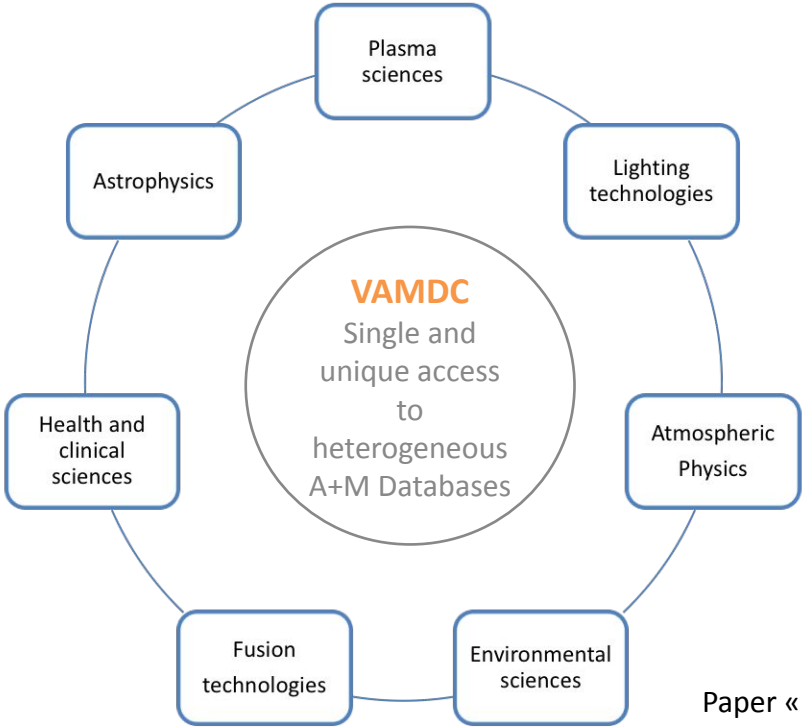
An overview of pyVAMDC functionalities

ASOV 2026 - 09/03/2025

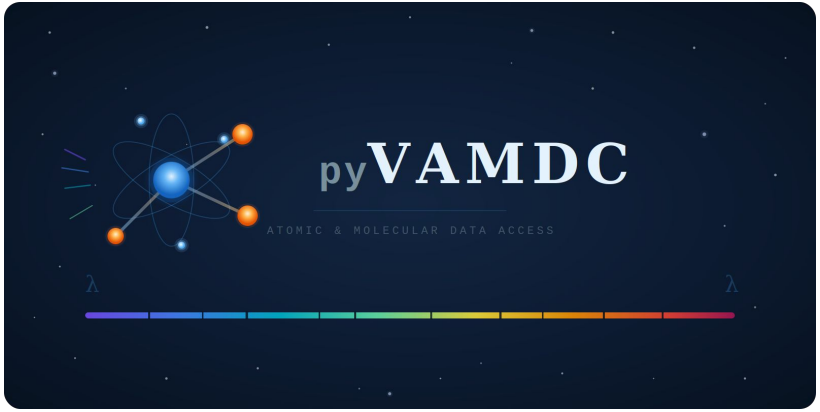
Carlo Maria Zwölf, Nicolas Moreau, Yaye-Awa Ba and VAMDC Contributors

The Virtual Atomic and Molecular Data Centre in a nutshell



- E-infrastructure connecting about 40 heterogeneous databases (*Nodes*) that can be accessed from http://portal.vamdc.org/ or any VAMDC compatible tools
- Consortium of 25 partners
- High quality scientific data come from different Physical/Chemical Communities
- Provides a large dissemination platform to data producers
- **Interoperability of queries and output**

Paper « A decade with VAMDC : results and ambition, Atoms, 2020 »
http://dx.doi.org/10.3390/atoms8040076



- Making access and usage of VAMDC infrastructure and data as easy as possible for astronomers

- Overcoming certain limitations of the VAMDC
  - Data truncation in case of big queries
  - Classic VO-protocols (from 2000 and 2010) badly scale for big-data

pyVAMC - more than just a Python library...

pyVAMDC

pyVAMC - more than just a Python library...

pyVAMDC

Species
Discover species by name, formula, mass, chemical intelligence

Data-Nodes
Information about available TAP endpoints

Lines
Get Spectral lines by wavelength ranges or telescopes bands

Collision (beta)
Get collisional data by species (target and collider)

