RDKit from Java

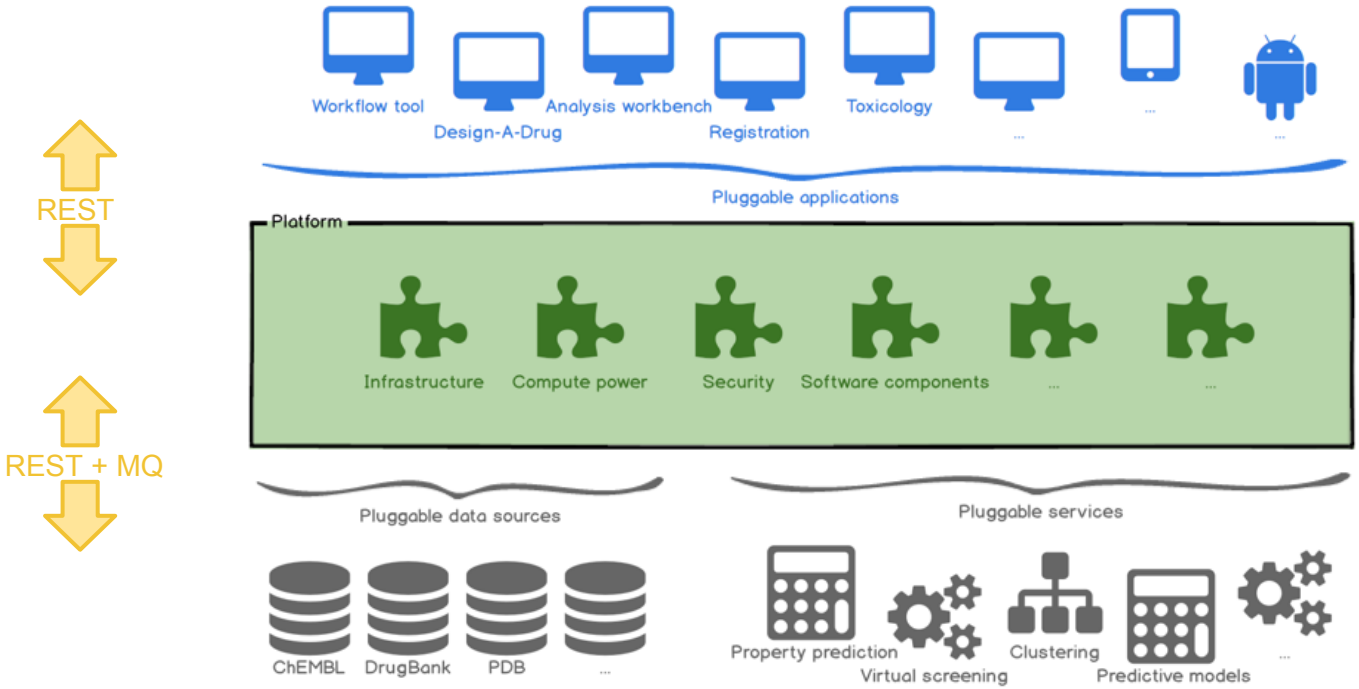
lessons learnt and a plea for help

Tim Dudgeon
Informatics Matters

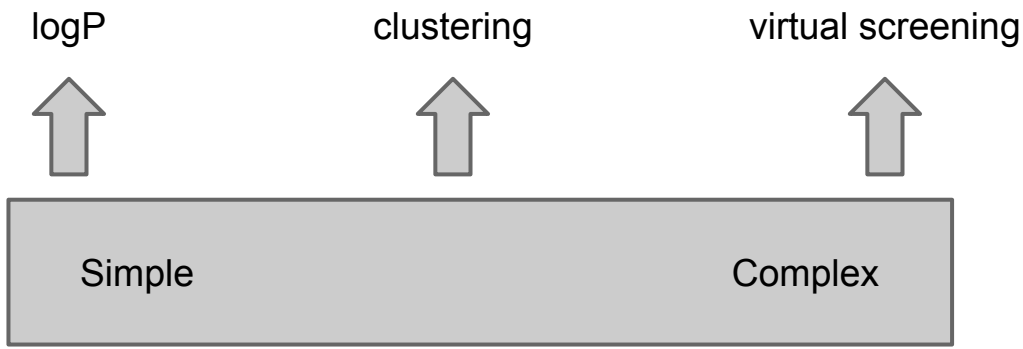
Our aim is to provide ...

