

**PyLightwaw**

# **ANACONDA, CPYTHON, MYPY?**

## **IMPLEMENTACJE PYTHONA**

*I INNE TAKIE RÓŻNE PODOBNE, ALE JUŻ NIE IMPLEMENTACJE*

## **CO TO, PO CO I DLACZEGO?**

@bartekbrak, 2018-03-15

# GUIDO VAN ROSSUM



By Doc Searls [[CC BY-SA 2.0](#)], via Wikimedia Commons

# KRÓTKA, WESOŁA HISTORIA POWSTANIA PYTHON

# KRÓTKA, WESOŁA HISTORIA POWSTANIA PYTHON

- powstał w Głowie wesołego Holendra o trudnym imieniu Guido

# KRÓTKA, WESOŁA HISTORIA POWSTANIA PYTHON

- powstał w Głowie wesołego Holendra o trudnym imieniu Guido
- gwiazdka 1989, trochę wolnego czasu, jako hobby

# KRÓTKA, WESOŁA HISTORIA POWSTANIA PYTHON

- powstał w Głowie wesołego Holendra o trudnym imieniu Guido
- gwiazdka 1989, trochę wolnego czasu, jako hobby
- wybrał nazwę po ulubionym Latającym Cyrku Monty Pythona...



Screen from one of the episode's of Monty Python's Flying Circus, image stolen from [here](#)

- dwa lata później (luty 1991) opublikował pierwszą działającą wersję

- dwa lata później (luty 1991) opublikował pierwszą działającą wersję
- kolejne dwa lata minęły zanim uformowała się społeczność i powstała pierwsza, kompletna wersja Pythona... tak powstał CPython.



Oryginalne logo Pythona, używane do 2005 roku



Obecne logo Pythona

**CO TO JEST CPYTHON I CZY TO TO SAMO CO PYTHON?**

## **CO TO JEST CPYTHON I CZY TO TO SAMO CO PYTHON?**

- Python i CPython to prawie to samo i zwykle mówiąc Python mamy na myśli właśnie CPython.

## CO TO JEST CPYTHON I CZY TO TO SAMO CO PYTHON?

- Python i CPython to prawie to samo i zwykle mówiąc Python mamy na myśli właśnie CPython.
- CPython to tak zwana *reference implementation*, implementacja modelowa

## CO TO JEST CPYTHON I CZY TO TO SAMO CO PYTHON?

- Python i CPython to prawie to samo i zwykle mówiąc Python mamy na myśli właśnie CPython.
- CPython to tak zwana *reference implementation*, implementacja modelowa
- Na jego podstawie można pisać inne Pythony



IronPython

pypy IP[y]:

