Nama : Ahmad Irvandi

Npm : 2117051089

Kelas : B

Daftar aturan (rules)

No	Aturan (*Rule*)

1	*IF*Sakit kepala *is True*
	AND Keluar Cairan *is True*
	AND Ada tanda-tanda radang diliang telinga *is True*
	*THEN*Penyakit Otitis Eksterna

2	*IF* Demam *is True* *AND* Sakit kepala *is True* *AND* PUS dan di meatus media *is True* *AND* Hidung tersumbat *is True* *AND* Hidung meler *is True* *AND* Nyeri pipi dibawah mata *is True* *AND* Selaput lendir merah dan bengkak *is True* *Then* Sinusitis
3	*IF* Bersin-bersin *is True* *AND* Hidung meler *is True* *AND* Hidung tersumbat *is True* *AND* Lendir di tenggorokan *is True* *Then* Renitis Non – Alergika
4	*IF* Demam *is True* *AND* Sakit kepala *is True* *AND* Nyeri saat berbicara atau menelan *is True* *AND* Sakit pada telinga *is True* *AND* Pembengkakan kelenjar getah bening *is True* *AND* Tenggorokan gatal *is True* *AND* Adanya tonsil yang membengkak *is True* *AND* Suara serak *is True* *Then* Farangitis (Radang Tenggorokan)

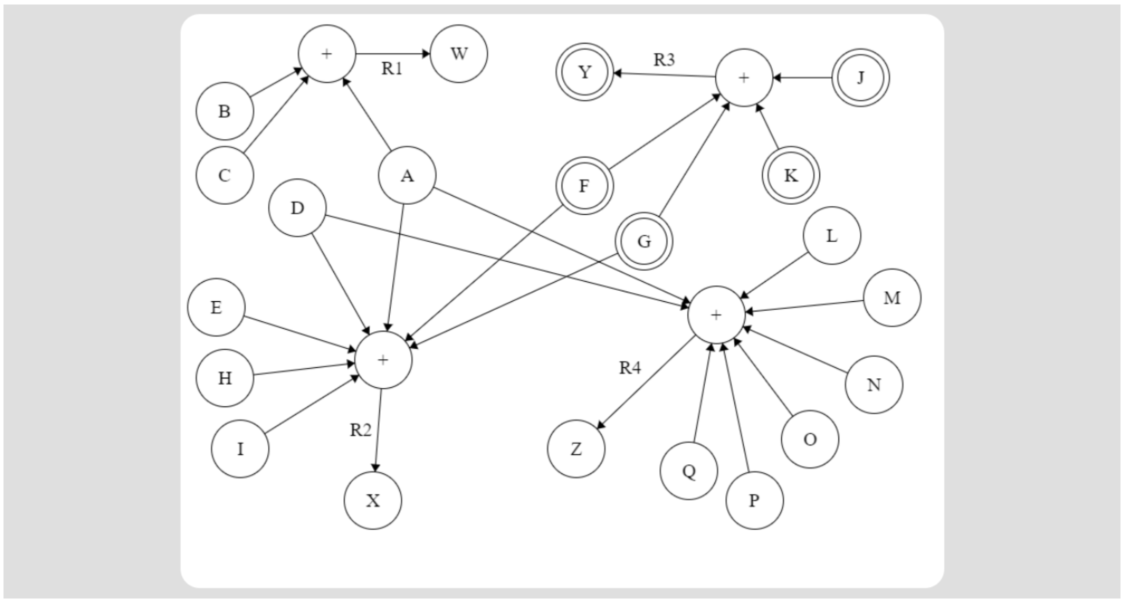
No	Aturan (rules)
R1	IF A & B & C THEN W
R2	IF D & A & E & F & G & H & I Then X
R3	IF J & G & F & K & Then Y
R4	IF D & A & L & M & N & O & P & Q Then Z

Fakta dan query :