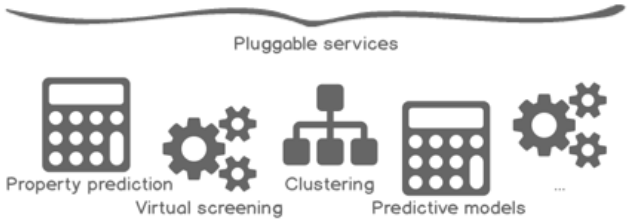
- An interoperable cheminformatics platform …
- integrating commercial and OS tools ...
- letting you use those tools without needing to know how they are implemented
- Aimed at users not geeks

Platform overview



REST services





RDKit
Open-Source Cheminformatics and Machine Learning

CDK

ChemAxon

OpenChemLib

Why Java?

- Leading language for enterprise software
- High performance
- Strongly typed
- Strong tooling support
- Wide range of 3rd party libraries
- Many languages run on JVM
- Its what we know best

So how to access RDKit from Java?

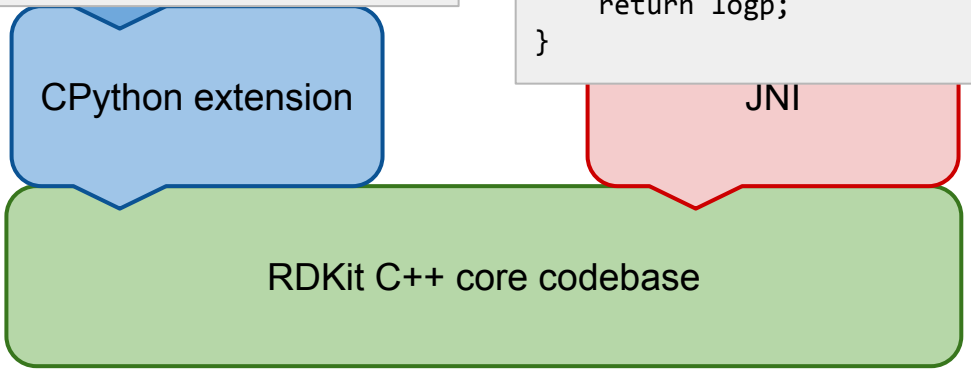
Core RDKit Architecture

```
from rdkit import Chem
from rdkit.Chem import Descriptors

m = Chem.MolFromSmiles('Cc1ccccc1')
logp = Descriptors.MolLogP(m)
print "LogP = " + str(logp)
```

```
import org.RDKit.RWMol;
import org.RDKit.RDKFuncs;

public static double calcLogp(String smiles) {
    double logp = RDKFuncs.calcMolLogP(mol);
    System.out.println("LogP = " + logp);
    return logp;
}
```



How to access the Java stuff?

- Must build from source (#)
- “Just” a matter of stating to create the SWIG wrappers
- Dockerfiles and images available