Energy converter
Unit conversion

Filters
Filtering utilities: Numeric range, string matching

pyVAMC - more than just a Python library...

pyVAMDC

Species
Discover species by name, formula, mass, chemical intelligence

Data-Nodes
Information about available TAP endpoints

Lines
Get Spectral lines by wavelength ranges or telescopes bands

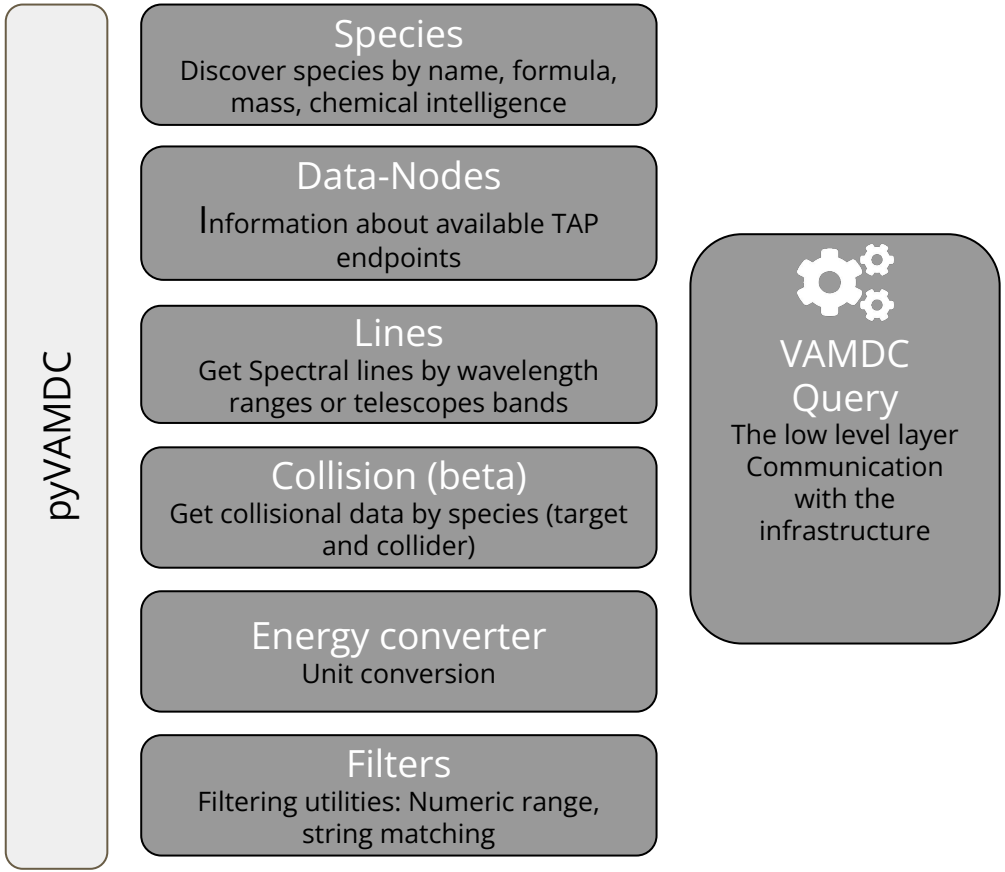
Collision (beta)
Get collisional data by species (target and collider)

