What's New in Python 3.6

Andrey Vlasovskikh

PyCharm Community Lead
JetBrains

Thank You for Inviting Me

—

- I'm glad to speak at PyCon JP
- Big thanks to the organizers and all the attendees!

- Andrey Vlasovskikh
  - My name is アンドレイ ヴラソフスキ


@vlasovskikh

- I'm the PyCharm Community Lead at JetBrains
  - IDE for Python and Web development
- I'm the maintainer of IdeaVim
  - Vim emulation plugin for PyCharm, IntelliJ, Android Studio

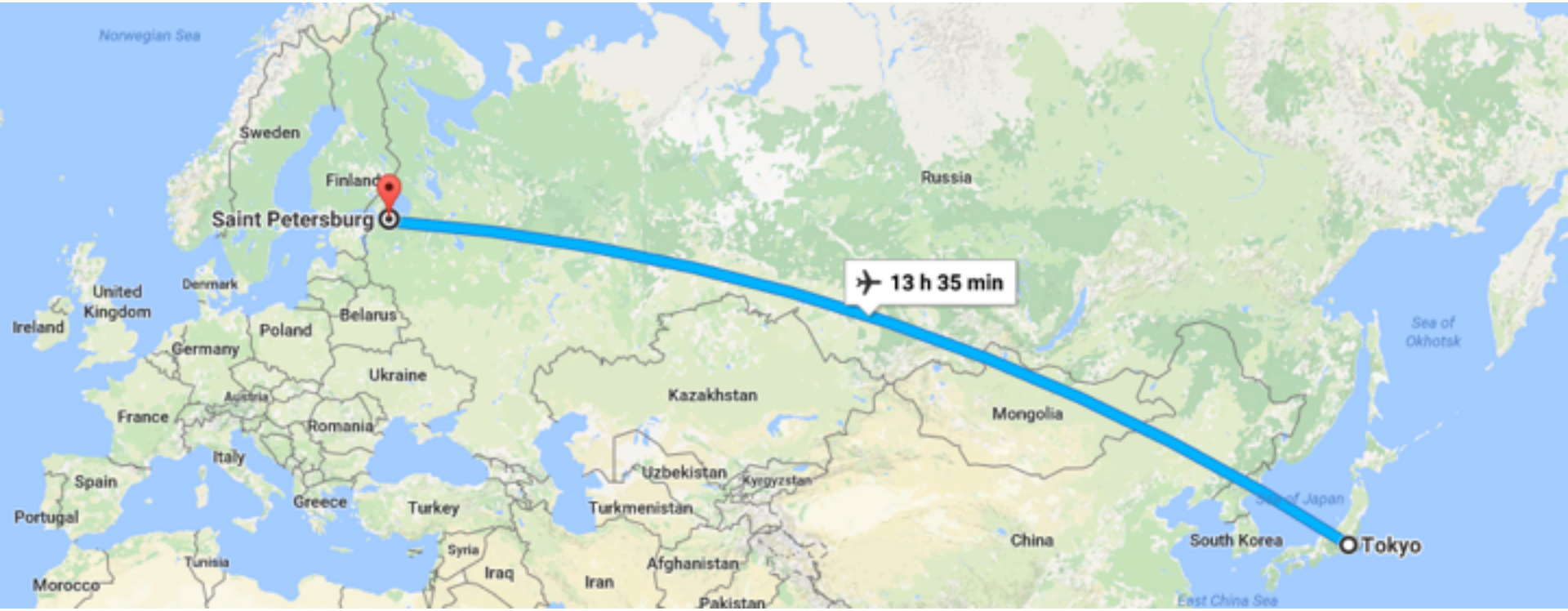
Contributor to Type Hints in Python

—

- I've contributed to PEP 484: Type Hints
- I've been working on Python 2 and 3 compatibility
- I've ported the `typing` module to Python 2.7
  - `pip install typing`

