

# Zostań „wężoustym” czyli jak rozmawiać z Pythonem

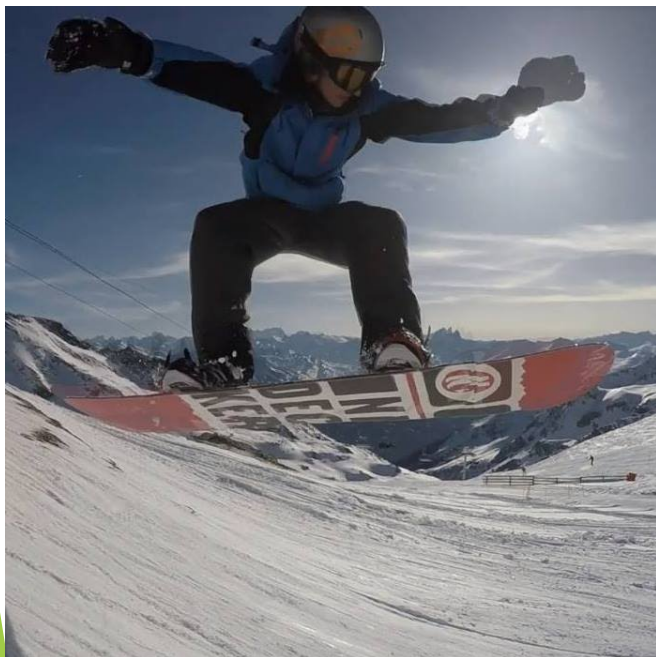
Daniel Sienkiewicz



Trochę o mnie...