Windows downloads with Java are available

Basic RDKit Dockerfile (no Java)

```
FROM debian:jessie
MAINTAINER Tim Dudgeon <tdudgeon@informaticsmatters.com>
# WARNING this takes about an hour to build

ENV RDKIT_BRANCH=Release_2015_03_1

RUN apt-get update && apt-get install -y \
 flex\
 bison\
 build-essential\
 python-numpy\
 cmake\
 python-dev\
 sqlite3\
 libsqlite3-dev\
 libboost-dev\
 libboost-python-dev\
 libboost-regex-dev\
 swig2.0\
 git

RUN git clone -b $RDKIT_BRANCH --single-branch https://github.
com/rdkit/rdkit.git

ENV RDBASE=/rdkit

RUN mkdir $RDBASE/build
WORKDIR $RDBASE/build
RUN cmake ..
RUN make
RUN make install

ENV LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$RDBASE/lib
ENV PYTHONPATH=$PYTHONPATH:$RDBASE

WORKDIR $RDBASE
```

https://hub.docker.com/r/informaticsmatters/rdkit/

RDKit + Java Dockerfile

```
# currently need backports as openjdk-8-jdk not available on vanilla jessie
FROM debian:jessie-backports
MAINTAINER Tim Dudgeon <tdudgeon@informaticsmatters.com>
# WARNING this takes about an hour to build

ENV RDKIT_BRANCH=Release_2015_03_1

RUN apt-get update && apt-get install -y \
 flex\
 bison\
 build-essential\
 python-numpy\
 cmake\
 python-dev\
 sqlite3\
 libsqlite3-dev\
 libboost-dev\
 libboost-python-dev\
 libboost-regex-dev\
 swig2.0\
 git\
 openjdk-8-jdk\
 wget

RUN git clone -b $RDKIT_BRANCH --single-branch https://github.com/rdkit/rdkit.git

ENV RDBASE=/rdkit
ENV JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

RUN mkdir $RDBASE/External/java_lib
RUN wget -O $RDBASE/External/java_lib/junit.jar http://search.maven.
org/remotecontent?filepath=junit/junit/4.12/junit-4.12.jar
RUN wget -O $RDBASE/External/java_lib/hamcrest-core.jar http://search.maven.
org/remotecontent?filepath=org/hamcrest/hamcrest-core/1.3/hamcrest-core-1.3.jar

RUN mkdir $RDBASE/build
WORKDIR $RDBASE/build
RUN cmake -D RDK_BUILD_SWIG_WRAPPERS=ON ..
RUN make
RUN make install

ENV LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$RDBASE/lib:
/rdkit/lib:$RDBASE/Code/JavaWrappers/gmwrapper
ENV PYTHONPATH=$PYTHONPATH:$RDBASE
ENV CLASSPATH=$RDBASE/Code/JavaWrappers/gmwrapper/org.RDKit.jar

WORKDIR $RDBASE
```

https://hub.docker.com/r/informaticsmatters/rdkit_java/

Run Java stuff

- Set environment variables:
  - export LD_LIBRARY_PATH=/rdkit/lib:/rdkit/Code/JavaWrappers/gmwrapper
  - export CLASSPATH=<*your_libs*>:/rdkit/Code/JavaWrappers/gmwrapper/org.RDKit.jar

- Load the native RDKit libs into Java

```
static {
    System.loadLibrary("GraphMolWrap");
}
```

What is available from Java

- Not all functionality is available through Java
- Finding what is available is quite hard!
  - Documentation and examples are limited
  - JavaDocs are pretty basic (#)
- Need to study the Python and C++ docs
- “Package” structure can be quite different, but the underlying functions are much the same

JavaDocs are built under /rdkit/Code/JavaWrappers/gmwrapper/doc

1. Molecule IO

- RWMol.MolFrom*()
  - RWMol.MolFromSmiles()
  - RWMol.MolFromSmarts()
  - RWMol.MolFromMolfile()
  - RWMol.MolFromPDBFile()
  - ...

```
import org.RDKit.RWMol;

public class SimpleSmiles {

   static {
      System.loadLibrary("GraphMolWrap");
   }

   public static void main(String[] args) {
      RWMol mol = RWMol.MolFromSmiles("Cc1ccccc1");
      System.out.println("Mol = " + mol);
   }
}
```