Energy converter
Unit conversion

Filters
Filtering utilities: Numeric range, string matching

VAMDC Query
The low level layer Communication with the infrastructure

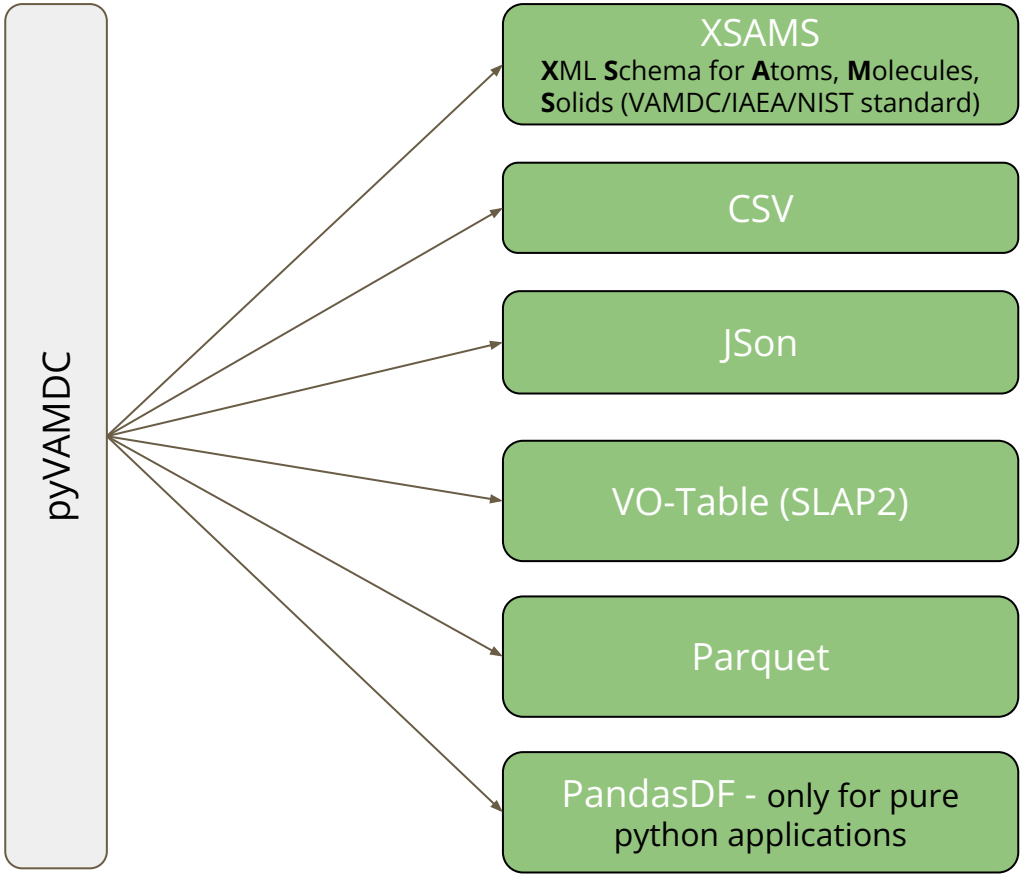
pyVAMC - more than just a Python library...



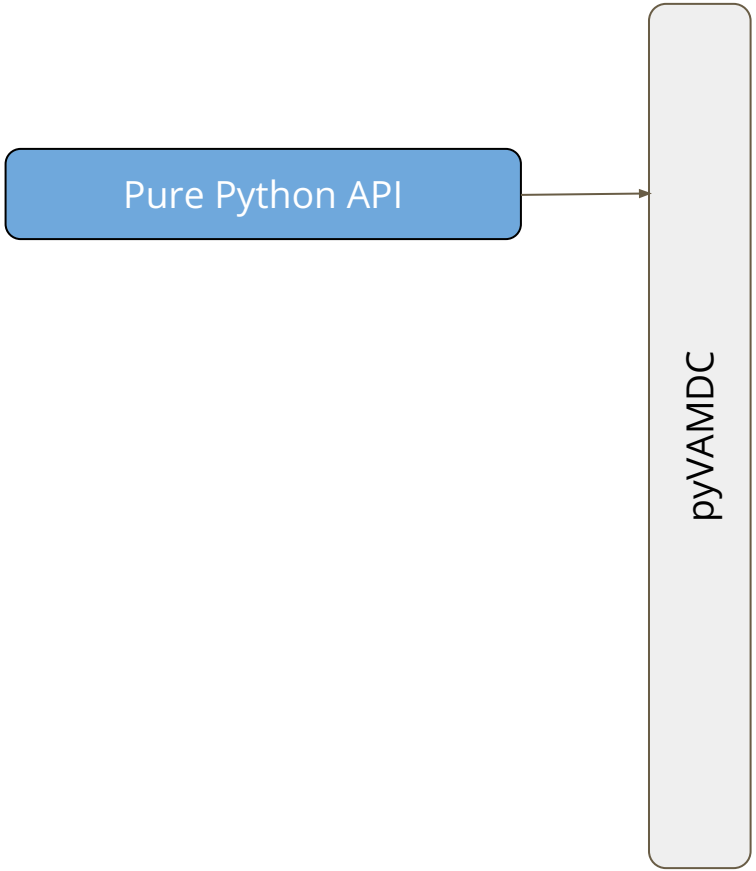
The game changer: recursvive paralle query splitting

- When a query would return truncated results, pyVAMDC automatically splits the wavelength range in half and retries recursively until every sub-query completes without truncation
- Non-truncated sub-queries are executed and reassembled using Parquet → **Complete dataset, memory-efficient**
- Automatic process invisible to the users
- Parrallel execution with semaphore per-node concurrency control → **Fast without overloading the data-nodes**.

