PROGRAMOWANIE WSPÓŁBIEŻNE W PYTHONIE

Mikołaj Lewandowski

STX Next

mikolaj.lewandowski@stxnext.pl

STXNEXT
python powerhouse
ZAWINĘLIŚMY
DO
GDAŃSKA
SMILE
ŻEGLUGA
GDAŃSKI

Współbieżność - dlaczego?

Responsywność

Wydajność (*)

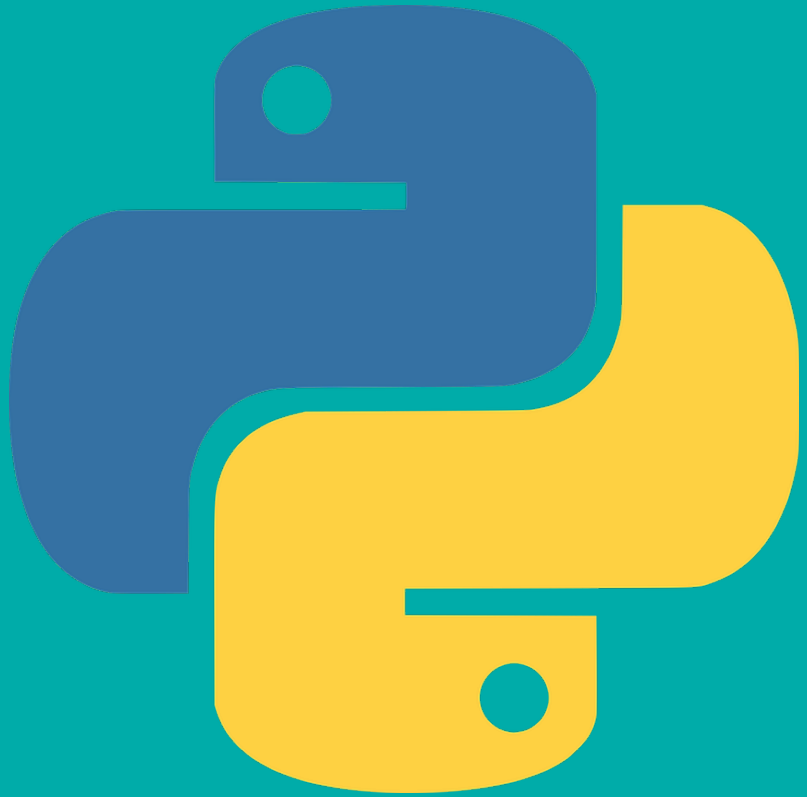
Tak działa świat

Niebezpieczeństwa

Deadlock

Livelock

Starvation



https://github.com/mikolevy/concurrent-python

git clone https://github.com/mikolevy/concurrent-python.git

Welcome to PyCharm

PyCharm

Version 2018.1.1

Create New Project

Open

Check out from Version Control

- CVS
- Git
- Mercurial
- Subversion

Configure

Get Help

Preferences

- Appearance & Behavior
- Keymap
- Editor
- Plugins
- Version Control
- Project: concurrent-python
  - Project Interpreter
  - Project Structure
- Build, Execution, Deployment
- Languages & Frameworks
- Tools

Project: concurrent-python › **Project Interpreter** For current project

Project Interpreter: Python 2.7 /usr/bin/python

Add...
Show All...

Package	Version	Latest
altgraph	0.10.2	0.15
bdist-mpkg	0.5.0	
bonjour-py	0.3	
macholib	1.5.1	1.9
matplotlib	1.3.1	2.2.2
modulegraph	0.10.4	0.16
numpy	1.8.0rc1	1.14.2
py2app	0.7.3	0.14
pyOpenSSL	0.13.1	17.5.0
pyobjc-core	2.5.1	4.2.1
pyobjc-framework-Accounts	2.5.1	4.2.1
pyobjc-framework-AddressBook	2.5.1	4.2.1
pyobjc-framework-AppleScriptKit	2.5.1	4.2.1
pyobjc-framework-AppleScriptObjC	2.5.1	4.2.1
pyobjc-framework-Automator	2.5.1	4.2.1
pyobjc-framework-CFNetwork	2.5.1	4.2.1
pyobjc-framework-Cocoa	2.5.1	4.2.1
pyobjc-framework-Collaboration	2.5.1	4.2.1
pyobjc-framework-CoreData	2.5.1	4.2.1
pyobjc-framework-CoreLocation	2.5.1	4.2.1
pyobjc-framework-CoreText	2.5.1	4.2.1
pyobjc-framework-DictionaryServices	2.5.1	4.2.1
pyobjc-framework-EventKit	2.5.1	4.2.1
pyobjc-framework-ExceptionHandling	2.5.1	4.2.1
pyobjc-framework-FSEvents	2.5.1	4.2.1
pyobjc-framework-InputMethodKit	2.5.1	4.2.1