1. Molecule IO

- *MolSupplier classes
  - SDMolSupplier
  - SmilesMolSupplier
  - PDBMolSupplier
  - TDTMolSupplier

```
import org.RDKit.ROMol;
import org.RDKit.RDKFuncs;
import org.RDKit.SDMolSupplier;

public class SdfReader {

   static { System.loadLibrary("GraphMolWrap"); }

   public static void main(String[] args) {

      String filename = args[0];
      long t0 = System.currentTimeMillis();
      int count = 0;
      int errors = 0;

      SDMolSupplier sdf = new SDMolSupplier(filename, true, false);
      while (!sdf.atEnd()) {
         count++;
         ROMol mol = sdf.next();
         if (mol == null) {
            errors++;
         }
      }

      long t1 = System.currentTimeMillis();
      System.out.println("Processed " + count + " mols in " + (t1-t0) +
          "ms. " + errors + " errors");
   }
}
```

RDKFuncs

- Lots of useful stuff resides here
- Molecule operations
- Property calculations/predictions
- Descriptors/Fingerprints
- MCS

RDKFuncs: Molecule operations

- Aromatize/Kekulize
- Hydrogen addition/removal
- Canonicalization
- Export
- … and much more

```
import org.RDKit.RDKFuncs;
import org.RDKit.RWMol;

public class MoleculeFunctions {

    static { System.loadLibrary("GraphMolWrap"); }

    public static void main(String[] args) {
        RWMol mol = RWMol.MolFromSmiles(args.length == 0 ?
            "Cc1ccccc1" : args[0]);
        System.out.println("Input: " + mol.MolToSmiles());
        RDKFuncs.Kekulize(mol);
        System.out.println("Kekule: " + mol.MolToSmiles());
        RDKFuncs.setAromaticity(mol);
        System.out.println("Aromatic: " + mol.MolToSmiles());
        RDKFuncs.addHs(mol);
        System.out.println("Hydrogens: " + mol.MolToSmiles());
    }
}
```

```
Input: Cc1ccccc1
Kekule: CC1=CC=CC=C1
Aromatic: Cc1ccccc1
Hydrogens: [H]c1c([H])c([H])c(C([H])([H])[H])c([H])c1[H]
```

RDKFuncs: Property calcs

- LogP
- TPSA
- Counts of *
- … and much more

mostly using the calcXxx() functions

```
public static void calculate(ROMol rdkitMol) {
    double logp = RDKFuncs.calcMolLogP(rdkitMol);
    double mw = RDKFuncs.calcExactMW(rdkitMol);
    double fsp3 = RDKFuncs.calcFractionCSP3(rdkitMol);
    long hba = RDKFuncs.calcNumHBA(rdkitMol);
    long hbd = RDKFuncs.calcNumHBD(rdkitMol);
    long lhba = RDKFuncs.calcLipinskiHBA(rdkitMol);
    long lhbd = RDKFuncs.calcLipinskiHBD(rdkitMol);
    String mf = RDKFuncs.calcMolFormula(rdkitMol);
    double mr = RDKFuncs.calcMolMR(rdkitMol);
    long numHetero = RDKFuncs.calcNumHeteroatoms(rdkitMol);
    long numRings = RDKFuncs.calcNumRings(rdkitMol);
    long numAromRings = RDKFuncs.calcNumAromaticRings(rdkitMol);
    long numRotBonds = RDKFuncs.calcNumRotatableBonds(rdkitMol);
    double tpsa = RDKFuncs.calcTPSA(rdkitMol);
}
```

