Our code receive configuration options from the *config.yaml* in the Inputs folder,

To run our code, run the environment.py code with the relevant *config.yaml* fields.

All queries and evidence of the program are given during run time.

Method for constructing the BN

- We created a network that contains a node dictionary. It maps a *node.name* to the *node* class itself for each node in the graph.
- Each *node* contains a *name, probs_table, parents, children* field
  The *name* identifies the node in the network
  The *probs_table* is the probability table for each node. It contains a probability value for each single conditional parent node.
  For example:

  ```
  P(mild|empty) = 0.1
  P(blocked|stormy) = 0.4
  P(Evacuees|Blockage 2) = 0.8
  ```

  Where in the weather probability: *empty* is like being non-conditional
  And in the *Evacuees* probability: We treat each other parent *blocking property* as *not blockage*. And now we can calculate all other probabilities in a 'noisy-or' manner.

  The *parents* are the nodes that this node is conditioned on

  The *children* are the nodes that conditioned on this node

Reasoning algorithm

- We used *enumeration_ask* and *enumerate_all* for our reasoning algorithm.

We ran 2 examples, one for input1.txt and another for input2.txt. the running results of the programs are attached in the zip folder.