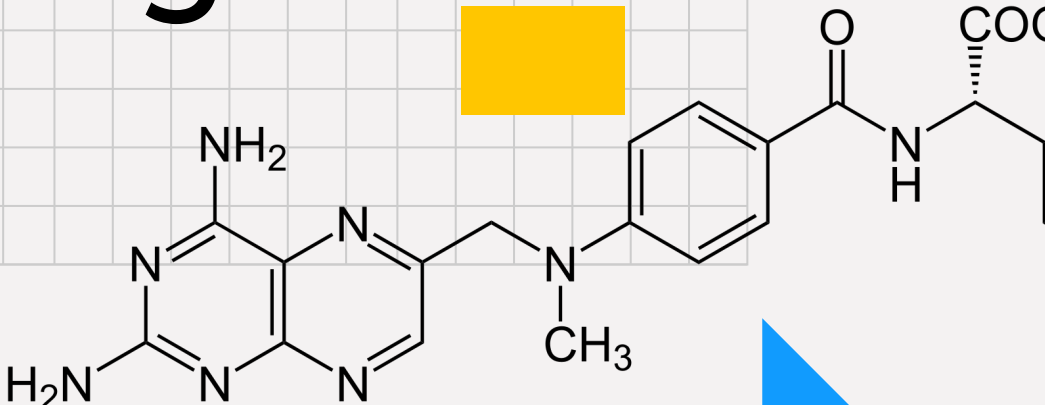
Molecular Fingerprinting

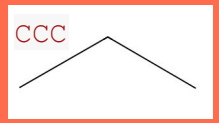
Cheminformatics assignment

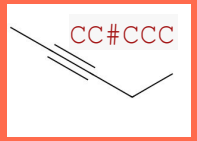


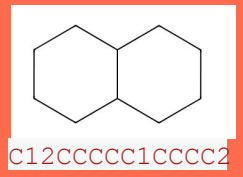
Piyush R.Maharana

SMILES

```
!pip install rdkit-pypi
```

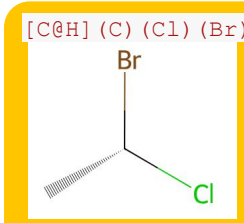


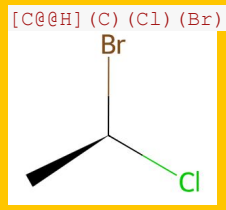


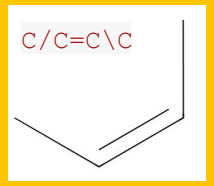


Basics

Canonical SMILES Bonds Cyclic

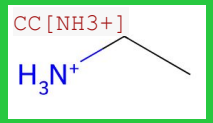


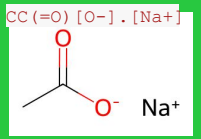


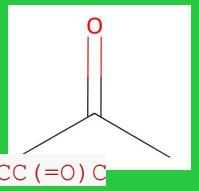


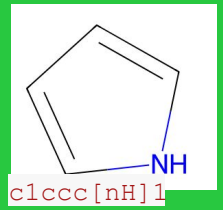
Stereochemistry

Wedge Dash E/Z Isomerisation





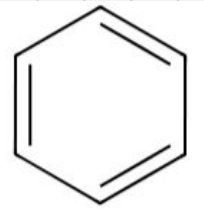




Others

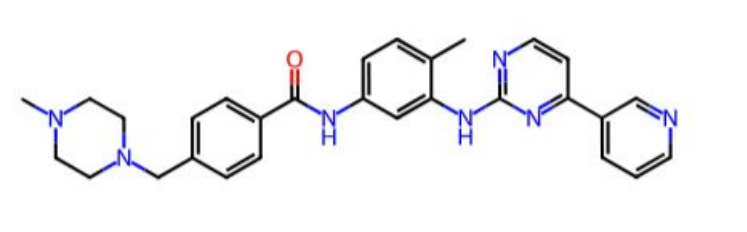
Branching Charges Disconnected Aromaticity

