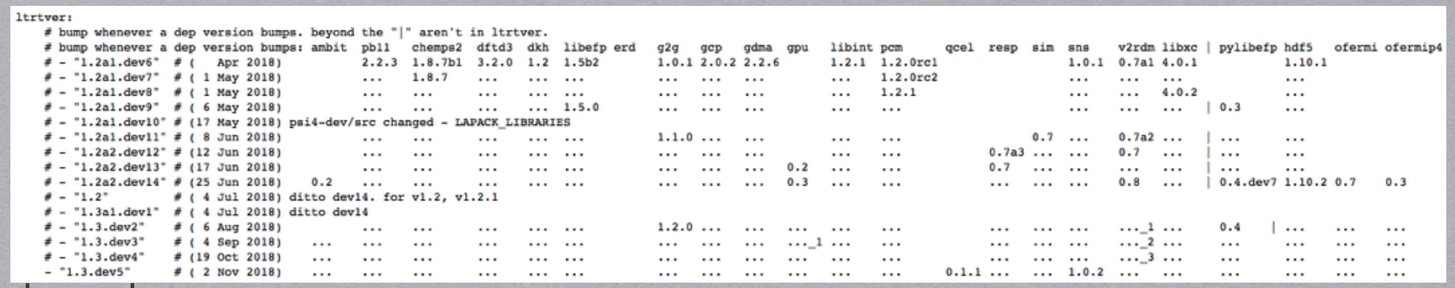
2018 Psi4 Ordnance Survey

LORI A. BURNS

9 NOV 2018

TESTING
PROJECT COV & PYTEST

- Ecosystem growing. How do we know when it's broken in general (syntax fault/incompatibility)? How do we know when a binary distribution is broken (build fault/environment fault)?
- For "broken in general",
  - Psi4 Travis tests several addons (d<addon>/d<psi4 commit>)
  - Addon Travis tests against Psi4 (d<psi4>/d<addon commit>)
  - Psi4 conda build tests all addons.



```
ltrtver:
    # bump whenever a dep version bumps. beyond the "|" aren't in ltrtver.
    # bump whenever a dep version bumps: ambit  pbll   chemps2  dftd3  dkh  libefp erd   g2g   gcp   gdma  gpu   libint pcm      qcel  resp  sim  sns    v2rdm libxc | pylibefp hdf5   ofermi ofermip4
    # - "1.2a1.dev6"  # (   Apr 2018)           2.2.3  1.8.7b1  3.2.0  1.2  1.5b2       1.0.1 2.0.2 2.2.6       1.2.1  1.2.0rc1                 1.0.1  0.7a1 4.0.1           1.10.1
    # - "1.2a1.dev7"  # ( 1 May 2018)           ...    1.8.7    ...    ...  ...         ...   ...   ...         ...    1.2.0rc2                 ...    ...   ...             ...
    # - "1.2a1.dev8"  # ( 1 May 2018)           ...    ...      ...    ...  ...         ...   ...   ...         ...    1.2.1                    ...    ...   4.0.2           ...
    # - "1.2a1.dev9"  # ( 6 May 2018)           ...    ...      ...    ...  1.5.0       ...   ...   ...         ...    ...                      ...    ...   ...   | 0.3       ...
    # - "1.2a1.dev10" # (17 May 2018) psi4-dev/src changed - LAPACK_LIBRARIES
    # - "1.2a1.dev11" # ( 8 Jun 2018)           ...    ...      ...    ...  ...         1.1.0 ...   ...         ...    ...                      0.7    ...   0.7a2 ...   | ...       ...
    # - "1.2a2.dev12" # (12 Jun 2018)           ...    ...      ...    ...  ...         ...   ...   ...         ...    ...              0.7a3   ...    ...   0.7   ...   | ...       ...
    # - "1.2a2.dev13" # (17 Jun 2018)           ...    ...      ...    ...  ...         ...   ...   ...   0.2   ...    ...              0.7     ...    ...   ...   ...   | ...       ...
    # - "1.2a2.dev14" # (25 Jun 2018)    0.2    ...    ...      ...    ...  ...         ...   ...   ...   0.3   ...    ...              ...     ...    ...   0.8   ...   | 0.4.dev7  1.10.2 0.7    0.3
    # - "1.2"         # ( 4 Jul 2018) ditto dev14. for v1.2, v1.2.1
    # - "1.3a1.dev1"  # ( 4 Jul 2018) ditto dev14
    # - "1.3.dev2"    # ( 6 Aug 2018)           ...    ...      ...    ...  ...         1.2.0 ...   ...   ...   ...    ...              ...     ...    ...   ...._1 ...   0.4     | ...       ...    ...
    # - "1.3.dev3"    # ( 4 Sep 2018)    ...    ...    ...      ...    ...  ...         ...   ...   ...   ..._1 ...    ...              ...     ...    ...   ...._2 ...   ...       ...       ...    ...
    # - "1.3.dev4"    # (19 Oct 2018)    ...    ...    ...      ...    ...  ...         ...   ...   ...   ...   ...    ...              ...     ...    ...   ...._3 ...   ...       ...       ...    ...
    - "1.3.dev5"      # ( 2 Nov 2018)    ...    ...    ...      ...    ...  ...         ...   ...   ...   ...   ...    ...      0.1.1   ...     ...   1.0.2 ...   ...   ...       ...       ...    ...
```