Multiple output formats



Multiple interfaces



Multiple interfaces

Pure Python API

pyVAMDC

```
def getAllSpecies():
```

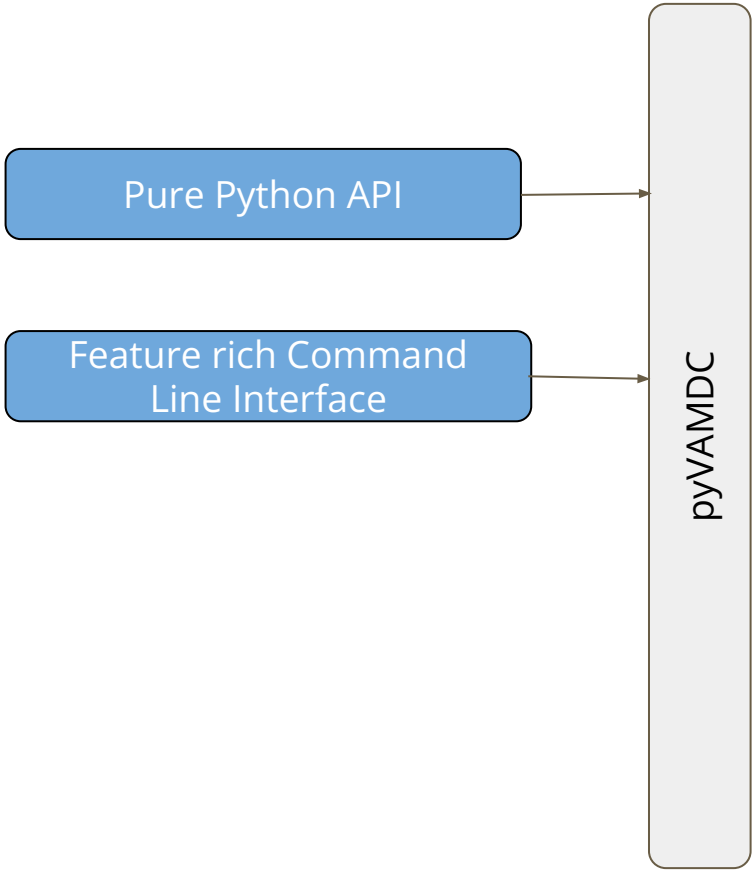
```
def getNodeHavingSpecies():
```

```
def get_metadata_for_lines(lambdaMin, lambdaMax, species_dataframe = None, nodes_dataframe = None, max_concurrent_per_node = 3):
```

