# pyGtoP Documentation

***Release 2.1.1***

**Sam Ireland**

**Aug 23, 2017**

# Contents

pyGtoP is a Python wrapper for the IUPHAR/BPS Guide to PHARMACOLOGY API. It provides a Python interface for access to the GtoP database.

# CHAPTER 1

## Example

```
>>> import pygtop
>>> my_drug = pygtop.get_ligand_by_id(5239)
>>> my_drug.name()
'paracetamol'
>>> my_drug.ligand_type()
'Synthetic organic'
```

See the *overview page* for more examples, or the *full documentation* for a full listing of features.

CHAPTER 2

Table of Contents

# Installing

## pip

pyGtoP can be installed using pip:

```
$ pip3 install pygtop
```

pyGtoP is written for Python 3, and as of version 1.0.0 is incompatible with Python 2. Versions 0.4.1 and earlier will continue to support Python 2.

## Requirements

PyGtoP requires the Python librares requests and molecuPy. These will be installed automatically if pyGtoP is installed with pip.

Otherwise pyGtoP has no external dependencies, and is pure Python.

# About the Guide to PHARMACOLOGY

The Guide to PHARMACOLOGY (GtoP), originally a collaboration between The British Pharmacological Society (BPS) and the International Union of Basic and Clinical Pharmacology (IUPHAR), acts as a "one-stop shop" portal to pharmacological information. One of its main aims is to provide a searchable database with quantitative information on drug targets and the prescription medicines and experimental drugs that act on them.

For more information, see the website itself. Information on the GtoP web services which pyGtoP accesses can be found here.

# Overview

## Ligands

The simplest way to create a ligand is via its GtoP ID:

```
>>> import pygtop
>>> my_drug = pygtop.get_ligand_by_id(5239)
>>> my_drug.name()
'paracetamol'
>>> my_drug.ligand_type()
'Synthetic organic'
```

Unlike previous versions of pyGtoP, all ligand (and target and interaction) properties can be accessed without requesting them separately:

```
>>> my_drug.rotatable_bonds()
2
>>> my_drug.molecular_weight()
151.0633286
>>> my_drug.smiles()
'CC(=O)Nc1ccc(cc1)O'
```

Some properties, such as name and synonyms, contain HTML entities. To get these without these often unceccessary additions, the `strip_html` argument can be used:

```
>>> my_drug = pygtop.get_ligand_by_id(2424)
>>> my_drug.name()
'&Delta;<sup>9</sup>-tetrahydrocannabinol'
>>> my_drug.name(strip_html=True)
'Δ9-tetrahydrocannabinol'
>>> my_drug.synonyms()
['Abbott 40566', 'delta9-THC', '&Delta;<sup>9</sup>-THC', 'Marinol&reg;', 't
etrahydrocannabinol']
>>> my_drug.synonyms(strip_html=True)
['Abbott 40566', 'delta9-THC', 'Δ9-THC', 'Marinol®', 'tetrahydrocannabinol']
```

Ligands can also be accessed by name:

```
>>> pygtop.get_ligand_by_name('caffeine')
<Ligand 407 (caffeine)>
```

You can get a list of ligands by either requesting all ligands, or providing a query:

```
>>> all_ligands = pygtop.get_all_ligands()
>>> len(all_ligands) # There are 8,400 ligands as of July 2016
8400
>>> all_ligands[0]
<'10,10-difluoro TXA<sub>2</sub>' Ligand (Synthetic organic)>
>>> query = {"type": "Approved", "molWeightGt": 50, "molWeightLt": 200}
>>> ligands = pygtop.get_ligands_by(query) # Get approved ligands between 50 and 200␣
↪Da
>>> len(ligands)
106
```

## Targets

The API for targets works in much the same way as for ligands:

```
>>> import pygtop
>>> my_target = pygtop.get_target_by_id(297)
>>> my_target.name()
'motilin receptor'
>>> my_target.target_type()
'GPCR'
>>> pygtop.get_target_by_name('CYP3A4')
<Target 1337 (CYP3A4)>
```

```
>>> all_targets = pygtop.get_all_targets()
>>> len(all_targets) # There are 2,866 ligands as of July 2016
2866
>>> all_targets[-1]
<Target 2893 (Branched chain amino acid transaminase 2)>
>>> query = {"type": "NHR"}
>>> targets = pygtop.get_targets_by(query) # Get all NHR targets
>>> len(targets)
49
```

There is a class representing target families, which are arranged hierarchically:

```
>>> my_target.families()
[<'Motilin receptor' TargetFamily>]
>>> my_target.families()[0].parent_families()
[<'G protein-coupled receptors' TargetFamily>]
>>> len(my_target.families()[0].parent_families()[0].sub_families())
69
```

