Scaffold analysis in Python with RDKit and pandas

Pandas (http://pandas.pydata.org/) is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.
RDKit (http://www.rdkit.org/) is an open source chemistry toolkit.

In [20]:
```
import pandas as pd
import rdkit.Chem as Chem
from rdkit.Chem import PandasTools
from rdkit.Chem import Draw
from rdkit.Chem import Descriptors
from rdkit.Chem.Draw import IPythonConsole # Enables RDKit IPython integration
```

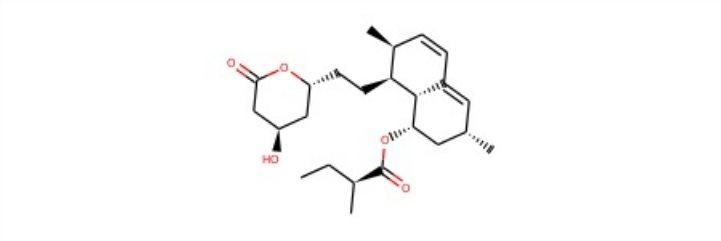
IPython and RDKit

RDKit provides IPython integration

In [21]:
```
mol = Chem.MolFromSmiles('O=C(O[C@@H]1[C@H]3C(=C/[C@H](C)C1)\C=C/[C@@H]([C@@H]3CC[C@H]2OC(=O)C[C@H
](O)C2)C)[C@@H](C)CC')
```

In [22]:
```
mol
```

Out[22]:



In [23]:
```
Descriptors.NumHDonors(mol)
```

Out[23]: 1

In [24]:
```
Descriptors.MolLogP(mol)
```

Out[24]: 4.195500000000004

RDKit and pandas

Load 'approved drugs' downloaded from www.drugbank.ca:
`% time` is a ipython magic function that tells you how much time did certain operation take to finish. It will be used to give you a feeling about speed of certain functions

In [25]:
```
% time cpds = PandasTools.LoadSDF('approved.sdf', includeFingerprints=False)
```

```
CPU times: user 6.5 s, sys: 23.3 ms, total: 6.53 s
Wall time: 6.54 s
```

In [26]:
```
cpds.columns
```

Out[26]:
```
Index([u'ALOGPS_LOGP', u'ALOGPS_LOGS', u'ALOGPS_SOLUBILITY', u'BRANDS', u'CHEMICAL_FORMULA', u'DRU
GBANK_ID', u'DRUG_GROUPS', u'EXACT_MASS', u'GENERIC_NAME', u'ID', u'INCHI_IDENTIFIER', u'INCHI_KEY
', u'IUPAC_NAME', u'JCHEM_ACCEPTOR_COUNT', u'JCHEM_ACIDIC_PKA', u'JCHEM_BASIC_PKA', u'JCHEM_DONOR_
COUNT', u'JCHEM_LOGP', u'JCHEM_PHYSIOLOGICAL_CHARGE', u'JCHEM_POLARIZABILITY', u'JCHEM_POLAR_SURFA
CE_AREA', u'JCHEM_REFRACTIVITY', u'JCHEM_ROTATABLE_BOND_COUNT', u'MOLECULAR_WEIGHT', u'SALTS', u'S
MILES', u'SYNONYMS', u'ROMol'], dtype=object)
```

In [27]:
```
len(cpds)
```

Out[27]: 1485

Assign the values of molnames and smiles (makes it easier to use this notebook on other sets with different col names)

In [28]:
```
molnames = 'DRUGBANK_ID'
smiles = 'SMILES'
```

Keep only columns 'DRUGBANK_ID', 'SMILES' and 'ROMol'

In [29]:
```
cpds = cpds[[molnames, smiles, 'ROMol']]
```

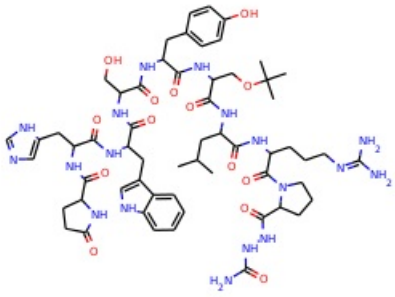
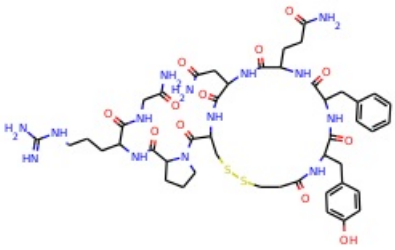
In [30]:
```
cpds.columns
```

