Module 8: Interacting with Databases: Desktop and web based applications

Andrew Lang (alang@oru.edu) & Jordi Cuadros, (jordi.cuadros@iqs.url.edu)

Learning Objectives

- Have a basic understanding of the main interfaces and technologies involved in using Web APIs
- Gain a knowledge of the main chemistry Web APIs
- Be able to pull chemical information programmatically from some chemistry online databases and services

Module Structure

Week I

8.1 Why using databases programmatically?
8.2 Technological aspects behind using a web API
8.3 Main chemistry APIs
Week I activities

Week II

8.4 Other relevant sources for data: Other web APIs & web scrapping
8.5 An introduction to programming and to using string functions
8.6 Calling the APIs programmatically from desktop (spreadsheets) and web based applications (Javascript & Google scripts)
Week II activities

Module 8: Interacting with Databases: Desktop and web based applications (Week I)

Andrew Lang (alang@oru.edu) & Jordi Cuadros, (jordi.cuadros@iqs.url.edu)

8.1 Why using databases programmatically?

Information is essential to many professions and chemistry is no exception. As chemists we frequently have to look up chemical information in handbooks or databases; you can name it: formation enthalpies and entropies, solubilities, ionization constants, densities of mixtures, spectra...

Much of this information, although available, is spread among several handbooks, databases and/or papers in scientific journals. Sometimes we are able to find the information in well organized data sources and occasionally these are available online and can be accessed programmatically.

When this is the case, accessing programmatically to these sources offers the following advantages:

- Availability and convenience. Curated and updated information is at your fingerprints. The information can be retrieved from your computer or your mobile device without requiring heavy applications or huge databases.
- Economy. Many of these source offers data on licensing options that allow using it at no-cost in different contexts and environments.
- Knowledge and data sharing, reusing and repurposing. Many of the sources that are available fit onto the open access policies which are slowly being adopted by the scientific communities. Likewise, this open data allows you to build other applications and personal mashups.

Some example projects that make use of open databases are:

- Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/
- JSmol: an open-source HTML5 viewer for chemical structures in 3D. http://wiki.jmol.org/index.php/JSmol
- Open Notebook Science Web Services. http://onswebservices.wikispaces.com/
- SpectralGame. http://lxsrv7.oru.edu/~alang/
- MolView. http://molview.org/

- CheMagic. http://chemagic.com/molecules/VMKMINInotes.htm

Further reading

- O'Boyle, N. M., Guha, R., Willighagen, E. L., Adams, S. E., Alvarsson, J., Bradley, J. C., ... & Murray-Rust, P. (2011). Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on. *J. Cheminformatics*, *3*, 37. Available at http://www.biomedcentral.com/content/pdf/1758-2946-3-37.pdf

8.2 Technological aspects behind using a web API

One common way to offer open data to be used programmatically is through a web API. Before digging into the most relevant chemistry APIs, we will spend a moment discussing some technological aspects that should be known in order to use it.

What's a web API?

An **API** (Application Programming Interface) is the set of elements that a programming library or service makes available to other programmers to be used remotely to access their services and data.

This API is called a **web API** when these functionalities are delivered via the HTTP protocol through the Internet.

How is a web API used?

In order to be able to get information from an online database, we will need to:

1. Send a query to the database through the Internet (HTTP protocol)
2. Understand the format in which the response will be received
3. Process this response to extract the desired information

Sending the query... - The HTTP protocol and the REST architecture

HTTP stands for HyperText Transfer Protocol and it comprises the set of rules that control the transfer of a resource between a web server and a web client (usually a browser).

The basic elements of an HTTP transaction are:

- A request message
- A response message (which is usually the resource being transferred)

The HTTP request message

The request message consists in some text data sent to an URL (Uniform Resource Locator). An URL is a way of specifying an address in the Internet.

The general structure for an URL consists of

```
scheme://[user:password@]domain:port/path?query_string#fragment_id
```

although in HTTP this commonly simplified to

```
http://domain/path or https://domain/path
```

The request message, which is usually handled by the HTTP client, has the following structure

- A first line which includes method, path and HTTP protocol version, usually
  ```
  METHOD /path HTTP/x.x
  ```
- Several header lines
- A blank line
- And an optional message body

The HTTP request methods: GET and POST

Although other methods exist, the most common request methods are GET and POST.