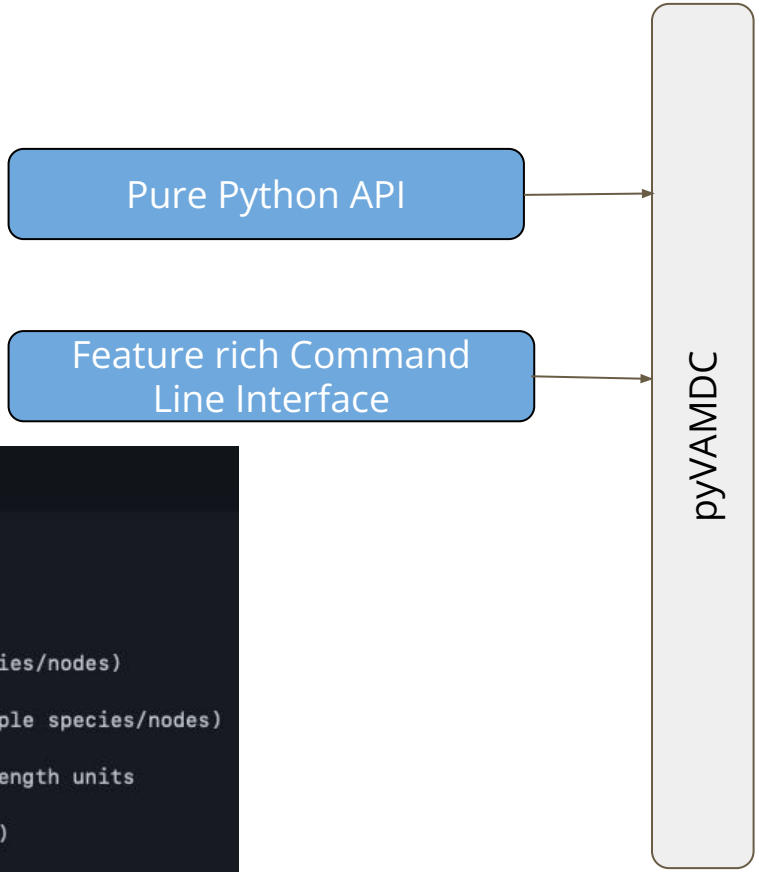
```
def getLines(lambdaMin, lambdaMax, species_dataframe = None, nodes_dataframe = None, acceptTruncation = False, max_concurrent_per_node = 3):
```

```
def getLinesAsDataFrames(lambdaMin, lambdaMax, species_dataframe=None, nodes_dataframe=None, acceptTruncation=False, max_concurrent_per_node=3):
```

Multiple interfaces



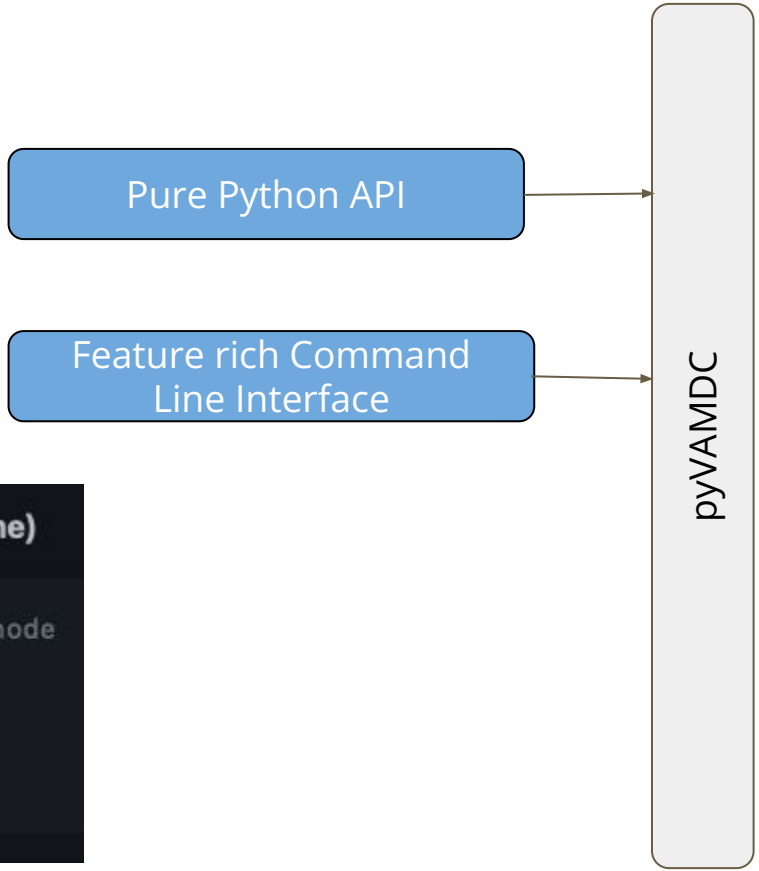
Multiple interfaces



The CLI is organized into command groups:

```
vamdc
├── get          # Retrieve data from VAMDC
│   ├── nodes    # List available data nodes
│   ├── species  # List chemical species
│   └── lines    # Query spectral lines (supports multiple species/nodes)
├── count        # Inspect metadata without downloading
│   └── lines    # Get line counts and metadata (supports multiple species/nodes)
├── convert      # Perform unit conversions
│   └── energy   # Convert between energy, frequency, and wavelength units
└── cache        # Manage local cache
    ├── status   # Show cache information (includes XSAMS files)
    └── clear    # Remove cached data
```

