PyDaylight

Rapid Software Development with the Daylight Toolkit

Python

- Very High-Level Programming Language
- Modules, Functions, Classes
- Automatic Garbage Collection
- Exceptions, Iterators, Unicode, ...
- “Batteries Included”
- Good Integration with C libraries

dalke scientific
More science. Less time.

Daylight Toolkit

- C/Fortran Libraries
- “oopish”
- No language support for objects, iterators, garbage collection, or exceptions

dalke scientific
More science. Less time.

Bindings to C Libraries

- Python, Perl, Tcl, Ruby, ... can integrate with C libraries (DayPerl, DayTcl)
- SWIG automates the interface generation (DaySWIG)
- But it still acts like the C library

PyDaylight

- Makes the Daylight libraries "pythonic"
  - convert oopish functions into classes
  - sequences and streams into Python lists or iterators
  - check function returns and raise exceptions if there was an error
  - use Python to automate dt_dealloc calls

Simple Example

```
from daylight import Smiles

mol = Smiles.smilin("c1(CO)cc(C)nn1c2cc(OC)nc(C)n2")
print len(mol.atoms), len(mol.bonds), len(mol.cycles)
print mol.atoms[0].symbol, mol.atoms[0].aromatic
print mol.cansmiles()
```

```
17 18 2
C 1
COc1cc(C)nn1c2cc(OC)nc(C)n2
```

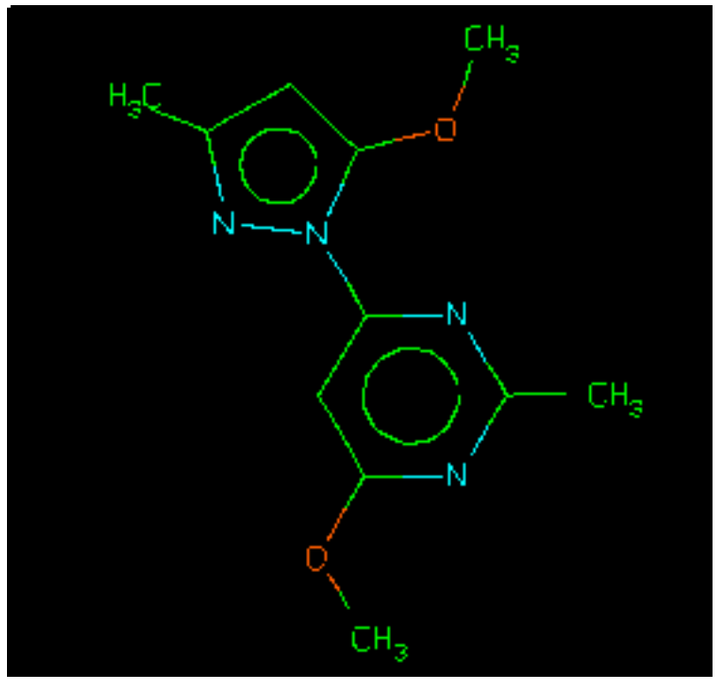
dalke scientific
More science. Less time.

Smarts matching

```
from daylight import Smiles, Smarts
mol = Smiles.smilin("c1(CO)cc(C)nn1c2cc(OC)nc(C)n2")

pat = Smarts.compile("[#6](*)(*)*")
for match in pat.umatch(mol):
    for atom in match.atoms:
        print atom.symbol,
    print
```

```
C N C C
C C C N
C N N C
C C O N
C N C N
```



dalke scientific
More science. Less time.

Reactions

```
>>> from daylight import Smiles, Smirks
>>> mol = Smiles.smilin("CC(=O)Cl.NCCC")
>>> trans = Smirks.compile("[C:1](=[O:2])Cl.[H][N:4][C:5]>>[C:1](=[O:2])[N:4][C:5] ")
>>> reactions = trans.transform(mol)
>>> reactions
[Reaction(826), Reaction(752)]
>>> for reaction in reactions:
...     print reaction.cansmiles()
...
CCCN.CC(=O)Cl>>CCCNC(=O)C
CCCN.CC(=O)Cl>>CCCNC(=O)C
>>> reactions[0].reactant.cansmiles()
'CCCN.CC(=O)Cl'
>>> reactions[0].product.cansmiles()
'CCCNC(=O)C'
>>>
```

dalke scientific
More science. Less time.