- For "broken in particular", need **runtime** testing.
  - Suggest keeping existing test suite but run it through pytest, not ctest
  - Test suite can be installed and run after `conda install`, for example.
  - Also can accumulate more units tests to aid refactoring
  - Pytest allows parameterization, testing for errors, env-dependent test activation

GLOBALS: THE LONG WAR

WE'RE WINNING

EFP, efp_, efp_torque_, efp(), set_efp(), efp_torque(), set_efp_torque()

ExternalPotential

frequencies_, set_frequencies(), frequencies()

P::e.globals, P::e.options_

KILLED THIS YEAR

REMAIN TROUBLESOME

OTHER ITEMS YOU MIGHT GO LOOKING FOR AND NOT FIND

thermo
→
qcdb.vib

HF::iterations()
→
psi4.driver.procrouting.scf.scf_iterate()

Molecule::create_molecule_from_string
→
qcel.molparse

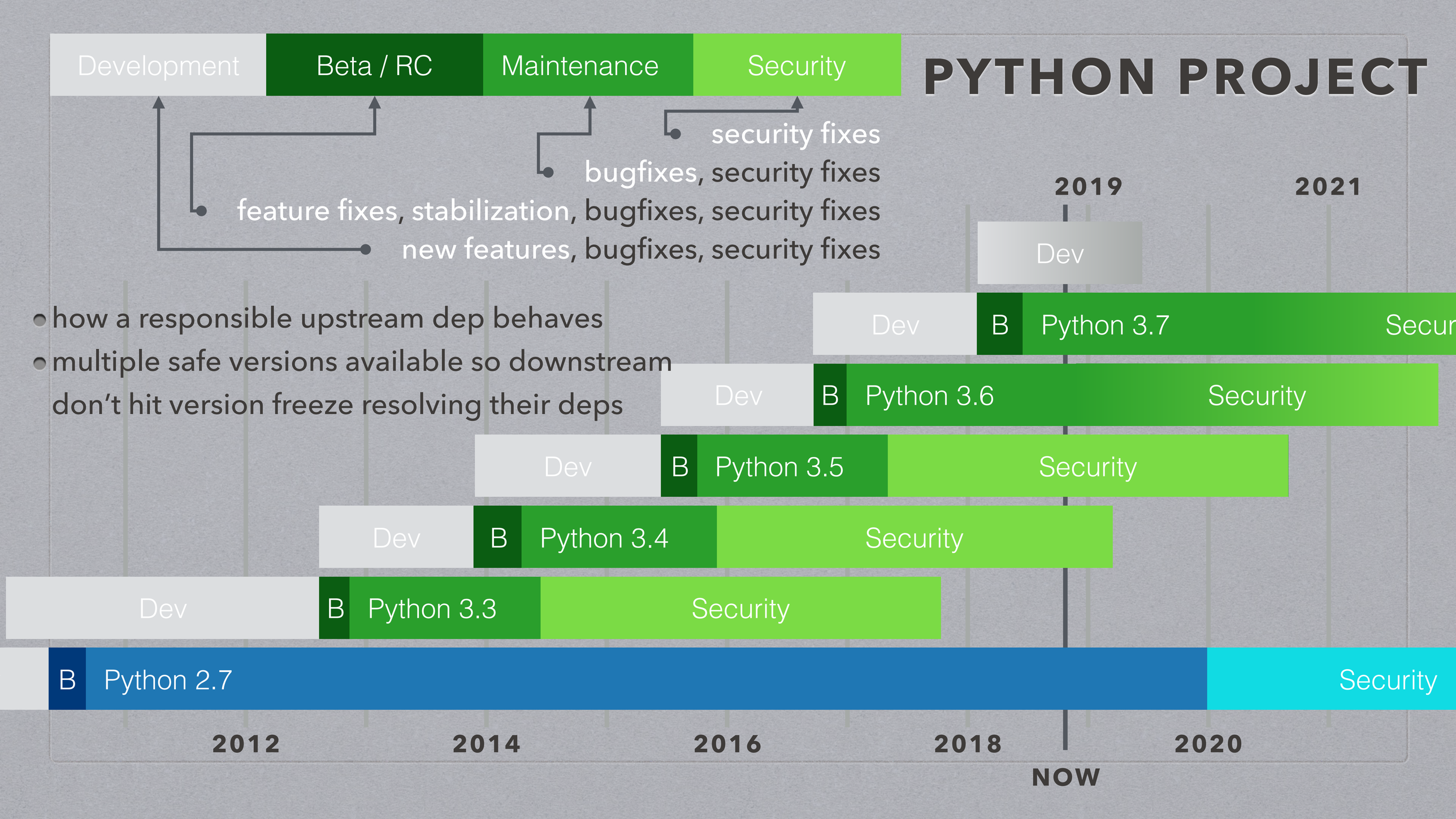
findif/vibinfo
→
qcdb.vib

findif/findif
→
driver.driver_findif

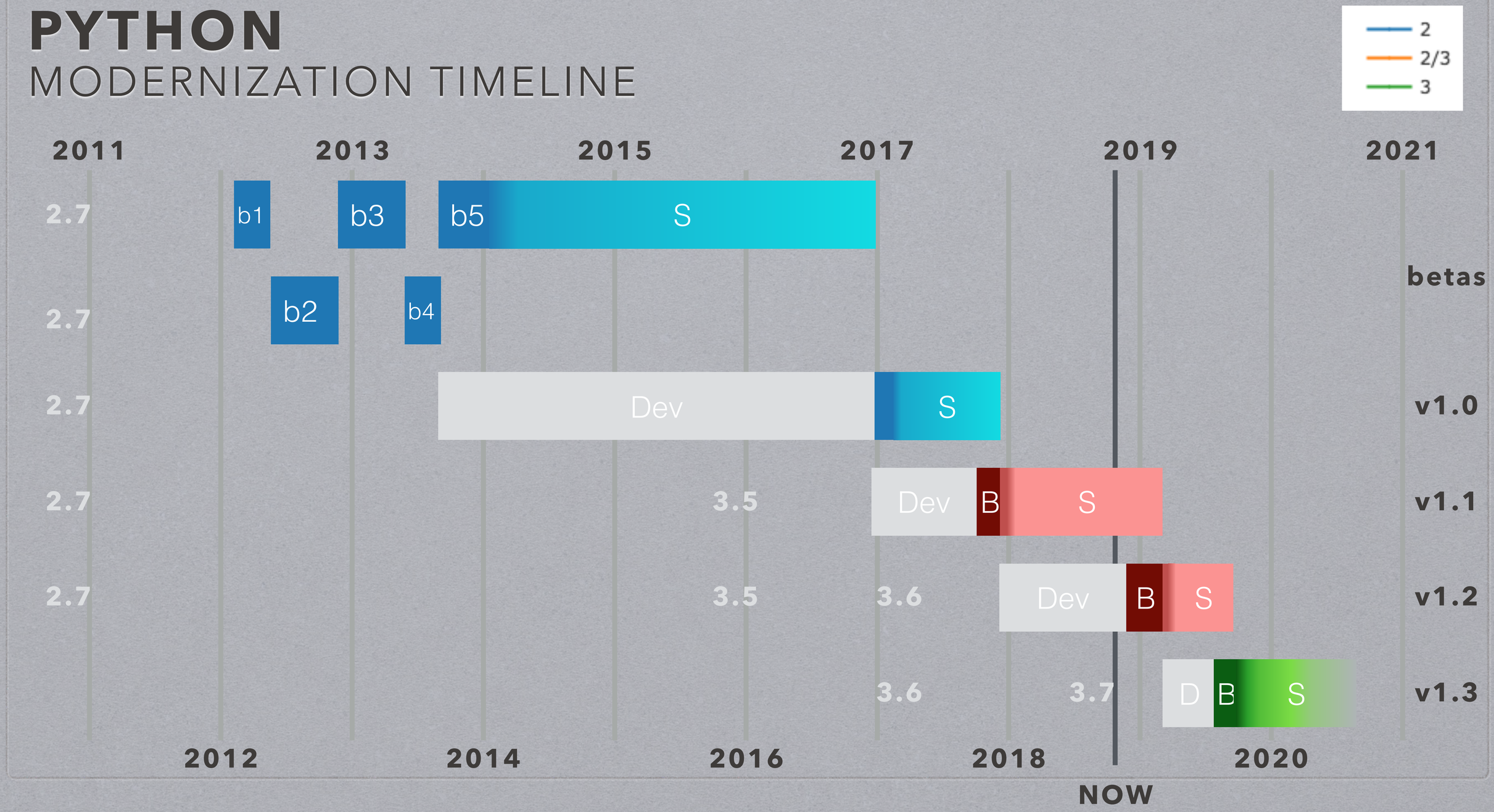
VERSIONS DEVS CARE ABOUT

TOOLS & CONDA
PSI NOW CONFORMIST, EXCEPT FOR INTEL COMPILERS

