# CGSB Cheat Sheet

## Basic syntax

CGSB can be used in a **Python script**:
Single command input:
```
CGSB.CGSB(command_line = "SET1:VAL1 SET2:VAL2")
```

Multiple command inputs:
```
CGSB.CGSB(command_line = ["SET1:VAL1", "SET2:VAL2"])
```
or
```
CGSB.CGSB(command_line1 = "SET1:VAL1", command_line2 = "SET2:VAL2")
```

or in the **terminal command line**:
```
python CGSB.py --command_line SET1:VAL1 SET2:VAL2
```
or
```
python CGSB.py --command_line SET1:VAL1 --comand_line SET2:VAL2
```

## Box commands

Box types:

| | |
|---|---|
| `box_type = "rectangular"` | min 3 dimensions (default). |
| `box_type = "hexagonal"` | min 2 dimensions. |
| `box_type = "dodecahedron"` | min 1 dimension. |
| `box_type = "optimal"` | same as `dodecahedron` |

Box dimensions:

| | |
|---|---|
| `box = [10, 10, 10]` | Box dimensions `x, y, z` (in nm). |
| `pbc = [10, 10, 10]` | Box dimensions `x, y, z` (in nm). |
| `x = 10, y = 10, z = 10` | Box dimensions `x, y, z` (in nm). |

`box_types = "rectangular"`,

| | |
|---|---|
| `box = [10]` | converted to `box = [10, 10, 10]` |
| `x = 10` | converted to `box = [10, 10, 10]` |
| `box = [10, 12]` | converted to `box = [10, 10, 12]` |
| `x = 10, y = 12` | converted to `box = [10, 10, 12]` |

## Environment variables

Output:

| | |
|---|---|
| `out_sys` | (str) Output .pdb and .gro file. |
| `out_sys_pdb` | (str) Output .pdb file. |
| `out_sys_gro` | (str) Output .gro file. |
| `out_top` | (str) Output .top file. |
| `out_log` | (str) Output .log file. |

Topology:

| | |
|---|---|
| `itp_input` | (str) Input .top file. |

Parameters:

| | |
|---|---|
| `sys_params` | (str) Parameter library for system. |
| `lipid_params` | (str) Parameter library for lipids. |
| `solv_params` | (str) Parameter library for solvents. |
| `prot_params` | (str) Parameter library for protein. |

Miscellaneous:

| | |
|---|---|
| `sn` | System name under `[ system ]`. |
| `backup` | (bool) Backup overwritten files (default: `True`). |
| `rand` | (int) Random seed. |

## Membrane building

Syntax:
```
membrane = "SET1:VAL1 SET2:VAL2"
```

Membrane types:

| | |
|---|---|
| `type:bilayer` | default. |
| `type:mono_upper` | upwards-facing monolayer. |
| `type:mono_lower` | downwards-facing monolayer. |

Lipid composition:

| | |
|---|---|
| `POPC` | lipid_type:1. |
| `POPE:5` | lipid_type:ratio. |
| `CHOL:3:dev18` | lipid_type:ratio:params. |

Area per lipid:

| | |
|---|---|
| `apl:0.6` | (float) default: 0.6 |

Leaflets:

| | |
|---|---|
| `leaflet:upper` | Upper leaflet subcommands follow. |
| `leaflet:lower` | Lower leaflet subcommands follow. |
| `leaflet:both` | Following subcommands apply to both. |

Placement and size:

| | |
|---|---|
| `x, y` | (float) x/y dimensions of the membrane |
| `center = [0, 0, 0]` | (list of floats) membrane position in xyz |
| `cx, xy, xz` | (float) membrane position in x, y, or z |

Lipid placement optimisation:

| | |
|---|---|
| `lipid_optim:avg_optimal` | default |
| `lipid_optim:abs_val` | Ratios as actual numbers. |
| `lipid_optim:fill` | apl- and ratio-based filling |
| `lipid_optim:force_fill` | apl-based filling |
| `lipid_optim:no` | no optimisation |

### Asymmetric bilayer

```
membrane = ["apl:0.5 leaflet:upper POPC:5 CHOL:1 leaflet:lower POPC:3 CHOL:2 leaflet:both params:Dev18"]
```

### Multiple bilayer

```
membrane = ["type:bilayer cz=5 POPC", "type:bilayer cz=-5 DOPC]
```

### Phase-separated membrane

```
membrane = ["POPC:5 CHOL:1 x:5 cx:2.5", "POPC:4 CHOL:2 x:5 cx:-2.5"]
```

## Protein insertion

Syntax:
```
protein = "prot_file:pdbfile SET2:VAL2"
```

Add topology:

| | |
|---|---|
| `mol_names:NAME` | `[ moleculetype ]` name |
| `mol_names:NAME1:NAME2` | Multiple `[ moleculetype ]` allowed. |

Placement:
`tx, ty, tz` (float) translation (in nm)
`rx, ry, rz` (float) rotation (in degrees)

Centering:

| | |
|---|---|
| `cen_method:cog` | Protein COG (default). |
| `cen_method:axis` | Axial mean coordinate. |
| `cen_method:res:5-7:10-20` | Residues 5-7 and 10-20. |
| `cen_method:point:0:0:0` | a coordinate point (0,0,0). |

## Solvation

Syntax:
```
solvation = "SET:TYPE:RATIO:PARAMS"
```
`SET: solv or pos or neg`. Only `TYPE` is mandatory.
Solvent composition:

| | |
|---|---|
| `solv:W` | `solv:solvent_type` (default: W). |
| `solv:W:DevWater5` | `solv:solvent_type:params`. |
| `solv:W:2` | `solv:solvent_type:ratio`. |
| `solv:W:2:DevWater5` | `solv:solvent_type:ratio:params`. |
| `pos:NA` | Positive ions (default: NA). |
| `neg:CL` | Negative ions (default: CL). |

Molarity:

| | |
|---|---|
| `solv_molarity` | (float) default: 55.56 mol/L. |
| `salt_molarity` | (float) default: 0.15 mol/L. |

Examples:
```
solvation = "solv:W pos:NA neg:CL"
solvation = "" # same as above
```

## Flooding commands

Syntax:
```
flooding = "NAME1:COUNT1 NAME2:COUNT2"
```
Example:
```
flooding = "RHO:30"
```
Solute topology:
```
solute_input = "solute_file:pdbfile mol:NAME:NRES:CHG"
```

| | |
|---|---|
| `mol` | Followed by NAME:NRES:CHARGE. |
| `names` | (string) Solute name. |
| `n_residues` | (int) No of residues per solute. |
| `nres` | (int) Same as `nres`. |
| `charges` | (int/float) of charge or (string) of `[ moleculetype ]` name. Default: 0. |

Example:
```
flooding = ["LIG1:50", "LIG2:10"],
solute_input = ["solute_file:ligands.pdb mol:LIG1:2:2 mol:LIG2:3:Ligand2"]
```
equivalent to
```
flooding = [ "LIG1:50", "LIG2:10"], solute_input = ["file:LIGANDS.pdb names:LIG1:LIG2 nres:2:3 charges:2:Ligand2"]
```

## References

http://github.com/MikkelDA/CGSB