RDKFuncs: MCS

```
public static ROMol_Vect createMolVectorFromSmiles(String[] smiles) {
    ROMol_Vect rovect = new ROMol_Vect();
    for (String smi : smiles) {
        RWMol mol = RWMol.MolFromSmiles(smi);
        RDKFuncs.setAromaticity(mol);
        if (mol != null) {
            rovect.add(mol);
        } else {
            throw new IllegalArgumentException("Could not read " + smi);
        }
    }
    return rovect;
}

public static MCSResult findMCS(ROMol_Vect mols) {
    return RDKFuncs.findMCS(mols);
}
```

```
public static MCSResult findMCS(
    ROMol_Vect mols,
    boolean maximizeBonds,
    double threshold,
    long timeout,
    boolean verbose,
    boolean matchValences,
    boolean ringMatchesRingOnly,
    boolean completeRingsOnly,
    boolean matchChiralTag,
    AtomComparator atomComp,
    BondComparator bondComp) {

        return RDKFuncs.findMCS(
        mols, maximizeBonds, threshold, timeout,
        verbose, matchValences,
        ringMatchesRingOnly, completeRingsOnly,
        matchChiralTag, atomComp, bondComp);
    }
```

RDKFuncs: Other

- Fingerprints
- Descriptors
- Similarity
- Path operations

… and lots more

Performance

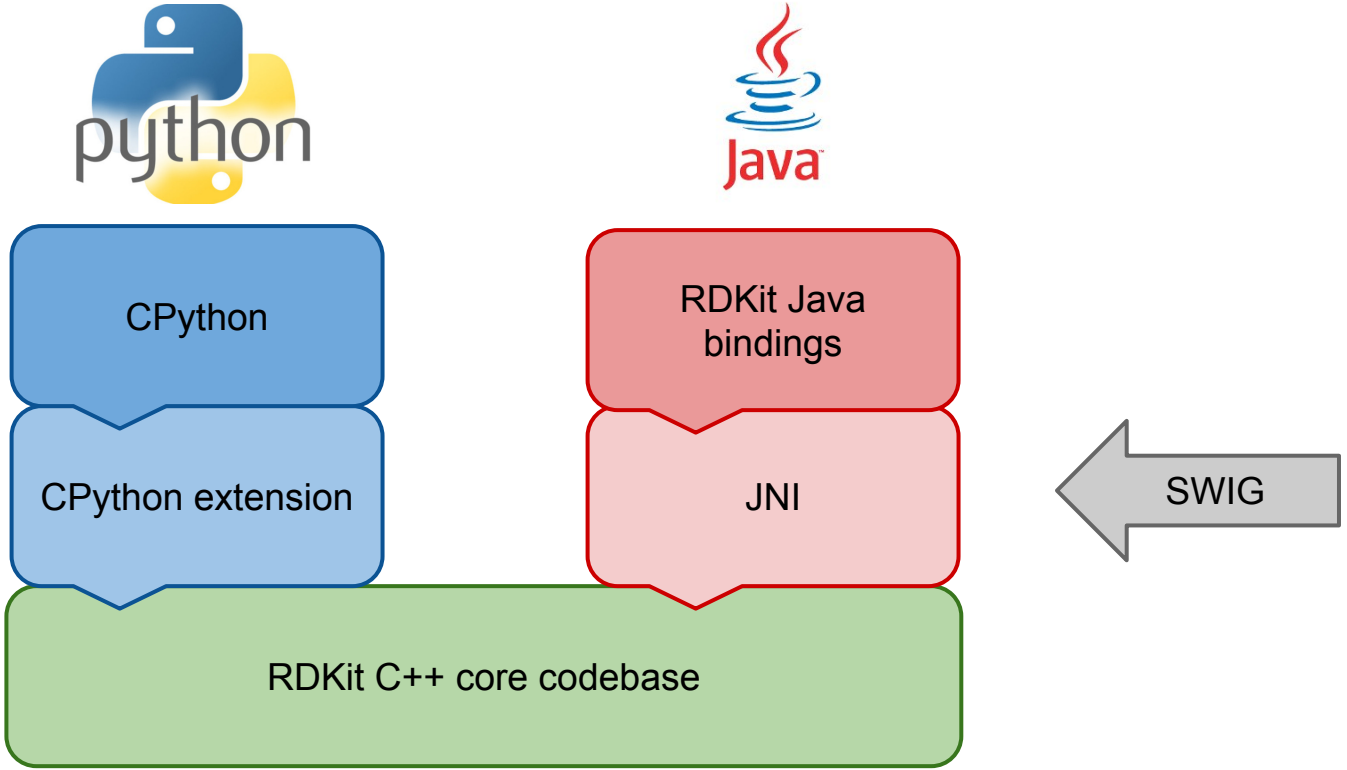
Task	Python	Java
Parse SDF	15.7	16.5
Parse SDF + calc Molar refractivity	55.6	54.7
Parse SDF + calc Labute ASA	16.6	17.2
Parse SDF + calc Murcko framework	23.0	23.2