- **CONDA-BUILD 3** c. 2018
  - internal sysroot so old GLIBC linked automatically
  - Psi4 completely compatible with defaults channel. Including mkl_rt linking matching defaults numpy.
  - Psi4 channel hosts all non-defaults dependencies so `-c psi4` always sufficient
  - Far better version templating allowing projects to suggest compatible version constraints for downstream
  - New set of conda recipes in GH:psi4/psi4meta/conda-recipes
- **COMPILERS**
  - **LINUX** GCC 7.3 for C, C++ (~C++17), Fortran. GCC 5.4 & 8.2 (prelim) also available.
  - Psi uses ICPC 2018.3 (C++14) atop GCC 7.3 for multiarch optimization.
  - **MAC** Clang 4.0.1 for C, C++; GCC Fortran.
  - Psi would use ICPC but it (or mac-psinet) is too slow. And licensing issues.
  - **WIN** MSVC for C, C++; IFORT (not distributed) for Fortran.
- **OPENMP**
  - **LINUX** GCC+gomp supports v4.5. ICPC+GCC+iomp5 supports v5.0.
  - **MAC** Clang+omp supports v3.1. Clang+iomp5 supports v3.1. ICPC+Clang+iomp5 supports v5.0?
  - **WIN** MSVC supports v2.0.