UNIWERSYTET GDAŃSKI

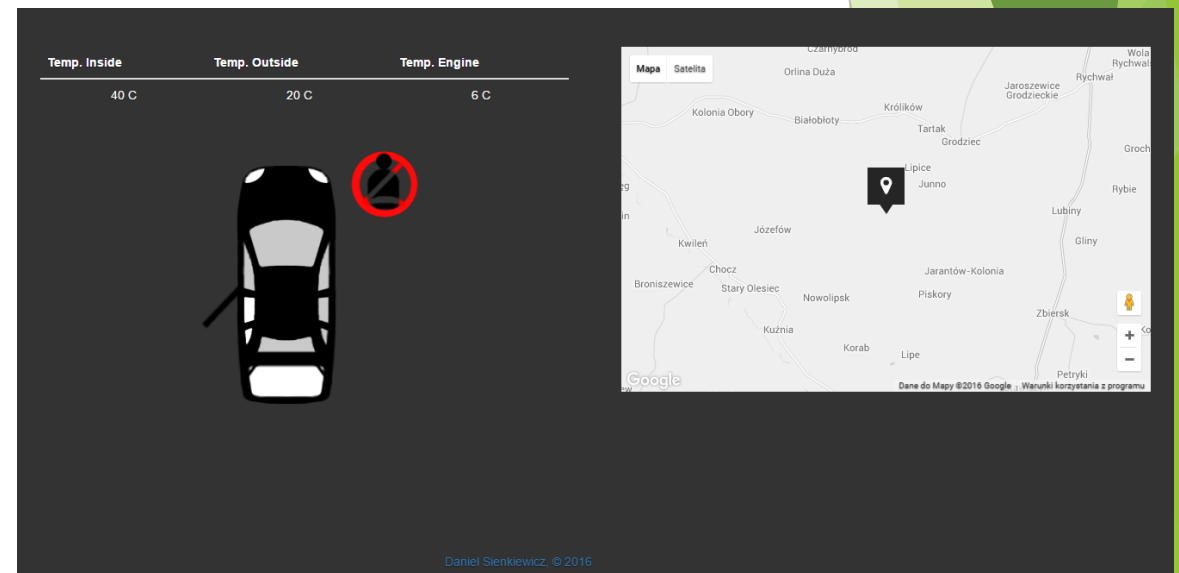
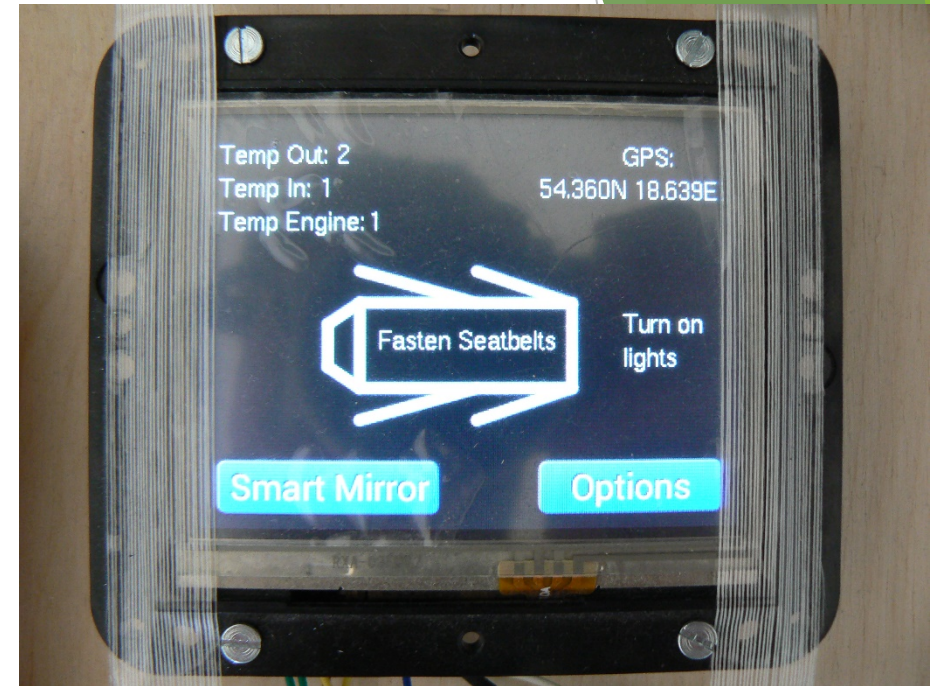
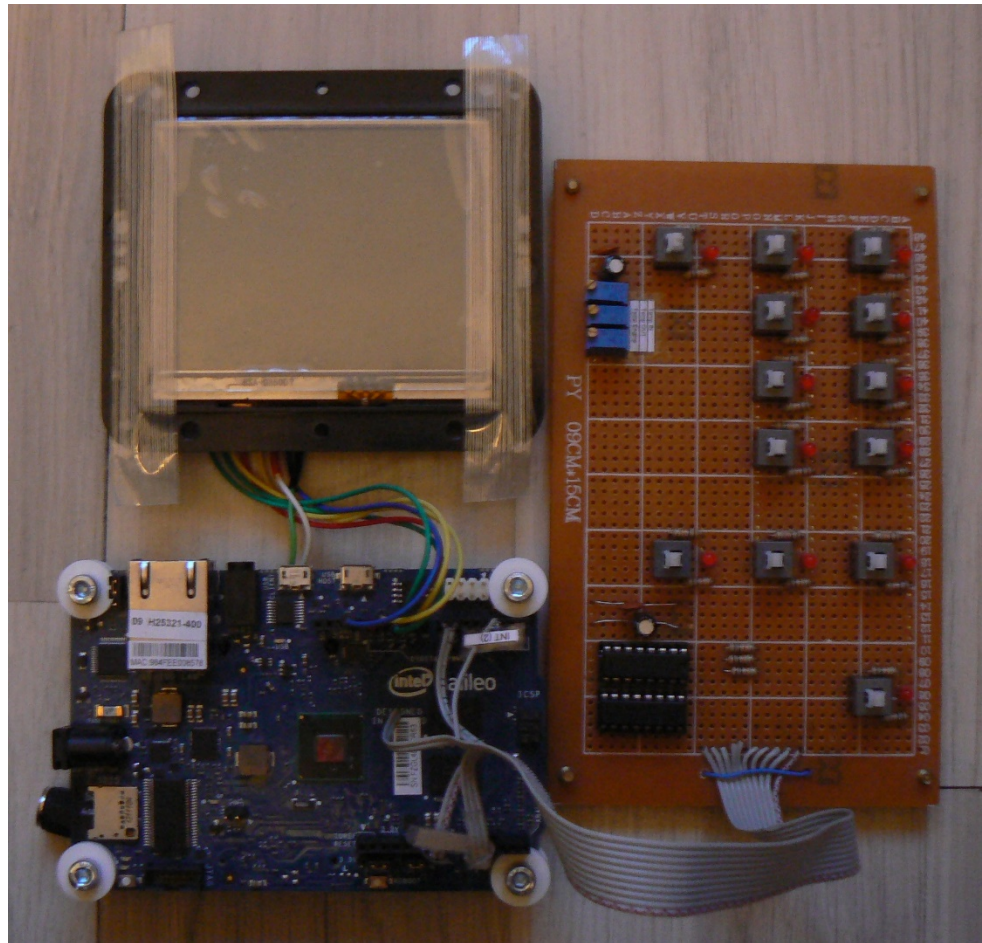