Out[30]: Index([u'DRUGBANK_ID', u'SMILES', u'ROMol'], dtype=object)

Look at two columns and only first 2 lines:

In [31]:
```
cpds[[molnames, 'ROMol']].head(2)
```

Out[31]:

	DRUGBANK_ID	ROMol
0	DB00014	
1	DB00035	

Remove lines with NaN (empty) values and duplicates

In [32]:
```
cpds = cpds.dropna()
cpds = cpds.drop_duplicates(molnames)
cpds = cpds.drop_duplicates(smiles)
len(cpds)
```

Out[32]: 1462

Descriptors

Add some descriptors

In [33]:
```
from rdkit.Chem import Descriptors
cpds['logp'] = cpds['ROMol'].map(Descriptors.MolLogP)
cpds['mw'] = cpds['ROMol'].map(Descriptors.MolWt)
```

Remove compounds with logp >= 5 and MW >= 500

In [34]:
```
cpds = cpds[cpds['logp'] <= 5]
cpds = cpds[cpds['mw'] <= 500]
len(cpds)
```

Out[34]: 1143

In [35]:
```
cpds[[molnames, 'logp', 'mw', smiles]].head()
```

Out[35]:

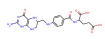
	DRUGBANK_ID	logp	mw	SMILES
6	DB00116	-0.2820	445.436	Nc1nc(=O)c2c([nH]1)NCC(CNc1ccc(C(=O)NC(CCC(=O)O)C(=O)O)cc1)N2
7	DB00117	-0.6359	155.157	NC(Cc1cnc[nH]1)C(=O)O
8	DB00118	-1.9222	399.453	C[S+](CCC(N)C(=O)O)CC1OC(n2cnc3c2ncnc3N)C(O)C1O
9	DB00119	-0.3400	88.062	CC(=O)C(=O)O
10	DB00120	0.6410	165.192	NC(Cc1ccccc1)C(=O)O

Alternative visualisation of a table

Default takes a lot of space

In [36]:
```
cpds.head(1)
```

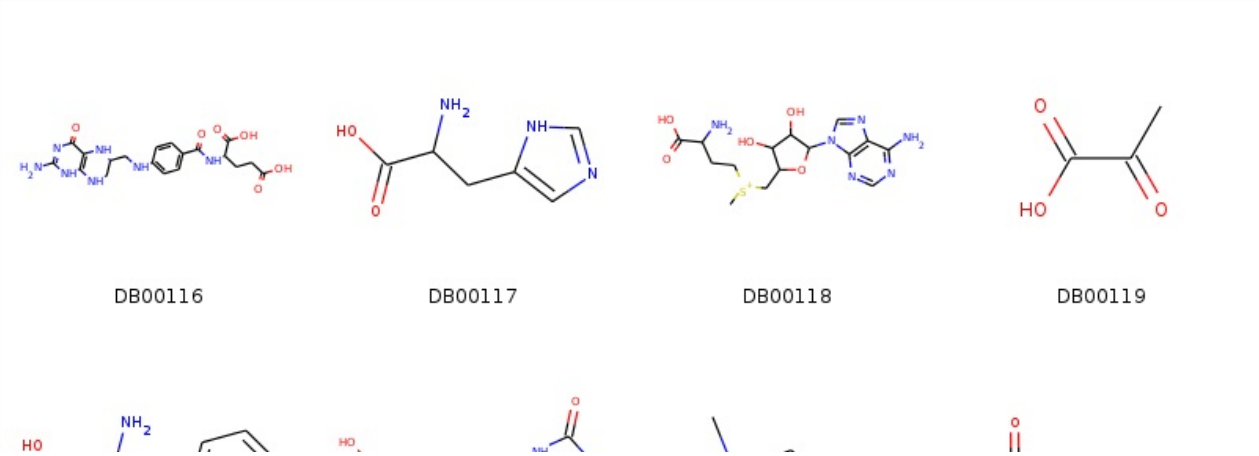
Out[36]:

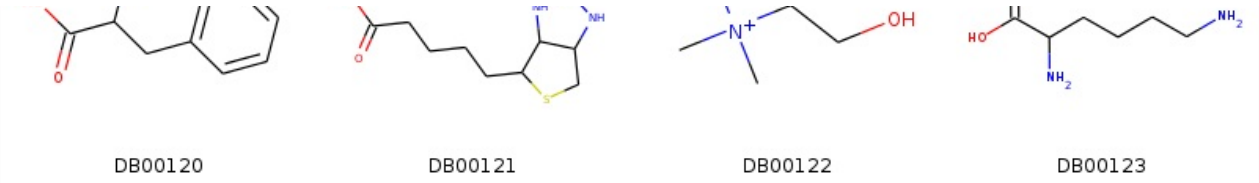
	DRUGBANK_ID	SMILES	ROMol	logp	mw
6	DB00116	Nc1nc(=O)c2c([nH]1)NCC(CNc1ccc(C(=O)NC(CCC(=O)O)C(=O)O)cc1)N2		-0.282	445.436

FrameToGridImage(pandasFrame, legendsCol=, molsPerRow=)

In [38]:
```
PandasTools.FrameToGridImage(cpds.head(8), legendsCol=molnames, molsPerRow=4)
```

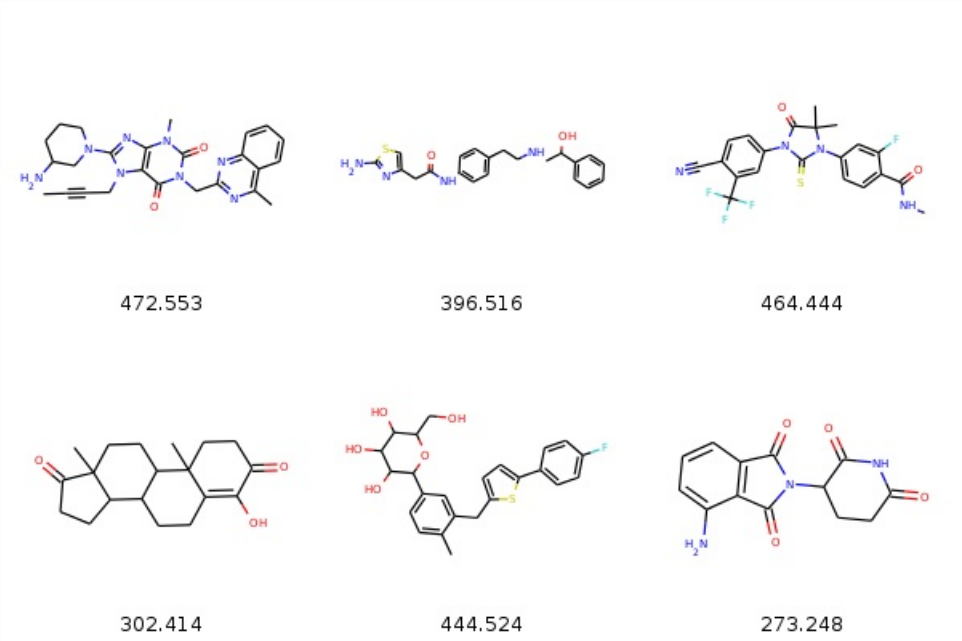
Out[38]:





In [39]:
```
PandasTools.FrameToGridImage(cpds.tail(6), legendsCol='mw', molsPerRow=3)
```

Out[39]:



Murcko scaffold decomposition

Removes side chain atoms

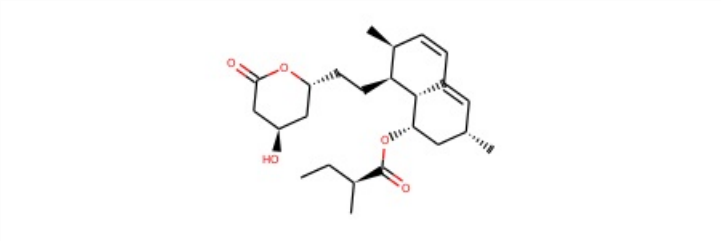
In [40]:
```
from rdkit.Chem.Scaffolds import MurckoScaffold
```

How it works with RDKit:

In [41]:
```
scaffold = MurckoScaffold.GetScaffoldForMol(mol)
generic = MurckoScaffold.MakeScaffoldGeneric(MurckoScaffold.GetScaffoldForMol(mol))
```