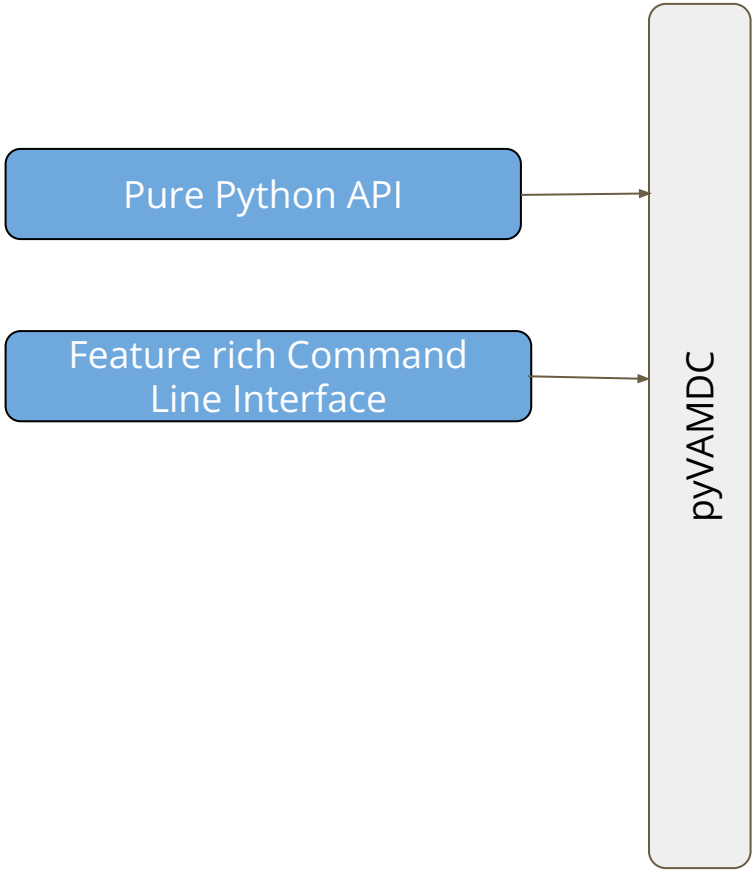
Multiple interfaces



Query all species from a specific node (using short name)

```
# Get metadata for all species from a specific node
vamdc count lines \
  --node=topbase \
  --lambda-min=0 \
  --lambda-max=90009076900
```