• **A P T I V** •





# Trochę o mnie...



# Trochę o mnie...



daniel@sienkiewicz.ovh



daniel-sienkiewicz



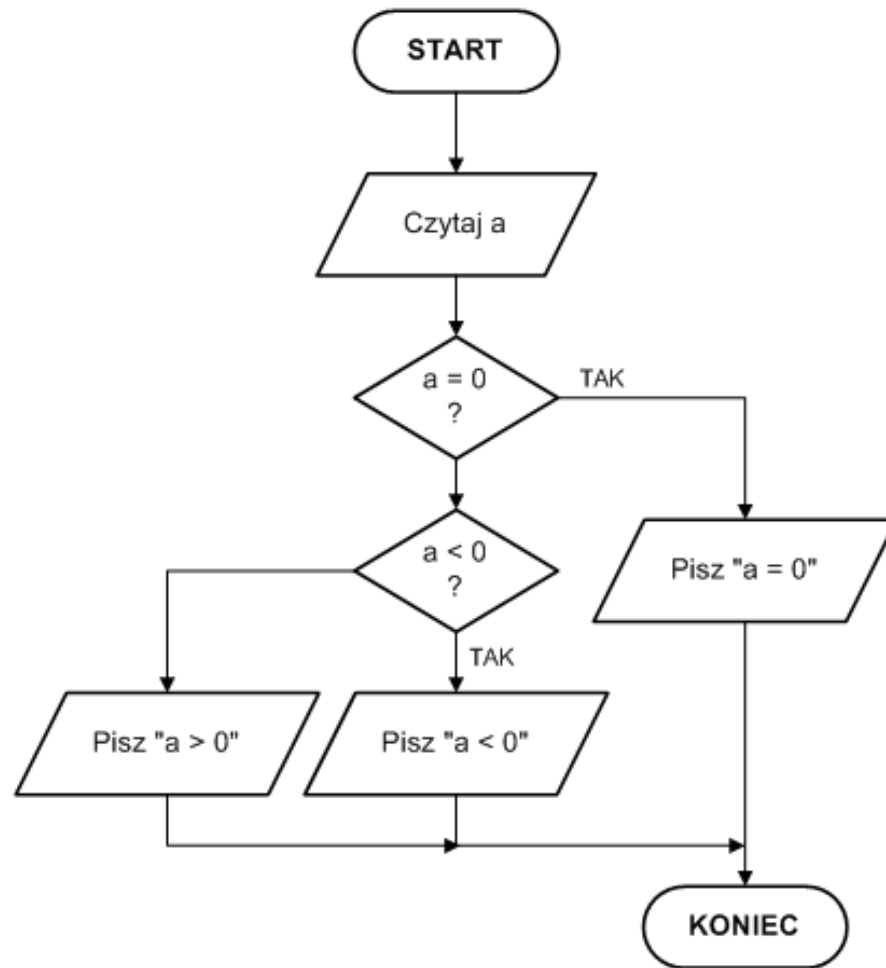
daniel-sienkiewicz

# Sentencja na dziś...

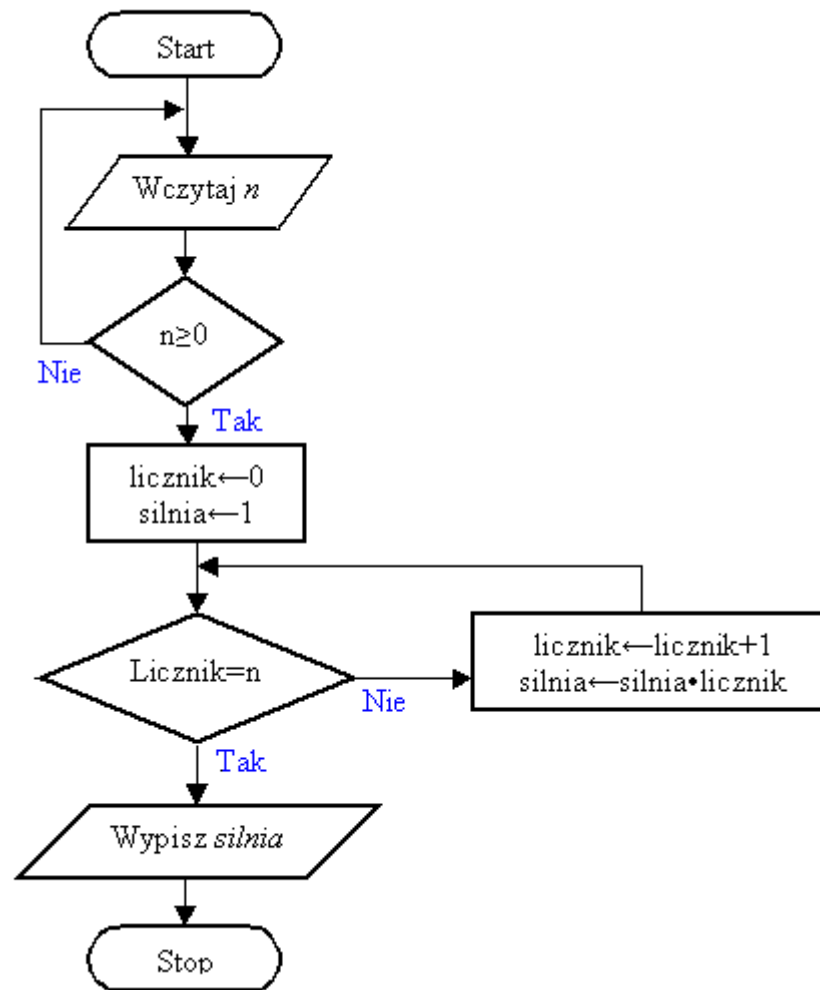
Komputer robi zawsze to co programista mu każe, ale nie zawsze to co ma na myśli...