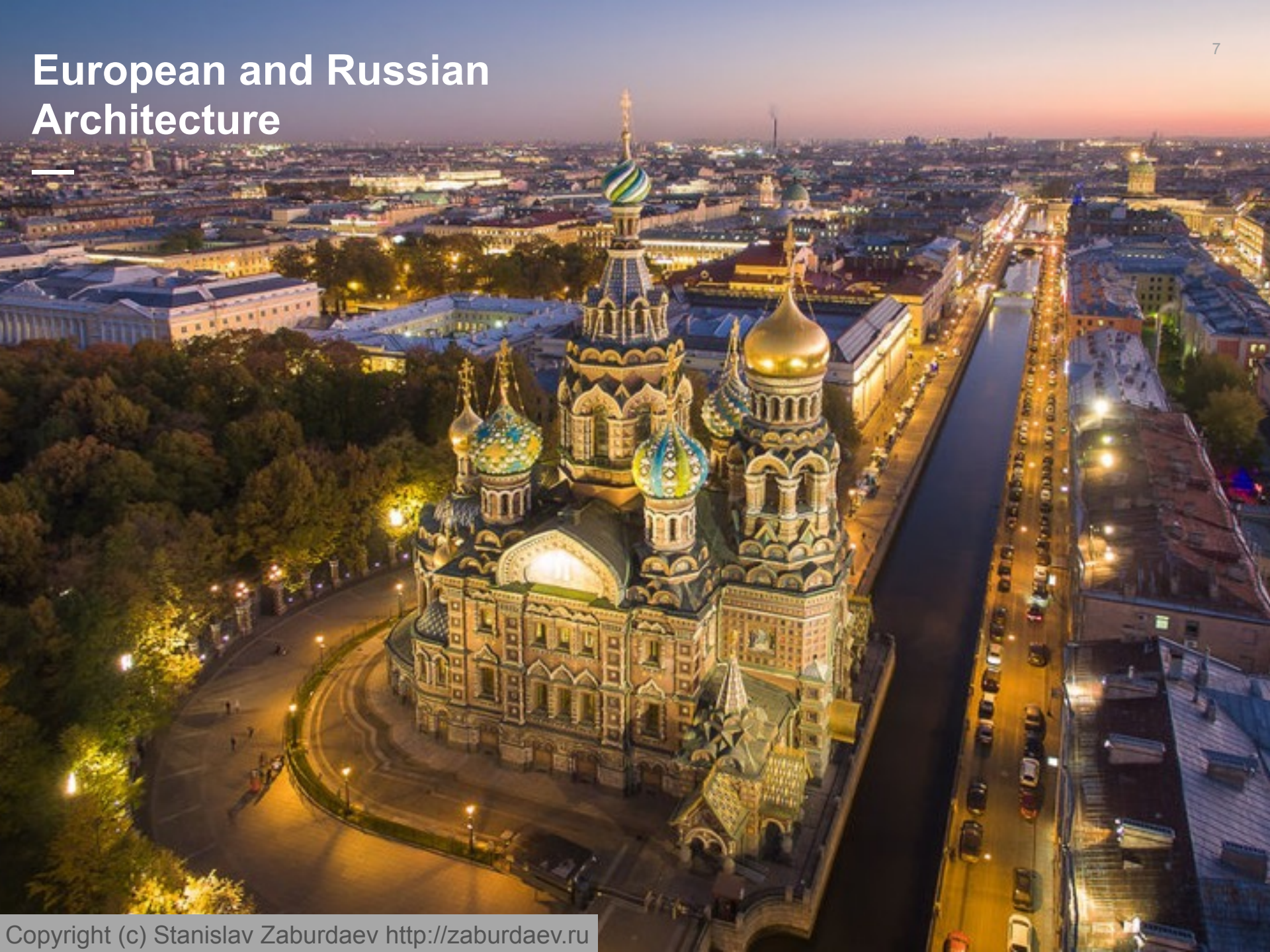
Program Committee Member of PyCon RU

Saint Petersburg, Russia



- The second largest Russian city after Moscow
- Northern Europe, at the Baltic Sea, next to Finland and Estonia

European and Russian Architecture



White Nights in St.Petersburg

What's New in Python 3.6

What's New in 3.6: Summary

Python 3.6b1 released on 2016-09-13

- Formatted string literals
  - `print(f'Hello, {city}')`
- Syntax for variable annotations
  - `city: str = '東京'`
- Underscores in numeric literals
  - `population = 13_617_445`
- Asynchronous generators and comprehensions

What's Out of Scope for Today

- I will not walk through the documentation
  - https://docs.python.org/3.6/whatsnew/3.6.html
  - It's a good read, take a look at it
- I'm not going to convince you that Python 3 is great
  - I will try to stay objective despite my love for Python 3

What this Talk is Really About

—

- I'm going to tell you a story about Python 3
  - I will follow the tradition of the famous Russian writers like Tolstoy, Dostoyevsky, Chekhov
- Explore the character of Python 3
  - I will describe how it changes over time
  - I will mention different points of view, both its upsides and downsides
  - It will be up to you to decide if Python 3 is good for you

Part 1.