Because so many properties of targets are specific to species variants, many properties have a `species` argument for only returning relevant results:

```
>>> my_target = pygtop.get_target_by_id(300)
>>> my_target.database_links()
[<ChEMBL Target link (102733) for Human>, <Ensembl Gene link (ENSMUSG0000002
0090) for Mouse>, <Ensembl Gene link (ENSRNOG00000000559) for Rat>, <Ensembl
 Gene link (ENSG00000148734) for Human>, <Entrez Gene link (237362) for Mous
e>, <Entrez Gene link (64106) for Human>, <Entrez Gene link (64107) for Rat>
, <GPCRDB link (Q9EP86) for Rat>, <GPCRDB link (Q9GZQ6) for Human>, <HomoloG
ene link (23348) for Human>, <Human Protein Reference Database link (12120)
for Human>, <OMIM link (607448) for Human>, <PharmGKB Gene link (PA134934991
) for Human>, <PhosphoSitePlus link (Q9GZQ6) for Human>, <PhosphoSitePlus li
nk (Q9EP86) for Rat>, <PhosphoSitePlus link (E9Q468) for Mouse>, <Protein GI
 link (11545887) for Human>, <Protein GI link (294661833) for Mouse>, <Prote
in GI link (294661831) for Rat>, <Protein Ontology (PRO) link (PRO:000001620
) for Human>, <RefSeq Nucleotide link (NM_022291) for Rat>, <RefSeq Nucleoti
de link (NM_022146) for Human>, <RefSeq Nucleotide link (NM_001177511) for M
ouse>, <RefSeq Protein link (NP_071627) for Rat>, <RefSeq Protein link (NP_0
71429) for Human>, <RefSeq Protein link (NP_001170982) for Mouse>, <UniGene
Hs. link (302026) for Human>, <UniProtKB link (Q9GZQ6) for Human>, <UniProtK
B link (Q9EP86) for Rat>, <UniProtKB ID/Entry name link (NPFF1_HUMAN) for Hu
man>, <UniProtKB ID/Entry name link (NPFF1_RAT) for Rat>]
>>> my_target.database_links(species="rat")
[<Ensembl Gene link (ENSRNOG00000000559) for Rat>, <Entrez Gene link (64107)
 for Rat>, <GPCRDB link (Q9EP86) for Rat>, <PhosphoSitePlus link (Q9EP86) fo
```

```
r Rat>, <Protein GI link (294661831) for Rat>, <RefSeq Nucleotide link (NM_0
22291) for Rat>, <RefSeq Protein link (NP_071627) for Rat>, <UniProtKB link
(Q9EP86) for Rat>, <UniProtKB ID/Entry name link (NPFF1_RAT) for Rat>]
```

## Interactions

The interactions of a ligand can be accessed as follows:

```
>>> import pygtop
>>> ligand = pygtop.get_ligand_by_id(5239)
>>> ligand.interactions()
[<Interaction (5239 --> Human 1375)>, <Interaction (5239 --> Human 1376)>]
```

Alternatively you can request the interacting targets instead:

```
>>> ligand.targets()
[<Target 1375 (COX-1 )>, <Target 1376 (COX-2 )>]
```

Targets can access interactions in much the same way:

```
>>> target = pygtop.get_target_by_id(50)
>>> target.interactions()
[<Interaction (681 --> Human 50)>, <Interaction (682 --> Human 50)>, <Intera
ction (683 --> Human 50)>, <Interaction (684 --> Human 50)>, <Interaction (6
95 --> Mouse 50)>, <Interaction (695 --> Rat 50)>, <Interaction (696 --> Rat
 50)>, <Interaction (697 --> Mouse 50)>, <Interaction (697 --> Rat 50)>, <In
teraction (3768 --> Human 50)>, <Interaction (700 --> Human 50)>, <Interacti
on (701 --> Mouse 50)>, <Interaction (701 --> Rat 50)>, <Interaction (705 --o
> Mouse 50)>, <Interaction (705 --> Rat 50)>, <Interaction (706 --> Human 50
)>]
>>> target.interactions(species="rat")
[<Interaction (695 --> Rat 50)>, <Interaction (696 --> Rat 50)>, <Interactio
n (697 --> Rat 50)>, <Interaction (701 --> Rat 50)>, <Interaction (705 --> R
at 50)>]
>>> target.ligands()
[<Ligand 681 (&alpha;-CGRP)>, <Ligand 682 (&beta;-CGRP)>, <Ligand 683 (adren
omedullin)>, <Ligand 684 (adrenomedullin 2/intermedin)>, <Ligand 695 (&alpha
;-CGRP)>, <Ligand 695 (&alpha;-CGRP)>, <Ligand 696 (&beta;-CGRP)>, <Ligand 6
97 (adrenomedullin)>, <Ligand 697 (adrenomedullin)>, <Ligand 3768 ([<sup>125
</sup>I]AM (rat))>, <Ligand 700 (&alpha;-CGRP-(8-37) (human))>, <Ligand 701
(&alpha;-CGRP-(8-37) (rat))>, <Ligand 701 (&alpha;-CGRP-(8-37) (rat))>, <Lig
and 705 (AM-(20-50) (rat))>, <Ligand 705 (AM-(20-50) (rat))>, <Ligand 706 (A
M-(22-52) (human))>]
>>> target.ligands(species="rat")
[<Ligand 695 (&alpha;-CGRP)>, <Ligand 696 (&beta;-CGRP)>, <Ligand 697 (adren
omedullin)>, <Ligand 701 (&alpha;-CGRP-(8-37) (rat))>, <Ligand 705 (AM-(20-5
0) (rat))>]
```