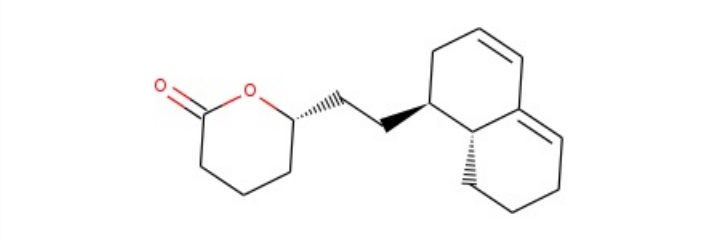
In [42]:
```
mol
```

Out[42]:



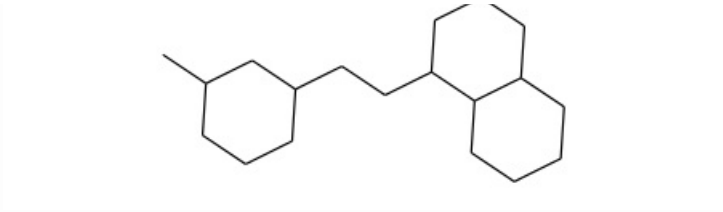
In [43]:
```
scaffold
```

Out[43]:



In [44]:
```
generic
```

Out[44]:



AddMurckoToFrame(pandasFrame, MurcoCol=, Generic=False)

Returns SMILES of scaffolds

In [45]:
```
% time PandasTools.AddMurckoToFrame(cpds)
```

```
CPU times: user 403 ms, sys: 6.67 ms, total: 410 ms
Wall time: 408 ms
```

In [46]:
```
% time PandasTools.AddMurckoToFrame(cpds, MurckoCol='Murcko_GENERIC', Generic=True)
```

```
CPU times: user 967 ms, sys: 10 ms, total: 977 ms
Wall time: 961 ms
```

In [47]:
```
cpds.head(1)
```

Out[47]:

	DRUGBANK_ID	SMILES	ROMol	logp	mw	Murcko_
6	DB00116	Nc1nc(=O)c2c([nH]1)NCC(CNc1ccc(C(=O)NC(CCC(=O)O)C(=O)O)cc1)N2		-0.282	445.436	O=c1nc[

Now we can use pandas groupby() functionality and group by scaffolds and create a new frame with scaffolds sorted by number of members

In [48]:
```
sortedScaffolds = cpds.groupby(['Murcko_SMILES']).count().sort(smiles, ascending=False)
```

In [49]:
```
sortedScaffolds = sortedScaffolds[[smiles]] # Keep only smiles column
sortedScaffolds = sortedScaffolds.rename(columns={smiles:'count'}) # rename smiles column to count
sortedScaffolds['Murcko_SMILES'] = sortedScaffolds.index # actual SMILES are only in index column,
 move it
sortedScaffolds.head()
```

Out[49]:

	count	Murcko_SMILES
Murcko_SMILES		
c1ccccc1	112	c1ccccc1
	111	
O=C1C=CC2C(=C1)CCC1C3CCCC3CCC21	17	O=C1C=CC2C(=C1)CCC1C3CCCC3CCC21
O=C1C=C2CCC3C4CCCC4CCC3C2CC1	12	O=C1C=C2CCC3C4CCCC4CCC3C2CC1
O=C1CN=C(c2ccccc2)c2ccccc2N1	12	O=C1CN=C(c2ccccc2)c2ccccc2N1

In [50]:
```
PandasTools.AddMoleculeColumnToFrame(sortedScaffolds, smilesCol='Murcko_SMILES')
```