Development
Beta / RC
Maintenance
Security
PYTHON PROJECT
security fixes
bugfixes, security fixes
feature fixes, stabilization, bugfixes, security fixes
new features, bugfixes, security fixes
2019
2021
how a responsible upstream dep behaves
multiple safe versions available so downstream don't hit version freeze resolving their deps
Dev
Dev
B
Python 3.7
Secur
Dev
B
Python 3.6
Security
Dev
B
Python 3.5
Security
Dev
B
Python 3.4
Security
Dev
B
Python 3.3
Security
B
Python 2.7
Security
2012
2014
2016
2018
2020
NOW

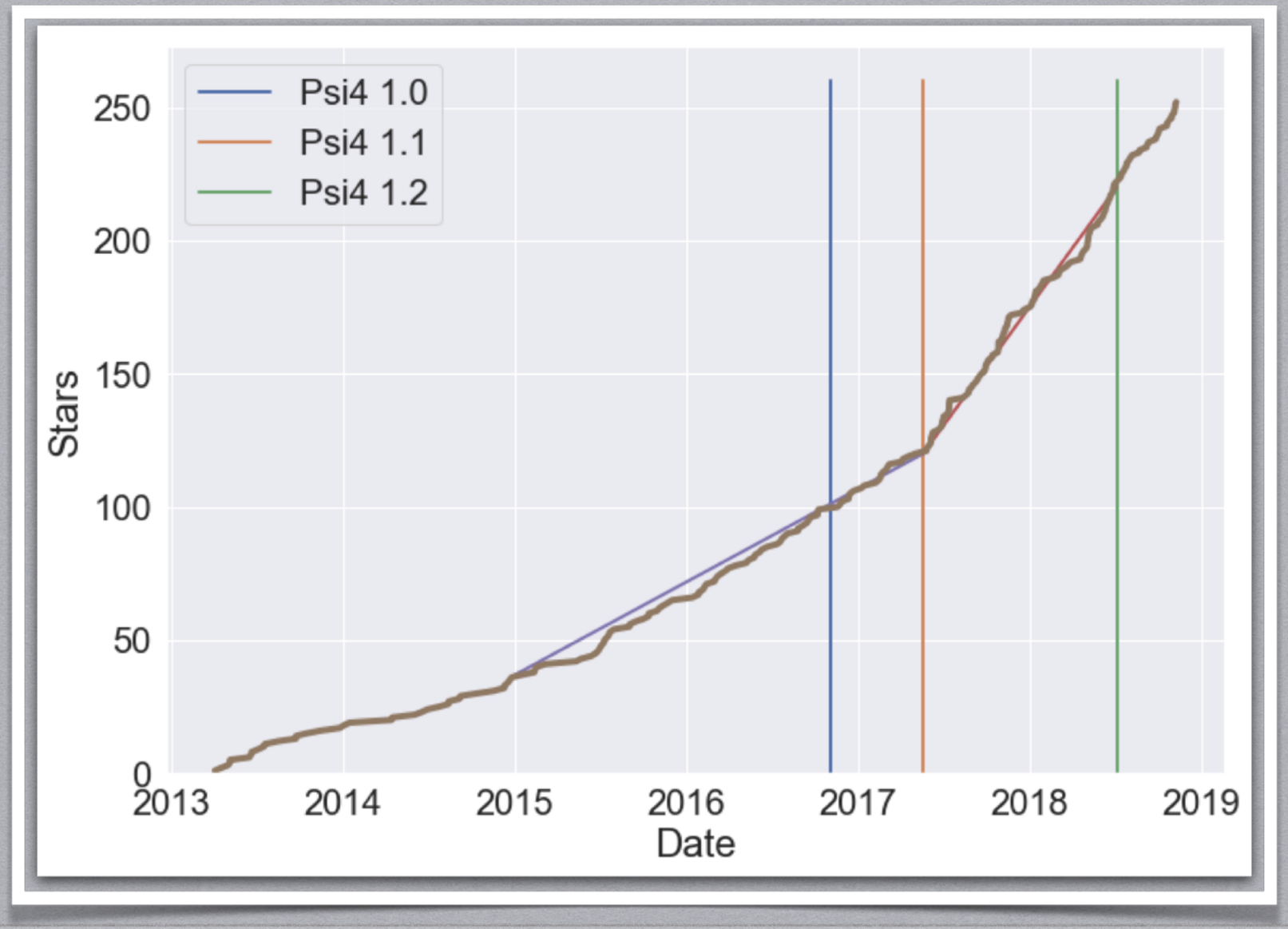
PYTHON