In the GET method, any additional information required to specify the request is included in the URL (which means in the path included in the first line of the message). The parameters are included after the path and a question mark as pairs name=value. Parameters are separated by the ampersand symbol (&)

```
/path?name1=value1&name2=value2
```

In some cases, as we will see in PubChem API, the parameters are included in the path. This technique is called URL rewriting and shortens the URLs and makes it more usable.

```
/path/name1/value1/name2/value2
```

or even

```
/path/value1/value2
```

In POST methods, any additional information is not included in the URL but in the body of the message and is usually submitted via a web form. We won't go further into this method since most of the chemistry APIs can be queried via GET calls.

To end this brief introduction to HTTP, note that some characters are either disallowed or have special meaning in URLs. When these characters appear they will need to be encoded; for example a space is encoded as %20, and a sharp (#) is %23. A reference can be found at http://www.w3schools.com/tags/ref_urlencode.asp.

The HTTP response message

As we have seen for the request, the response contains:

- A first line
- Several header lines
- A blank line
- And a message body which is the transferred resource

Two aspects of the response to be highlighted are the status code (included in the first line) and the content-type information (which is one of the headers).

HTTP status codes are indicated as a three-digit number

- 200 means transaction completed successfully
- 3xx means some type of redirection
- 4xx means a client-side error
    - 404 means resource not found
- 5xx means a server-side error
    - 500 means unexpected server error

The content type parameter indicates what data is included in the message body. Each type of resource is indicated with a standardized string (Internet media type; aka MIME type). For example, some common response formats are

- HTML file: text/html
- XML file: application/xml or text/xml
- CSV file: text/csv
- JSON file: application/json
- PNG image: image/png

Querying the databases

We have so far seen how the information is transmitted over the WWW. But how are databases accessed over the WWW? There are two main architectures used to access databases over the Net: SOAP and REST. Given its prevalence, we will center our discussion on the REST architecture.

Some major characteristics of the REST (REspresentational State Transfer) architecture are:

- It is usually implemented over HTTP.
- The queries are implemented as GET requests.
- The communication is stateless. Any query must contain all required information. Nothing is saved in the server.
- The query result is obtained as a response in a predefined format: XML, JSON, CSV, HTML, plain text...

Further reading

- http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177
- http://code.tutsplus.com/tutorials/a-beginners-guide-to-http-and-rest--net-16340
- https://www.addedbytes.com/articles/for-beginners/url-rewriting-for-beginners/
- https://www.iana.org/assignments/media-types/media-types.xhtml
- http://www.restapitutorial.com/

Understanding the responses... - Common structured data filetypes

Common formats in which a web API may provide its response are HTML, XML, CSV, JSON and plain text. A short explanation of the first four types follows next. Feel free to skip it if you already know their characteristics.

HTML (HyperText Markup Language)

HTML is the standard language used to create web pages. When a web API outputs its result in HTML, this resource is usually intended to be viewed a browser. When no better option is available, HTML can be processed (parsed) to extract a specific information.

Likewise, any web page is published as HTML and so can be processed to extract any relevant information contained therein.

An example of HTML can be found here: http://kinetics.nist.gov/janaf/html/Cl-054.html
This the output of a search in the NIST-JANAF Thermodynamical Tables online database. The full HTML code can be read looking at the source of the page. A fragment is reproduced in Figure 1.

```
163 <TR>
164 <TD>500</TD>
165 <TD WIDTH=10></TD>
166 <TD>53.940</TD>
167 <TD WIDTH=10></TD>
168 <TD>122.019</TD>
169 <TD WIDTH=10></TD>
170 <TD>100.905</TD>
171 <TD WIDTH=10></TD>
172 <TD>10.557</TD>
173 <TD WIDTH=10></TD>
174 <TD>-387.693</TD>
175 <TD WIDTH=10></TD>
176 <TD>-351.283</TD>
177 <TD WIDTH=10></TD>
178 <TD>36.698</TD>
179 </TR>
```

Figure 1. Part of the HTML code of a page

More information on HTML can be found at http://www.w3schools.com/html/.

