Manual

Contents

Introduction to the GIT repository

The Python pipeline is used to create the first framework of a unified semantic database for the biobricks in the current iGEM Registry. This Wikibase instance is stored in a Resource Description Framework (RDF) format that makes the data easily retrievable using a triple format query. The pipeline annotates the biological parts with protein functions, restriction sites from restriction enzymes, and interconnects information of parts of the biobricks that are composed of other biological parts. Wikibase is a semantic database. The BioParts Wikibase is an instance that helps navigate the iGEM registry and it is constructed in the iGEM 2021 competition and to be updated upon by the iGEM community. This git repository contains a current version and previous versions used to upload to the BioParts Wikibase. Additionally, this git repository also contains query examples to retrieve information intelligently from the BioParts Wikibase.

Installation of packages

To use the pipeline as a whole, several packages are required. Their use differs slightly per script within the repository.

WikidataIntegrator 0.7.4

This package is used to read and write to the Wikibase. It is specifically created to upload biological information to Wikibase instances. The package is used to create statements in the correct format and write these to the designated item page. Additionally, it also acts as a wrapper for the queries performed to the BioParts Wikibase to retrieve information.

Biopython 1.77

This package is used to find restriction sites in the nucleotide sequences of the biological parts and makes sure that the annotation of this is correct with the standards as indicated on the iGEM registry their assembly information pages.

XlsxWriter 1.3.7

This package is used to write the output from the queries to the Wikibase instance to the excel file.

SPARQLWrapper 1.8.5

This package is used as a python SPARQL wrapper when querying external databases, such as the UniProt database to retrieve additional information on the biological part.

Beautiful Soup 4.9.1

This package is used to parse XML and HTML formatted data. It is used both for parsing of the point in time datadump file and for the webpages used for storage of nucleotide sequences of the biological parts.

The DIAMOND protein aligner

This external programme is used as a Basic Local Alignment Search Tool (BLAST). To install, follow the start guide on their website to set up DIAMOND.

Creation of the Wikibase instance

Description of MAIN.py

The MAIN.py script orchestrates the entirety of the parsing, blasting, and uploading. It contains functions that call imported functions from both the other .py files and the modules. The gist of the pipeline is based upon dictionaries that are transferred and pickle files used to store the objects. The original XML datadump file, containing all the items to be uploaded and related statements is first parsed by the BB_parser() function, which uses functions provided in BB_parser_functions.py file. Two outputs are created (*Figure 1*). A Temp_pickle file for each of the items in a Temp_directory, later to be used again, and a Fasta file (.fna) for the DIAMOND BLAST, external programme. This performs a BLAST and has a Blasted file (.XML) as output. The Temp_pickle files are iterated and matched with the corresponding BLAST hit in the WDI_dict_blast_add(), again using functions from BB_parser_functions.py (*Figure 1*). This is also where additional information is retrieved from an external source (UniProt in this example). These dictionaries are then stored as Final_pickle files in the Final_directory. Lastly, this directory is iterated, and using functions from WDI_writer.py the objects are uploaded to the Wikibase instance.