MODERNIZATION TIMELINE



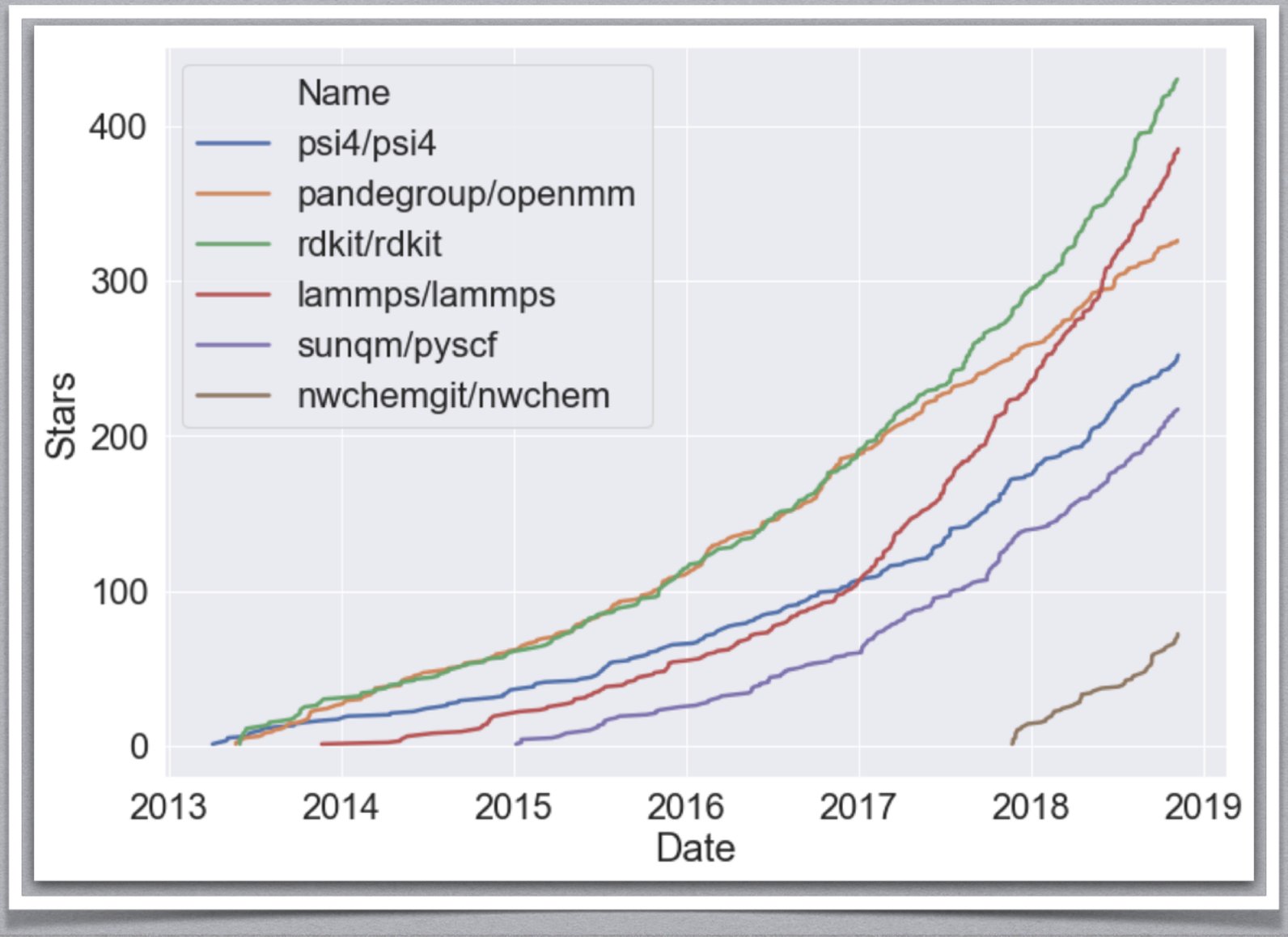
RELEASE SCHEDULE
FASTER PACE

- v1.1 + ~800 & Nov 2018 when decided to “do minimal polishing” and mint v1.2
- v1.2 turned into v1.1 +1555 released July 2018
- partially this was my fault to renovate recipes for conda-build 3
- pyscf had 8 releases this year
- many people want to use “stable releases”
- releases would be less painful with fewer last-minute features if more frequent
- suggest twice a year
- that means Feature Freeze as soon as all tests pass for py-optking and Windows; before end of 2018.