The interaction objects themselves have methods for returning the relevant ligand or target object:

```
>>> interaction = ligand.interactions()[0]
>>> interaction.ligand()
<Ligand 5239 (paracetamol)>
>>> interaction.target()
<Target 1375 (COX-1 )>
```

### Structural Data

The Guide to PHARMACOLOGY has PDB codes annotated on some ligands and targets. These can be accessed as follows:

```
>>> ligand = pygtop.get_ligand_by_id(149)
>>> ligand.gtop_pdbs()
['4IB4']
>>> target = pygtop.get_target_by_id(595)
>>> target.gtop_pdbs()
['1NYX']
```

In addition, ligands and targets can query the RSCB PDB Web Services to find other PDB codes:

```
>>> ligand.smiles_pdbs()
['4IAR', '4IB4', '4NC3']
>>> target.uniprot_pdbs()
['1FM6', '1FM9', '1I7I', '1K74', '1KNU', '1NYX', '1PRG', '1RDT', '1WM0', '1Z
EO', '1ZGY', '2ATH', '2F4B', '2FVJ', '2G0G', '2G0H', '2GTK', '2HFP', '2HWQ',
'2HWR', '2I4J', '2I4P', '2I4Z', '2OM9', '2P4Y', '2POB', '2PRG', '2Q59', '2Q5
9', '2Q5P', '2Q5S', '2Q61', '2Q6R', '2Q6S', '2Q8S', '2QMV', '2VSR', '2VST',
'2VV0', '2VV1', '2VV1', '2VV2', '2VV3', '2VV4', '2VV4', '2XKW', '2YFE', '2ZK
0', '2ZK1', '2ZK2', '2ZK3', '2ZK4', '2ZK5', '2ZK6', '2ZNO', '2ZVT', '3ADS',
'3ADT', '3ADU', '3ADV', '3ADW', '3ADX', '3AN3', '3AN4', '3B0Q', '3B0R', '3B1
M', '3B3K', '3BC5', '3CDP', '3CDS', '3CS8', '3CWD', '3D6D', '3DZU', '3DZY',
'3E00', '3ET0', '3ET3', '3FEJ', '3FUR', '3G9E', '3GBK', '3H0A', '3HO0', '3HO
D', '3IA6', '3K8S', '3KMG', '3LMP', '3NOA', '3OSI', '3OSW', '3PBA', '3PO9',
'3PRG', '3QT0', '3R5N', '3R8A', '3R8I', '3S9S', '3SZ1', '3T03', '3TY0', '3U9
Q', '3V9T', '3V9V', '3V9Y', '3VJH', '3VJI', '3VN2', '3VSO', '3VSP', '3WJ4',
'3WJ5', '3WMH', '3X1H', '3X1I', '4A4V', '4A4W', '4CI5', '4E4K', '4E4Q', '4EM
9', '4EMA', '4F9M', '4FGY', '4HEE', '4JAZ', '4JL4', '4L96', '4L98', '4O8F',
'4OJ4', '4PRG', '4PVU', '4PWL', '4R06', '4R2U', '4R6S', '4XLD', '4XTA', '4XU
M', '4Y29', '4YT1']
```

See the full documentation for a list of all the ways to search for PDB codes.

pyGtoP can now also use the molecuPy library to return PDBs as PDB objects. To do this, simply provide `molecupy=True` to any of the PDB requesting methods:

```
>>> ligand.smiles_pdbs(molecupy=True)
[<Pdb (4IAR)>, <Pdb (4IB4)>, <Pdb (4NC3)>]
```

See the molecuPy documentation for a full accounting of the functionality this offers. pyGtoP requires molecuPy 1.0.0 or higher.