XML (eXtensible Markup Language)

XML is a markup language designed to contain structured data in format easy to be read by computers and for humans. An example of an XML response can be found running the following query:

https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/1/property/MolecularFormula,MolecularWeight,InChIKey/XML. Figure 2 shows its response.

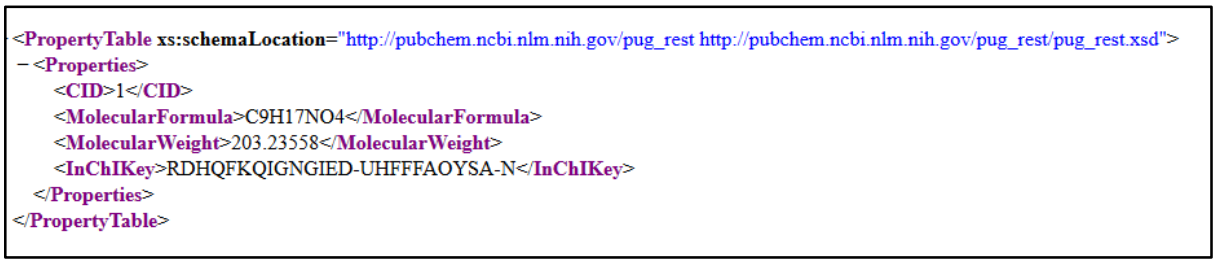

Figure 2. An XML response

More information on XML can be found at http://www.w3schools.com/xml/.

CSV (Comma Separated Values)

CSV, for Comma Separated Values, is a plain-text file format use to store tabular data. In practice, CSV is not a unique format but different variations depending on the implementation and the regional setup of the operating system.

CSV main features:

- Data is stored as plain text.
- Records are separated by new-line characters.
- Each record contains several fields. Columns (fields) are separated by a delimiting character (usually a comma, a semicolon or a tab).
- Each record must have the same number of fields.

CSV main variations:

- Any encoding may be used for the text file: ASCII, Latin-1, UTF-8...
- Values may be quoted or not.
- End-of-line character, delimiting character, quotes and escaping sequences may vary.
- There may be header lines.

An example of a response in CSV is obtained when running the following query http://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/1000,1001/assaysummary/CSV. The response is shown in Figure 3.

```
1  "AID","AID Version","AID Revision","Panel Member ID","SID","CID","Bioactivity
   Outcome","Target GI","Target GeneID","Activity Value [uM]","Activity
   Name","Assay Name","Bioassay Type","PubMed ID","RNAi"
2  6408,3,8,"",103183980,1000,"Inconclusive","","","","","Affinity for
   5-hydroxytryptamine receptor was determined using male Dawley rat fundus
   preparation; Not determined","Other",7069716,""
3  22830,3,3,"",103183980,1000,"Unspecified","","","","","Vmax value was
   measured","Other",3336016,""
4  155315,3,7,"",103183980,1000,"Unspecified","","","","","Vmax was determined
   against bovine phenylethanolamine N-Methyltransferase (PNMT)","Other",14695818,""
5  155338,4,5,"",103183980,1000,"Unspecified","","","","","Km ratio for human and
   bovine Phenylethanolamine N-methyl-transferase","Other",11412985,""
6  156061,3,7,"",103183980,1000,"Unspecified","","","","","In vitro enzyme kinetics
   (PNMT)of the compound expressed as turnover rate isolated from bovine adrenal
   glands","Other",3336033,""
```

Figure 3. Fragment of a CSV response

More information can be obtained at https://en.wikipedia.org/wiki/Comma-separated_values.

JSON (JavaScript Object Notation)

JSON, for JavaScript Object Notation, is a format that encodes data object as human-readable text, organized as pairs attribute-value (name:value). Its main features are being a quite compact notation with easy conversion to Javascript objects (`eval()`, `JSON.parse()`).