~ Sfrustrowany Programista

# Program komputerowy

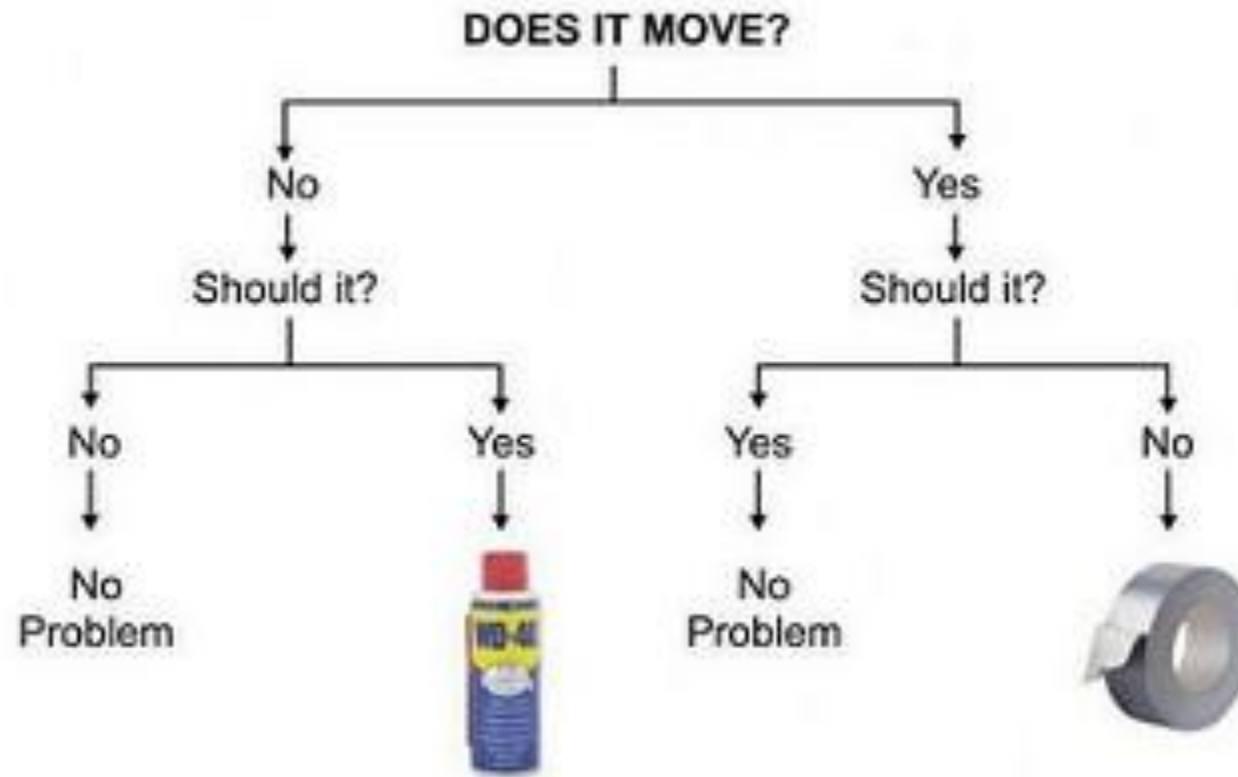


# Program komputerowy



# Program komputerowy

## Engineering Flowchart





# „Hello Word”

```
def main():  
    print „Hello Word”
```

```
if __name__ == '__main__':  
    main()
```

```
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println(„Hello Word”);  
    }  
}
```

```
#include<stdio.h>  
int main(void) {  
    printf(„Hello Word”);  
    return 0;  
}
```

