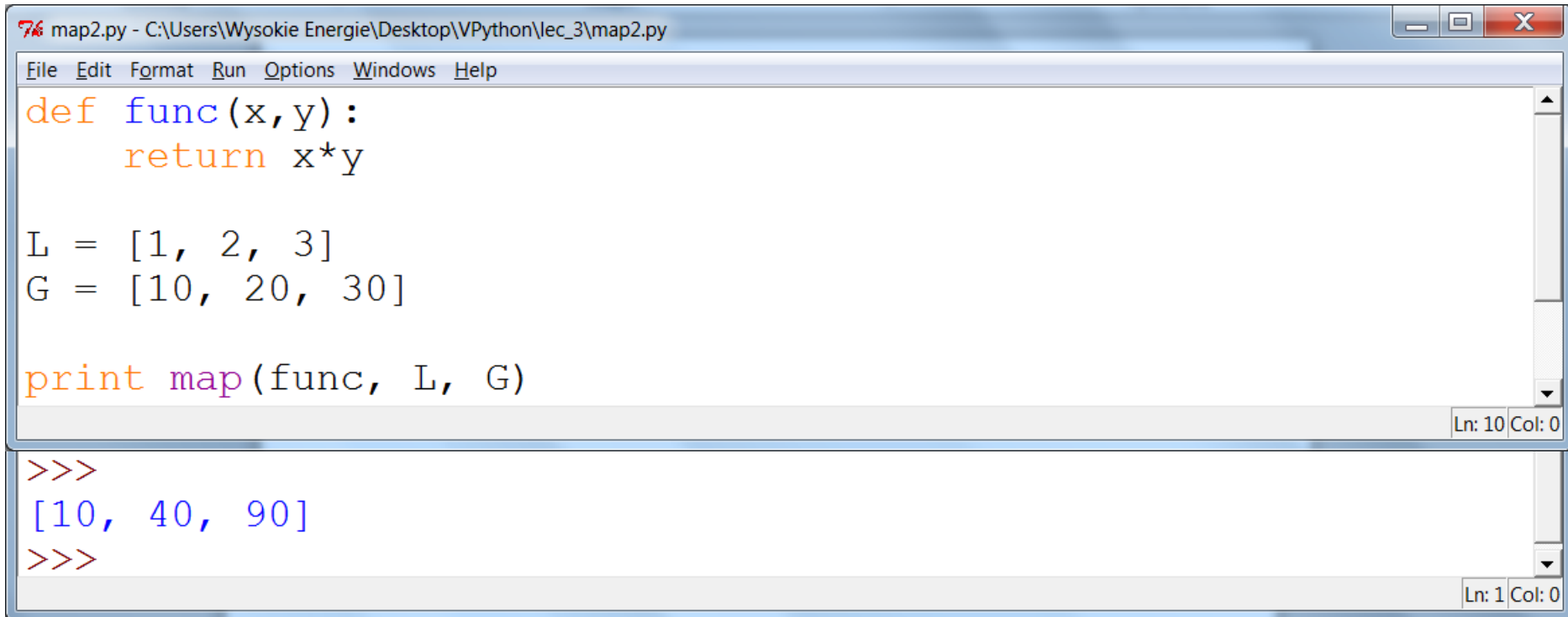
```
>>>  
[1.1, 4.1, 9.1]  
>>>
```

On the right side of the IDE, the text **list(map(f, L)) w Python 3** is displayed. The status bar at the bottom right of the script editor shows "Ln: 8 Col: 0", and the status bar at the bottom right of the output window shows "Ln: 1 Col: 0".

map działa $f(x)$ na każdy element L

to samo z $[f(x) \text{ for } x \text{ in } L]$

map



The image shows a screenshot of a Python IDE window titled "map2.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\map2.py". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The main editor area contains the following Python code:

```
def func(x, y):  
    return x*y  
  
L = [1, 2, 3]  
G = [10, 20, 30]  
  
print map(func, L, G)
```