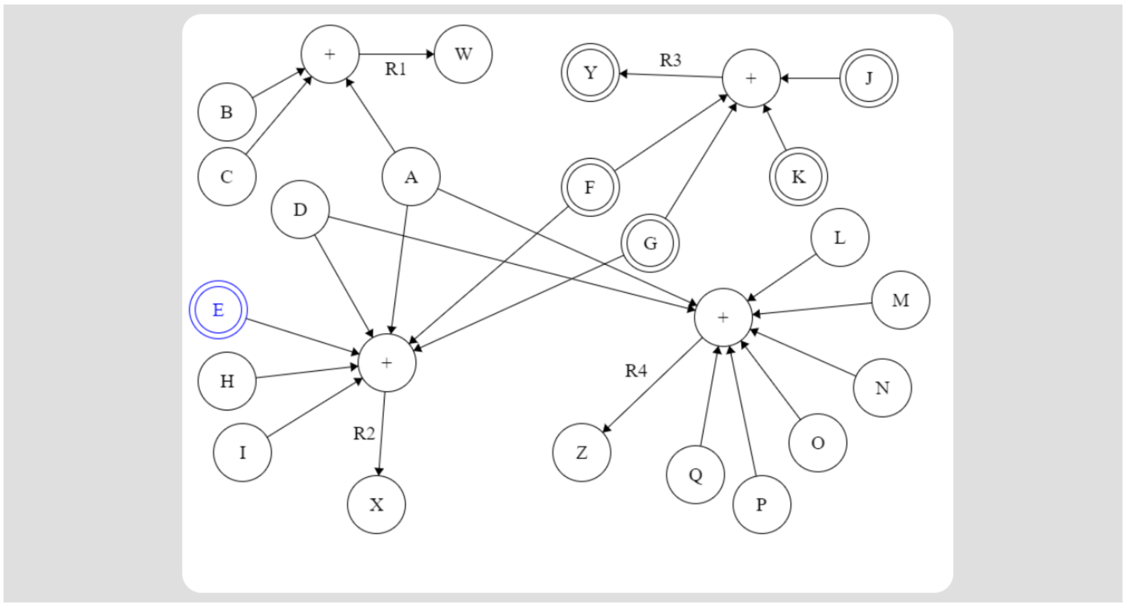
1. Fakta adalah J,G,F dan K bernilai true
   Pertanyaan (query) : apakah Y bernilai true?
2. Fakta adalah J,G,F, K, dan E bernilai true
   Pertanyaan (query) : apakah X bernilai true?