Python packaging tools not found. Install packaging tools

Cancel Apply OK

Add Python Interpreter

- Virtualenv Environment
- Conda Environment
- System Interpreter

New environment

Location: /Users/mikolevy/.virtualenvs/concurrent-python

Base interpreter: /usr/local/bin/python3.6

- Python 2.7 /usr/bin/python
- /usr/local/bin/python3.6
- /usr/bin/python2.7
- /usr/local/bin/python3
- /System/Library/Frameworks/Python.framework/Versions/2.7/bin/python
- /System/Library/Frameworks/Python.framework/Versions/2.6/bin/python
- /System/Library/Frameworks/Python.framework/Versions/2.5/bin/python
- /System/Library/Frameworks/Python.framework/Versions/2.3/bin/python

Inherit global s

Make available

Existing environme

Interpreter:

Make available

Cancel OK

class `threading.`**`Thread`**(*group=None*, *target=None*, *name=None*, *args=()*, *kwargs={}*, **,*
daemon=None)

```python
import threading


def do_some_work():
    print(f'I am thread number: {threading.get_ident()} '
          f'and it is so hard work!')


thread = threading.Thread(target=do_some_work)
thread.start()
thread.join()

print(f'I am the main thread. My number is: {threading.get_ident()})'
      f'All work is done - bye bye!')

```

```
I am thread number: 123145446957056 and it is so hard work!
I am the main thread. My number is: 140735846814528)All work is done - bye bye!
```

```python
rabbits_colony_size: int = 0


def rabbit_counter(number_of_rabbits: int):
    global rabbits_colony_size
    rabbits_colony_size += number_of_rabbits
    print(f'Now we have {rabbits_colony_size} rabbits')
    print('----------------')


for _ in range(10):
    rabbit_counter(1)

print('All rabbits are counted')

```

Run exercise_1

```
/Users/mikolevy/.virtualenvs/beit/bin/python
  /Users/mikolevy/Documents/Praca/Projects/co
  /exercise_1.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run solution_1

```
/Users/mikolevy/.virtualenvs/beit
  /Users/mikolevy/Documents/Praca/
  /solution_1.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run exercise_2

```
/Users/mikolevy/.virtualenvs/beit/bin
  /Users/mikolevy/Documents/Praca/Proj
  /exercise_2.py
Now we have 1 rabbits
Now we have 3 rabbits
---------------
---------------
Now we have 5 rabbits
---------------
Now we have 7 rabbits
Now we have 7 rabbits
Now we have 7 rabbits
---------------
---------------
Now we have 9 rabbits
Now we have 9 rabbits
---------------
Now we have 9 rabbits
---------------
---------------
---------------
All rabbits are counted
Now we have 10 rabbits
---------------

Process finished with exit code 0
```

`Queue.put(item, block=True, timeout=None)`

Put *item* into the queue. If optional args *block* is true and *timeout* is `None` (the default), block if necessary until a free slot is available. If *timeout* is a positive number, it blocks at most *timeout* seconds and raises the `Full` exception if no free slot was available within that time. Otherwise (*block* is false), put an item on the queue if a free slot is immediately available, else raise the `Full` exception (*timeout* is ignored in that case).

`Queue.put_nowait(item)`

Equivalent to `put(item, False)`.

`Queue.get(block=True, timeout=None)`

Remove and return an item from the queue. If optional args *block* is true and *timeout* is `None` (the default), block if necessary until an item is available. If *timeout* is a positive number, it blocks at most *timeout* seconds and raises the `Empty` exception if no item was available within that time. Otherwise (*block* is false), return an item if one is immediately available, else raise the `Empty` exception (*timeout* is ignored in that case).