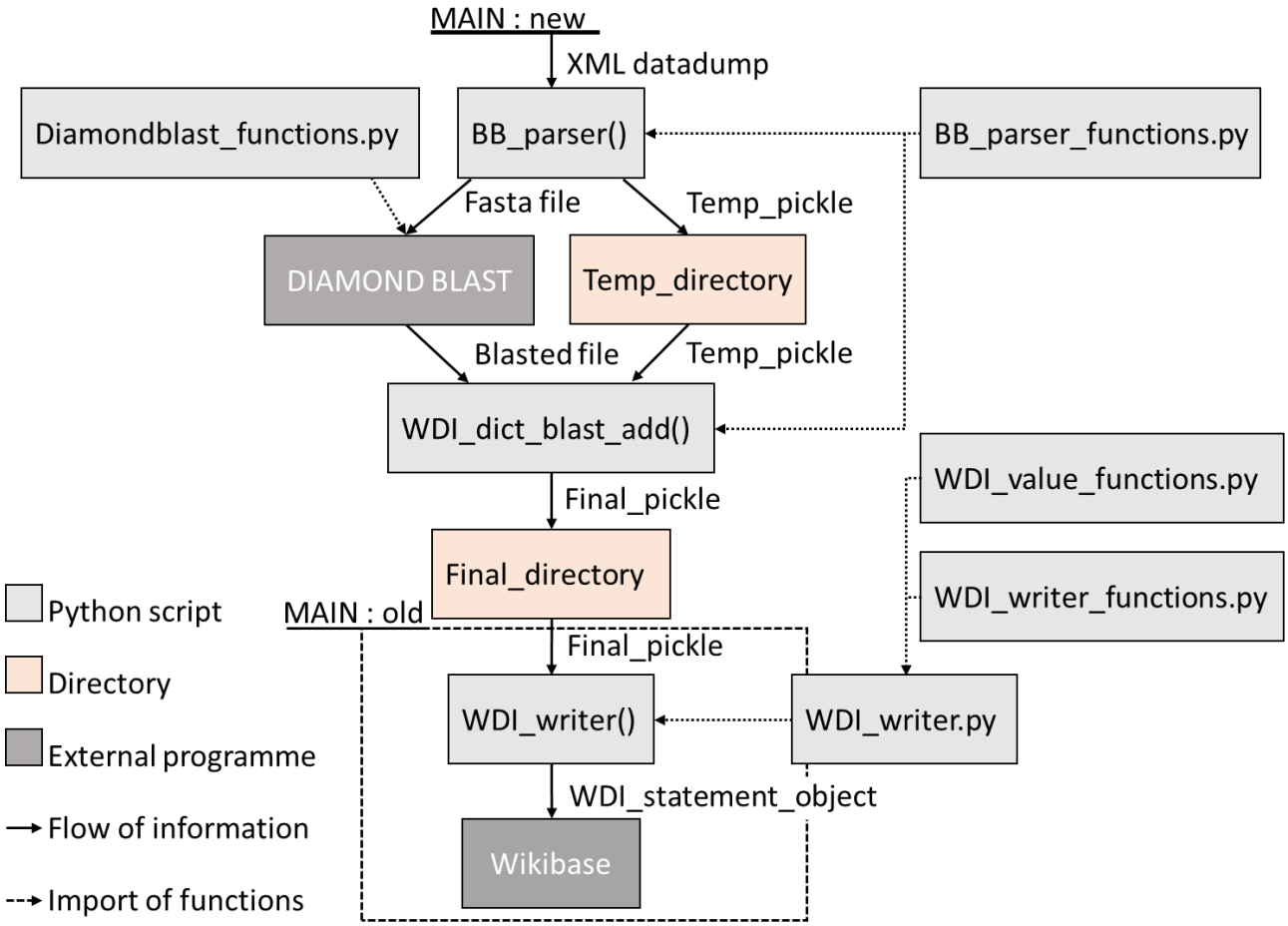

Figure 1: Pseudocode of MAIN.py *including external programmes and called functions from the scripts in GIT. The external packages that are used are not visible here.*

Use of MAIN.py

The MAIN.py file has two methods of usage. In addition to a Wikibase username and password, a “new”/”old” parameter should also be provided. As can be seen in *Figure 1*. The “new” parameter indicates that all of the functions and the BLAST are performed. The “old” parameter indicates that only the WDI_writer() is performed. This can be used when small changes have to be made to the WDI_writer.py or whenever the output and rerunning of previous steps are redundant.

Before the script can be run the following paths have to be provided

```
input_path      = # path to XML file
blasted_file    = # path to blasted file location
(standard is '../Parts/Db_output.xml')
fasta_loc       = # path to fasta file location
(standard is '../Parts/fastafile.fna')
T_directory     = # path to temporary pickle file directory
(standard is '../Parts/Temp_pickle/')
F_directory     = # path to final pickle file directory
(standard is '../Parts/Temp_pickle/')
database        = # path to database .dmnd filer
```

Additionally, the following URLs have to be provided

For the Wikibase

```
endpoint_url        = # Wikibase SPARQL endpoint
mediawiki_api_url   = # Wikibase API
```

For UniProt

```
Sparql_endpoint     = # UniProt SPARQL endpoint
(standard is 'http://purl.uniprot.org/uniprot/')
```

For iGEM

```
iGEM_sequence_url = # iGEM HTML sequence
(standard is
'http://parts.iGEM.org/cgi/partsdb/composite_edit/putseq.cgi?part=
')
```