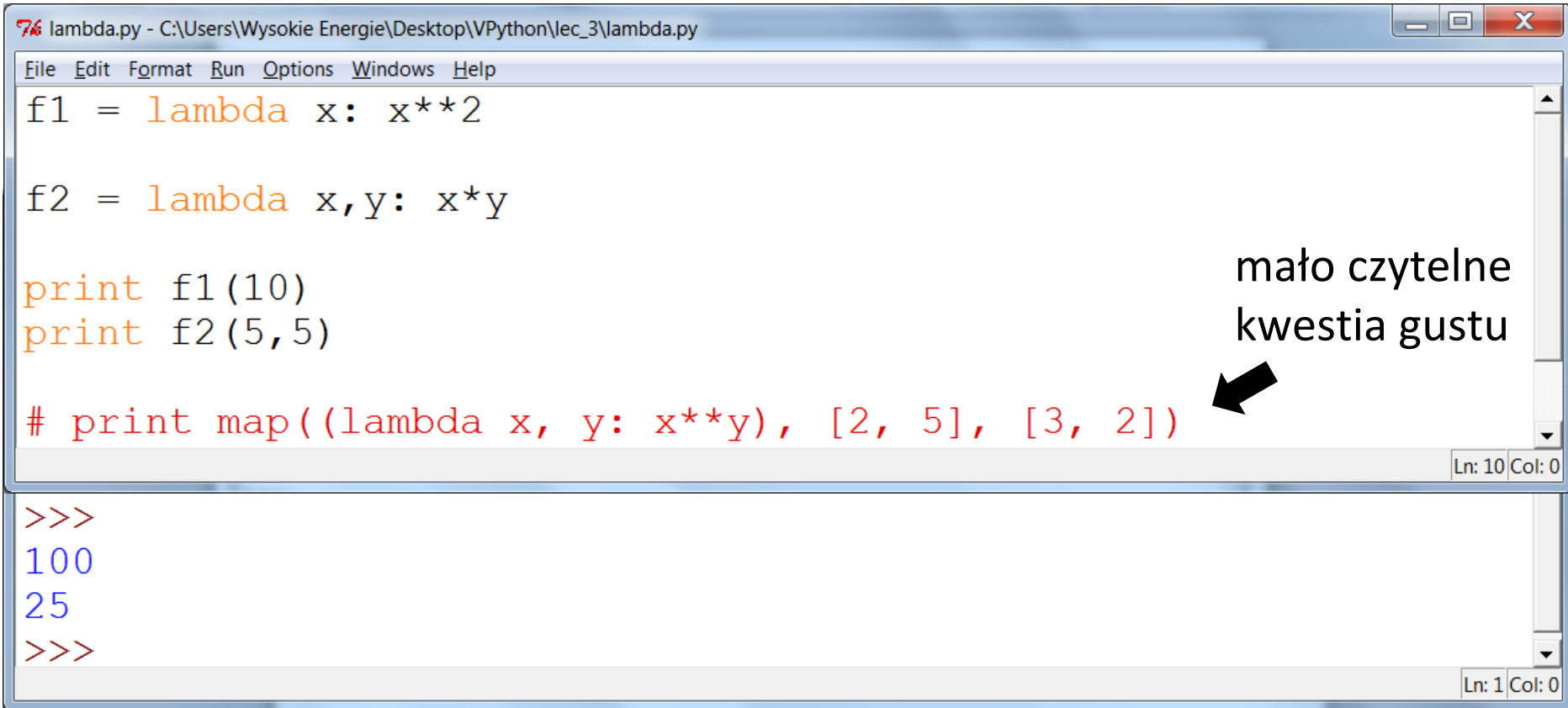
The status bar at the bottom right of the editor shows "Ln: 10 Col: 0". Below the editor is a console window showing the execution output:

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

The status bar at the bottom right of the console shows "Ln: 1 Col: 0".

map działa $\text{func}(x,y)$ na pierwsze elementy L i G, potem na drugie itd.

lambda (inline functions)



The screenshot shows a Python IDE window titled 'lambda.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\lambda.py'. The editor contains the following code:

```
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])
```

The code is color-coded: 'lambda' is orange, 'x' and 'y' are black, and 'x**2' and 'x*y' are black. The comment line is red. The IDE status bar at the bottom right shows 'Ln: 10 Col: 0'.

Below the editor, the Python shell output is shown:

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

The IDE status bar at the bottom right shows 'Ln: 1 Col: 0'.

An annotation 'mało czytelne kwestia gustu' (a matter of taste, not very readable) with a black arrow points to the commented-out line in the code.