## Full documentation

### `pygtop.ligands` (Ligands)

Contains ligand-specific objects and functions.

`pygtop.ligands.`**`get_ligand_by_id`**(*ligand_id*)

Returns a Ligand object of the ligand with the given ID.

**Parameters** **`ligand_id`** (*int*) – The GtoP ID of the Ligand desired.

**Return type** *Ligand*

**Raises** *NoSuchLigandError* if no such ligand exists in the database

pygtop.ligands.**get_all_ligands**()
Returns a list of all ligands in the Guide to PHARMACOLOGY database. This can take a few seconds.

**Returns** list of *Ligand* objects

pygtop.ligands.**get_ligands_by**(*criteria*)
Get all ligands which specify the criteria dictionary.

**Parameters** **criteria** (*dict*) – A dictionary of *field=value* pairs. See the GtoP ligand web services page for key/value pairs which can be supplied.

**Returns** list of *Ligand* objects.

pygtop.ligands.**get_ligand_by_name**(*name*)
Returns the ligand which matches the name given.

**Parameters** **name** (*str*) – The name of the ligand to search for. Note that synonyms will not be searched.

**Return type** *Ligand*

**Raises** *NoSuchLigandError* if no such ligand exists in the database.

pygtop.ligands.**get_ligands_by_smiles**(*smiles*, *search_type='exact'*, *cutoff=0.8*)
Search for ligands by SMILES string.

**Parameters**

- **smiles** (*str*) – The SMILES string to search with.
- **search_type** (*str*) – The type of search. Viable options are `"exact"`, `"substructure"` or `"similarity"`.
- **cutoff** (*float*) – If performing a similarity search, this is the cutoff used for similarity. The default is 0.8 and the maximum is 1.

**Returns** list of *Ligand* objects.

**class** pygtop.ligands.**Ligand**(*json_data*)
A Guide to PHARMACOLOGY ligand object.

**Parameters** **json_data** – A dictionary obtained from the web services.

**ligand_id**()
Returns the ligand's GtoP ID.

**Return type** int

**name**(**args*, *strip_html=False*, ***kwargs*)
Returns the ligand's name.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**abbreviation**(**args*, *strip_html=False*, ***kwargs*)
Returns the ligand's abbreviated name.

**Parameters** **strip_html** (*bool*) – If `True`, the abbreviation will have HTML entities stripped.

**Return type** str

**inn**(**args*, *strip_html=False*, ***kwargs*)
Returns the ligand's INN name.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**ligand_type**()
Returns the ligand's type.

**Return type** str

**species**()
Returns the ligand's species, where appropriate.

**Return type** str

**radioactive**()
Returns True if the ligand is radioactive.

**Return type** bool

**labelled**()
Returns True if the ligand is labelled.

**Return type** bool

**approved**()
Returns True if the ligand is approved.

**Return type** bool

**withdrawn**()
Returns True if the ligand has been withdrawn.

**Return type** bool

**approval_source**(**args*, *strip_html=False*, ***kwargs*)
Returns the regulatory body that approved the ligand, where appropriate.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**subunit_ids**()
Returns the the ligand IDs of all ligands which are subunits of this target.