Graf

1. J, G, F dan K bernilai True

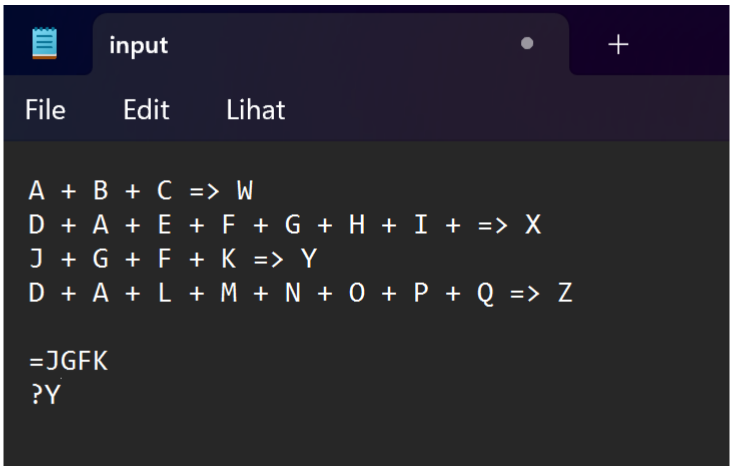


2. J, G, F, K DAN E bernilai True



Inputan Program

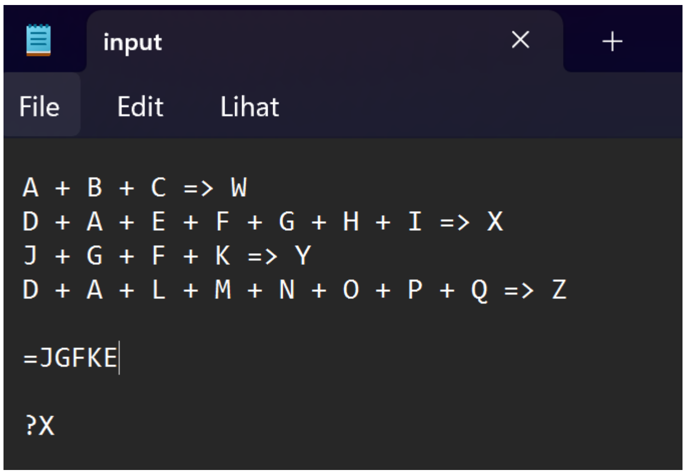
Fakta : J,G,F,K
Query : Y



```
A + B + C => W
D + A + E + F + G + H + I + => X
J + G + F + K => Y
D + A + L + M + N + O + P + Q => Z

=JGFK
?Y
```

Fakta : J,G,F,K,E

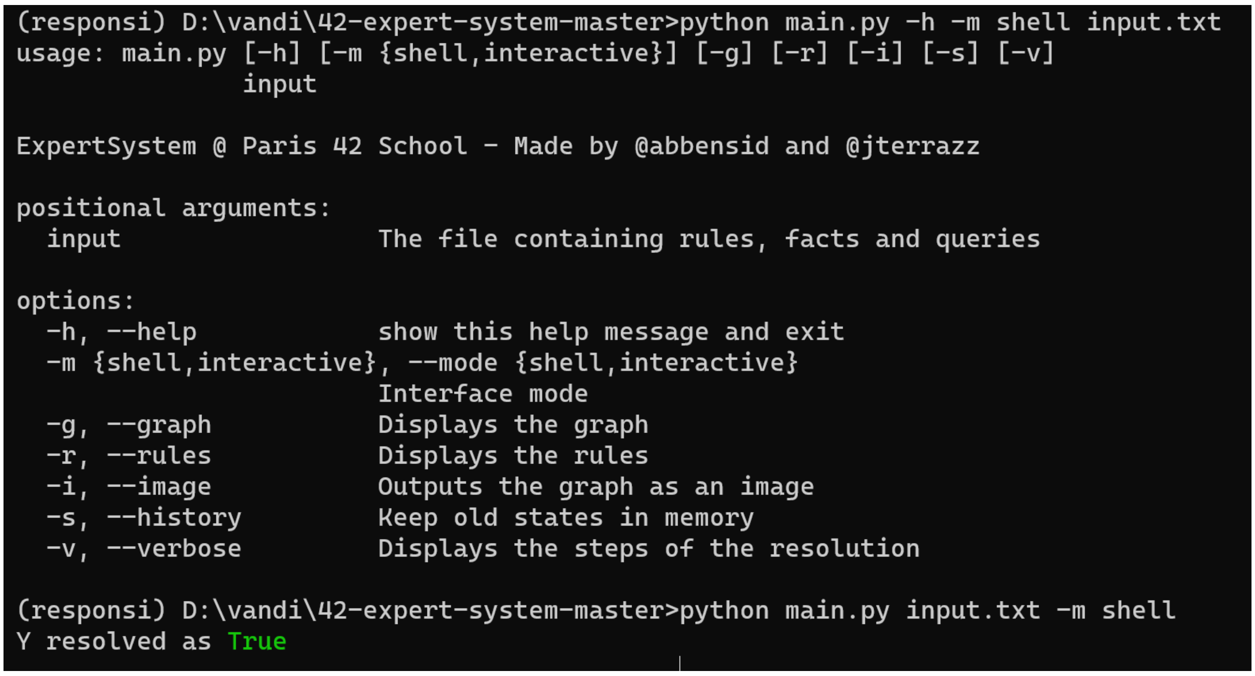
Query : X



```
A + B + C => W
D + A + E + F + G + H + I => X
J + G + F + K => Y
D + A + L + M + N + O + P + Q => Z

=JGFKE

?X
```