IPython



Numba

python™

python



ANACONDA®



Mython

typhon



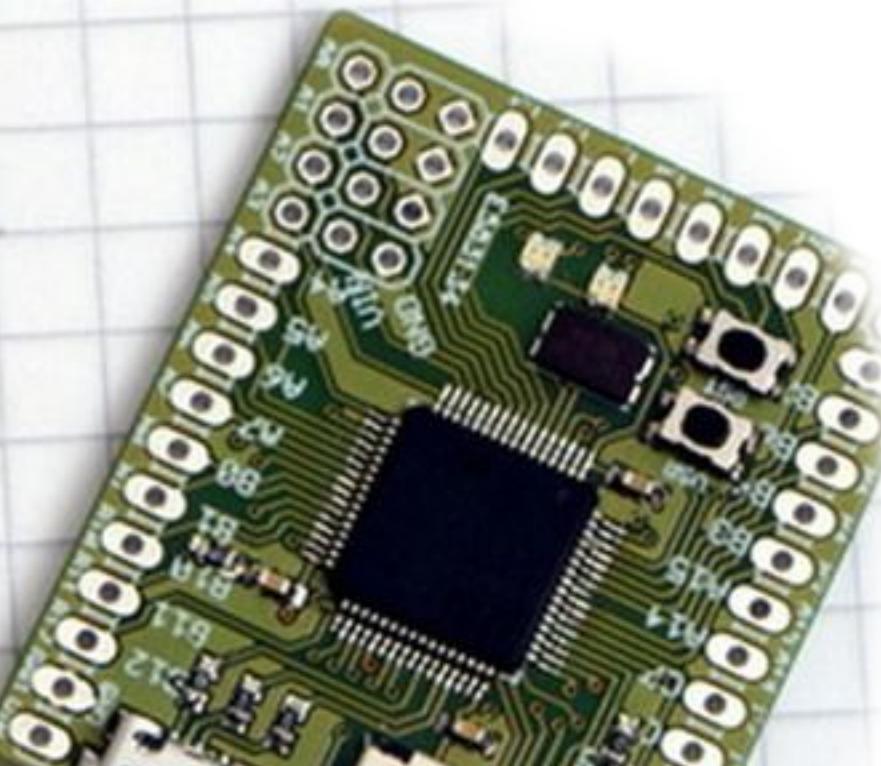
Jython

python  
stackless

Nuitka

Psyco

CL PYTHON



Cython  
SKULPT

BRYTHON

**JAKIE INNE IMPLEMENTACJE ISTNIEJĄ I DO CZEGO, OGÓLNIE SŁUŻĄ? KTO Z NICH KORZYSTA?**

## **JAKIE INNE IMPLEMENTACJE ISTNIEJĄ I DO CZEGO, OGÓLNIE SŁUŻĄ? KTO Z NICH KORZYSTA?**

- CPython

## **JAKIE INNE IMPLEMENTACJE ISTNIEJĄ I DO CZEGO, OGÓLNIE SŁUŻĄ? KTO Z NICH KORZYSTA?**

- CPython
- Anaconda

## **JAKIE INNE IMPLEMENTACJE ISTNIEJĄ I DO CZEGO, OGÓLNIE SŁUŻĄ? KTO Z NICH KORZYSTA?**

- CPython
- Anaconda
- PyPy

## **JAKIE INNE IMPLEMENTACJE ISTNIEJĄ I DO CZEGO, OGÓLNIE SŁUŻĄ? KTO Z NICH KORZYSTA?**

- CPython
- Anaconda
- PyPy
- IPython

- MicroPython

- MicroPython
- IronPython

- MicroPython
- IronPython
- Jython

- MicroPython
- IronPython
- Jython
- Numba

- MicroPython
- IronPython
- Jython
- Numba
- Stackless

- MicroPython

- IronPython

- Jython

- Numba

- Stackless

- Cython

# CZY PYTHON 2 I 3 TO TEŻ RÓŻNE IMPLEMENTACJE, CZY 2 UMRZE?



Guido Van Rossum on [Python dev mailing list](#)

*Nie ma co wchodzić w gierki słowne, Python 2 kończy życie 1 stycznia 2020, nie będzie aktualizacji, nawet latek bezpieczeństwa. [...] Programiści Python włożyli wystarczająco dużo wysiłku w podtrzymanie pacjenta przy życiu, było wystarczająco dużo czasu, żeby się przesiąść (pierwsza data pogrzebu to 2015). Zupełnie mnie to nie boli, że kończymy z 2.*

# **ANACONDA, JAK UŁATWIA ŻYCIE**

## **ANACONDA, JAK UŁATWIA ŻYCIE**

- Anaconda zawiera setki zbudowanych pakietów, których instalacja sprawiłaby trudność nie tylko początkującym

# ANACONDA, JAK UŁATWIA ŻYCIE

- Anaconda zawiera setki zbudowanych pakietów, których instalacja sprawiłaby trudność nie tylko początkującym
- Instalacja nowych pakietów jest prostsza, wystarczy `conda install PACKAGE NAME`, na każdym systemie