SDF contained 58855 structures.
Times in seconds, average of 3 runs.

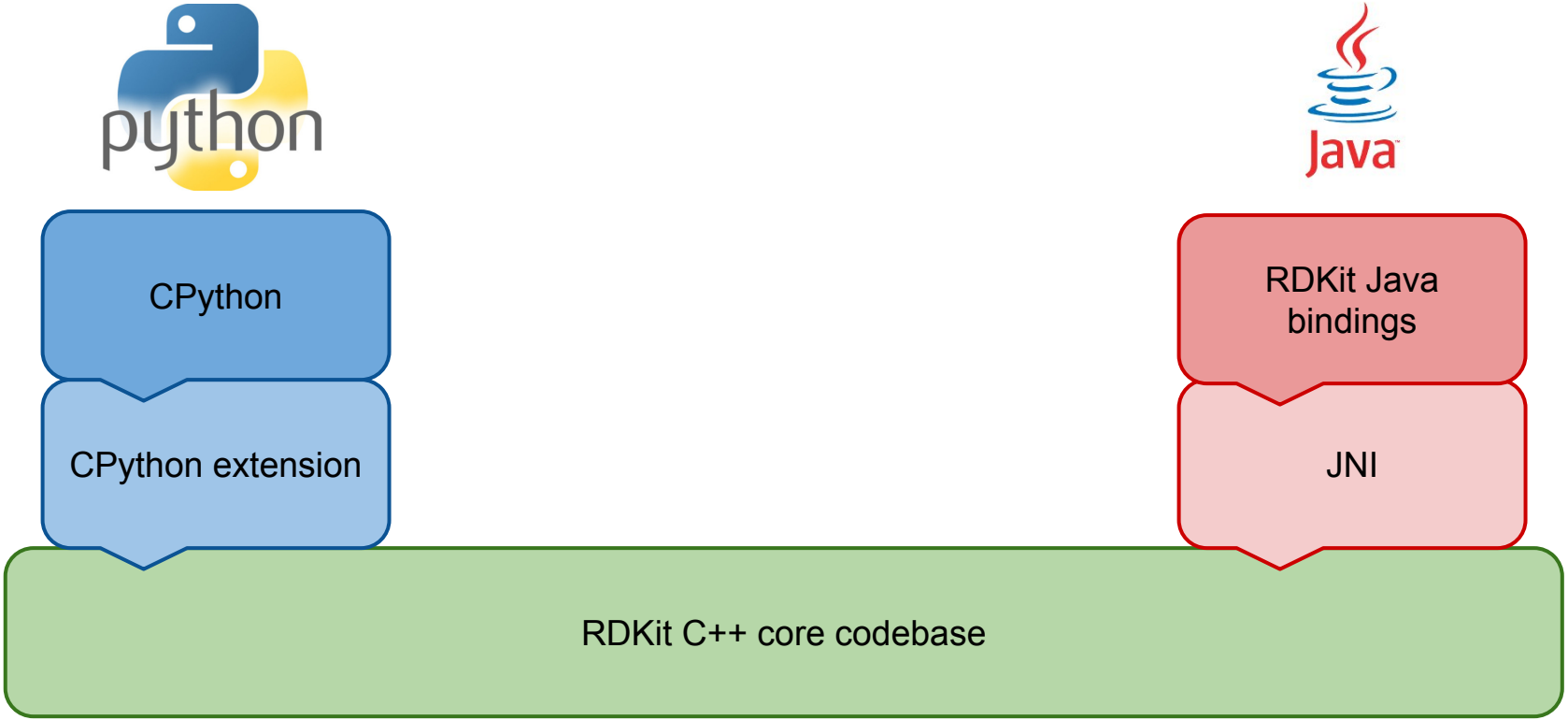
How to use RDKit?