An example query is https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/aspirin/property/MolecularWeight/JSON. The response is shown in Figure 4.

```
{
  "PropertyTable": {
    "Properties": [
      {
        "CID": 2244,
        "MolecularWeight": 180.15742
      }
    ]
  }
}
```

Figure 4. A JSON response

More information can be found at http://www.json.org/.

Processing the responses... - Parsers and string functions

Ultimately the response will need to be processed to extract the desired information. This can be done using specific functions or parsers or through the use of some simple string manipulation functions.

Sometimes you want to parse XML or HTML using custom string functions for deleting whitespace, finding the position of certain characters within strings and for extracting substrings. These functions (plus many more) appear in all commonly used languages.

Table 1 summarizes some of these functions in the programming environments we will be discussing next week.

Table 1. Main string functions

Spreadsheet (Excel, Calc)	**Basic (VBA, Libre Basic)**	**Javascript**
SEARCH, FIND	InStr	search, indexOf
MID	Mid	substr, substring
SUBSTITUTE	Replace	replace
TRIM	Trim	trim
CONCATENATE	& (operator),	concat

8.3 Main chemistry APIs

CIR (Chemical Identifier Resolver)

The Chemical Identifier Resolver is one of several online tools offered by the CADD Group Chemoinformatics Tools and User Services at the National Cancer Institute.

The CIR web service will convert one identifier (e.g. SMILES) to another (e.g. formula). It can be accessed via a web form with an inbuilt structure editor at http://cactus.nci.nih.gov/chemical/structure or via its web API.

An incomplete web API reference is found at http://cactus.nci.nih.gov/chemical/structure_documentation. Further information can be obtained either by using the form version of the tool or by reading the web services blog: http://cactus.nci.nih.gov/blog.

The general structure for querying the web service is
`http://cactus.nci.nih.gov/chemical/structure/"structure identifier"/"representation"`
where "structure identifier" is the input and "representation" indicates the desired identifier for output.

An example query is http://cactus.nci.nih.gov/chemical/structure/aspirin/smiles which returns a unique SMILES for aspirin.

Default output format is plain text, but XML output can be obtained by appending `/xml` to the URL, e.g. http://cactus.nci.nih.gov/chemical/structure/aspirin/names/xml.

OPSIN (Open Parser for Systematic IUPAC nomenclature)

OPSIN (Open Parser for Systematic IUPAC nomenclature) is service provided by the Centre for Molecular Informatics at the University of Cambridge. It can be used to generate a structure (and some common identifiers) from a systematic chemical name.

The reference for the web service can be found at http://opsin.ch.cam.ac.uk/instructions.html. Its entry point is http://opsin.ch.cam.ac.uk/opsin.

The query should be set as
`http://opsin.ch.cam.ac.uk/opsin/"name"."type"`
where "name" refers to the chemical to be processed and "type" indicates the expected response.

For example, http://opsin.ch.cam.ac.uk/opsin/benzoic+acid.smi returns the SMILES representation of benzoic acid.

CDK (Chemistry Development Kit)

The CDK is an open source program that can generate descriptors from structure. It can be used online as a web API from a server at Uppsala University or by installing it on your own host.

Documentation is available at http://rest.rguha.net/. Servers are provided at the very bottom of the page.

For example, http://ws1.bmc.uu.se:8182/cdk/fingerprint/std/CCO shows a fingerprint for ethanol (plain text response) and http://ws1.bmc.uu.se:8182/cdk/depict/200/200/CCO provides a 2D image of this molecule in PNG.

The list of available molecular descriptors can be looked up at `/cdk/descriptors`.

ChemSpider

ChemSpider APIs allow programmatic access to part of the ChemSpider databases at The Royal Society of Chemistry. They can be accessed through SOAP and/or a REST interface. Some operations require a security token which can be obtained by registering (free) and looking it up in the user profile page.

ChemSpider web services includes four different APIs: Search API, InChI API, MassSpec API and Spectra API. The general documentation is available at http://www.chemspider.com/aboutservices.aspx.

ChemSpider Search API allows searching ChemSpider databases by chemical identifier, structure or properties and retrieve information about the associated records. All operations require a security token and some require a service subscriber role. Not all operations are available through a REST interface. The documentation can be found at http://www.chemspider.com/Search.asmx.