**Returns** list of `int`

**subunits**()
Returns a list of all ligands which are subunits of this ligand.

**Returns** list of *Ligand* objects

**complex_ids**()
Returns the the ligand IDs of all ligands of which this target is a subunit.

**Returns** list of `int`

**complexes**()
Returns a list of all ligands of which this ligand is a subunit.

**Returns** list of *Ligand* objects

**prodrug_ids**()
Returns the the ligand IDs of all ligands which are prodrugs of this ligand.

**Returns** list of `int`

**prodrugs**()
Returns a list of all ligands which are prodrugs of this ligand.

**Returns** list of *Ligand* objects

**active_drug_ids**()
Returns the the ligand IDs of all ligands which are active equivalents of this ligand.

**Returns** list of int

**active_drugs**()
Returns a list of all ligands which are active equivalents of this ligand.

**Returns** list of *Ligand* objects

**iupac_name**()
Returns the ligand's IUPAC name.

**Return type** str

**smiles**()
Returns the ligand's SMILES string.

**Return type** str

**inchi**()
Returns the ligand's InChI string.

**Return type** str

**inchi_key**()
Returns the ligand's InChI key.

**Return type** str

**one_letter_sequence**()
Returns the ligand's single letter amino acid sequence where appropriate.

**Return type** str

**three_letter_sequence**()
Returns the ligand's three letter amino acid sequence where appropriate.

**Return type** str

**post_translational_modifications**()
Returns any post-translational modifications.

**Return type** str

**chemical_modifications**()
Returns any chemical modifications.

**Return type** str

**hydrogen_bond_acceptors**()
Returns the number of hydrogen bond accepting atoms.

**Return type** int

**hydrogen_bond_donors**()
Returns the number of hydrogen bond donor atoms.

**Return type** int

**rotatable_bonds**()
Returns the number of rotatable bonds in the ligand.

**Return type** int

**topological_polar_surface_area**()
Returns the polar surface area of the ligand in Angstroms.

**Return type** float

**molecular_weight**()
Returns the ligand's mass in Daltons.

**Return type** float

**log_p**()
Returns the logP value of the ligand.

**Return type** int

**lipinski_rules_broken**()
Returns the number of Lipinski's Rules the ligand breaks.

**Return type** int

**synonyms**(**args*, *strip_html=False*, ***kwargs*)
Returns the number ligand's synonyms

**Returns** list of `str`

**general_comments**()
Returns general comments pertaining to the ligand.

**Return type** str

**bioactivity_comments**()
Returns comments pertaining to bioactivity.

**Return type** str

**clinical_use_comments**()
Returns comments pertaining to clinical use.

**Return type** str

**mechanism_of_action_comments**()
Returns comments pertaining to mechanism.

**Return type** str

**absorption_and_distribution_comments**()
Returns comments pertaining to absorption and distribution.

**Return type** str

**metabolism_comments**()
Returns comments pertaining to metabolism.

**Return type** str

**elimination_comments**()
Returns comments pertaining to elimination from the body.

**Return type** str

**population_pharmacokinetics_comments**()
Returns comments pertaining to population pharmacokinetics.

**Return type** str

**organ_function_impairments_comments**()
Returns comments pertaining to organ function impairment.

**Return type** str

**mutations_and_pathophysiology_comments**()
Returns comments pertaining to mutations and pathophysiology.

**Return type** str

**database_links**()
Returns a list of database links for this ligand.

**Return type** list of *DatabaseLink*

**interactions**()
Returns a list of interactions for this ligand.

**Return type** list of *Interaction*

**get_interaction_by_id**(*interaction_id*)
Returns an Interaction object of a given ID belonging to the ligand.

**Parameters** **interaction_id** (*int*) – The interactions's ID.

**Return type** *Interaction*

**Raises** *NoSuchInteractionError*: if no such interaction exists in the database.

**targets**()
Returns a list of all targets which this ligand interacts with.

**Returns** list of *Target* objects

**gtop_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Returns a list of PDBs which the Guide to PHARMACOLOGY says contain this ligand.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**smiles_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the ligand's SMILES string.

**Parameters**

- **search_type** (*str*) – The type of search to run - whether exact matches only should be returned.
- **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**inchi_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the ligand's InChI string.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**name_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the ligand's name.

**Parameters**

- **comparator** (*str*) – The type of search to run - whether exact matches only should be returned, or substrings etc.

- **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**sequence_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the ligand's amino acid sequence, if that ligand is a peptide.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**het_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the ligand's amino acid sequence, if that ligand is a peptide.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**all_external_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database by all parameters.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**all_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Get a list of PDB codes using all means available - annotated and external.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**find_in_pdb_by_smiles**(*molecupy_pdb*)
Searches for the ligand in a molecuPy PDB object by SMILES string and returns the small molecule it finds.

**Parameters** **molecupy_pdb** – The molecuPy PDB object.

**Return type** `SmallMolecule`

**find_in_pdb_by_name**(*molecupy_pdb*)
Searches for the ligand in a molecuPy PDB object by ligand name and returns the small molecule it finds.

**Parameters** **molecupy_pdb** – The molecuPy PDB object.

**Return type** `SmallMolecule`

**find_in_pdb_by_mass**(*molecupy_pdb*)
Searches for the ligand in a molecuPy PDB object by ligand mass and returns the small molecule it finds.

**Parameters** **molecupy_pdb** – The molecuPy PDB object.

**Return type** `SmallMolecule`

**find_in_pdb_by_peptide_string**(*molecupy_pdb*)
Searches for the ligand in a molecuPy PDB object by peptide sequence and returns the chain it finds.

**Parameters** **molecupy_pdb** – The molecuPy PDB object.

**Return type** `Chain`

## pygtop.targets (Targets)

Contains target-specific objects and functions.

pygtop.targets.**get_target_by_id**(*target_id*)
Returns a Target object of the target with the given ID.