`Queue.get_nowait()`

Equivalent to `get(False)`.

Two methods are offered to support tracking whether enqueued tasks have been fully processed by daemon consumer threads.

`Queue.task_done()`

Indicate that a formerly enqueued task is complete. Used by queue consumer threads. For each `get()` used to fetch a task, a subsequent call to `task_done()` tells the queue that the processing on the task is complete.

If a `join()` is currently blocking, it will resume when all items have been processed (meaning that a `task_done()` call was received for every item that had been `put()` into the queue).

Raises a `ValueError` if called more times than there were items placed in the queue.

`Queue.join()`

Blocks until all items in the queue have been gotten and processed.

The count of unfinished tasks goes up whenever an item is added to the queue. The count goes down whenever a consumer thread calls `task_done()` to indicate that the item was retrieved and all work on it is complete. When the count of unfinished tasks drops to zero, `join()` unblocks.

Run queue_example

```
/Users/mikolevy/.virtualenvs/beit/bin/pythor
 /Users/mikolevy/Documents/Praca/Projects/c
Doing task_data: Task number: 0...
Doing task_data: Task number: 1...
Doing task_data: Task number: 2...
Doing task_data: Task number: 3...
Doing task_data: Task number: 4...
Doing task_data: Task number: 5...
Doing task_data: Task number: 6...
Doing task_data: Task number: 7...
Doing task_data: Task number: 8...
Doing task_data: Task number: 9...
All work is done - bye bye!

Process finished with exit code 0
```

Run solution_2

```
/Users/mikolevy/.virtualenvs/beit/
  /Users/mikolevy/Documents/Praca/
  /solution_2.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run solution_2_no_fuzzy

```
/Users/mikolevy/.virtualenvs/beit/bi
  /Users/mikolevy/Documents/Praca/Pr
  /solution_2_no_fuzzy.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run queue_example_hang

```
/Users/mikolevy/.virtualenvs/beit/bin/pyt
 /Users/mikolevy/Documents/Praca/Projects
Doing task_data: Task number: 0...
Doing task_data: Task number: 1...
Doing task_data: Task number: 2...
Doing task_data: Task number: 3...
Doing task_data: Task number: 4...
Doing task_data: Task number: 5...
Doing task_data: Task number: 6...
Doing task_data: Task number: 7...
Doing task_data: Task number: 8...
Doing task_data: Task number: 9...
```

Run lock_example

```
/Users/mikolevy/.virtualenvs/beit/bin/python
  /Users/mikolevy/Documents/Praca/Projects/concurrent-python/examples/lock_example.py
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
I have lock when I am writing it!
All work is done - bye bye!

Process finished with exit code 0
```

Run exercise_3

```
/Users/mikolevy/.virtualenvs/beit/b
  /Users/mikolevy/Documents/Praca/Pr
  /exercise_3.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run solution_3

```
/Users/mikolevy/.virtualenvs/beit/bi
  /Users/mikolevy/Documents/Praca/Pr
  /solution_3.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run solution_3_no_fuzzy

```
/Users/mikolevy/Documents/Praca/P
/solution_3_no_fuzzy.py
Now we have 1 rabbits
---------------
Now we have 2 rabbits
---------------
Now we have 3 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run lock_mean_example

```
/Users/mikolevy/Documents/Praca/Projects/concu
Now we have 1 rabbits
---------------
Now we have 4 rabbits
---------------
Now we have 4 rabbits
Now we have 5 rabbits
Now we have 5 rabbits
---------------
Now we have 6 rabbits
---------------
---------------
---------------
Now we have 7 rabbits
---------------
Now we have 8 rabbits
---------------
Now we have 9 rabbits
---------------
Now we have 10 rabbits
---------------
All rabbits are counted

Process finished with exit code 0
```

Run exercise_cpu

```
/Users/mikolevy/.virtualenvs/beit/bin/
  /Users/mikolevy/Documents/Praca/Proj
  /exercise_cpu.py
All work is done - bye bye!
--- 20.66218590736389 seconds ---