Response is usually XML and binary information is base64 encoded. Note that some operations are asynchronous. A first call launches the calculation and an id is produced. The id is then used to access the response in a second call to GetAsyncSearchResult.

An example is `http://www.chemspider.com/Search.asmx/GetCompoundInfo?csid=1&token=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` where token must be replaced with the user's one. Results are shown in Figure 5.

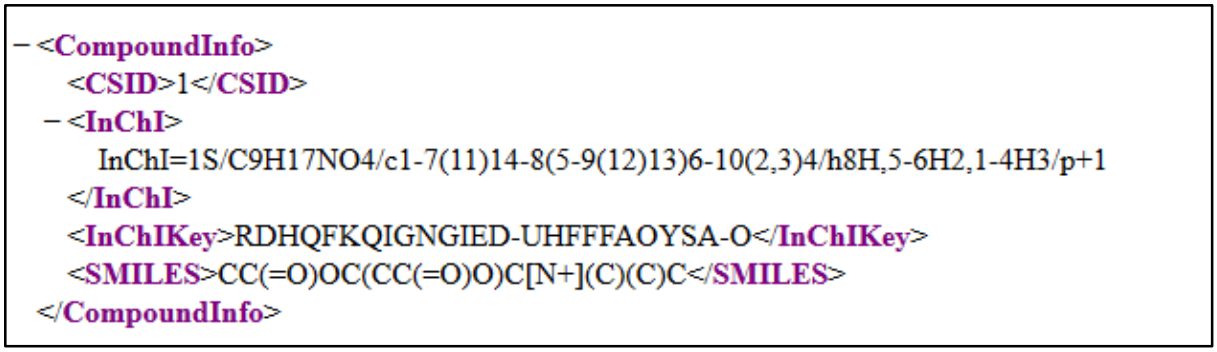

Figure 5. A search result from ChemSpider Search API.

ChemSpider InChI API allows interconversion between some chemical identifiers, namely CSID, mol, InChI, InChIKey and SMILES. Few operations require a security token and/or are not available through a REST interface. Documentation is found at http://www.chemspider.com/InChI.asmx. Response is usually in XML.

An example is http://www.chemspider.com/InChI.asmx/SMILESToInChI?smiles=CCO.

ChemSpider MassSpec API allows searching ChemSpider by formula or by mass. All operations require a security token and some require a service subscriber role. Not all operations are available through a REST interface. Documentation is available at http://www.chemspider.com/MassSpecAPI.asmx.

Response is usually XML. Some operations are asynchronous; a first call launches the calculation and an id is produced. The id is then used to access the response in a second call to GetAsyncSearchResult (Search API).

An example is the call
http://www.chemspider.com/MassSpecAPI.asmx/SearchByMass2?mass=1888&range=0.1.

ChemSpider Spectra API allows searching spectra in ChemSpider database. All operations require a security token with a service subscriber role. The documentation is available at http://www.chemspider.com/Spectra.asmx.

PubChem

PubChem, a product of the National Center for Biotechnology Information, is accessed programmatically through the PUG (Power User Gateway) service. This service can be used either directly or via any of its two interfaces: PUG SOAP or PUG REST. We will, for now, ignore the SOAP interface and focus on the PUG REST service.

When accessed directly, PUG is a queued service that allows searching and downloading of records in the PubChem database: substances and compounds, structures, bioassays... Queries are sent via an HTTP POST call with request and response messages being XML

documents. Documentation can be found at https://pubchem.ncbi.nlm.nih.gov/pug/pughelp.html and http://www.ncbi.nlm.nih.gov/home/api.shtml.

The PUG REST interface offers a way to access the PubChem database with the convenience of the REST architecture. It allows us to query the PubChem database for data on substances (uncurated data), compounds (curated data) or assays. Documentation is available at https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST_Tutorial.html and https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST.html. Although most operations are synchronous, some (the asynchronous ones) require a two-step query for obtaining the response (a listkey is returned).

The general structure for using PUG REST is

http://pubchem.ncbi.nlm.nih.gov/rest/pug	**/compound/name/vioxx**	**/property/InChI**	**/TXT**
prolog	*input*	*operation*	*output*

PUG REST output format can be any of the formats shown in Table 2.