Output Program

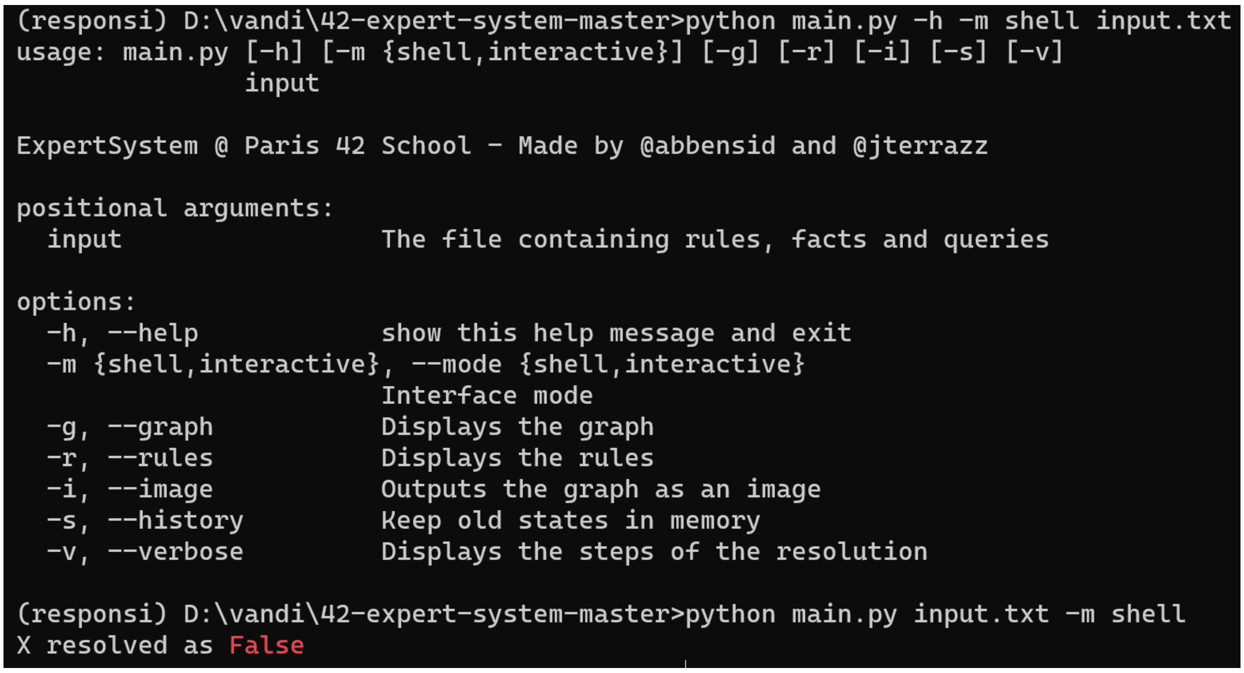
Fakta : J,G,F,K
Query : Y



```
(responsi) D:\vandi\42-expert-system-master>python main.py -h -m shell input.txt
usage: main.py [-h] [-m {shell,interactive}] [-g] [-r] [-i] [-s] [-v]
               input

ExpertSystem @ Paris 42 School - Made by @abbensid and @jterrazz

positional arguments:
  input                 The file containing rules, facts and queries

options:
  -h, --help            show this help message and exit
  -m {shell,interactive}, --mode {shell,interactive}
                        Interface mode
  -g, --graph           Displays the graph
  -r, --rules           Displays the rules
  -i, --image           Outputs the graph as an image
  -s, --history         Keep old states in memory
  -v, --verbose         Displays the steps of the resolution

(responsi) D:\vandi\42-expert-system-master>python main.py input.txt -m shell
Y resolved as True
```