Displaying chemical structures in RDKit



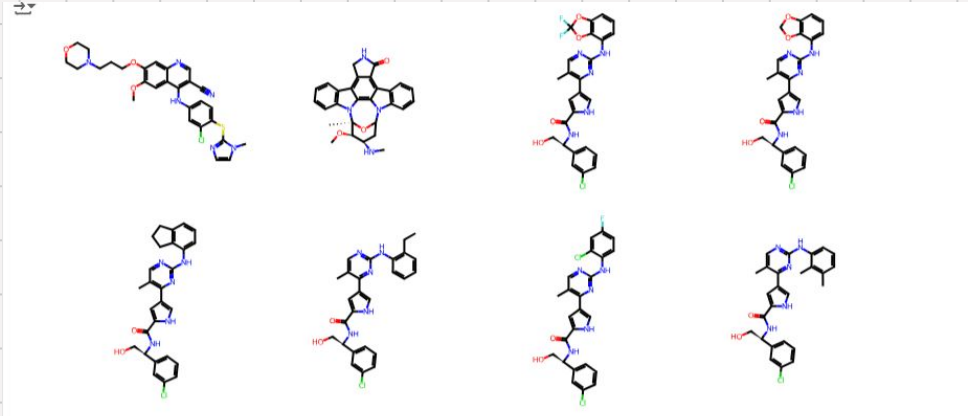
```
mol = Chem.MolFromSmiles("c1ccccc1")
```

```
from rdkit.Chem import AllChem
from rdkit.Chem import rdMolDescriptors
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw
from rdkit import DataStructs
```



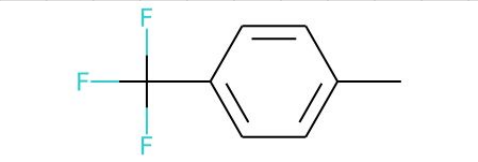
```
glvc =
Chem.MolFromSmiles("Cc1ccc(cc1Nc2nccc(n2)c3cc
cnc3)NC(=O)c4ccc(cc4)CN5CCN(CC5)C")
```

```
Draw.MolsToGridImage(mols,molsPerRow= 4,useSVG=True)
```

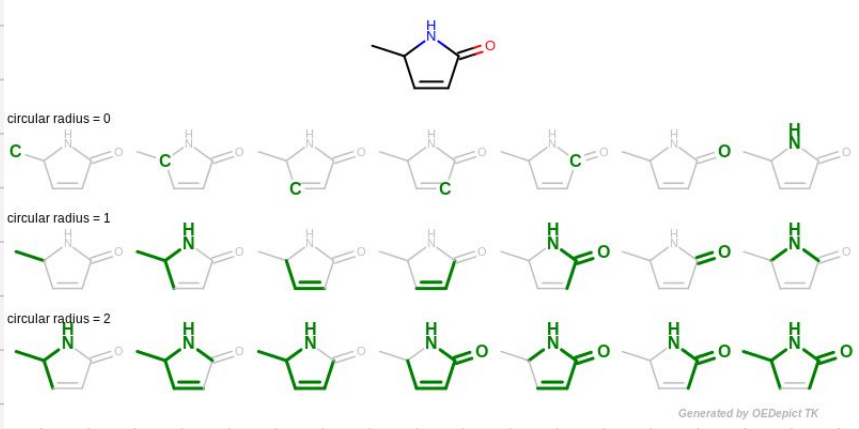


Fingerprints

Molecular fingerprints are representations of molecular structures that encode information about their chemical properties and potential biological activity.



1. Substructure Fingerprints
   - MACCS Keys
   - Extended Connectivity Fingerprints
2. Path-Based Fingerprints
   - Topological
   - Circular or Morgan
   - Atom Pair
3. Machine Learning-Based Fingerprints