Process finished with exit code 0
```

Run solution_threading

```
/Users/mikolevy/.virtualenvs/beit/bin/
  /Users/mikolevy/Documents/Praca/Proj
  /solution_threading.py
All work is done - bye bye!
--- 20.08674192428589 seconds ---

Process finished with exit code 0
```

```c
// pythread.h
typedef void *PyThread_type_lock;


// ceval.c
static PyThread_type_lock interpreter_lock = 0; /* This is the GIL */

void
PyEval_InitThreads(void)
{
    if (interpreter_lock)
        return;
    interpreter_lock = PyThread_allocate_lock();
    PyThread_acquire_lock(interpreter_lock, 1);
    main_thread = PyThread_get_thread_ident();
}


// thread_pthread.h: PyThread_allocate_lock

lock = (pthread_lock *) malloc(sizeof(pthread_lock));
if (lock) {
    memset((void *)lock, '\0', sizeof(pthread_lock));
    lock->locked = 0;

    status = pthread_mutex_init(&lock->mut,
                                pthread_mutexattr_default);
```

```python
from multiprocessing import Process

def f(name):
    print('hello', name)

if __name__ == '__main__':
    p = Process(target=f, args=('bob',))
    p.start()
    p.join()
```

```python
from multiprocessing import Pool
from time import sleep

def f(x):
    return x*x

if __name__ == '__main__':
    # start 4 worker processes
    with Pool(processes=4) as pool:

        # print "[0, 1, 4,..., 81]"
        print(pool.map(f, range(10)))
```

Run solution_multiprocessing

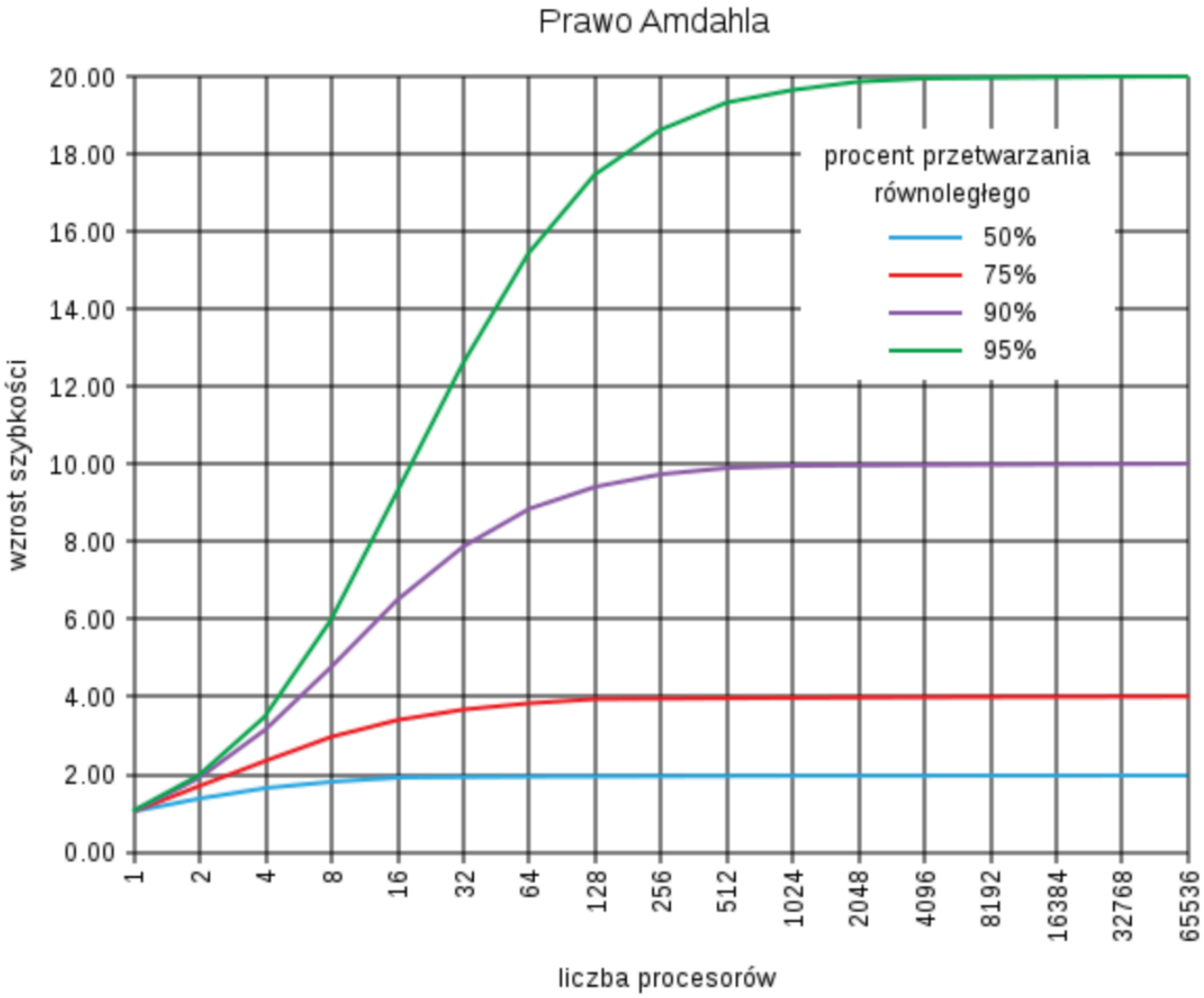
```
/Users/mikolevy/.virtualenvs/beit/bin/p
  /Users/mikolevy/Documents/Praca/Projec
  /solution_multiprocessing.py
All work is done - bye bye!
--- 10.439208030700684 seconds ---