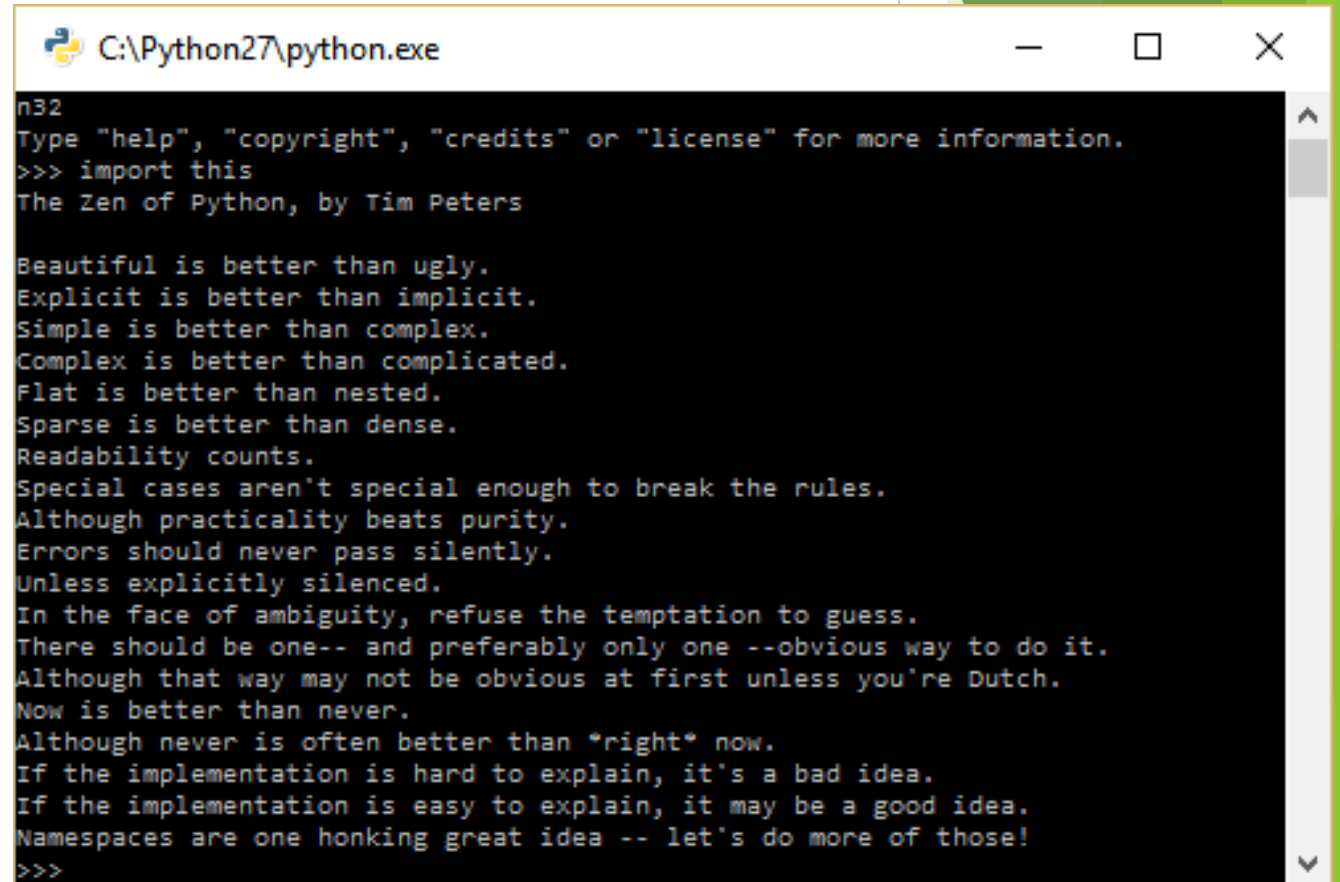
```
#include <iostream>  
using namespace std;  
int main(void) {  
    cout << "Hello World";  
    return 0;  
}
```

# Typy zmiennych

Nazwa typu	Rozmiar (w bajtach)	Rozmiar (w bitach)	Zakres wartości
bool	1	8	false lub true
char	1	8	-128 do 127
short	2	16	-32.768 do 32.767
int	4	32	-2 147 483 648 do 2 147 483 647
long	4	32	-2 147 483 648 do 2 147 483 647
float	4	32	-3.4E38 do 3.4E38
double	8	64	$-1.8 \cdot 10^{308}$ do $1.8 \cdot 10^{308}$

# Kilka ciekawostek

- Dwie gałęzie: 2.x and 3.x
- Zawiera dużo humoru z Monthly Python
- Język interpretowany
- Duck typing
- Garbage collector



```
C:\Python27\python.exe
n32
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

# Konsola Python

To jest Python prompt