Python or Java

Core RDKit Architecture

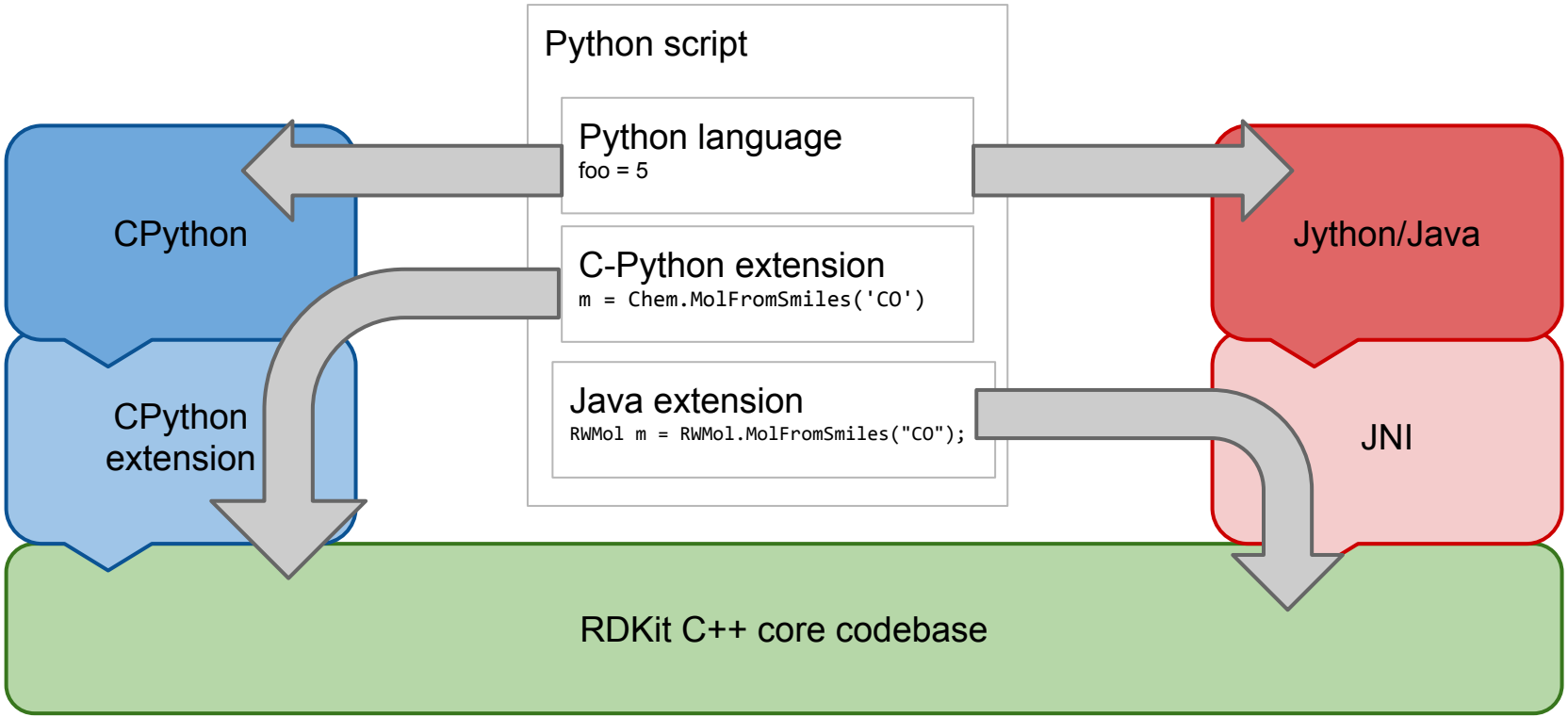


Can the 2 worlds meet?