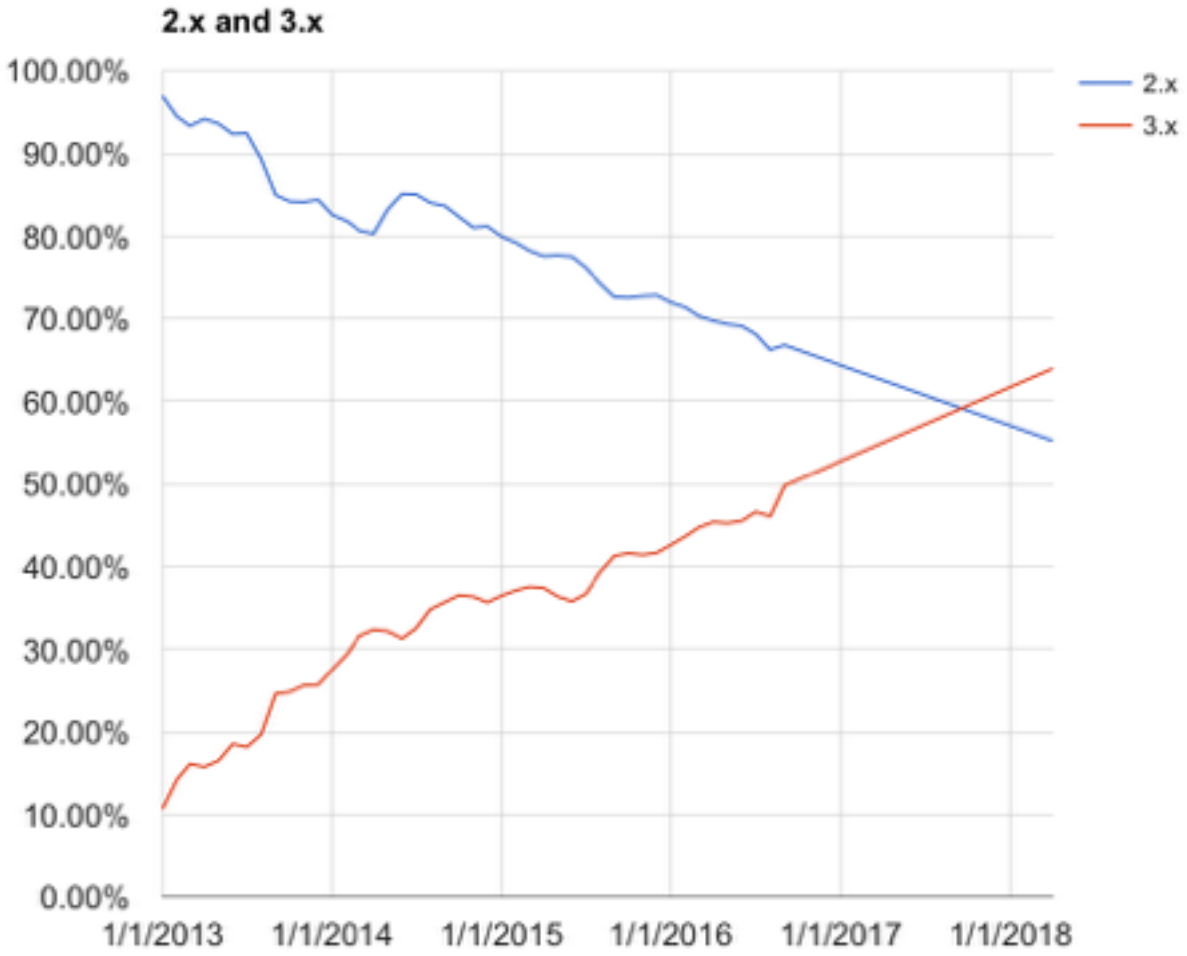
Python 2 vs 3 in Numbers

PyCon JP: 2 vs 3 Survey

- Which Python version(s) do you regularly use?
  - Python 2 only
  - Python 2 and 3
  - Python 3 only

Python 2 vs 3 in PyCharm

- Predicted equal usage in 2017-09
- Up to 20% use both Python 2 and Python 3



Python 2 End-of-Life is 2020

—

- Python 2 retirement party at PyCon US 2020
  - Announced by Guido van Rossum
  - http://pythonclock.org/
- Django 2.0 will drop Python 2 in 2017-12
  - 1.11 LTS until 2020-04

Part 2.