Fakta : J,G,F,K,E

Query : X



```
(responsi) D:\vandi\42-expert-system-master>python main.py -h -m shell input.txt
usage: main.py [-h] [-m {shell,interactive}] [-g] [-r] [-i] [-s] [-v]
               input

ExpertSystem @ Paris 42 School - Made by @abbensid and @jterrazz

positional arguments:
  input                 The file containing rules, facts and queries

options:
  -h, --help            show this help message and exit
  -m {shell,interactive}, --mode {shell,interactive}
                        Interface mode
  -g, --graph           Displays the graph
  -r, --rules           Displays the rules
  -i, --image           Outputs the graph as an image
  -s, --history         Keep old states in memory
  -v, --verbose         Displays the steps of the resolution

(responsi) D:\vandi\42-expert-system-master>python main.py input.txt -m shell
X resolved as False
```

Penjelasan alur kerja program

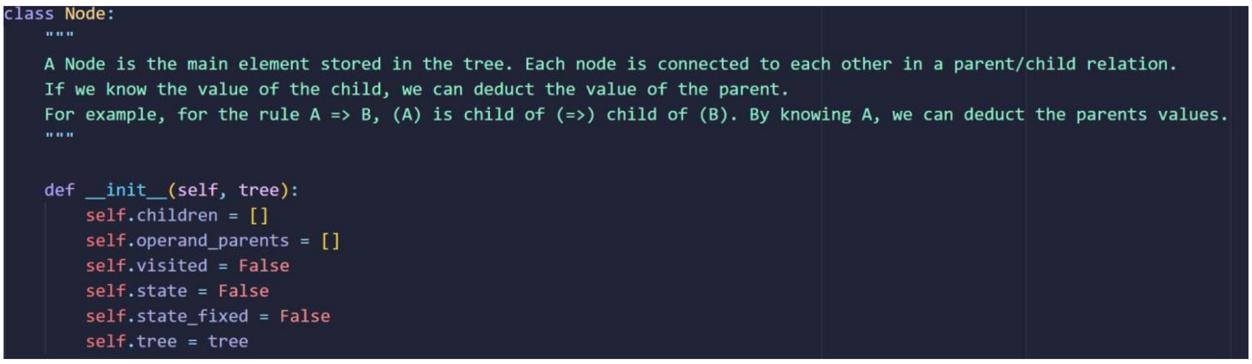
Program ini dibuat dengan 2 sub-tipe node yaitu

- AtomNode yaitu node yang berdiri sendiri seperti G,J,K, dst
- dan ConnectorNode yaitu operator penghubung antara Atomnote seperti AND, OR,dst

Dimana setiap facts (fakta) diawali dengan tanda (=)
contoh :
=GJFK : bahwa node G,J,F,K bernilai true
Sedangkan query diawali dengan (?)
contoh :
?Y : untuk mengetahui kebenaran nilai dari Y

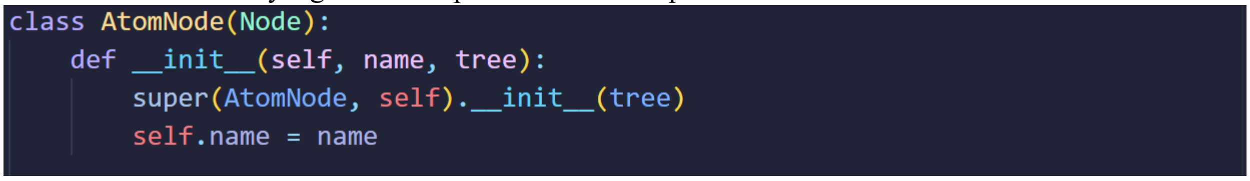
Alur kerja program :