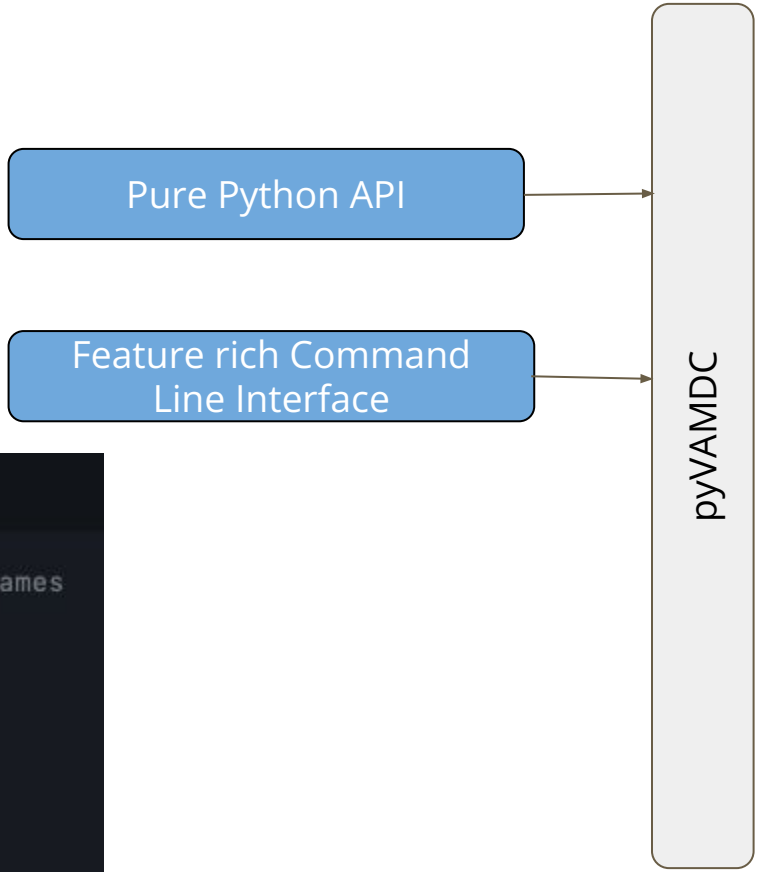
Multiple interfaces



Multiple species, single node (using short name)

```
# Query CO and H2O from CDMS using short name
vamdc get lines \
  --inchikey=LFQSCWFLJHTTHZ-UHFFFAOYSA-N \
  --inchikey=XLYOFNOQVPJJNP-UHFFFAOYSA-N \
  --node=cdms \
  --lambda-min=100000 \
  --lambda-max=200000
```

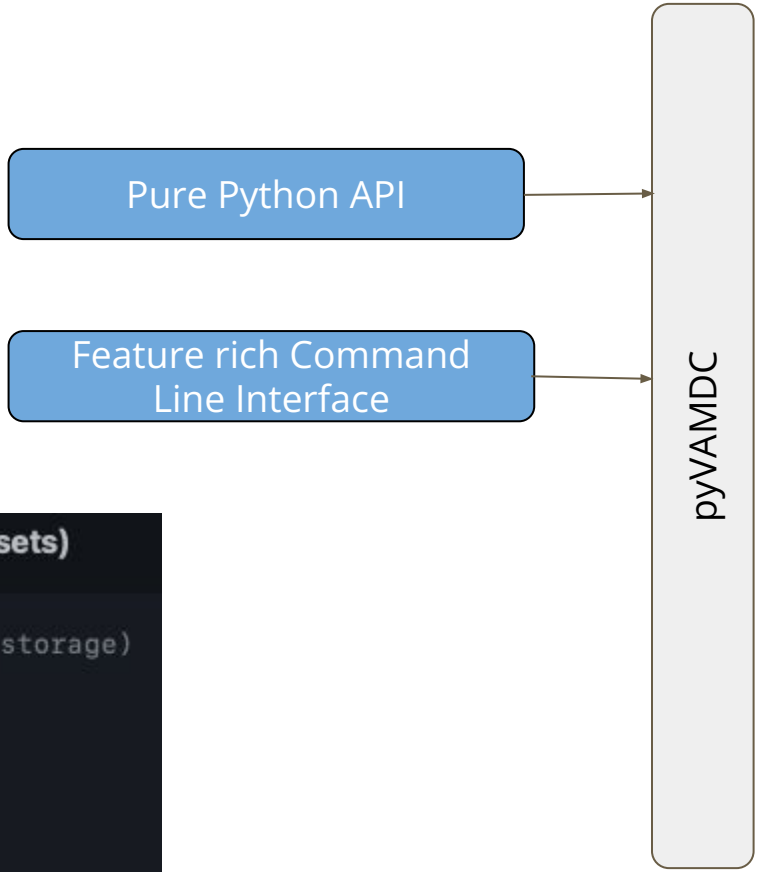
Multiple interfaces



Single species, multiple nodes (using short names)

```
# Query CO from multiple databases using short names
vamdc get lines \
  --inchikey=LFQSCWFLJHTTHZ-UHFFFAOYSA-N \
  --node=cdms \
  --node=jpl \
  --node=basecol2015 \
  --lambda-min=100000 \
  --lambda-max=200000
```

