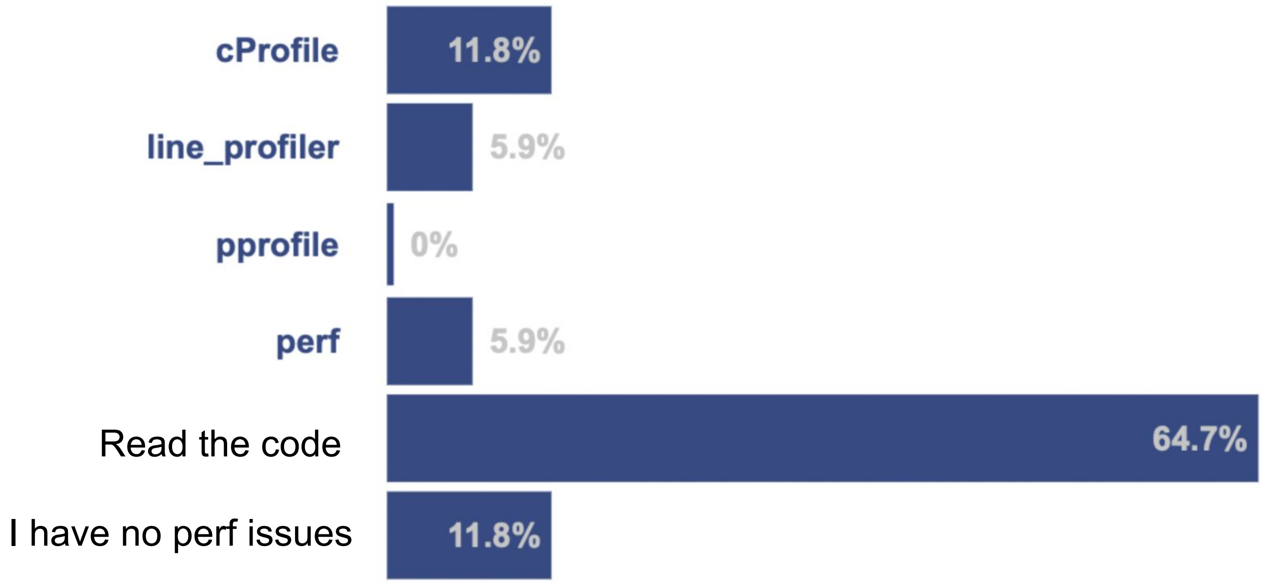
EUROPYTHON PRAGUE & REMOTE 8-14 July 2024

Profiling Python code

Pavel Filonov

Once upon a time at Python meetup

What do you use to solve performance issues in your code?



Another poll attempt

What do you use to solve performance issues in your code?

About speaker

- DS independent consultant
- Former C++ Developer
- Favorite python topics:
  - DS instruments
  - performance
  - Interop
  - tooling

Email: filonovpv@gmail.com
Telegram: @pavel_filonov

Plan

Time	Topic	Tools
13:45	Measure 7 times	pytest-benchmark, cProfile
12:15	Between the lines	line_profiler
15:15	Coffee break	Coffee and Cookies
15:30	Hungry for RAM	memory_profiler
16:00	Profiling on the fly	py-spy
16:30	Conclusion	Reflection and feedback

Link to this slides

Telegram group for communication

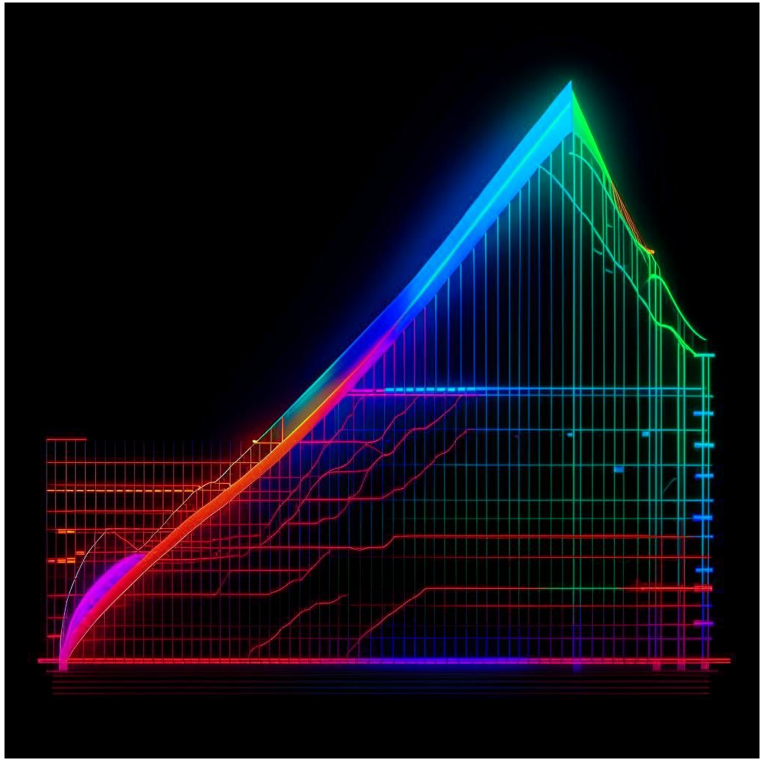
- go to the workshop chat
- ask questions
- share your results