To use the script and when new final pickle files have to be created, run the following code. The "new" parameter indicates that the parsing and aligning of biological parts has not been done prior and therefore no dictionary objects are present (or they are to be replaced). The username and password parameters are to authenticate the bot and make sure that changes can be made to the Wikibase instance. They can be created on the BioParts page.

```
python3 MAIN.py new username password
```

On the contrary, if final pickle files can be used again, run the following code. The "old" parameter indicates that the previous pickled dictionary objects can be used. This has decreased runtime and can be used when changed to WDI_writer.py is updated.

```
python3 MAIN.py old username password
```

Use of Add_assembly.py

The Add_assembly.py file is used to create the linkage between BioParts item pages. The script adds the deep_u_list assembly to the Wikibase for each of the pickled dictionaries in the final directory. It uses the "contains" property on the Wikibase to connect the files. Running Add_assembly.py is always done after pickled dictionary objects have been created. The script adds links between biobrick item pages via the "contains" statement. Recommended to run after the previous script, but running in tandem is also possible. Again the username and password have to be provided as arguments.

```
python3 Add_assembly.py username password
```

Datamodel created by WDI_writer.py

The current WDI_writer version creates the following data model. The WikidataIntegrator package provides documentation regarding the syntax used. A visual representation can be seen in *Figure 2.* Further versions might change or add to this model.