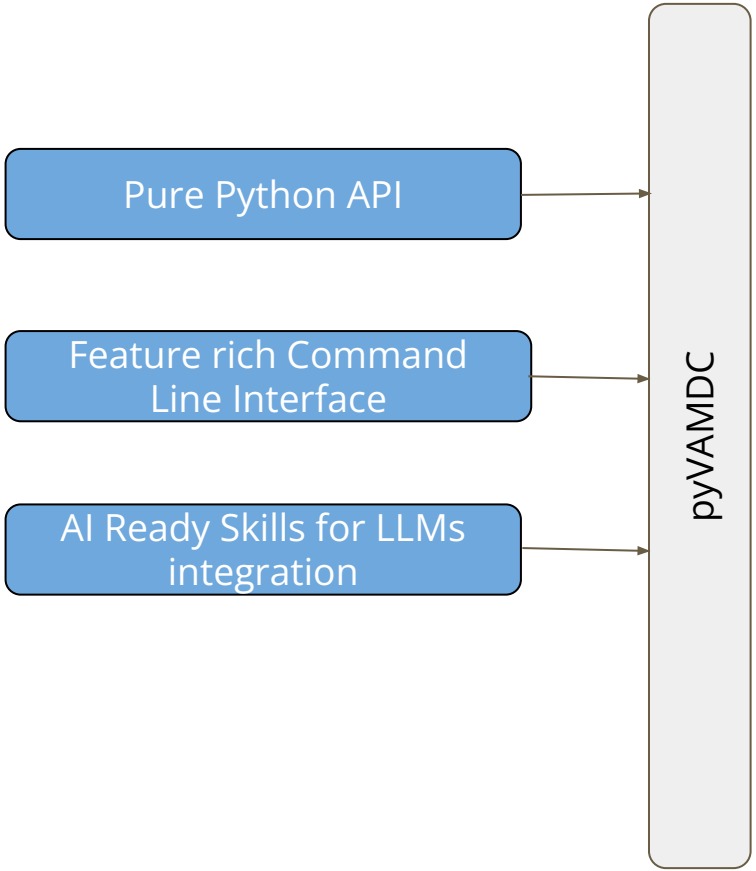
Multiple interfaces



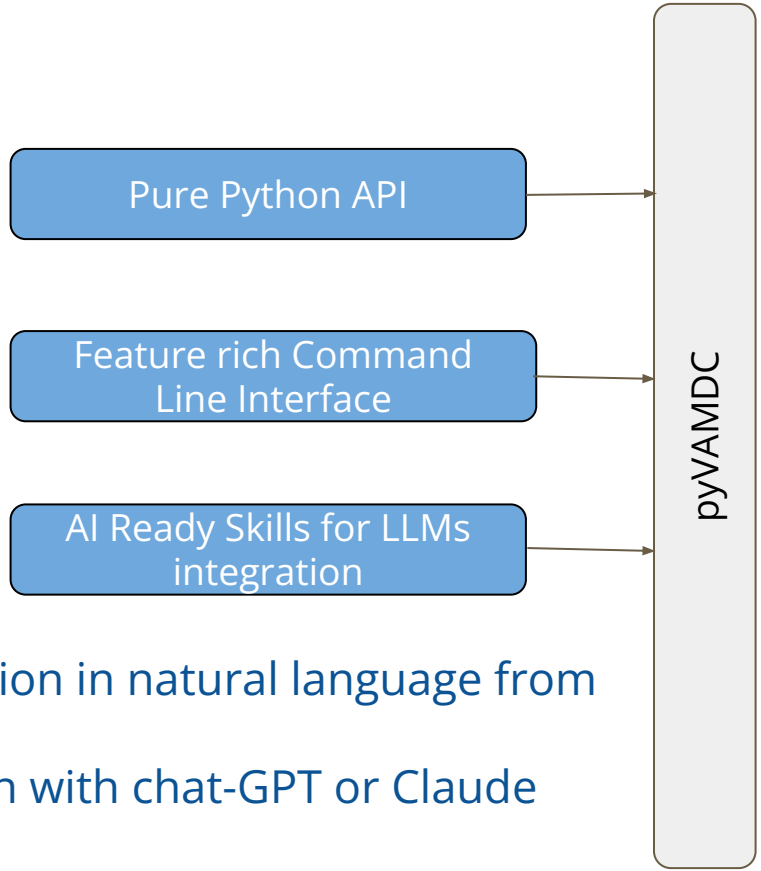
Parquet format output (memory-efficient for large datasets)

```
# Get data as parquet files (efficient columnar storage)
vamdc get lines \
  --inchikey=LFQSCWFLJHTTHZ-UHFFFAOYSA-N \
  --lambda-min=100000 \
  --lambda-max=200000 \
  --format parquet
```

Multiple interfaces



Multiple interfaces



- Drive pyVAMDC execution in natural language from LLMs.
- Complex task execution with chat-GPT or Claude integration
- *Are you intrigued? Come to the semi-hack-at-on ;-)*

Conclusion and discussions

- pyVAMDC is a comprehensive toolkit for accessing atomic and molecular data
- It combines a Python library, a command-line interface and AI-ready skills for seamless integration into modern research workflows.
- **https://github.com/VAMDC/pyVAMDC/**

- **Discussion…**