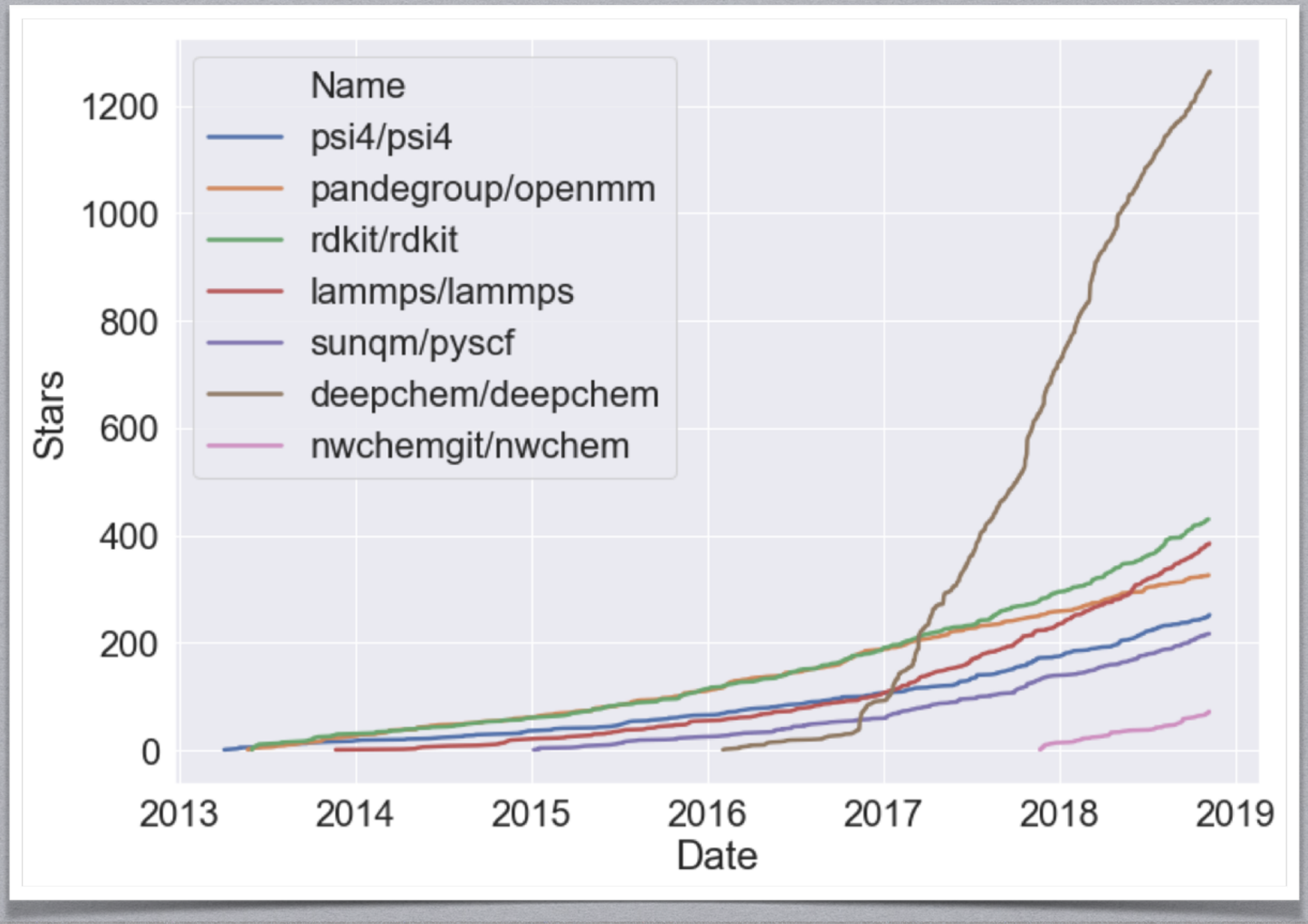
PSI4 STARS


Psi4 1.0
Psi4 1.1
Psi4 1.2
Stars
0
50
100
150
200
250
2013
2014
2015
2016
2017
2018
2019
Date

PSI4 STARS


Name
psi4/psi4
pandegroup/openmm
rdkit/rdkit
lammps/lammps
sunqm/pyscf
nwchemgit/nwchem
Stars
400
300
200
100
0
2013
2014
2015
2016
2017
2018
2019
Date