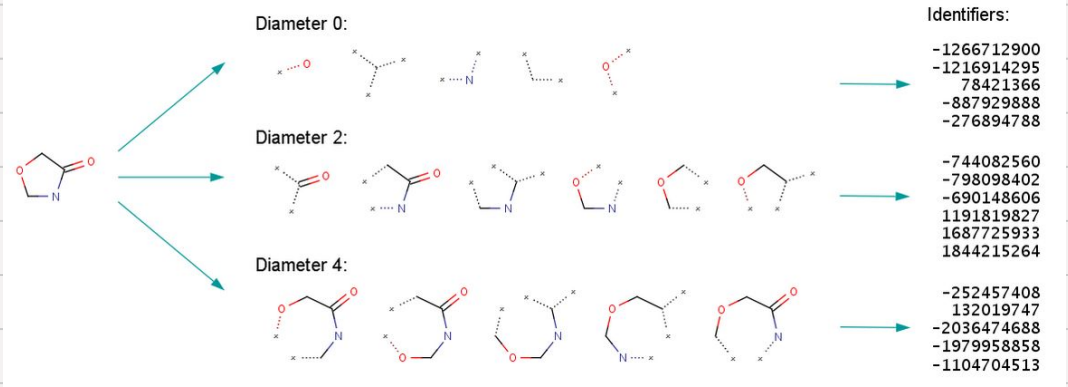
List of Available Fingerprints

Fingerprint Type	Notes	Language
RDKit	a Daylight-like fingerprint based on hashing molecular subgraphs	C++
Atom Pairs	*JCICS* **25**:64–73 (1985)	C++
Topological Torsions	*JCICS* **27**:82–5 (1987)	C++
MACCS keys	Using the 166 public keys implemented as SMARTS	C++
Morgan/Circular	Fingerprints based on the Morgan algorithm, similar to the ECFP/FCFP fingerprints *JCIM* **50**:742–54 (2010).	C++
2D Pharmacophore	Uses topological distances between pharmacophoric points.	C++
Pattern	a topological fingerprint optimized for substructure screening	C++
Extended Reduced Graphs	Derived from the ErG fingerprint published by Stiefl et al. in *JCIM* **46**:208-20 (2006). NOTE: these functions return an array of floats, not the usual fingerprint types	C++
MHFP and SECFP	Derived from the ErG fingerprint published by Probst et al. in *J Cheminformatics* **10** (2018). NOTE: these functions return different types of values	C++

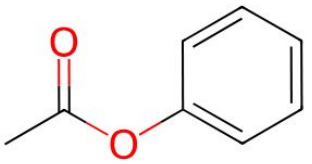
```python
from rdkit import Chem, DataStructs
from rdkit.Chem import rdMolDescriptors

# these are SMARTS patterns corresponding to the MDL MACCS keys
smartsPatts = {
  1: ('?', 0),  # ISOTOPE
  #2:('[#104,#105,#106,#107,#106,#109,#110,#111,#112]',0),  # atomic num >103 Not complete
  2: ('[#104]', 0),  # limit the above def'n since the RDKit only accepts up to #104
  3: ('[#32,#33,#34,#50,#51,#52,#82,#83,#84]', 0),  # Group IVa,Va,VIa Rows 4-6
  4: ('[Ac,Th,Pa,U,Np,Pu,Am,Cm,Bk,Cf,Es,Fm,Md,No,Lr]', 0),  # actinide
  5: ('[Sc,Ti,Y,Zr,Hf]', 0),  # Group IIIB,IVB (Sc...)
  6: ('[La,Ce,Pr,Nd,Pm,Sm,Eu,Gd,Tb,Dy,Ho,Er,Tm,Yb,Lu]', 0),  # Lanthanide
  7: ('[V,Cr,Mn,Nb,Mo,Tc,Ta,W,Re]', 0),  # Group VB,VIB,VIIB
  8: ('[!#6;!#1]1~*~*~*~1', 0),  # QAAA@1
  9: ('[Fe,Co,Ni,Ru,Rh,Pd,Os,Ir,Pt]', 0),  # Group VIII (Fe...)
  10: ('[Be,Mg,Ca,Sr,Ba,Ra]', 0),  # Group IIa (Alkaline earth)
  11: ('*1~*~*~*~1', 0),  # 4M Ring
  12: ('[Cu,Zn,Ag,Cd,Au,Hg]', 0),  # Group IB,IIB (Cu..)
  13: ('[#8]~[#7](~[#6])~[#6]', 0),  # ON(C)C
  14: ('[#16]-[#16]', 0),  # S-S
  15: ('[#8]~[#6](~[#8])~[#8]', 0),  # OC(O)O
  16: ('[!#6;!#1]1~*~*~1', 0),  # QAA@1
  17: ('[#6]#[#6]', 0),  #CTC
  18: ('[#5,#13,#31,#49,#81]', 0),  # Group IIIA (B...)
  19: ('*1~*~*~*~*~*~*~1', 0),  # 7M Ring
```