Process finished with exit code 0
```

Prawo Amdahla

Maksymalne przyśpieszenie = $$\frac{1}{(1-P)+\frac{P}{N}}.$$

P - część programu, którą można zrównoleglić

N - liczba rdzeni


Prawo Amdahla
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
wzrost szybkości
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536
liczba procesorów
procent przetwarzania równoległego
50%
75%
90%
95%

Run exercise

```
/Users/mikolevy/.virtualenvs/beit/bi
  /Users/mikolevy/Documents/Praca/Pr
  /exercise.py
--- 20.02696704864502 seconds ---

Process finished with exit code 0
```

Run solution_threading (1)

```
/Users/mikolevy/.virtualenvs/beit/bin/pyth
  /Users/mikolevy/Documents/Praca/Projects
  /solution_threading.py
All work is done - bye bye!
--- 2.0067811012268066 seconds ---

Process finished with exit code 0
```

Wątki vs procesy

Współdzielony stan	Komunikacja IPC
GIL -> 1 core -> IO	Wykorzystanie wielu rdzeni
Współbieżność	Równoległość

Wątki vs async

Preemptively	Cooperatively
Locki	Await
Łatwo dodać do kodu	Cały kod asynchroniczny

Run async_example

```
/Users/mikolevy/.virtualenvs/beit/bin/python
  /Users/mikolevy/Documents/Praca/Projects/concurrent-python/examp
Rabbit breeder is WORKING
Snake breeder is WORKING
They are waiting for their animals, so I will do some work...
Farmer: DONE!
Rabbit breeder report: DONE
Snake breeder report: DONE

Process finished with exit code 0
```

Run async_exercise

```
/Users/mikolevy/.virtualenvs/beit/bin/
  /Users/mikolevy/Documents/Praca/Proj
  .py
Fetchning url: funny-url-0...
Data from: funny-url-0
Fetchning url: funny-url-1...
Data from: funny-url-1
Fetchning url: funny-url-2...
Data from: funny-url-2
Doing some hard work
Ufff - now will fetch one URL...
Fetchning url: HARD-WORK-URL...
Data from: HARD-WORK-URL
--- 13.389259099960327 seconds ---

Process finished with exit code 0
```

Run async_solution_2

```
/Users/mikolevy/.virtualenvs/beit/bin/
  /Users/mikolevy/Documents/Praca/Proje
  /async_solution_2.py
Doing some hard work
Ufff - now will fetch one URL...
Fetchning url: HARD-WORK-URL...
Fetchning url: funny-url-0...
Fetchning url: funny-url-1...
Fetchning url: funny-url-2...
Data from: HARD-WORK-URL
--- 4.286681652069092 seconds ---

Process finished with exit code 0
```