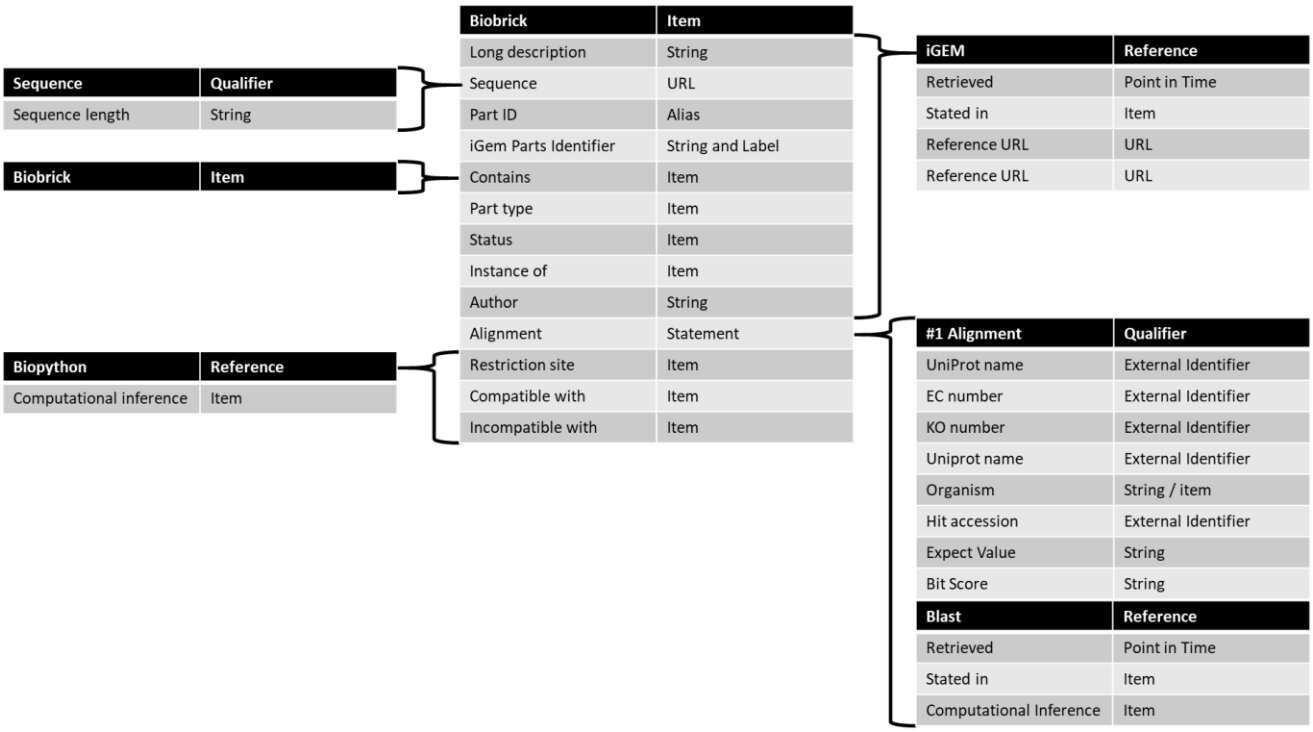

Figure 2: Datamodel as created by the WDI_writer() using WDI_writer.py functions*. To change this use the WikidataIntegrator syntax and change the statements in WDI_writer.py. This model is to be expanded upon and can also be used as guidelines when drafting queries for the Wikibase.*

Entity schema created by WDI_writer.py

The following provides insight into the current version of the WDI_writer.py script. It shows how the current statements are outputted from this script. The first three lines indicate the primary statements that are present on any item page and do not have property IDs. The following tables indicate the statements as written and outputted from WDI_writer.py using the dictionary stored in the pickle files as input. It is suggested to use the following tables when writing a personalised query.

Label: string: {part name: string}

Description: string {short desc: string}

Alias: strings {part id: string, nickname: string}

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**iGem Parts ID**	P38	String	-	Part name	String
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Long description**	P9	String	-	Description	String
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Sequence**	**P23**	URL	-	Biobrick sequence page URL	-
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Author**	P12	String	-	Authors	List(string)
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Restriction site**	P22	Item(s)		RS_dict	{string : list[string]}
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-
Reference 3	Computational inference	P29	Item	Q27	Biopython	-
Qualifier i	At site	P31	String	-	Location of restriction site	{string : list[string]}

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Compatible with**	P26	Item(s)	[Q19-Q24]	Assembly standard	list[string]
Reference 1	Computational inference	P29	Item	Q27	Biopython	-
Qualifier 1	Retrieved	P2	Datetime	-	Datetime	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Incompatible with**	P28	Item(s)	[Q19-Q24]	Assembly standard	list[string]
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-

	Property	#	Data Type	#	Value / label	Data type in script
Statement	**Part type**	**P27**	Item	-	Part type	String
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Alignment**	P47	String	-	UniProt dict [Hit_nmr]	Dict{string : string}
Reference 1	Stated in	P1	Item	Q3	TrEMBL	-
Reference 2	Computational inference	P29	Item	Q26	The DIAMOND protein aligner	-
Qualifier 1	UniProt name	P17	External Identifier	-	UniProt name	Dict{string : string}
Qualifier 2	EC number	P10	External Identifier	-	Enzyme Commission number	Dict{string : string}
Qualifier 3	KO number	P45	External Identifier	-	KEGG Ontology number	Dict{string : string}
Qualifier 4	Organism	P40	string	-	Organism	Dict{string : string}
Qualifier 5	UniProt protein ID	P16	External Identifier	-	Hit_accession	Dict{string : string}
Qualifier 6	Expect Value	P18	String	-	Hsp_evalue	Dict{string : string}
Qualifier 7	Bit Score	P42	String	-	Hsp_bit-score	Dict{string : string}

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Status**	**P44**	item	-	Part ID	String
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-
Qualifier 1	Retrieved	P2	Datetime	-	Datetime	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Part ID**	**P11**	External ID	-	Part ID	String
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-
Qualifier 1	Retrieved	P2	Datetime	-	Datetime	-

	Property	#	Data Type	#	Value / label	Data type in the script
Statement	**Instance of**	P20	Item	Q6	Biobrick	-
Reference 1	Stated in	P1	Item	Q5	iGEM Repository of Standard Biological Parts	-
Reference 2	Reference URL	P3	URL	-	Biobrick page URL	-

Query the Wikibase

Query_example.py

The Wikibase uses a triple formatted query. At present only the query method used by WikidataIntegrator can be used to read from the Wikibase and no other SPARQL wrappers. Additionally, the prefixes need to be set manually. For the BioParts Wikibase, the following prefixes have to be set.