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

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

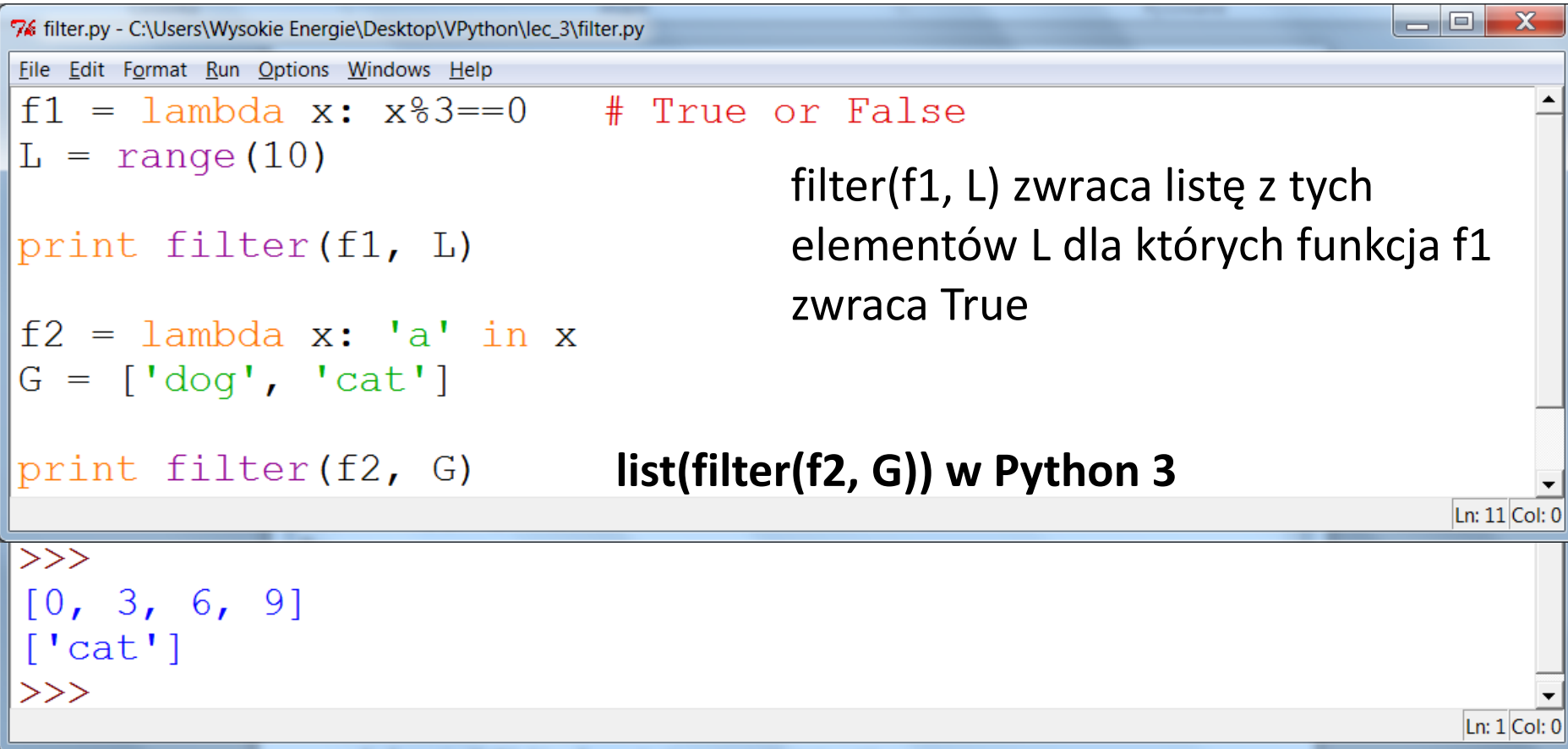
- Previous message: [\[Python-Dev\] math.areclose ...?](#)
- Next message: [\[Python-Dev\] Let's just *keep* lambda](#)
- Messages sorted by: [\[date \]](#) [\[thread \]](#) [\[subject \]](#) [\[author \]](#)

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



The screenshot shows a Python IDE window titled 'filter.py - C:\Users\Wysokie Energie\Desktop\VPython\lec_3\filter.py'. The code defines two lambda functions, f1 and f2, and lists L and G. f1 filters numbers divisible by 3, and f2 filters strings containing 'a'. The execution results show the output of filter(f1, L) as [0, 3, 6, 9] and filter(f2, G) as ['cat'].

```
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)
```

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

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

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