**Parameters** **target_id** (*int*) – The GtoP ID of the Target desired.

**Return type** *Target*

**Raises** *NoSuchTargetError*: if no such target exists in the database

pygtop.targets.**get_all_targets**()
Returns a list of all targets in the Guide to PHARMACOLOGY database. This can take a few seconds.

**Returns** list of *Target* objects

pygtop.targets.**get_targets_by**(*criteria*)
Get all targets which specify the criteria dictionary.

**Parameters** **criteria** (*dict*) – A dictionary of *field=value* pairs. See the GtoP target web services page for key/value pairs which can be supplied.

**Returns** list of *Target* objects.

pygtop.targets.**get_target_by_name**(*name*)
Returns the target which matches the name given.

**Parameters** **name** (*str*) – The name of the target to search for. Note that synonyms will not be searched.

**Return type** *Target*

**Raises** *NoSuchTargetError*: if no such target exists in the database.

pygtop.targets.**get_target_family_by_id**(*family_id*)
Returns a TargetFamily object of the family with the given ID.

**Parameters** **family_id** (*int*) – The GtoP ID of the TargetFamily desired.

**Return type** *TargetFamily*

**Raises** *NoSuchTargetFamilyError*: if no such family exists in the database

pygtop.targets.**get_all_target_families**()
Returns a list of all target families in the Guide to PHARMACOLOGY database.

**Returns** list of *TargetFamily* objects

**class** pygtop.targets.**Target**(*json_data*)
A Guide to PHARMACOLOGY target object.

**Parameters** **json_data** – A dictionary obtained from the web services.

**target_id**()
Returns the target's GtoP ID.

**Return type** int

**name**(**args*, *strip_html=False*, ***kwargs*)
Returns the target's name.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**abbreviation**(**args*, *strip_html=False*, ***kwargs*)
Returns the target's abbreviated name.

**Parameters** **strip_html** (*bool*) – If `True`, the abbreviation will have HTML entities stripped.

**Return type** str

**systematic_name**(**args*, *strip_html=False*, ***kwargs*)
Returns the target's systematic name.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**target_type**()
Returns the target's type.

**Return type** str

**family_ids**()
Returns the the family IDs of any families this target is a member of.

**Returns** list of `int`

**families**()
Returns a list of all target families of which this target is a member.

**Returns** list of *TargetFamily* objects

**subunit_ids**()
Returns the the target IDs of all targets which are subunits of this target.

**Returns** list of `int`

**subunits**()
Returns a list of all targets which are subunits of this target.

**Returns** list of *Target* objects

**complex_ids**()
Returns the the target IDs of all targets of which this target is a subunit.

**Returns** list of `int`

**complexes**()
Returns a list of all targets of which this target is a subunit.

**Returns** list of *Target* objects

**synonyms**(**args*, *strip_html=False*, ***kwargs*)
Returns any synonyms for this target.

**Parameters** **strip_html** (*bool*) – If `True`, the synonyms will have HTML entities stripped.

**Returns** list of str

**database_links**(*species=None*)
Returns any database links for this target.

**Parameters** **species** (*str*) – If given, only links belonging to this species will be returned.

**Returns** list of *DatabaseLink* objects.

**genes**(*species=None*)
Returns any genes for this target.

**Parameters** **species** (*str*) – If given, only genes belonging to this species will be returned.

**Returns** list of *Gene* objects.

**interactions**(*species=None*)
Returns any interactions for this target.

**Parameters** **species** (*str*) – If given, only interactions belonging to this species will be returned.

**Returns** list of *Interaction* objects.

**get_interaction_by_id**(*interaction_id*)
Returns an Interaction object of a given ID belonging to the target.

**Parameters** **interaction_id** (*int*) – The interactions's ID.

**Return type** *Interaction*

**Raises** *NoSuchInteractionError*: if no such interaction exists in the database.

**ligands**(*species=None*)
Returns any ligands that this target interacts with.

**Parameters** **species** (*str*) – If given, only ligands belonging to this species will be returned.

**Returns** list of *DatabaseLink* objects.

**gtop_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Returns a list of PDBs which the Guide to PHARMACOLOGY says contain this target.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**uniprot_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database with the targets's uniprot accessions.

**Parameters**

- **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.
- **species** (*str*) – If given, only PDBs belonging to this species will be returned.

**Returns** list of `str` PDB codes

**all_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Get a list of PDB codes using all means available - annotated and external.

**Parameters**

- **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.
- **species** (*str*) – If given, only PDBs belonging to this species will be returned.