Type Hints

History of Type Hints

- Python 3.0: function annotations
  - Just syntax, no semantics
- Python 3.5: standard notation for type hints
  - Based on function annotations
  - The `typing` module for type system constructs
  - Third-party type checkers: Mypy, PyCharm
- Python 3.6: variable annotations
  - Native syntax instead of type hints in comments

Type Hints: Example

```
from typing import Dict, Iterator

class Element:
    text: str
    attrib: Dict[str, str]

    def itertext(self) -> Iterator[str]: ...

def uppercase_heading(b: bytes) -> Element:
    element = Element('h1')
    element.text = b.upper()  # Type error: expected 'str'
    return element

heading = uppercase_heading(b'Hello')
print(heading.itertext()[0])  # Type error: no '__getitem__'
```

Type Hints: Upsides

- Tools for type checking, code completion, refactoring
  - Type information is crucial for static code analysis
  - More like TypeScript
- Better documentation
  - More compact and than lengthy native language descriptions
- It's enough to annotate APIs
  - Type checkers can infer types

Type Hints: Downsides

- May look like static typing
  - More code to write with no apparent benefits for small projects
- Tools don't support all the features of type hints
  - Mypy and PyCharm are not 100% compatible
  - Pylint doesn't support type hints yet
- Only a few libraries come with type hints
  - Third-party annotations via Python stub files
  - https://github.com/python/typeshed

Case Study: Porting to Python 3

- Porting to Python 3 is expensive
  - Porting Twisted required about $60000
- In a statically typed language porting is a lot simpler
  - Automated code checks and refactorings can guarantee correctness
- Start porting to Python 3 by adding type hints
  - Python 2+3 is the common subset of the two versions
  - First to Python 2+3, then to Python 3
  - Type hints help you find both Python 2 and 3 errors
  - Dropbox is adding type hints to their Python 2 code as the first step

Learn More about Type Hints

—

- 型ヒントについて考えよう!
  - Talk by Yusuke Miyazaki
  - Today at 16:20 in the room 203

Part 3.

Async-Await

History of Async-Await

- Python 3.4: sequential syntax for async code
  - No "callback hell" thanks to the `yield from` syntax
  - The `asyncio` module for event loop
- Python 3.5: new syntax for async code
  - `async def` for coroutine functions and `await` for awaitable objects
- Python 3.6: asynchronous generators and comprehensions
  - `yield` in coroutine functions
  - `async for` and `await` in comprehensions

Async-Await: Example

```
async def stream_events(url):
  while True:
    try:
      response = await aiohttp_request('GET', url)
      async for chunk in aiohttp_read(response.content):
        event = json.loads(chunk.decode('utf-8'))
        yield event
    except Exception as e:
      log.error(e)

async def filter_events(url, f):
  return (e async for e in stream_events(url) if f(e))
```

Async-Await: Upsides

- Sequential syntax for async code
  - Tornado and Twisted code is becoming more readable
  - New aio* family of libraries: `aiohttp`, `aiopg`, etc.
- Coroutine and awaitable as language concepts
  - Async code looks pythonic and native now
  - Fewer reasons to write blocking code
  - You can write async code by default
  - More like Scala, Go, Erlang

Async-Await: Downsides

- Async versions for all blocking I/O libraries
  - Huge duplication of development efforts
- Django and Flask use blocking I/O
  - It's a big step to switch to Tornado or `aiohttp`
- No async for CPU-intensive computations in threads
  - GIL prevents it, only in separate processes

Learn More about Async-Await

—

- You Might Not Want Async (in Python)
  - Talk by Tzu-ping Chung
  - Today at 10:45 in the room 204

Epilogue

The Character of Python 3

- It has its complicated past, but it keeps evolving
- If you want to start your relationship with it, you have to appreciate both its strengths and its weaknesses

The End

@vlasovskikh