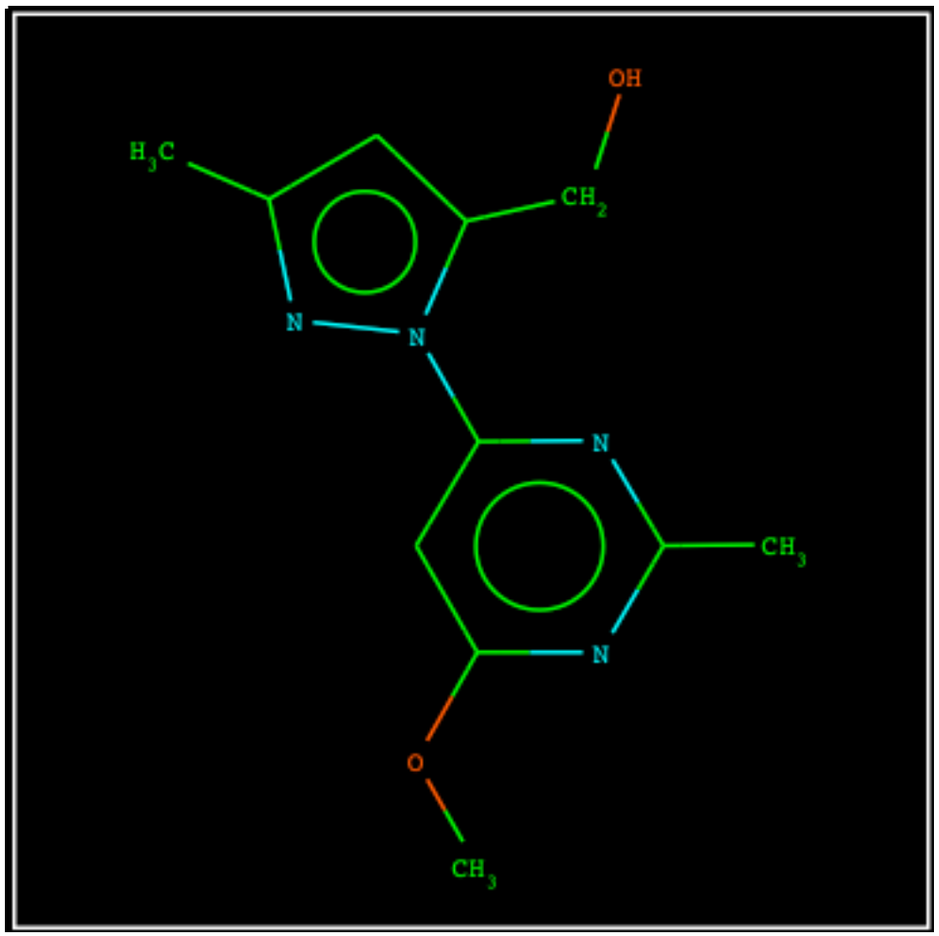
Depictions (PIL, PDF, Qt, Tk)

```
from daylight import Smiles, Depict
from daylight.Depict import PDFCallback, Colors
from reportlab.pdfgen import canvas

mol = Smiles.smilin("c1(CO)cc(C)nn1c2cc(OC)nc(C)n2")
dep = Depict.Depiction(mol)
dep.calcxy()
dep.color_atoms()

canv = canvas.Canvas("depict.pdf")
for k, v in PDFCallback.linewidths.items():
    PDFCallback.linewidths[k] = v * 3
cb = PDFCallback.PDFCallback(canv,
          colors = Colors.screen_colors, size=(500, 500))
dep.depict(cb)

canv.showPage()
canv.save()
```



dalke scientific
More science. Less time.