Table 2. PubChem output formats

Output Format	Description
XML	standard XML, for which a schema is available
JSON	JSON, JavaScript Object Notation
JSONP	JSONP, like JSON but wrapped in a callback function
ASNB	standard binary ASN.1, NCBI's native format in many cases
ASNT	NCBI's human-readable text flavor of ASN.1
SDF	chemical structure data
CSV	comma-separated values, spreadsheet compatible
PNG	standard PNG image data
TXT	plain text

Two examples queries are https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/aceticacid/PNG and https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/methanol/property/IUPACName,MolecularWeight/CSV. The responses are shown in Figures 6 and 7.

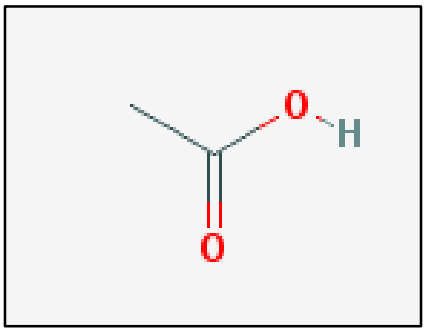

Figure 6. A PNG response from PubChem PUG REST API.

```
1  "CID","IUPACName","MolecularWeight"
2  887,"methanol",32.041860
```

Figure 7. A CSV response from PubChem PUG REST API.

PDB (Protein DataBank)

RSCB (Research Collaboratory for Structural Bioinformatics) PDB (Protein DataBank) offers three web services to ease accessing to its data. These are documented at http://www.rcsb.org/pdb/software/rest.do.

Currently they include a HTTP POST query method for advanced searching, a simple search REST API for ligands using SMILES as identifier, and a REST API for fetching descriptions of ligands, structures, files, annotations from a PDB id.

The simple search REST API returns the identifiers and descriptions for the ligands matching the search query. For example, http://www.rcsb.org/pdb/rest/smilesQuery?smiles=CC%23C&search_type=substructure performs a substructure search including the group CC#C. Part of the results are shown in Figure 8.

Figure 8. Part of the response to substructure search in PDB.

The fetch API allows the retrieval of the data in the database from its identifiers in PDB. For example, http://www.rcsb.org/pdb/rest/describeHet?chemicalID=CB3 recovers the data for the chemical with id CB3. Responses are commonly in XML.

ChEMBL

The last API accessible database we will consider is ChEMBL, which is a database of bioactive drug-like small molecules hosted at the European Bioinformatics Institute. It provides a RESTful API which responds in XML, JSON or YAML. The main documentation is available at https://www.ebi.ac.uk/chembl/ws.

The API is split into two services:

- data, for accessing the database
- utils, for accessing some cheminformatics tools

The data API allows searching in the ChEMBL database for assays, molecules, target, cell-lines... It includes options for filtering and searching by similarity or substructure. Note that results are paginated. If the number of records returned is larger than 20, you may have to take care of this. This API is documented at https://www.ebi.ac.uk/chembl/api/data/docs.