MACCS

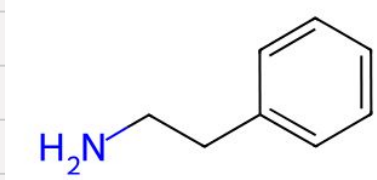


ECFP



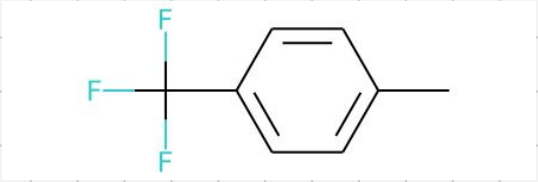
```
No of ones:  13
No of zeros:  1011

11      1
33      1
64      1
322     1
356     1
650     1
695     1
705     1
726     1
807     1
849     1
893     1
1017    1
```

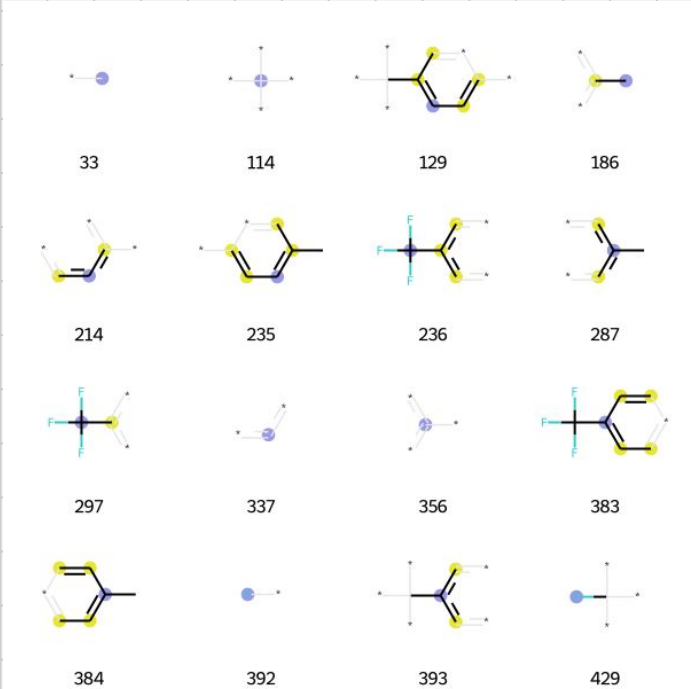


```
No of ones:  10
No of zeros:  1014

64      1
80      1
147     1
219     1
356     1
726     1
730     1
816     1
849     1
981     1
```

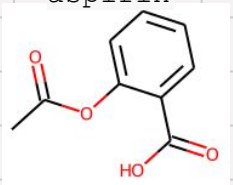


```
all_fragments = [(dataset.Structure[0], x, onbits) for x in mf.GetOnBits()]
Draw.DrawMorganBits(all_fragments[:],molsPerRow=4, legends=[str(x) for x in mf.GetOnBits()][:])
```

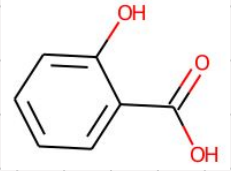


Molecular Similarity

aspirin



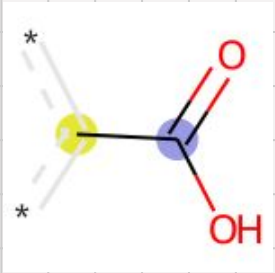
salicylic_acid



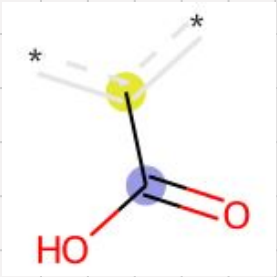

```
bit_asp = {}
bit_sal = {}
```

```
aspirin_fp = AllChem.GetMorganFingerprintAsBitVect(aspirin, 2, nBits=2048, bitInfo=bit_asp)
salicylic_acid_fp = AllChem.GetMorganFingerprintAsBitVect(salicylic_acid, 2, nBits=2048, bitInfo=bit_sal)
```

```
Draw.DrawMorganBit(salicylic_acid, 456, bit_sal)
```




```
Draw.DrawMorganBit(aspirin, 456, bit_asp)
```