Calling Program Objects

```
>>> import os
>>> from daylight import Program
>>> clogptalk_path = os.path.expandvars(
...                         "${DY_ROOT}/bin/clogptalk")
>>> prog = Program.Program(clogptalk_path)
>>> print prog("c1ccccc1")
['c1ccccc1 2.142 0 LogPstar: 2.13']
>>>
```

Includes RestartableProgram and MsgList

Writing Program Objects

```
from daylight import Pipetalk

class Length(Pipetalk.SimpleServer):
    def do(self, s, *args):
        return [str(len(s))]
    def do_Say_Program(self, message_type, *args):
        return ["String length server"]
    def do_Say_VERSION(self, message_type, *args):
        return ["1.0"]
    def do_Say_NOTICE(self, message_type, *args):
        return ["Placed in the public domain."]
    def do_Say_HELP(self, message_type, *args):
        return ["Returns the length of the string passed in to it."]

if __name__ == "__main__":
    server = LengthServer()
    server.serve_forever()
```

Thor and Merlin

- Full support for Thor and Merlin
- MCL to PyDaylight converter

http://daylight.daylight.com/meetings/mug00/Dalke/overview/

- Thor ping and ls
- Listing aliases
- Printing TDTs
- Searching Merlin

dalke scientific
More science. Less time.

New since MUG 2001

dalke scientific
More science. Less time.

Fingerprints

```
>>> from daylight import Smiles, Fingerprint
>>> mol1 = Smiles.smilin("c1ccccc1O")
>>> mol2 = Smiles.smilin("c1ccccc1")
>>> fp1 = Fingerprint.generatefp(mol1)
>>> fp2 = Fingerprint.generatefp(mol2)
>>> Fingerprint.fingertest(fp1, fp2)
0
>>> Fingerprint.fingertest(fp2, fp1)
1
>>> Fingerprint.euclid(fp2, fp1)
0.0068359375
>>> len(fp1), len(fp2)
(2048, 2048)
>>> fp1.bitcount, fp2.bitcount
(33, 19)
```

“daylight-fp” encoding

```
>>> "Hello MUG!".encode("base64")
'SGVsbG8gTVVHIQ==\n'
>>>
>>> import daylight
>>> from daylight import Smiles, Fingerprint
>>> mol = Smiles.smilin("C#N")
>>> fp = Fingerprint.generatefp(mol, size=64)
>>> s = fp.stringvalue
>>> s
'\x01\x81x\x18d \x02\x88'
>>> s.encode("daylight-fp")
'.M3s44EU.cU.2'
>>> s.encode("daylight-fp").decode("daylight-fp")
'\x01\x81x\x18d \x02\x88'
>>>
>>> fp[0], fp[1], fp[2]
(1, 0, 0)
>>> "".join([str(bit) for bit in fp])
'1000000010000001000111100001100000100110000001000100000000010001'
```

HTTP toolkit

```
>>> from daylight import HTTP
>>> server = HTTP.HTTPServer(8080)
>>> req = server.get_request()
>>> for k, v in req.prop.items():
...     print k, "=", v
...
htt_recv_Host = localhost
        ....
htt_mime_type = text/html
htt_code = 404
        ....
htt_path = /Mug2003/example
htt_method = GET
>>> req.prop["htt_code"] = 200
>>> req.prop["htt_mime_type"] = "text/plain"
>>> req.wfile.write("This was served via ")
>>> req.wfile.write("a Daylight web server.\n")
>>> req.close()
```

```
% telnet localhost 8080
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
GET /Mug2003/example HTTP/1.1
Host: localhost

HTTP/1.1 200 OK
Date: Thu, 27 Feb 2003 09:37:35 GMT
Server: Daylight/4.82
Content-Length: 43
Content-Type: text/plain

This was served via a Daylight web server.
Connection closed by foreign host.
%
```

dalke scientific
More science. Less time.

Qt Depictions

- Qt is a cross-platform C++ graphics library
- PyQt provides Python bindings
  - except not yet for Mac OS X
- Easier to write GUI apps using Daylight

dalke scientific
More science. Less time.

Updated to 4.81

- New functions (common bit count, xsmiles, appendstring, appendstringvalue)
- NOT compatible with earlier toolkits

dalke scientific
More science. Less time.

Resources

- http://www.dalkescientific.com/PyDaylight/
- Public CVS on Sourceforge
- pydaylight-general@lists.sourceforge.net
- Dr. Dobb's article, previous MUG talks
- Dalke Scientific Software, LLC

dalke scientific
More science. Less time.