```
>>>
```

To jest znak kontynuacji

```
...
```

To jest napis

```
'To ja Artur, syn Uthera Pendragona,  
z zamku Camelot. Król Brytyjczyków,  
pogromca Saksonów, władca całej Anglii!'
```

Przykład:

```
>>> # To jest komentarz
```

```
... 2 + 2
```

```
4
```

# Zmienne

```
>>> tax = 12.5 / 100
```

```
>>> price = 100.50
```

```
>>> price * tax
```

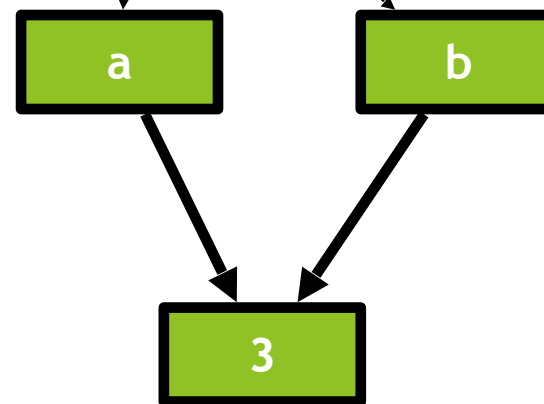
```
12.5625
```

```
>>> a = b = 3
```

```
>>> c
```

```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
NameError: name 'c' is not defined
```

Tylko  
nazwy...



... które wskazują na  
wartość w pamięci



# Napisy

```
>>> 'aaaa'  
'aaaa'
```

```
>>> "aaaa"  
'aaaa'
```

```
>>> """multi  
... line  
... string  
... """  
'multi\nline\nstring\n'
```

# Napisy

```
>>> word = 'Help' + 'A'
>>> word
'HelpA'
>>> '<' + word * 5 + '>'
'<HelpAHelpAHelpAHelpAHelpA>'
>>> word[4]
'A'
>>> word[0:2]
'He'
>>> word[:2]    # Pierwsze dwa znaki
'He'
```

# Napisy

```
>>> word[2:]      # Wszystko z pominięciem dwóch  
pierwszych  
'lpA'  
>>> word[-1]     # Ostatni  
'A'  
>>> word[-2:]    # Ostatnie 2  
'pA'  
>>> word[:-2]    # Wszystko oprócz dwóch ostatnich  
'Hel'
```

# Listy

