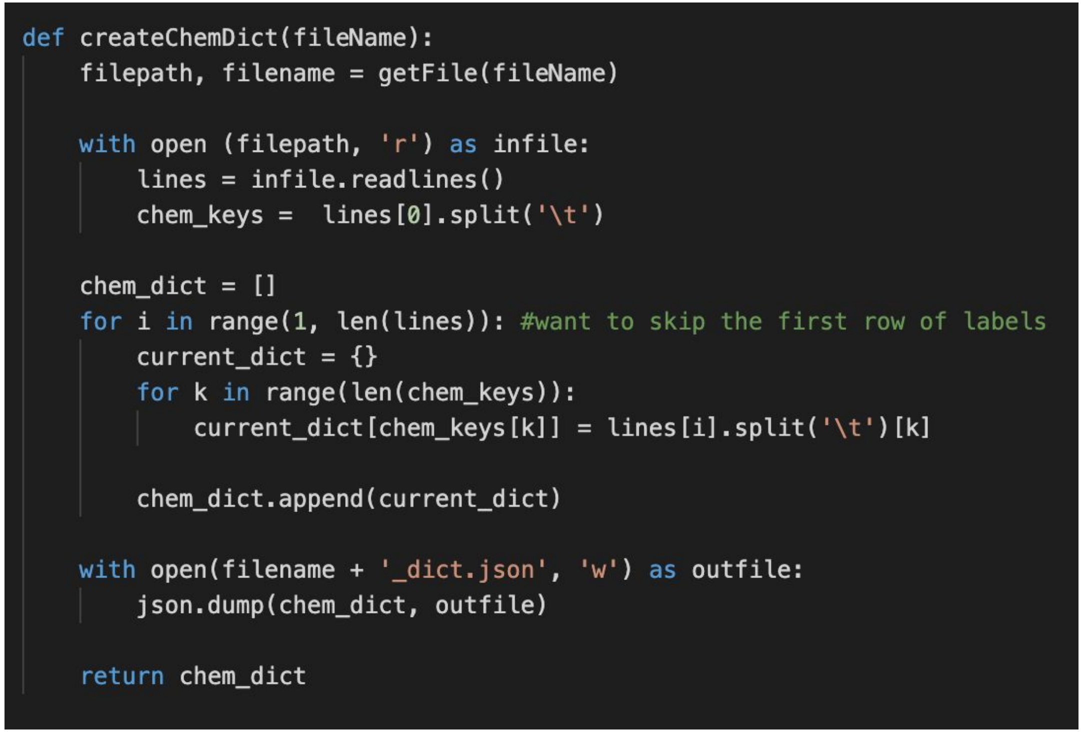
Hive 2021: Cheminformatics of UVCB substances

January 22, 2021

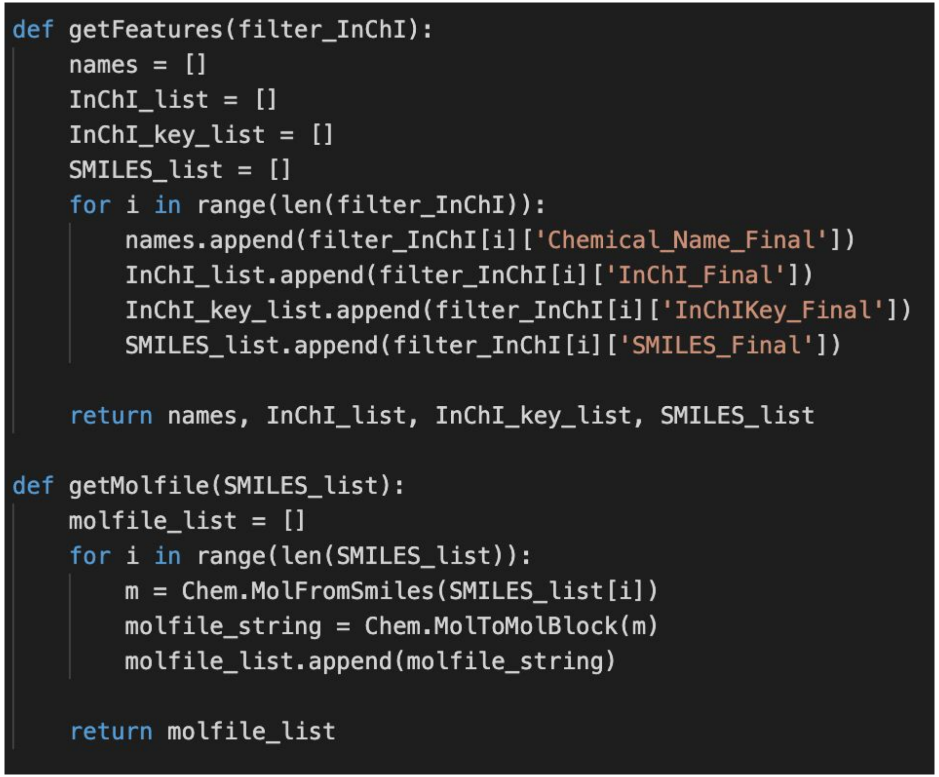
Processing mixture data



```python
def createChemDict(fileName):
    filepath, filename = getFile(fileName)

    with open (filepath, 'r') as infile:
        lines = infile.readlines()
        chem_keys =  lines[0].split('\t')

    chem_dict = []
    for i in range(1, len(lines)): #want to skip the first row of labels
        current_dict = {}
        for k in range(len(chem_keys)):
            current_dict[chem_keys[k]] = lines[i].split('\t')[k]

        chem_dict.append(current_dict)

    with open(filename + '_dict.json', 'w') as outfile:
        json.dump(chem_dict, outfile)

    return chem_dict
```

- Create a dictionary for each mixture
- Filter out mixtures that don’t have InChI

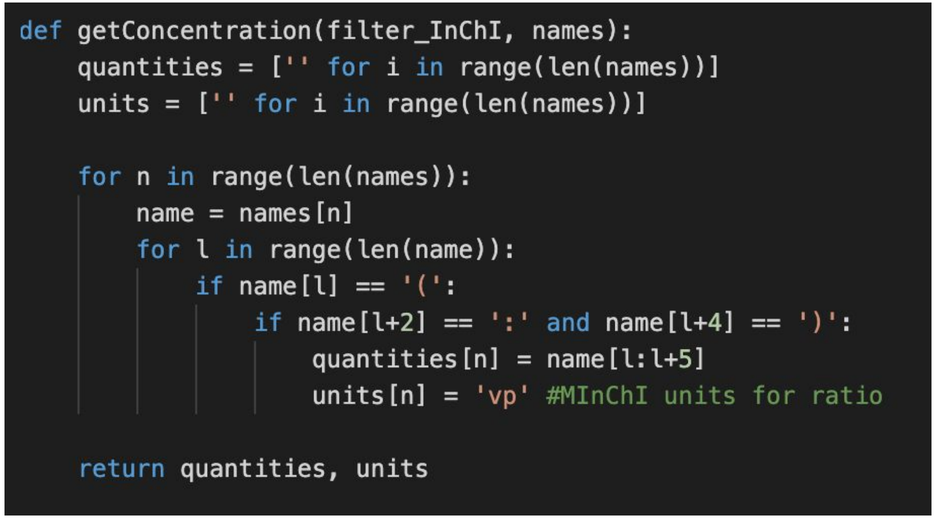
Extracting features for mixfile



```python
def getFeatures(filter_InChI):
    names = []
    InChI_list = []
    InChI_key_list = []
    SMILES_list = []
    for i in range(len(filter_InChI)):
        names.append(filter_InChI[i]['Chemical_Name_Final'])
        InChI_list.append(filter_InChI[i]['InChI_Final'])
        InChI_key_list.append(filter_InChI[i]['InChIKey_Final'])
        SMILES_list.append(filter_InChI[i]['SMILES_Final'])

    return names, InChI_list, InChI_key_list, SMILES_list

def getMolfile(SMILES_list):
    molfile_list = []
    for i in range(len(SMILES_list)):
        m = Chem.MolFromSmiles(SMILES_list[i])
        molfile_string = Chem.MolToMolBlock(m)
        molfile_list.append(molfile_string)

    return molfile_list
```

- Extract relevant information from each mixture dictionary
  - Chemical name
  - InChI
  - InChI key
  - SMILES
- Use rdkit to generate molfile from SMILES

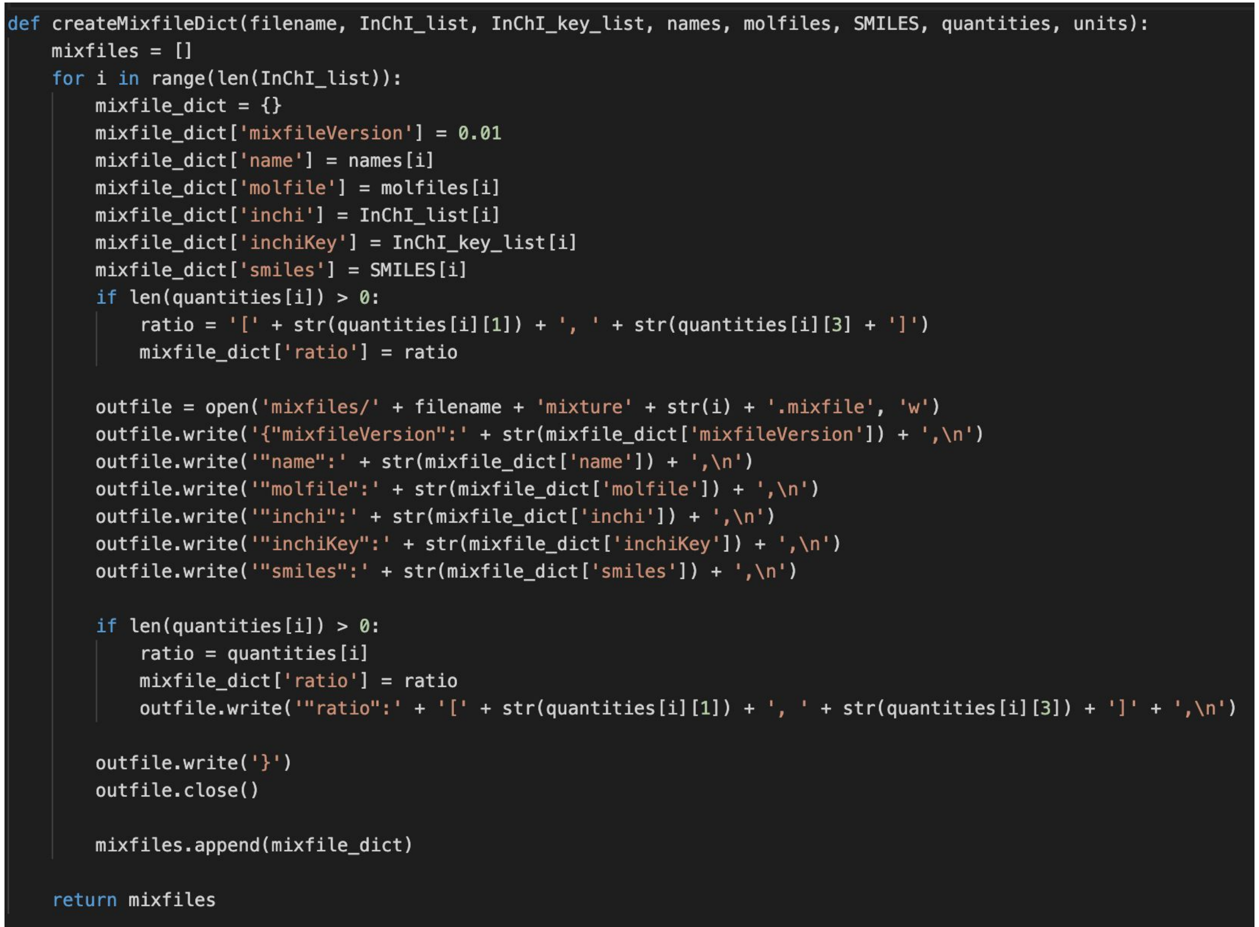
Determining concentration for each mixture



```python
def getConcentration(filter_InChI, names):
    quantities = ['' for i in range(len(names))]
    units = ['' for i in range(len(names))]

    for n in range(len(names)):
        name = names[n]
        for l in range(len(name)):
            if name[l] == '(':
                if name[l+2] == ':' and name[l+4] == ')':
                    quantities[n] = name[l:l+5]
                    units[n] = 'vp' #MInChI units for ratio

    return quantities, units
```

- Use concentrations for mixtures that have a concentration ratio like (1:1) or (1:2)
- Use “vp” for concentration units

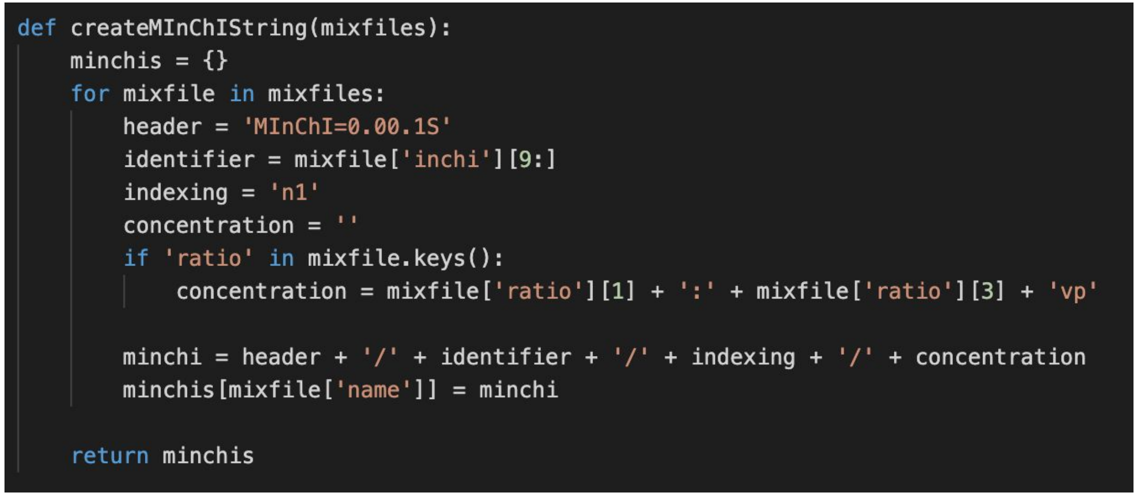
Creating the mixfile for each mixture



```python
def createMixfileDict(filename, InChI_list, InChI_key_list, names, molfiles, SMILES, quantities, units):
    mixfiles = []
    for i in range(len(InChI_list)):
        mixfile_dict = {}
        mixfile_dict['mixfileVersion'] = 0.01
        mixfile_dict['name'] = names[i]
        mixfile_dict['molfile'] = molfiles[i]
        mixfile_dict['inchi'] = InChI_list[i]
        mixfile_dict['inchiKey'] = InChI_key_list[i]
        mixfile_dict['smiles'] = SMILES[i]
        if len(quantities[i]) > 0:
            ratio = '[' + str(quantities[i][1]) + ', ' + str(quantities[i][3] + ']')
            mixfile_dict['ratio'] = ratio

        outfile = open('mixfiles/' + filename + 'mixture' + str(i) + '.mixfile', 'w')
        outfile.write('{"mixfileVersion":' + str(mixfile_dict['mixfileVersion']) + ',\n')
        outfile.write('"name":' + str(mixfile_dict['name']) + ',\n')
        outfile.write('"molfile":' + str(mixfile_dict['molfile']) + ',\n')
        outfile.write('"inchi":' + str(mixfile_dict['inchi']) + ',\n')
        outfile.write('"inchiKey":' + str(mixfile_dict['inchiKey']) + ',\n')
        outfile.write('"smiles":' + str(mixfile_dict['smiles']) + ',\n')

        if len(quantities[i]) > 0:
            ratio = quantities[i]
            mixfile_dict['ratio'] = ratio
            outfile.write('"ratio":' + '[' + str(quantities[i][1]) + ', ' + str(quantities[i][3]) + ']' + ',\n')

        outfile.write('}')
        outfile.close()

        mixfiles.append(mixfile_dict)

    return mixfiles
```

- Generate mixfile for each mixture that has an InChI value
- Unable to separate mixture into separate components (don't have SMILES for each component)

Generating MInChI string for each mixture



```python
def createMInChIString(mixfiles):
    minchis = {}
    for mixfile in mixfiles:
        header = 'MInChI=0.00.1S'
        identifier = mixfile['inchi'][9:]
        indexing = 'n1'
        concentration = ''
        if 'ratio' in mixfile.keys():
            concentration = mixfile['ratio'][1] + ':' + mixfile['ratio'][3] + 'vp'

        minchi = header + '/' + identifier + '/' + indexing + '/' + concentration
        minchis[mixfile['name']] = minchi

    return minchis
```

- Generate MInChI strings for all mixtures that have an InChI value
- Each MInChI has one component → index of 1

Examples of MInChI strings from EU_REACH.txt

Alcohols, C9-11':

'MInChI=0.00.1S/C10H22O/c1-2-3-4-5-6-7-8-9-10-11/h11H,2-10H2,1H3/n1/'

'1,2-Benzenedicarboxylic acid, di-C11-14-branched alkyl esters, C13-rich':

'MInChI=0.00.1S/C34H58O4/c1-29(2)23-17-13-9-5-7-11-15-21-27-37-33(35)31-25-19-20-26-32(31)34(36)38-28-22-16-12-8-6-10-14-18-24-30(3)4/h19-20,25-26,29-30H,5-18,21-24,27-28H2,1-4H3/n1/'

'9-Octadecenoic acid (9Z)-, compd. with 2-amino-2-methyl-1-propanol (1:1)':

'MInChI=0.00.1S/C18H34O2.C4H11NO/c1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18(19)20;1-4(2,5)3-6/h9-10H,2-8,11-17H2,1H3,(H,19,20);6H,3,5H2,1-2H3/b10-9-;/n1/1:1vp'

Examples of MInChI strings (cont.)

'Copper, complexes with diazotized 4-nitro-1,3-benzenediamine coupled with diazotized sodium 3-amino-4-methoxybenzenesulfonate (1:1) and resorcinol, sodium salts': 'MInChI=0.00.1S/C7H9NO4S.C6H7N3O2.C6H6O2.Cu.2Na/c1-12-7-3-2-5(4-6(7)8)13(9,10)11;7-4-1-2-6(9(10)11)5(8)3-4;7-5-2-1-3-6(8)4-5;;;/h2-4H,8H2,1H3,(H,9,10,11);1-3H,7-8H2;1-4,7-8H;;;/q;;;;2*+1/p-2/n1/1:1vp'

'4-acetamidobenzoic acid, compound with 1-(dimethylamino)propan-2-ol (1:1)': 'MInChI=0.00.1S/C9H9NO3.C5H13NO/c1-6(11)10-8-4-2-7(3-5-8)9(12)13;1-5(7)4-6(2)3/h2-5H,1H3,(H,10,11)(H,12,13);5,7H,4H2,1-3H3/n1/1:1vp'