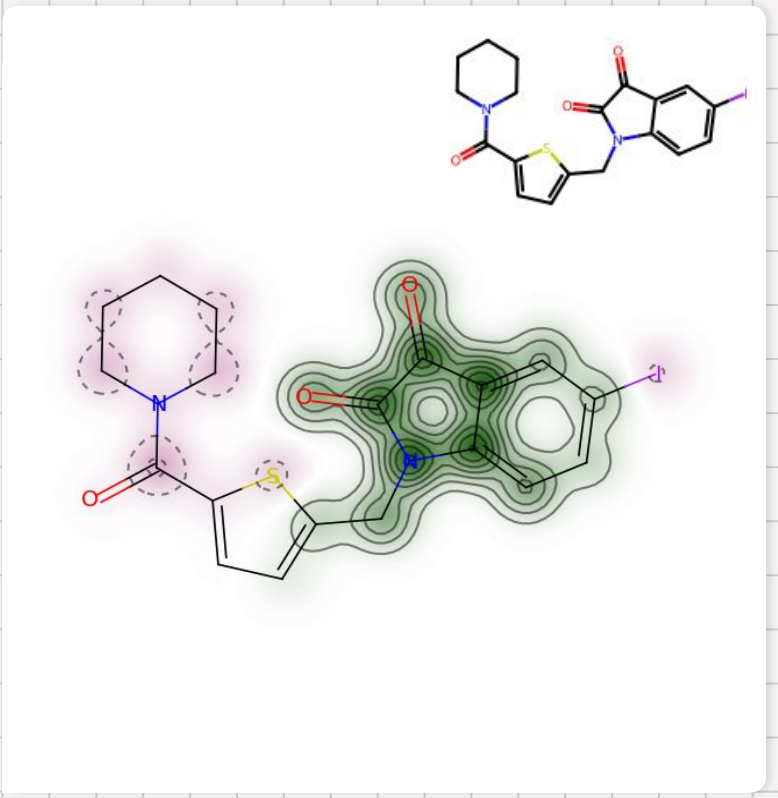
Measure	Expression	Range
Tanimoto/Jaccard coefficient	$\frac{c}{a+b-c}$	0 to 1
Euclidean distance	$\sqrt{a+b-2c}$	0 to N
City-block/Manhattan/Hamming distance	$a+b-2c$	0 to N
Dice coefficient	$\frac{2c}{a+b}$	0 to 1
Cosine similarity	$\frac{c}{\sqrt{ab}}$	0 to 1
Russell–RAO coefficient	$\frac{c}{m}$	0 to 1
Forbes coefficient	$\frac{cm}{ab}$	0 to 1
Soergel distance	$\frac{a+b-2c}{a+b-c}$	0 to 1

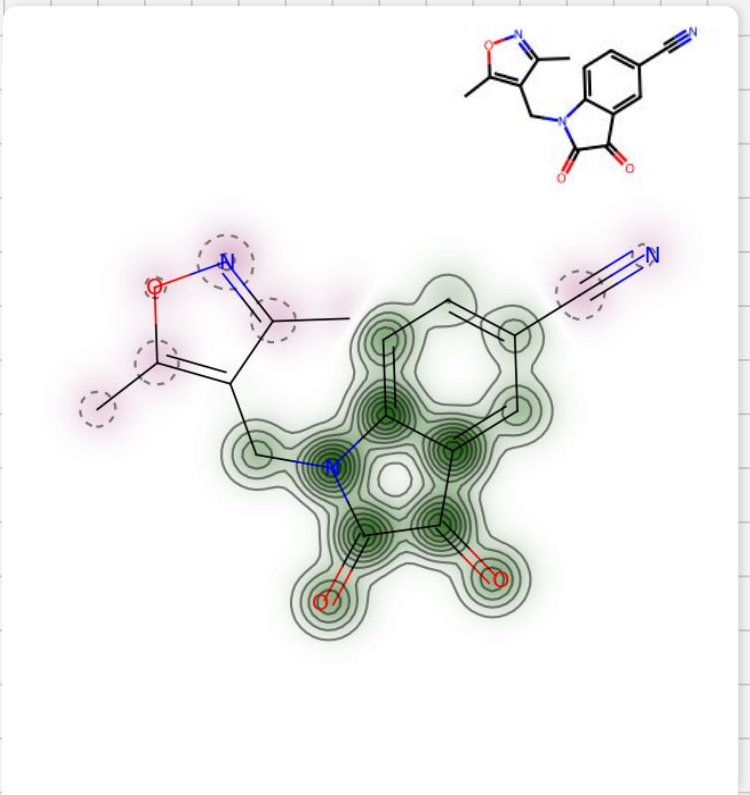
```
print("TanimotoSimilarity",DataStructs.FingerprintSimilarity(aspirin_fp, salicylic_acid_fp, metric=DataStructs.TanimotoSimilarity))
```

Tanimoto Similarity 0.44827586
Dice Similarity 0.619047619
Cosine Similarity 0.62546279

Comparing molecules

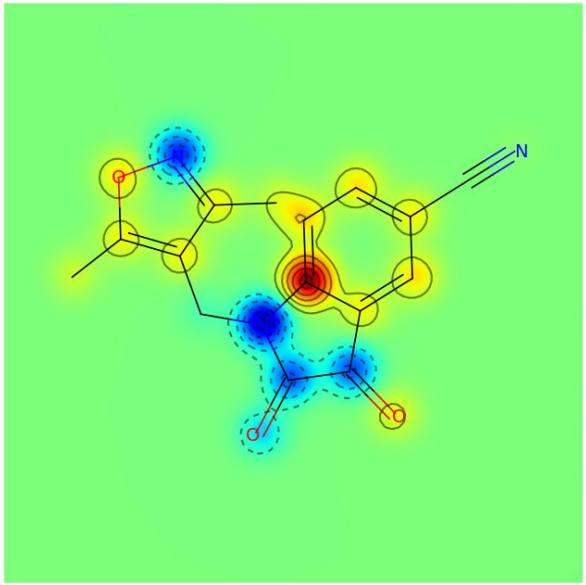
```
fig, maxweight =
SimilarityMaps.GetSimilarityMapForFingerprint(my_this_mol_obj,
my_that_mol_obj, SimilarityMaps.GetMorganFingerprint)
```





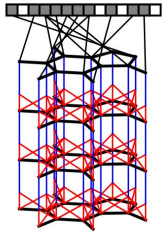
Descriptors similarity maps

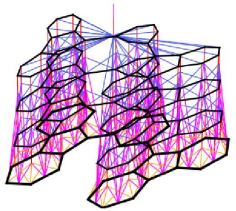
```
from rdkit.Chem import rdMolDescriptors
contribs = rdMolDescriptors._CalcCrippenContribs(my_this_mol_obj)
fig = SimilarityMaps.GetSimilarityMapFromWeights(my_this_mol_obj,[x for x,y in contribs], colorMap='jet',
contourLines=10)
```



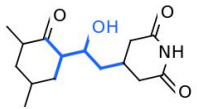
Convolutional Networks on Graphs for Learning Molecular Fingerprints

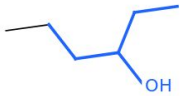
David Duvenaud†, Dougal Maclaurin†, Jorge Aguilera-Ipa
Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik,
Harvard University



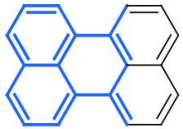


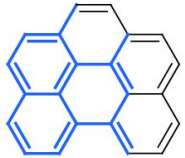
Fragments most activated by pro-solubility feature





Fragments most activated by anti-solubility feature





Thank you