**Returns** list of `str` PDB codes

**class** `pygtop.targets.`**TargetFamily**(*json_data*)
A Guide to PHARMACOLOGY target family object.

**Parameters** **json_data** – A dictionary obtained from the web services.

**family_id**()
Returns the family's GtoP ID.

**Return type** int

**name**(**args*, *strip_html=False*, ***kwargs*)
Returns the family's name.

**Parameters** **strip_html** (*bool*) – If `True`, the name will have HTML entities stripped.

**Return type** str

**target_ids**()
Returns the the target IDs of all targets in this family. Note that only immediate children are shown - if a family has subfamilies then it will not return any targets here - you must look in the sub-families.

**Returns** list of int

**targets**()
Returns a list of all targets in this family. Note that only immediate children are shown - if a family has subfamilies then it will not return any targets here - you must look in the sub-families.

**Returns** list of *Target* objects

**parent_family_ids**()
Returns the the target IDs of all target families of which this family is a member.

**Returns** list of int

**parent_families**()
Returns a list of all target families of which this family is a member.

**Returns** list of *TargetFamily* objects

**sub_family_ids**()
Returns the the target IDs of all arget families which are a member of this family.

**Returns** list of int

**sub_families**()
Returns a list of all target families which are a member of this family.

**Returns** list of *TargetFamily* objects

## pygtop.interactions (Interactions)

pygtop.interactions.**get_all_interactions**()
Returns a list of all interactions in the Guide to PHARMACOLOGY database. This can take a few seconds.

**Returns** list of *Interaction* objects

**class** pygtop.interactions.**Interaction**(*json_data*)
A Guide to PHARMACOLOGY interaction object.

**Parameters** **json_data** – A dictionary obtained from the web services.

**interaction_id**()
Returns the interaction's GtoP ID.

**Return type** int

**ligand_id**()
Returns the GtoP ID of the associated ligand.

**Return type** int

**ligand**()
Returns the Ligand object for this interaction.

**Return type** *Ligand*

**target_id**()
Returns the GtoP ID of the associated target.

**Return type** int

**target**()
Returns the Target object for this interaction.

**Return type** *Target*

**gtop_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Returns a list of PDBs which the Guide to PHARMACOLOGY says contain this interaction.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**all_external_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Queries the RSCB PDB database for PDBs containing this interaction by all parameters.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**all_pdbs**(**args*, *as_molecupy=False*, ***kwargs*)
Get a list of PDB codes containing this interaction using all means available - annotated and external.

**Parameters** **as_molecupy** (*bool*) – Returns the PDBs as molecuPy PDB objects.

**Returns** list of `str` PDB codes

**species**()
Returns the species in which the interaction takes place.

**Return type** str

**primary_target**()
Returns `True` if the the interaction represents a ligand interacting with its primary target.

**Return type** bool

**endogenous**()
Returns `True` if the the interaction is an endogenous interaction.

**Return type** bool

**interaction_type**()
Returns the type of interaction.

**Return type** str

**action**()
Returns the action of the interaction.

**Return type** str

**affinity_low**()
Returns the lowest reported affinity for this interaction.

**Return type** float

**affinity_high**()
Returns the highest reported affinity for this interaction.

**Return type** float

**affinity_type**()
Returns the units of the interaction.

**Return type** str

## pygtop.gtop (GtoP Interface)

Functions for interacting with the Guide to PHARMACOLOGY web services.

pygtop.gtop.**get_json_from_gtop**(*query*, *attempts=5*)
Issues a query to the GtoP web services, and returns the resulting JSON.

If it does not get a valid response, it will try again, and if it still doesn't get JSON back, it will return None.

**Parameters**

- **query** (*str*) – The query to append to the base URL.
- **attempts** (*int*) – The number of attempts to make before giving up (default is 5).

**Returns** JSON object or None

## pygtop.pdb (RSCB PDB Interface)

Functions for interacting with the RSCB PDB web services.

pygtop.pdb.**query_rcsb**(*query_type*, *criteria*)
Queries the RSCB PDB web services with a simple GET request.

**Parameters**

- **query_type** (*str*) – The type of query to make
- **criteria** (*str*) – The criteria for this query

**Return type** ElementTree XML element

pygtop.pdb.**query_rcsb_advanced**(*query_type*, *criteria*)
Queries the RSCB PDB web services as an advanced search with a POST request.

**Parameters**

- **query_type** (*str*) – The type of query to make
- **criteria** (*str*) – The criteria for this query

**Returns** list of str PDB codes

pygtop.pdb.**ask_about_molecupy**(*func*)
A decorator which, when applied to a function, will add a 'as_molecupy' keyword argument - if set to True this will convert any PDB codes the function returns to molecuPy PDB objects.