PSI4 STARS

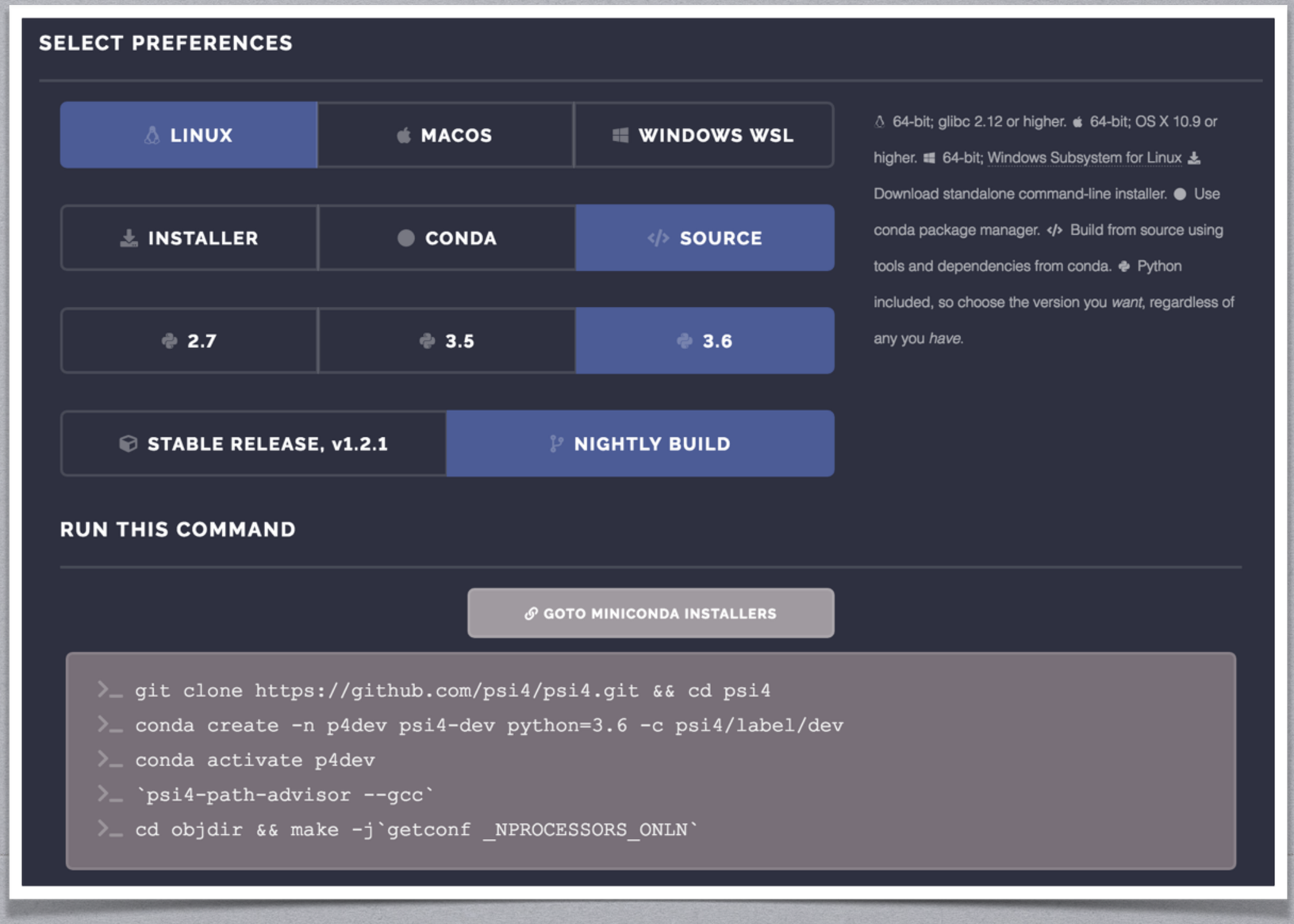

Name
psi4/psi4
pandegroup/openmm
rdkit/rdkit
lammps/lammps
sunqm/pyscf
deepchem/deepchem
nwchemgit/nwchem
Stars
0
200
400
600
800
1000
1200
2013
2014
2015
2016
2017
2018
2019
Date

ACQUIRING & BUILDING PSI

DON'T BUILD LIBINT YOURSELF

PSICODE.ORG/DOWNLOADS

- To build Psi4, you can either learn ordinary CMake commands, use the options in main CMakeLists.txt, and rebuild dependencies for every installation. –OR–
- Use the psi4-dev conda package that (1) brings prebuilt dependencies and (2) brings pre-generated `cmake` commands.
- Either equally valid; latter a great deal easier.

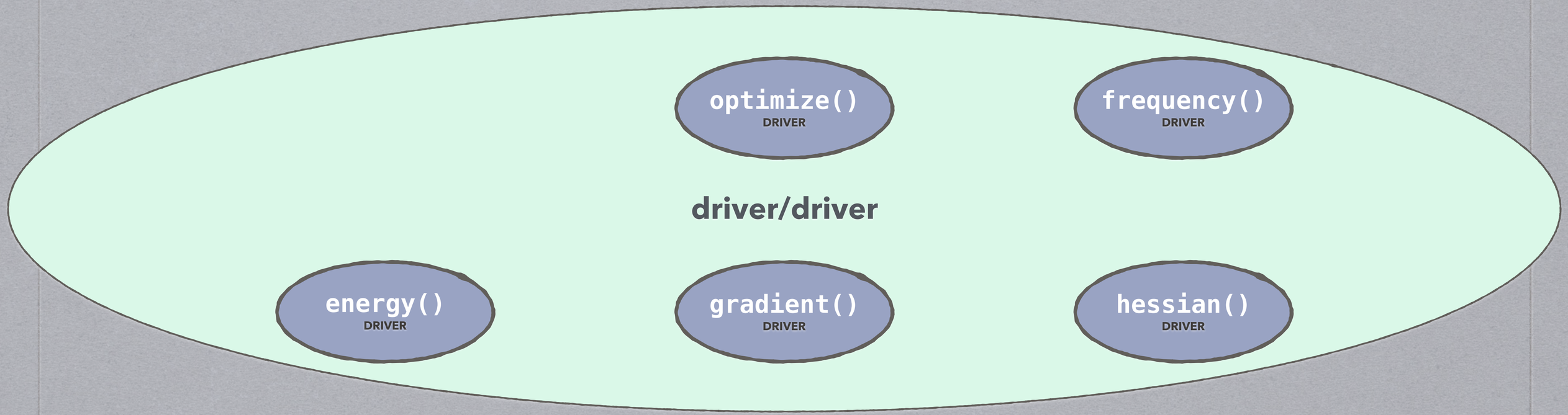


ROADMAP
HOW DO OTHERS FIND PLANS

- For everything, usually a core-dev or two has plans
- Sometimes, all of core-dev knows the plan
- Sometimes try to query full panel at monthly conference calls. Most often, panel just nods or looks uneasy. This is fine for approval but no discussion of best path.
- People outside conf call have no hope of knowing future plans.
- How can we improve communication?

USER-FACING DRIVER FUNCTIONS

MAGIC




```
psi4.energy('mp2')
```

```
psi4.energy('mp2/cc-pvtz')
```

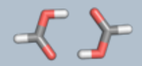
```
psi4.energy('mp2/cc-pv[dt]z')
```

```
psi4.gradient('mp2/cc-pvtz', bsse_type='cp')
```

```
psi4.optimize('mp2/cc-pv[dt]', bsse_type='uncp', dertype=0)
```

RECURSIVE DRIVER

```
def energy(name, molecule, opts):
```

'mp2' 

name is callable function?

no

BSSE in **opts**?

no

compound/composite **name**?

no

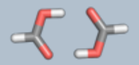
set convergence criteria

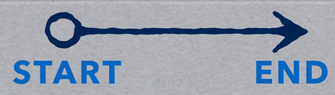
✓

```
wfn = procedures['energy'][name](molecule)
```

```
      HF TOTAL ENERGY / cc-pVTZ -377.701126
MP2 CORRELATION ENERGY / cc-pVTZ   -1.352510
     MP2 TOTAL ENERGY / cc-pVTZ -379.053636
```

```
return wfn['energy']
```