For this specific Wikibase

```
PREFIX wd: <http://bioparts.wiki.opencura.com/entity/>
PREFIX wdt: <http://bioparts.wiki.opencura.com/prop/direct/>
PREFIX p: <http://bioparts.wiki.opencura.com/prop/>
PREFIX ps: <http://bioparts.wiki.opencura.com/prop/statement/>
PREFIX pq: <http://bioparts.wiki.opencura.com/prop/qualifier/>
```

Standard unchanged prefixes

```
PREFIX wikibase: <http://wikiba.se/ontology#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX bd: <http://www.bigdata.com/rdf#>
```

A specific example of a query incorporated in a python script can be found under ./manual/query_example. Which performs a SPARQL query to the BioParts Wikibase using the wrapper provided by WikidataIntegrator. It searches for an item given an Enzyme Commission number (EC number) and can be used as guidelines to construct a personalised query script. The current query example uses the xlsxwriter package to create a .xlsx (excel) file as output. This can easily be changed if another output format is required. Before using the following has to be provided.

```
endpoint_url = # URL to Wikibase SPARQL endpoint
Out_name = # Output file name
```

Query_example output

The Query_example script is used to query through the Wikibase instance. In this particular example, a nested dictionary containing EC numbers is used as input to find all biobricks that are suggested to catalyse the reactions. It searches through the Wikibase instance using a query as previously explained, including the prefixes, to find all relevant biobricks. As can be seen in *Figure 3* the script extracts the iGEM parts ID (?ID), the item page link (?item), the part type (?type), and lastly the URL to the sequence (?Seq). Using the function get_sequence(), the actual nucleotide sequence is then extracted from the registry. This is all checked for duplicates and lastly written to an excel workbook, using the xlsxwriter module for Python. This gives an output that aligned 4 out of 8 EC numbers. Whenever an EC number has been aligned multiple hits can be found. For example EC: '1.11.1.7' gave *Figure 4* as an output. As functionality proves, the script can now be altered or adjusted to fit the needs of the user.

```
SELECT ?ID ?item ?type ?Seq
Where {
    ?item p:P47 ?alignment.
    ?alignment pq:P10 \"""+ ID +"""\".
    ?item wdt:P23 ?Seq.
    ?item wdt:P27 ?x.
    ?x rdfs:label ?type.
    ?item wdt:P38 ?ID.
} LIMIT 100
```

***Figure 3: The code shows the query used to find the iGEM parts ID, item page, part type, and Sequence URL**. The script extracts the iGEM parts ID (?ID), the item page link (?item), the part type (?type), and lastly the URL to the sequence (?Seq)*

ID	item	type	seq
BBa_K2809031	http://bioparts.wiki.opencura.com/entity/Q1877	composite	cga…
BBa_K1997011	http://bioparts.wiki.opencura.com/entity/Q3906	composite	caa…
BBa_K1997012	http://bioparts.wiki.opencura.com/entity/Q18222	composite	caa…
BBa_K3105671	http://bioparts.wiki.opencura.com/entity/Q19431	composite	atg…
BBa_K1291071	http://bioparts.wiki.opencura.com/entity/Q23501	composite	ttg…
BBa_K3105677	http://bioparts.wiki.opencura.com/entity/Q30604	composite	gat…
BBa_K2696003	http://bioparts.wiki.opencura.com/entity/Q34599	composite	cgg…
BBa_K2881006	http://bioparts.wiki.opencura.com/entity/Q2311	coding	atg…
BBa_K2696001	http://bioparts.wiki.opencura.com/entity/Q12350	coding	cag…
BBa_K1997000	http://bioparts.wiki.opencura.com/entity/Q14176	coding	aaa…
BBa_K2696006	http://bioparts.wiki.opencura.com/entity/Q23161	coding	acg…
BBa_K2354012	http://bioparts.wiki.opencura.com/entity/Q28049	coding	caa…
BBa_K2354000	http://bioparts.wiki.opencura.com/entity/Q33799	coding	atg…
BBa_K3105668	http://bioparts.wiki.opencura.com/entity/Q34813	coding	atg…
BBa_K2696009	http://bioparts.wiki.opencura.com/entity/Q35804	coding	atg…
BBa_K2809030	http://bioparts.wiki.opencura.com/entity/Q2860	protein_domain	caa…

***Figure 4: The table shows the output for query EC number 1.11.1.7**. As can be seen the Sequence, although found of all biobricks, has been omitted to save space. The excel document contains the EC number as sheet name and the retrieved information as columns and rows.*