```
>>> a = [ 'spam', 'eggs', 100, 1234 ]
```

```
>>> a
```

```
[ 'spam', 'eggs', 100, 1234 ]
```

```
>>> a[2] = a[2] + 23
```

```
>>> a
```

```
[ 'spam', 'eggs', 123, 1234 ]
```

```
>>> q = [2, 3]
```

```
>>> p = [1, q, 4]
```

```
>>> len(p)
```

```
3
```

```
>>> p[1]
```

```
[2, 3]
```

```
>>> p[1][0]
```

```
2
```

```
>>> p[1].append( 'extra' )
```

```
>>> p
```

```
[1, [2, 3, 'extra'], 4]
```

# Listy

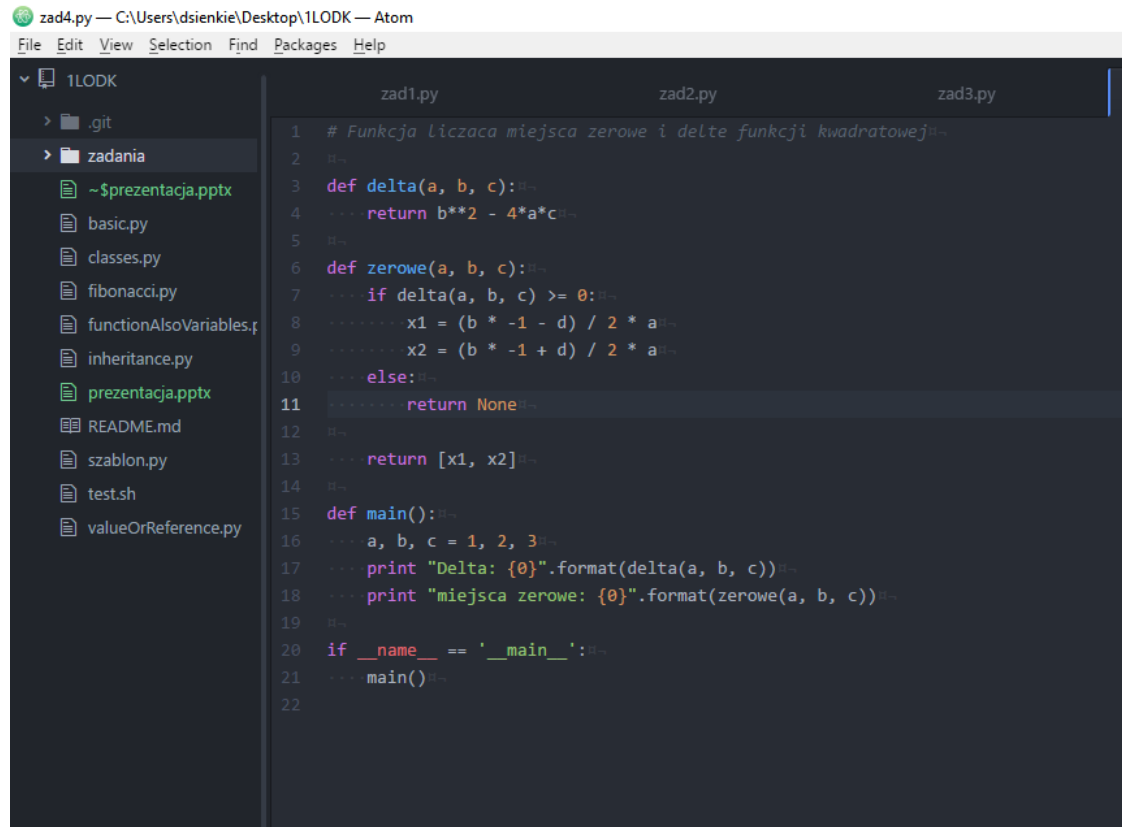
```
>>> q
[2, 3, 'xtra']
>>> q + [4, 5]
[2, 3, 'xtra', 4, 5]
>>> q.extend([2, 2])
[2, 3, 'xtra', 4, 5, 2, 2]
>>> q.count(2)
3
>>> q.insert(1, 9)
[2, 9, 3, 'xtra', 4, 5, 2, 2]
>>> q.remove(2)
[9, 3, 'xtra', 4, 5, 2, 2]
>>> q[:] = [x for x in q if x != 2]
[9, 3, 'xtra', 4, 5]
```



# Instrukcje warunkowe

```
>>> x = input("Musicie dać nam żywopłot... ")
Musicie dać nam żywopłot: Co takiego?
>>> if x == 'ładny i niezbyt drogi':
...     print 'Trelefere Kukuryku Cip-cip-cip'
... else:
...     print 'Ni, Ni, Ni!!!'
'Ni, Ni, Ni!!!'
```

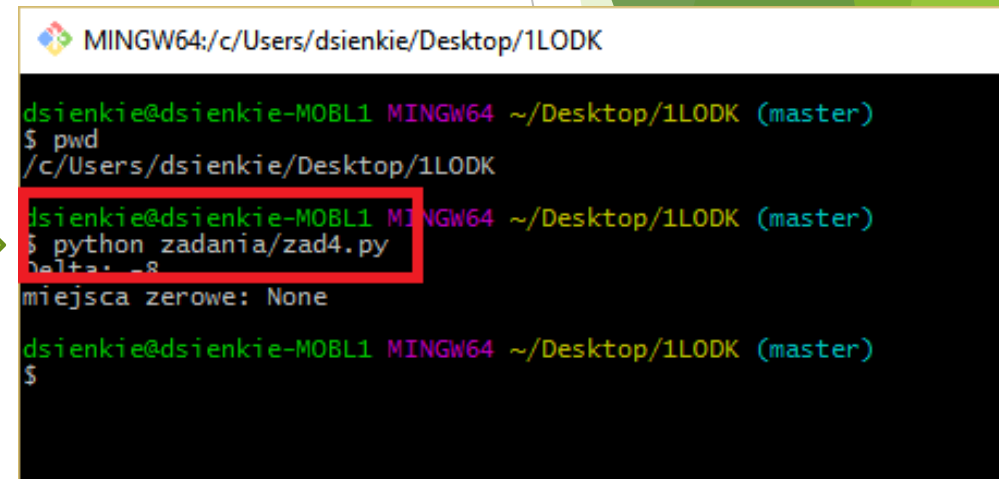
# Jak to uruchomić?