# ANACONDA, JAK UŁATWIA ŻYCIE

- Anaconda zawiera setki zbudowanych pakietów, których instalacja sprawiłaby trudność nie tylko początkującym
- Instalacja nowych pakietów jest prostsza, wystarczy `conda install PACKAGE NAME`, na każdym systemie
- Zawiera wbudowany virtualenv i różne wersje pythona: `conda create --name py35 python=3.5`

[https://conda.io/docs/\\_downloads/conda-cheatsheet.pdf](https://conda.io/docs/_downloads/conda-cheatsheet.pdf)

**CONDA®**  
**CONDA CHEAT SHEET**  
Command line package and environment manager

Learn to use conda in 30 minutes at [bit.ly/tryconda](http://bit.ly/tryconda)

**TIP:** Anaconda Navigator is a graphical interface to use conda.  
Double-click the Navigator icon on your desktop or in a Terminal or at  
the Anaconda prompt, type `anaconda-navigator`

**Conda basics**

Verify conda is installed, check version number	<code>conda info</code>
Update conda to the current version	<code>conda update conda</code>
Install a package included in Anaconda	<code>conda install PACKAGENAME</code>
Run a package after install, example Spyder*	<code>spyder</code>
Update any installed program	<code>conda update PACKAGENAME</code>
Command line help	<code>COMMANDNAME --help</code> <code>conda install --help</code>

\*Must be installed and have a deployable command,  
usually PACKAGENAME

**Using environments**

Create a new environment named py35, install Python 3.5	<code>conda create --name py35 python=3.5</code>
---	--

- zawiera anaconda navigator

# ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home

Environments

Projects (beta)

Learning

Community

Documentation

Developer Blog

Feedback

Applications on root Channels Refresh

**jupyter notebook**  
5.0.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

**IP[y]: qtconsole**  
4.3.0

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

**spyder**  
3.1.4

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

**glueviz**  
0.10.4

Multidimensional data visualization across files. Explore relationships within and among related datasets.

**orange3**  
3.4.1

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

**rstudio**  
1.0.136

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Launch Launch Launch Install

Install Install

# JAK WYGLĄDA CYTHON?

- trochę jak Python
- trochę jak C
- ale wcale jak żaden z nich

```
In [1]: # Python: integrate_py.py

from math import sin

def f(x):
    return sin(x**2)

def integrate_f(a, b, N):
    dx = (b-a)/N
    s = 0
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

```
In [2]: # Cython: integrate_cy.pyx

cdef extern from "math.h":
    double sin(double x)

cdef double f(double x):
    return sin(x**2)

cpdef double integrate_f(double a, double b, int N):
    cdef double dx, s
    cdef int i

    dx = (b-a)/N
    s = 0
    for i in range(N):
        s += f(a+i*dx)
    return s * dx
```

```
File "<ipython-input-2-13e709f1e8dc>", line 3
    cdef extern from "math.h":  
    ^
SyntaxError: invalid syntax
```

# JAK WYGLĄDA MYPY?

- jak Python 3.6 z zastosowanym typowaniem

```
In [ ]: class BankAccount:  
    def __init__(self, initial_balance=0):  
        self.balance = initial_balance  
    def deposit(self, amount):  
        self.balance += amount  
    def withdraw(self, amount):  
        self.balance -= amount  
    def overdrawn(self):  
        return self.balance < 0
```

```
In [ ]: class BankAccount:  
    def __init__(self, initial_balance=0):  
        self.balance = initial_balance  
    def deposit(self, amount):  
        self.balance += amount  
    def withdraw(self, amount):  
        self.balance -= amount  
    def overdrawn(self):  
        return self.balance < 0
```

```
In [ ]: # Mypy with static typing  
class BankAccount:  
    def __init__(self, initial_balance: int = 0) -> None:  
        self.balance = initial_balance  
    def deposit(self, amount: int) -> None:  
        self.balance += amount  
    def withdraw(self, amount: int) -> None:  
        self.balance -= amount  
    def overdrawn(self) -> bool:  
        return self.balance < 0
```

DZIĘKI