In [51]:
```
sortedScaffolds.head(1)
```

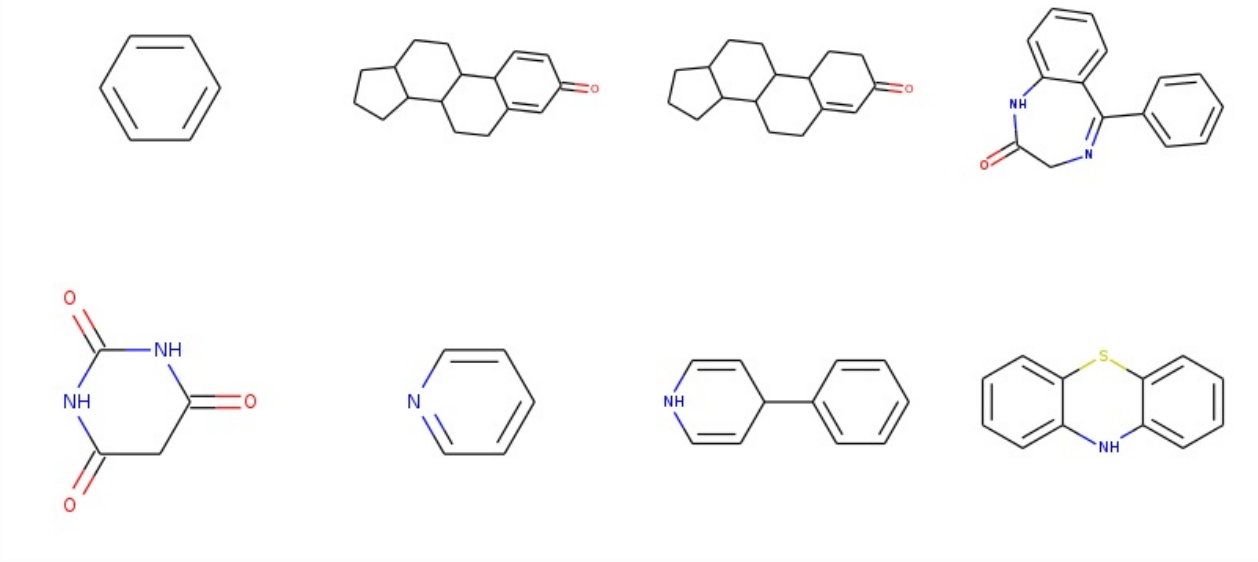
Out[51]:

	count	Murcko_SMILES	ROMol
Murcko_SMILES			

c1ccccc1	112	c1ccccc1	

In [53]:
```
PandasTools.FrameToGridImage(sortedScaffolds.dropna().head(8), molsPerRow=4) #dropna drops compoun
ds without scaffold
```

Out[53]:



We can also retrieve all compounds with certain scaffold from original table
Benzodiazepine scaffold #4 O=C1CN=C(c2ccccc2)c2ccccc2N1

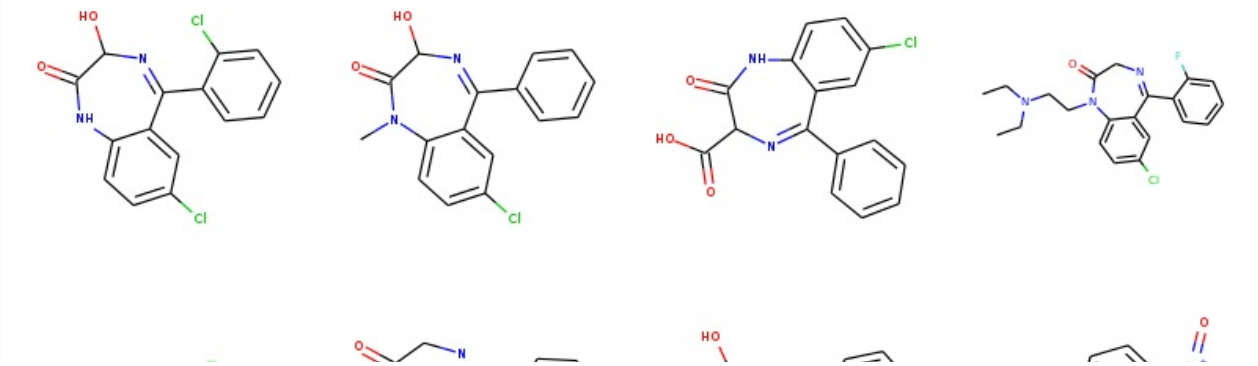
In [55]:
```
cpds[cpds['Murcko_SMILES'] == 'O=C1CN=C(c2ccccc2)c2ccccc2N1'].head(1)
```