Environment

```
git clone
https://github.com/sdukshis/python-profilin
g-workshop.git

python3.11 -m venv .venv

source .venv/bin/activate

pip install -U pip

pip install -e .
```

Measure 7 times

- what we measure
- average
- median
- minimum
- maximum
- percentile



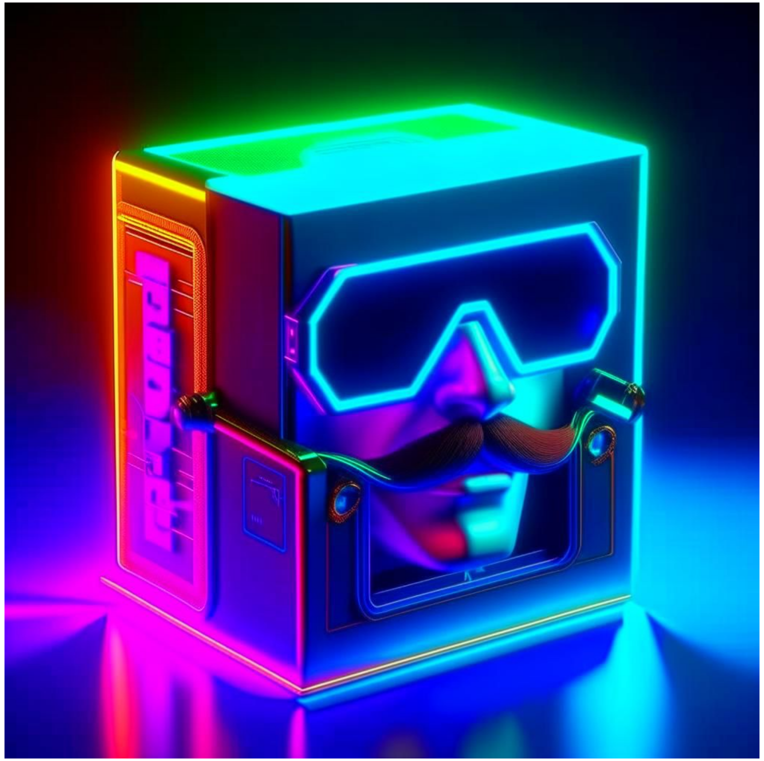
Exercise

1.	Run python 01_cprofile/game_of_chance.py 5 times
2.	Estimate the deviation of the average result
3.	Copy all 5 results and the deviation into the chat in one message

