1 Python

1.1 getting started with the code

in a python script:
Save your new python script in the folder `flow_polytopes/` at the top of it include the lines:

```
from graph_cal import *
from quiver_cal import *
```

Then use any of the functions listed below!

in a python shell:
navigate to the folder `flow_polytopes/`

```
>>> from graph_cal import *
>>> from quiver_cal import *
```

Building a first quiver:
In python, the quiver Q is represented using either a list of arrows $[(a_i, b_i)]$ for $a_i, b_i \in Q_0$, $i = 0, ..., |Q_1|$. or as a numpy matrix:

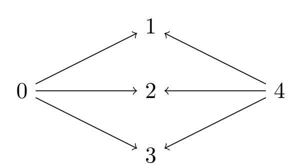
$$\left[a_{ij} = \begin{cases} 1, & \text{if head of arrow } j \text{ is vertex } i \\ -1, & \text{if tail of arrow } j \text{ is vertex } i \\ 0, & \text{otherwise} \end{cases} \right]$$

There is also funcionality to go between the two representations of a quiver, as shown below.
Example: The code to represent the quiver shown below is given in two ways as a sample:

`sampleScipt.py`:

```
from numpy import *
from graph_cal import *
from quiver_cal import *

Q_list = [(0,1),(0,2),(0,3),(4,1),(4,2),(4,3)]
Q_mat = matrix([
        [-1,-1,-1, 0, 0, 0],
        [ 1, 0, 0, 1, 0, 0],
        [ 0, 1, 0, 0, 1, 0],
        [ 0, 0, 1, 0, 0, 1],
        [ 0, 0, 0,-1,-1,-1]
        ])
```



1.2 available functions

To obtain all d-dimensional quivers:

```
Qs = all_possible_graphs(d)
```

Get the polytope associated to the quiver Q:

```
flow_polytope(Q)
```

Generate all the subquivers of Q

```
subquivers(Q)
```

Get all subsets of the vertices of Q that are closed under arrows:

```
subsets_closed(M)
```

Calculates weights of the vertices that are inherited from the weights on the arrows

```
theta(Q)
```

Is the subquiver $subQ$ stable?

```
is_stable(Q, subQ)
```