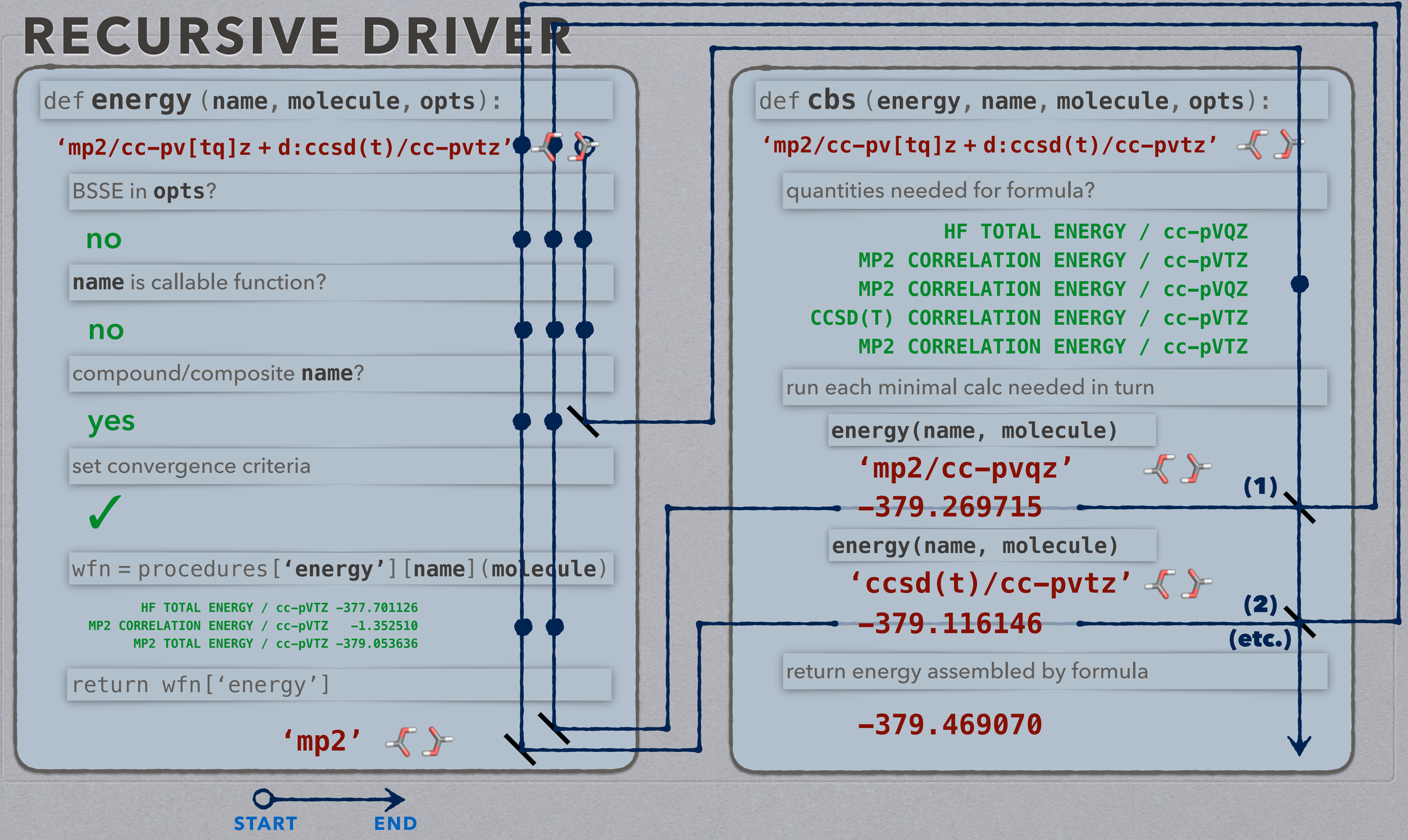
-379.053636 **'mp2'** 



- nothing psi-specific to these questions (except default convergence criteria)
- answering "yes" redirects to special function to handle complication and eventually return.
- localize work for distributed computing
- far easier for users to remember few fns.

⟵ **DO WORK HERE**

RECURSIVE DRIVER



RECURSIVE DRIVER ALL POSSIBLE ROUTES

```
def energy(name, molecule, opts):
def gradient(name, molecule, opts):
def hessian(name, molecule, opts):
```

BSSE in **opts**?

name is callable function?

compound/composite **name**?

if analytic Hessian avail./req.?

elif analytic gradient avail./req.?

else analytic energy req.?

```
return procedures[<func>][name](molecule)
```

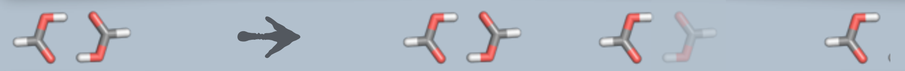
end or →

START → END

```
def optimize(name, molecule, opts):
    for cycle in range(opt_maxiter):
        procedures['gradient'][name](molecule)
```

```
def nbody(func, name, molecule):
```

separate into molecule & subsystem basis calls. method & func const.



run each in turn

```
func(name, molecule)
```

```
def user_def_func(func, name, molecule):
```

separate into pieces required for energy (etc.) formula. mol & func const.

'mp3.5/cc-pvtz' → MP4 TOTAL ENERGY/cc-pVTZ

run each in turn

```
func(name, molecule)
```

```
def cbs(func, name, molecule):
```

separate into method & basis calls. mol & func const.

'mp2/cc-pv[tq]z' → MP2 TOTAL ENERGY/cc-pVTZ
MP2 TOTAL ENERGY/cc-pVQZ

run each in turn

```
func(name, molecule)
```

QCDB, QCA ECOSYSTEMS