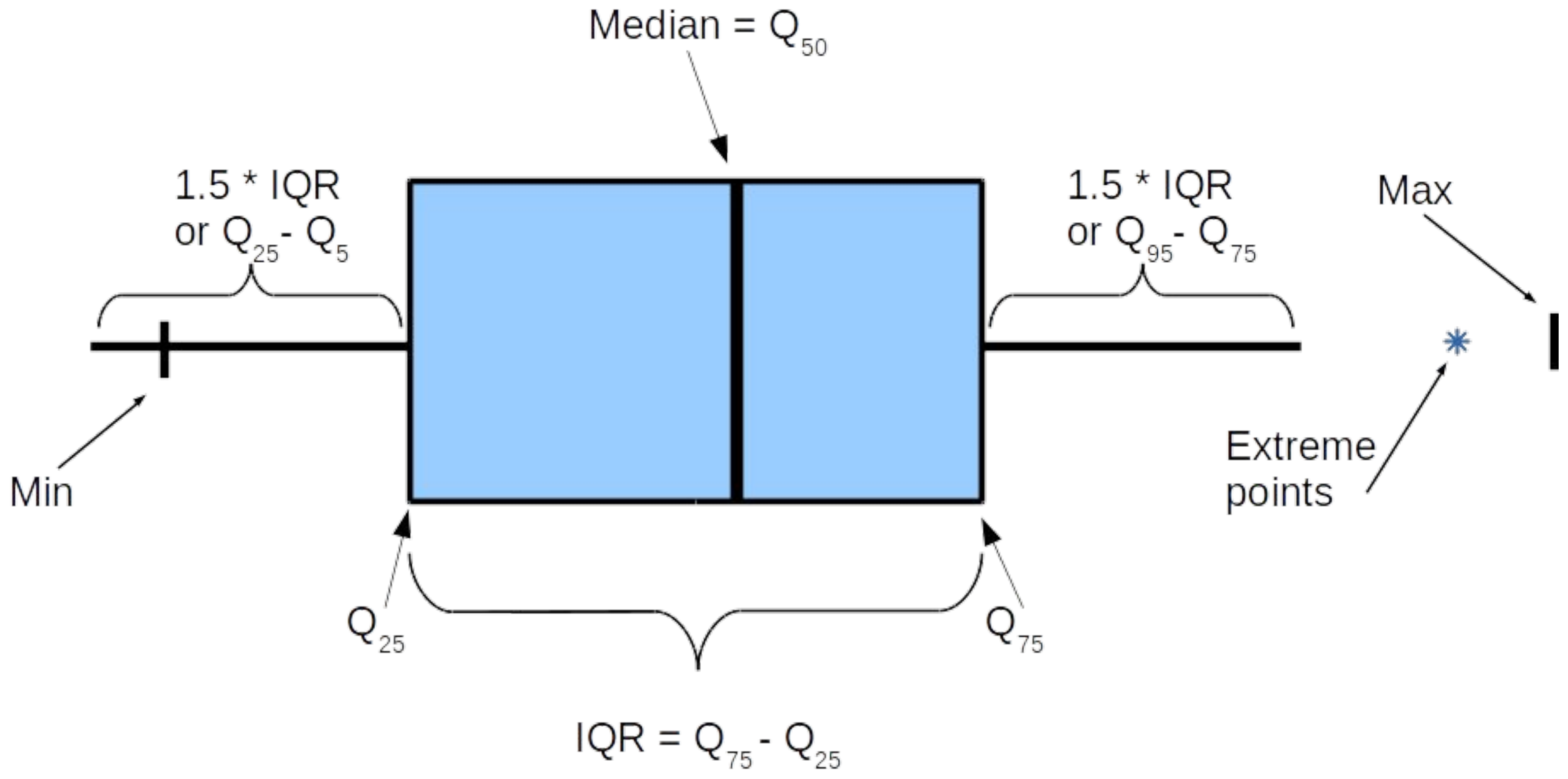
Time for exercise: 5 minutes

Such different statistics

- max will wobble very often
- CPU load
- task Manager
- accuracy of measurements
- Moon phases
- mutating neutrinos
- ...
- average is sensitive to max
- quantiles and min are more stable

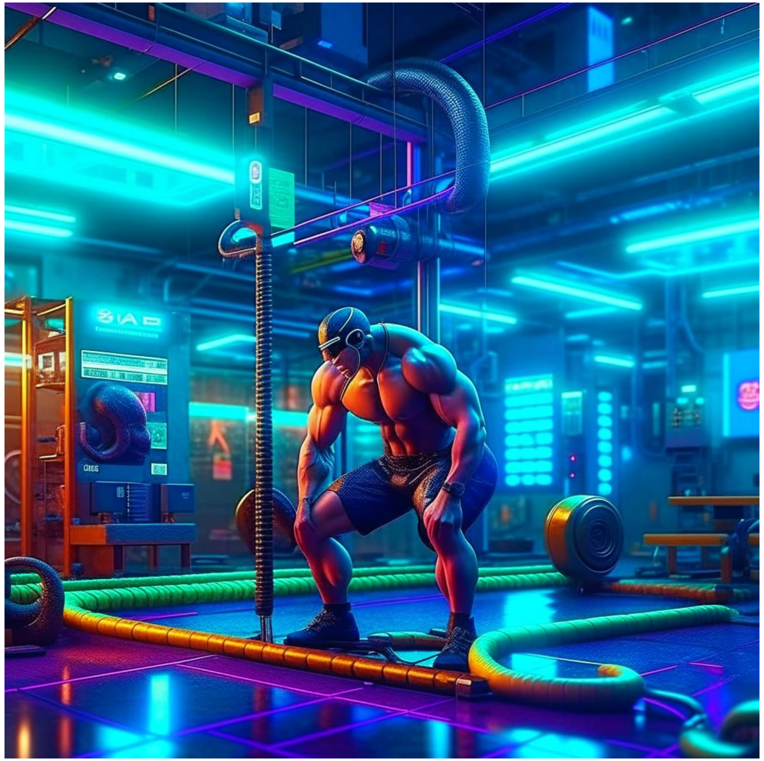


Mustache Box


Median = Q_{50}
1.5 * IQR or Q_{25} - Q_5
1.5 * IQR or Q_{95} - Q_{75}
Max
Min
Extreme points
Q_{25}
Q_{75}
IQR = Q_{75} - Q_{25}