An example of using this API is https://www.ebi.ac.uk/chembl/api/data/molecule.json?molecule_structures__canonical_smiles__flexmatch=OCC. The result in JSON format is shown in Figure 10.


```
{"molecules": [{"atc_classifications": ["D08AX08", "V03AB16", "V03AZ01"],
"availability_type": "2", "biotherapeutic": null, "black_box_warning": "0", "chebi_par_id":
16236, "chirality": "2", "dosed_ingredient": true, "first_approval": null, "first_in_class":
"0", "helm_notation": null, "indication_class": "Pharmaceutic Aid (solvent); Anti-Infective,
Topical", "inorganic_flag": "0", "max_phase": 4, "molecule_chembl_id": "CHEMBL545",
"molecule_hierarchy": {"molecule_chembl_id": "CHEMBL545", "parent_chembl_id": "CHEMBL545"},
"molecule_properties": {"acd_logd": "-0.18", "acd_logp": "-0.18", "acd_most_apka": null,
"acd_most_bpka": null, "alogp": "-0.01", "aromatic_rings": 0, "full_molformula": "C2H6O",
"full_mwt": "46.07", "hba": 1, "hbd": 1, "heavy_atoms": 3, "med_chem_friendly": "Y",
"molecular_species": "NEUTRAL", "mw_freebase": "46.07", "mw_monoisotopic": "46.0419",
"num_alerts": 0, "num_ro5_violations": 0, "psa": "20.23", "qed_weighted": "0.41",
"ro3_pass": "Y", "rtb": 0}, "molecule_structures": {"canonical_smiles": "CCO",
"standard_inchi": "InChI=1S/C2H6O/c1-2-3/h3H,2H2,1H3", "standard_inchi_key": "LFQSCWFLJHTTHZ-
UHFFFAOYSA-N"}, "molecule_synonyms": [{"syn_type": "FDA", "synonyms": "Alcohol"},
{"syn_type": "OTHER", "synonyms": "Alcohol"}, {"syn_type": "USAN", "synonyms": "Alcohol"},
{"syn_type": "USP", "synonyms": "Alcohol"}, {"syn_type": "NATIONAL_FORMULARY", "synonyms":
"Alcohol, Diluted"}, {"syn_type": "OTHER", "synonyms": "Alcohol, Diluted"}, {"syn_type":
"USAN", "synonyms": "Alcohol, Diluted"}, {"syn_type": "INN", "synonyms": "Alcohol,
Rubbing"}, {"syn_type": "USAN", "synonyms": "Alcohol, Rubbing"}, {"syn_type": "USP",
"synonyms": "Alcohol, Rubbing"}, {"syn_type": "E_NUMBER", "synonyms": "E1510"}, {"syn_type":
"INN", "synonyms": "Ethanol"}, {"syn_type": "JAN", "synonyms": "Ethanol"}, {"syn_type":
"TRADE_NAME", "synonyms": "Ethanol"}, {"syn_type": "OTHER", "synonyms": "Ethyl Alcohol"},
{"syn_type": "TRADE_NAME", "synonyms": "Ethyl Alcohol"}, {"syn_type": "OTHER_PC",
"synonyms": "SID17389892"}], "molecule_type": "Small molecule", "natural_product": "0",
"oral": false, "parenteral": true, "polymer_flag": false, "pref_name": "ALCOHOL", "prodrug":
"0", "structure_type": "MOL", "therapeutic_flag": false, "topical": true, "usan_stem": null,
"usan_stem_definition": null, "usan_substem": null, "usan_year": null}], "page_meta":
{"limit": 20, "next": null, "offset": 0, "previous": null, "total_count": 1}}
```


Figure 10. JSON response for the ChEMBL data API

The utils API offers some interesting function to obtain molecular descriptors and to generate or process graphical representations of molecules, but requires most inputs as base64 encoded strings. Documentation can be found at https://www.ebi.ac.uk/chembl/api/utils/docs.

As a side note, you may use https://www.base64encode.org/ to encode and decode base64 strings.

An example use of this utils API is https://www.ebi.ac.uk/chembl/api/data/image/CHEMBL25?format=svg where a SVG image of the molecule with id CHEMBL25 is obtained (see Figure 11).

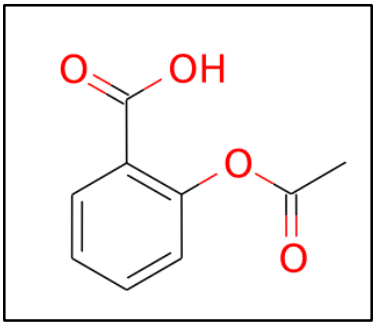

Figure 11. SVG image obtained from ChEMBL utils API.

Week I activities

Activity 1

Choose a chemical and, using the web services presented in this module, obtain as much information as you can about it. Present the collected information in a report indicating, for each piece the complete URL (parameters included) for the query and the response obtained.

Activity 2

Make a diagram or schema that summarizes which APIs allows you to convert from one information of a chemical to another. At least, the schema should include the following informations: name, IUPAC name, molecular weight, boiling point, SMILES, unique SMILES, mol file, InChI, formula, 2D molecular drawing...