```
zad4.py — C:\Users\dsienkie\Desktop\1LODK — Atom
File Edit View Selection Find Packages Help

1LODK
├── .git
├── zadania
│   ├── ~$prezentacja.pptx
│   ├── basic.py
│   ├── classes.py
│   ├── fibonacci.py
│   ├── functionAlsoVariables.py
│   ├── inheritance.py
│   ├── prezentacja.pptx
│   ├── README.md
│   ├── szablon.py
│   ├── test.sh
│   └── valueOrReference.py
└── zad1.py zad2.py zad3.py zad4.py

1 # Funkcja licząca miejsca zerowe i delte funkcji kwadratowej
2
3 def delta(a, b, c):
4     return b**2 - 4*a*c
5
6 def zerowe(a, b, c):
7     if delta(a, b, c) >= 0:
8         x1 = (b * -1 - d) / 2 * a
9         x2 = (b * -1 + d) / 2 * a
10    else:
11        return None
12
13    return [x1, x2]
14
15 def main():
16     a, b, c = 1, 2, 3
17     print "Delta: {}".format(delta(a, b, c))
18     print "miejsca zerowe: {}".format(zerowe(a, b, c))
19
20 if __name__ == '__main__':
21     main()
22
```



```
MINGW64:/c:/Users/dsienkie/Desktop/1LODK

dsienkie@dsienkie-MOBL1 MINGW64 ~/Desktop/1LODK (master)
$ pwd
/c:/Users/dsienkie/Desktop/1LODK
dsienkie@dsienkie-MOBL1 MINGW64 ~/Desktop/1LODK (master)
$ python zadania/zad4.py
Delta: -8
miejsca zerowe: None

dsienkie@dsienkie-MOBL1 MINGW64 ~/Desktop/1LODK (master)
$
```

**\$ python3 <nazwa\_pliku>.py**

# Petle

```
>>> a = ['sir Bedevere', 'sir Galahad', 'sir Robin']
>>> for x in a:
...     print x, len(x)
sir Bedevere 12
sir Galahad 11
sir Robin 9
>>> i = 0
>>> while i < len(a):
...     print a[i]
...     i = i + 1
sir Bedevere
sir Galahad
sir Robin
```

# Petle

```
>>> questions = ['name', 'quest', 'favourite  
color']  
>>> answers = ['lancelot', 'the holy grail',  
'blue']  
>>> for q, a in zip(questions, answers):  
...     print 'What is your {0}? It is  
{1}.'.format(q, a)  
...
```

What is your name? It is lancelot.

What is your quest? It is the holy grail.

What is your favorite color? It is blue.

# Funkcje

```
def fib(n):  
    """Return a list containing the  
    Fibonacci series up to n."""  
    result = []  
    a, b = 0, 1  
    while a < n:  
        result.append(a)  
        a, b = b, a + b  
    return result
```

```
f100 = fib(100)
```

```
print f100
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```



# Operacje na pliku

**Źle:**

```
f = open("abc.txt", "w")  
f.write("xyz")  
f.close()
```

**Dobrze:**

```
with open("abc.txt", "w") as f:  
    f.write("xyz")
```

# Praktyki

- **Źle**

```
temp = a  
a = b  
b = temp
```

- **Dobrze**

```
b, a = a, b
```

# Praktyki

- **Źle**

```
colors = ['red', 'blue']  
result = ''  
for c in colors:  
    result += c
```

- **Dobrze**

```
colors = ['red', 'blue']  
result = ''.join(colors)
```

# Praktyki

- **Źle**
  - `dir + "/" + file`

- **Dobrze**
  - `os.path.join(dir, file)`

# Praktyki

- **Źle**

```
if x == True:
    pass
if len(items) != 0:
    pass
```

- **Dobrze**

```
if x:
    pass
if items:
    pass
```



# Praktyki

# Przykładowy szablon prostego programu

```
def main():
```

```
    # KOD
```

```
if __name__ == '__main__':
```

```
    main()
```

# Quiz

1. Co to znaczy ze python jest interpretowany?
2. Co to znaczy, ze python jest dynamicznie typowany?
3. Z jaką prędkością porusza się jaskółka bez obciążenia?
4. Co robi metoda count w liście?
5. Jakie znasz typy zmiennych w językach programowania?
6. Jaka jest stolica Asyrii?

Pytania?



I co teraz z tym wszystkim?

**OK...** so what now?

# Zadanka do wykonania

1. Wczytanie tablicy od użytkownika (lub z pliku) i posortowanie jej wybranym algorytmem
2. Wyliczenie max, min, średniej arytmetycznej z tablicy
3. Zamiana liczby binarnej na dziesiętną / dziesiętnej na binarną
4. Funkcja licząca miejsca zerowe i deltę podanej funkcji kwadratowej
5. Dodanie dwóch wektorów
6. Prosta baza danych - w pliku
7. Wasz własny pomysł...

POWODZENIA ;-)



Dziękuję ;-)



daniel@sienkiewicz.ovh



daniel-sienkiewicz



daniel-sienkiewicz