## pygtop.shared (Shared objects)

Objects not specific to ligands or targets.

**class** pygtop.shared.**DatabaseLink**(*json_data*)
A link to an external database, containing accession and species information.

**Parameters** **json_data** – A dictionary obtained from the web services.

**accession**()
The Accession code.

**database**()
The Database being linked to.

**url**()
The URL for this database entry.

**species**()
The specific species the database entry refers to.

**class** pygtop.shared.**Gene**(*json_data*)
A gene for a pyGtoP target.

**Parameters** **json_data** – A dictionary obtained from the web services.

**target_id**()
The ID of the pyGtoP target derived from this gene.

**species**()
The species the gene is from.

**gene_symbol**()
The gene's code.

**gene_name**()
The gene's name.

**official_gene_id**()
The gene's official ID.

**genomic_location**()
The gene's location in its genome.

**amino_acids**()
The number of amino acids the gene codes for.

**transmembrane_domains**()
The number of amino acids in the resultant protein.

**pore_loops**()
The number of pore loops in the resultant protein.

pygtop.shared.**strip_html**(*func*)
A decorator which, when applied to a function, will add a 'strip_html' keyword argument - if set to True this will strip any HTML from the function's output.

## pygtop.exceptions (Exceptions)

pyGtoP custom exceptions.

**exception** pygtop.exceptions.**NoSuchLigandError**
The exception raised if a specific ligand is requested which does not exist.

**exception** pygtop.exceptions.**NoSuchTargetError**
The exception raised if a specific target is requested which does not exist.

**exception** pygtop.exceptions.**NoSuchTargetFamilyError**
The exception raised if a specific target family is requested which does not exist.

**exception** pygtop.exceptions.**NoSuchInteractionError**
The exception raised if a specific interaction is requested which does not exist.

**exception** pygtop.exceptions.**NoSuchTypeError**
The exception raised if a random ligand or target is requested of a type which does not exist.

# Changelog

## Release 2.1.0

*23 August 2017*

- Added search by HET method for ligand PDBs

## Release 2.0.1

*4 August 2016*

- pyGtoP now compatible for molecuPy 1.0.0 and higher.
- DatabaseLink and Gene objects now have method properties.

## Release 2.0.0

*9 July 2016*

- Most properties now accessible as methods.
  - Affects ligands, targets and interactions.
  - This removes the need to request these properties separately.
- Gene object added.
- Added ability to strip HTML from certain string outputs, such as name.
- Added extra safeguards to GtoP web services requester to make more stable.
- Changed handling of affinity values in Interactions.
  - Now provides a low and a high value as numbers.
  - String range accessible via the original JSON object.
- This release is backwards-incompatible with the 1.x releases.

## Release 1.0.1

*19 May 2016*

- Version number fix.

## Release 1.0.0

*18 May 2016*

- molecuPy integration
  - PDB retrieval can now be by 4-char string, or molecuPy PDB object.
  - Ligands now have methods for locating themselves in a PDB file.

As molecuPy is a Python 3 package, this is the first version of pyGtoP to be incompatible with Python 2, hence the new major version number.

## Release 0.4.0

*22 April 2016*

- PDB functionality
  - Access GtoP PDB annotations for ligands, targets and interactions
  - Query RSCB PDB web services for PDBs of ligands, targets and interactions
- Can now search for ligands by SMILES string

## Release 0.3.1

*31 March 2016*

- Bug fixes:
  - Interactions with median affinity values no longer throw errors
  - Interactions with string voltages no longer throw errors

## Release 0.3.0

*30 March 2016*

- Interaction functionality
  - Interaction objects now available, which can link to ligands and targets
  - Ligands can get their interactions, and by extension their targets
  - Targets can get their interactions, and by extension their ligands
- Other features
  - Python 2 json strings no longer throw errors if they contain special characters
  - All species names now lowercase, regardless of how they are stored in the database

## Release 0.2.0

*23 March 2016*

- Target functionality
  - Single target access (by ID, name, or at random)
  - Multiple target access (all, or by providing a query)
  - Target family manipulation
  - Target species-variant handling
- New Ligand features
  - Ligands now have methods for returning other ligands instead of lists of ligand IDs

## Release 0.1.1

*16 March 2016*

- Added Python 2 compatibility
- Bug fixes:
  - Ligand string repr no longer throws attribute error

## Release 0.1.0

*15 March 2016*

- Ligand functionality
  - Single ligand access (by ID, name, or at random)
  - Multiple ligand access (all, or by providing a query)

# Python Module Index

## p

# Index

## H



## I



## L



## M



## N



## O



## P



## Q



## R



## S

## T



## U



## W