Microbenchmarks

- measure before changes
- measure after changes
- compare
- let's repeat
- automate



Exercise

1.	Run 1st pytest **--benchmark-autosave 01_cprofile/bench_game_of_chance.py**
2.	Try speeding up the play_game code
3.	Run the 2nd benchmark with comparison **pytest --benchmark-compare=0001 01_cprofile/bench_game_of_chance.py**
4.	Take a screenshot and post it in chat

Time for exercise: 5 minutes

Python in profile or cProfile

- with batteries
- pstats for analysis
- snakeviz for visualization
- integration with pytest

Exercise

1. Measure the time of **pytest --benchmark-autosave 01_cprofile/corrector.py**
2. Find the bottleneck using cProfile
3. Make a change to the code
4. Measure the time after the change
5. Take a screenshot of the comparison and send it to the chat

Time for exercise: 10 minutes

Between the lines

- cProfile measures features
- line_profiler measures lines

Exercise

1.	Run 1st **pytest --benchmark-autosave 02_line_profiler/julia.py**
2.	Try speeding up julia's code
3.	Run the 2nd benchmark with comparison **pytest --benchmark-compare=0001 02_line_profiler/julia.py**
4.	Take a screenshot and post it in chat

Time for exercise: 10 minutes

Exercise

1.	Run kernprof for 02_memory_profiler/SimulatedDataset.py
2.	Building a flamegraph using snakeviz
3.	Find a way to speed up any 1 function
4.	Measure the result and write in the chat was/was for this function

Time for exercise: 10 minutes

Break
15 minutes

Python is hungry for memory

- python not only slow
- but also hungry for RAM

memory_profiler

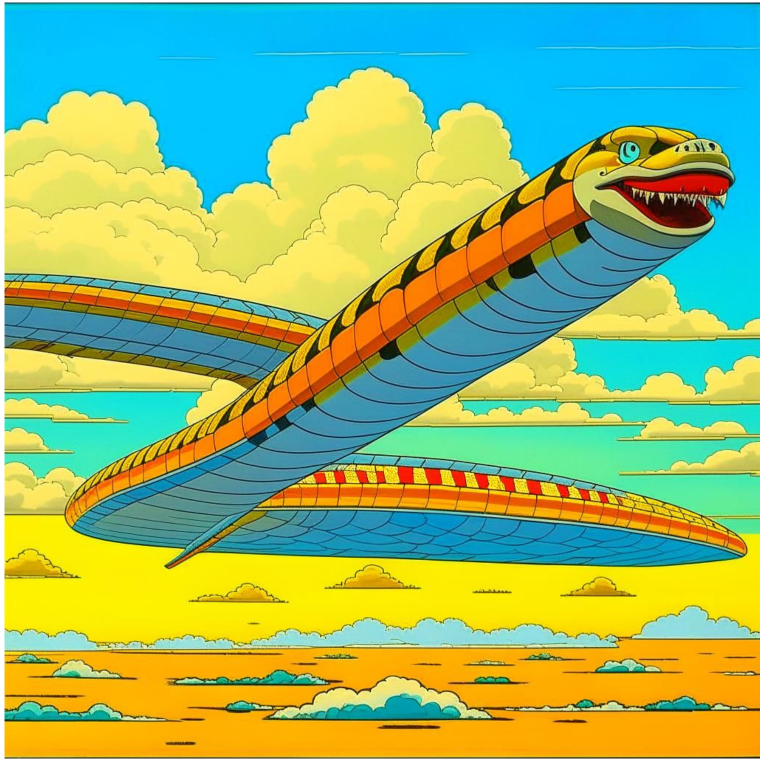
- memory consumption graph
- display highlighted functions
- line-by-line analysis

Practical task

1. Run memory_profiler for generate_scene1 function from file 03_memory_profiler/shapes.py
2. Record how much memory was consumed by Points allocation
3. Think of a way to reduce memory consumption
4. Measure the result
5. Write in chat how much Mb you managed to reduce

Profile on the fly

- sampling
- py-spy
- record
- top
- flamegraphs



Practical task

1.	Run social net client **python 04_profile_on_the_fly/social_net.py**
2.	Attach with py-spy and collect profile flamegraph
3.	Find out from flamegraph wich of service endpoints is slow

Conclusion