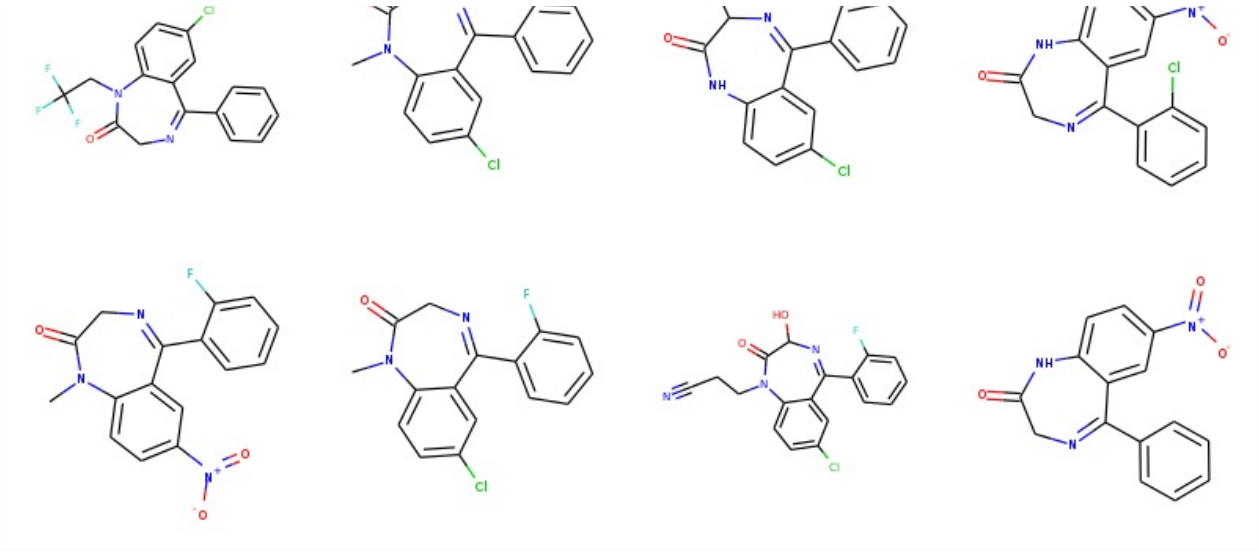
Out[55]:

	DRUGBANK_ID	SMILES	ROMol	logp	mw	Murcko_SMILES	M
74	DB00186	O=C1Nc2ccc(Cl)cc2C(c2ccccc2Cl)=NC1O		3.1013	321.163	O=C1CN=C(c2ccccc2)c2ccccc2N1	C

In [57]:
```
PandasTools.FrameToGridImage(cpds[cpds['Murcko_SMILES'] == 'O=C1CN=C(c2ccccc2)c2ccccc2N1'], molsPe
rRow=4)
```

Out[57]:





Aligning compounds to scaffolds

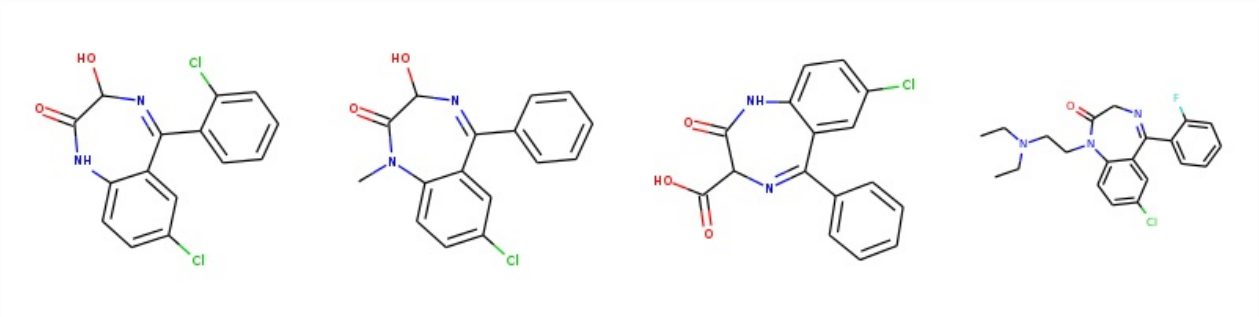
AlignToScaffold(dataframe, molCol=, scaffoldCol=)

In [58]:
```
somemols = cpds.groupby('Murcko_SMILES').get_group('O=C1CN=C(c2ccccc2)c2ccccc2N1')
```

Note how molecules are not alignied

In [59]:
```
PandasTools.FrameToGridImage(somemols.head(4), molsPerRow=4)
```

Out[59]:



Align them to scaffold

In [60]:
```
PandasTools.AlignToScaffold(somemols, molCol='ROMol', scaffoldCol='Murcko_SMILES')
```

In [61]:
```
PandasTools.FrameToGridImage(somemols.head(4), molsPerRow=4)
```

Out[61]: