2.7.2: Python Assignment 2B

Interconversion between PubChem records

Downloadable Files

Lecture03_list_conversion.ipynb

- Download the ipynb file and run your Jupyter notebook.
  - You can use the notebook you created in section 1.5 or the Jupyter hub at LibreText: https://jupyter.libretexts.org (see your instructor if you do not have access to the hub).
  - This page is an html version of the above .ipynb file.
    - If you have questions on this assignment you should use this web page and the hypothes.is annotation to post a question (or comment) to the 2019OLCCStu class group. Contact your instructor if you do not know how to access the 2019OLCCStu group within the hypothes.is system.

PUG-REST can be used to retrieve PubChem records related to another PubChem records. Basically, PUG-REST takes an input list of records in one of the three PubChem databases (Compound, Substance, and BioAssay) and returns a list of the related records in the same or different database. Here, the meaning of the relationship between the input and output records may be specified using an optional parameter. This allows one to do various tasks, including (but not limited to):

- Depositor-provided records (i.e., substances) that are standardized to a given compound.
- Mixture compounds that contain a given component compound.
- Stereoisomers/isotopomers of a given compound.
- Compounds that are tested to be active in a given assay.
- Compounds that have similar structures to a given compound.

Getting depositor-provided records for a given compound

First let's import the requests package necessary to make a web service request.

In [1]:

```
import requests
```

The code snippet below retrieves the substance record associated with a given CID (CID 129825914).

In [2]:

```
prolog    = "https://pubchem.ncbi.nlm.nih.gov/rest/pug"

pr_input  = "compound/cid/129825914"
pr_oper   = "sids"
pr_output = "txt"
url       = prolog + '/' + pr_input + '/' + pr_oper + '/' + pr_output

res = requests.get(url)
print(res.text)
```

```
341669951
```

It is also possible to provide a comma separated list of CIDs as input identifiers.

In [3]:

```
pugin   = "compound/cid/129825914,129742624,129783988"
pugoper = "sids"
pugout  = "txt"
url     = prolog + '/' + pugin + '/' + pugoper + '/' + pugout

res = requests.get(url)
print(res.text)
```

```
341669951
341492923
341577059
345261280
368769438
```

In the example above, the input list has three CIDs, but the PUG-REST request returned five SIDs. It means that some CID(s) must be associated with multiple SIDs, but it is hard to see which CID it is. Therefore, we want the SIDs grouped by the corresponding CIDs. This can be done using the optional parameter "**list_return=grouped**" and changing the output format to **json**.

In [4]:

```
pugin   = "compound/cid/129825914,129742624,129783988"
pugoper = "sids"
pugout  = "json"
pugopt  = "list_return=grouped"
url     = prolog + '/' + pugin + '/' + pugoper + '/' + pugout + "?" + pugopt

res = requests.get(url)
print(res.text)
```


```
{
  "InformationList": {
    "Information": [
      {
        "CID": 129825914,
        "SID": [
          341669951
        ]
      },
      {
        "CID": 129742624,
        "SID": [
          341492923
        ]
      },
      {
        "CID": 129783988,
        "SID": [
          341577059,
          345261280,
```

```
          368769438
        ]
      }
    ]
  }
}
```

Note that the **json** output format is used in the above request. The "**txt**" output format in PUG-REST returns data into a single column but the result from the above request cannot fit well into a single column.

If you want output records to be "flattened", rather than being grouped by the input identifiers, use "**list_return=flat**".

In [5]:

```
pugopt  = "list_return=flat"
url     = prolog + '/' + pugin + '/' + pugoper + '/' + pugout + "?" + pugopt

res = requests.get(url)
print(res.text)
```

```
{
  "IdentifierList": {
    "SID": [
      341492923,
      341577059,
      341669951,
      345261280,
      368769438
    ]
  }
}
```

The default value for the "list_return" parameter is:

- "flat" when the output format is TXT
- "grouped" when the output format is JSON and XML

It is also possible to specify the input list **implicitly**, rather than providing the input identifiers explicitly. For example, the following example uses a chemical name to specify the input list.

In [6]:

```
# Input CIDs are provided using a chemical name
url =
'https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/lactose/cids/txt'
res = requests.get(url)
cids = res.text.split()
print("# CIDs returned:", len(cids))
print(",".join(cids))

# Input CIDs are provided using the name, then coverted to SIDs.
url =
'https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/lactose/sids/txt'
```

```
res = requests.get(url)
sids = res.text.split()
print("# SIDs returned (method 1):", len(sids))
#print(",".join(sids))

# Input *SIDs* are provided using the name, and returned the input SIDs.
url =
'https://pubchem.ncbi.nlm.nih.gov/rest/pug/substance/name/lactose/sids/txt'
res = requests.get(url)
sids = res.text.split()
print("# SIDs returned (method 2):", len(sids))
#print(",".join(sids))
```

```
# CIDs returned: 7
6134,440995,84571,294,439186,49837892,69301022
# SIDs returned (method 1): 419
# SIDs returned (method 2): 125
```

The above example illustrates how the list conversion works.

- In the first request, the name "lactose" is searched for against the Compound database and the resulting 7 CIDs are returned.
- If you change the operation part from "cids" to "sids" (as in the second request), the same name search is done first against the **Compound** database, followed by the list conversion from the resulting 7 CIDs to associted 415 SIDs.
- In the third request, the name search is performed against the **Substance** database, and the resulting 125 SIDs are returned.

Exercise 1a Statins are a class of drugs that lower cholesterol levels in the blood. Retrieve in **JSON** the substance records associated with the compounds whose names contain the string "statin".

- Make only one PUG-REST request.
- For partial name matching, set the *name_type* parameter to "word" (See the PUG-REST document for an example).
- Group the substances by the corresponding compound records.
- Print the json output using print()

In [7]:

```
# Write your code in this cell.
```

Getting mixture/component molecules for a given molecule.

The list interconversion may be used to retrieve mixtures that contain a given molecule as a component. To do this, the input molecule should be a single-component compound (that is, with only one covalently-bound unit), and the optional parameter "**cids_type=component**" should be provided.

In [8]:

```
prolog    = "https://pubchem.ncbi.nlm.nih.gov/rest/pug"

url = prolog + "/compound/name/tylenol/cids/txt?cids_type=component"
res = requests.get(url)
cids = res.text.split()
print(len(cids))
print( cids )
```

```
349
['137528085', '137524012', '136090259', '134821662', '134821661', '134539182', '13223
```

It should be noted that, if the input molecule is a multi-component compound, the option "**cids_type=component**" returns the components of that compound. For example, the following example shows how to get all components of the first molecule in the "cids" list generated in the previous example.

In [9]:

```
url = prolog + "/compound/cid/" + cids[0] + "/cids/txt?cids_type=component"
res = requests.get(url)
component_cids = res.text.split()
print( "CID:", cids[0])
print( "Number of Components", len(component_cids))
print( component_cids )
```

```
CID: 137528085
Number of Components 3
['446155', '4891', '1983']
```

Exercise 2a: Many over-the-counter drugs contain more than one active ingredients. In this exercise, we want to find component molecules that occur with three common pain killers (aspirin, tylenol, advil) as a mixture.

Step 1. Define a list that contains three drug names (aspirin, tylenol, advil).

In [10]:

```
# Write your code in this cell.
```

Step 2. Using a for loop, retrieve PubChem CIDs corresponding to the three drugs and store them in a new list. In order not to overload the PubChem servers, stop the program for 0.2 second for each iteration in the for loop (using sleep()).

In [11]:

```
# Write your code in this cell.
```

Step 3. Using another for loop, do the following things for each drug:

- Get the PubChem CIDs of the mixture compounds that contain each drug and store them in a list.
- Get the PubChem CIDs of the components that occur in any of the returned mixtures, by setting the "list_return" parameter to "flat". This can be done with a single request.
- Print all the components.
- Stop the code for 0.2 second using sleep() each time a PUG-REST request is made.

In [12]:

```
# Write your code in this cell.
```

Getting compounds tested in a given assay

PUG-REST may be used to retrieve compounds tested in a given assay. For example, the following code cell shows how to get all compounds tested in AID 1207599.

In [13]:

```
url = prolog + "/assay/aid/" + "1207599" + "/cids/txt"
res = requests.get(url)
cids = res.text.split()
print(len(cids))
print(cids)
```

```
791
['6175', '6197', '8547', '10219', '14169', '17558', '21389', '68050', '84677', '95783
```

If you are interested in only the compounds that are tested "active" in a given assay, set the "**cids_type**" parameter to "**active**", as shown in the code below.

In [14]:

```
url = prolog + "/assay/aid/" + "1207599" + "/cids/txt?cids_type=active"
res = requests.get(url)
cids = res.text.split()
print(len(cids))
print(cids)
```

```
435
['6197', '10219', '14169', '17558', '68050', '177894', '182792', '253602', '348623',
```

It is also possible to specify the input assay list implicitly. For example, the following code cell retrieves compounds tested in any assays targeting human Carbonic anhydrase 2 (CA2), whose accession number is P00918.

In [15]:

```
url = prolog + "/assay/target/accession/" + "P00918" + "/cids/txt"
res = requests.get(url)
cids = res.text.split()
print(len(cids))
#print(cids)
```

```
23978
```

Exercise 3a: Find compounds that are tested to be active against human acetylcholinesterase (accession: P08173) and retrieve SMILES strings for those compounds.

- Split the CID list into smaller chunks (with a chunk size of 100).
- Print the retrieved data in a CSV format (CID and SMILES strings in the first and second columns, respectively).

In [16]:

```
# Write your code in this cell.
```

In [ ]:

```

```