Core RDKit Architecture

CPython != Jython



Approaches to bridging the chasm

JyNI: http://jyni.org/

- Jython extension providing access to C extensions using JNI

JPY: https://github.com/bcdev/jpy

JEP: https://github.com/mrj0/jep

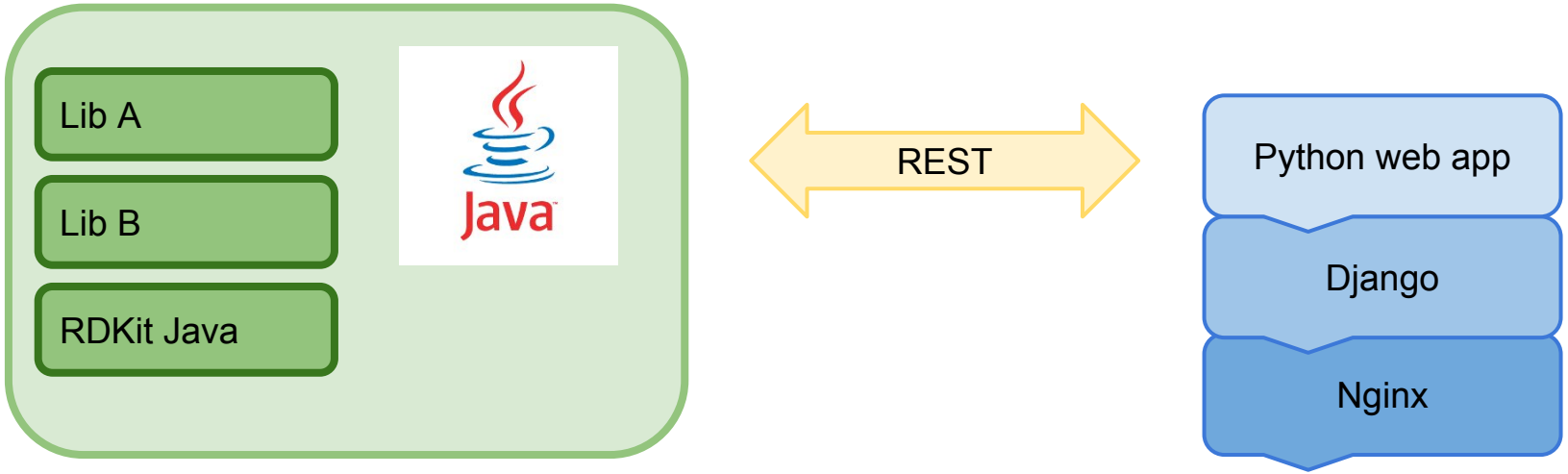
- embed CPython interpreter in Java

Anyone got any experience with these?

So what have we learnt?

- Using RDKit from Java is perfectly viable
  - But it's harder work than from Python
  - And not everything is available
- A better alternative can be to use Python and access as a service e.g. REST web service

So what have we learnt?



Faster/simpler operations

- property calculations
- molecule operations

Slower/complex operations

- screening
- clustering

The plea for help

- Who's interested in improving RDKit for Java?
  - General docs
  - Javadocs
  - Examples

Resources

Docker images:

- RDKit alone: https://hub.docker.com/r/informaticsmatters/rdkit/
- RDKit + Java: https://hub.docker.com/r/informaticsmatters/rdkit_java/
- RDKit + Java + Tomcat7: https://hub.docker.com/r/informaticsmatters/rdkit_tava_tomcat/
- PostgreSQL + RDKit cartridge: <in progress>

Code examples:

- https://github.com/InformaticsMatters/rdkit_java_examples

Contact me:

tdudgeon _at_ informaticsmatters _dot_ com