1. node – node akan dilihat apakah memiliki turunan, apakah sudah pernah dikunjungi atau belom, serta melihat statenya apakah true atau false dalam class node



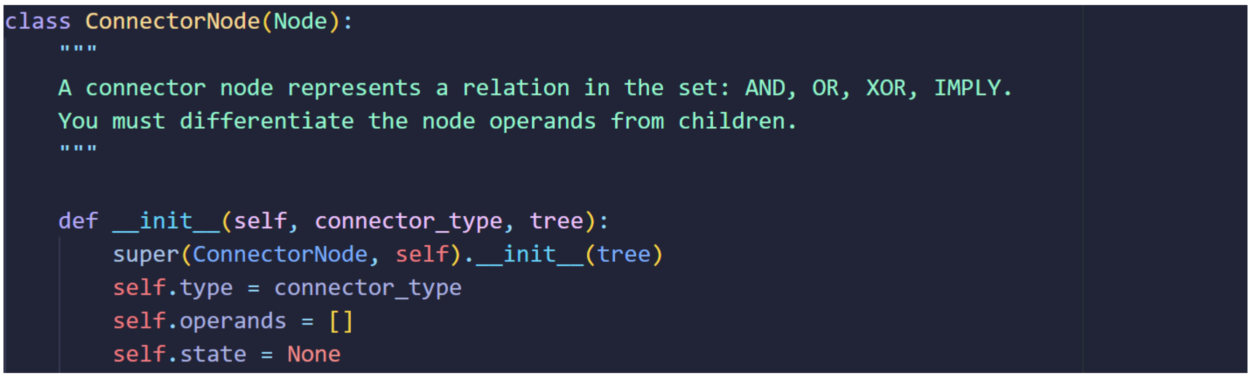
```
class Node:
    """
    A Node is the main element stored in the tree. Each node is connected to each other in a parent/child relation.
    If we know the value of the child, we can deduct the value of the parent.
    For example, for the rule A => B, (A) is child of (=>) child of (B). By knowing A, we can deduct the parents values.
    """

    def __init__(self, tree):
        self.children = []
        self.operand_parents = []
        self.visited = False
        self.state = False
        self.state_fixed = False
        self.tree = tree
```

2. Nama – nama node yang sudah diinputkan akan disimpan dalam class AtomNode



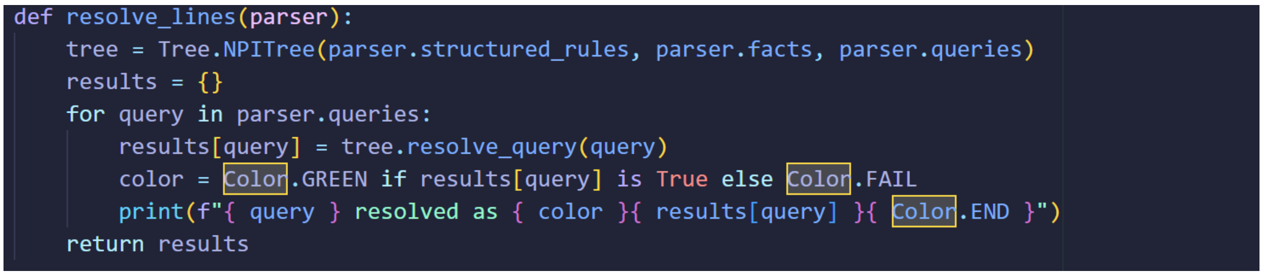
```
class AtomNode(Node):
    def __init__(self, name, tree):
        super(AtomNode, self).__init__(tree)
        self.name = name
```

3. Sedangkan tipe conector akan disimpan dalam class ConnectorNode, dalam class ini pula nilai operands akan disimpan yang nanti akan digunakan untuk menarik nilai kebenaran node conector itu sendiri.



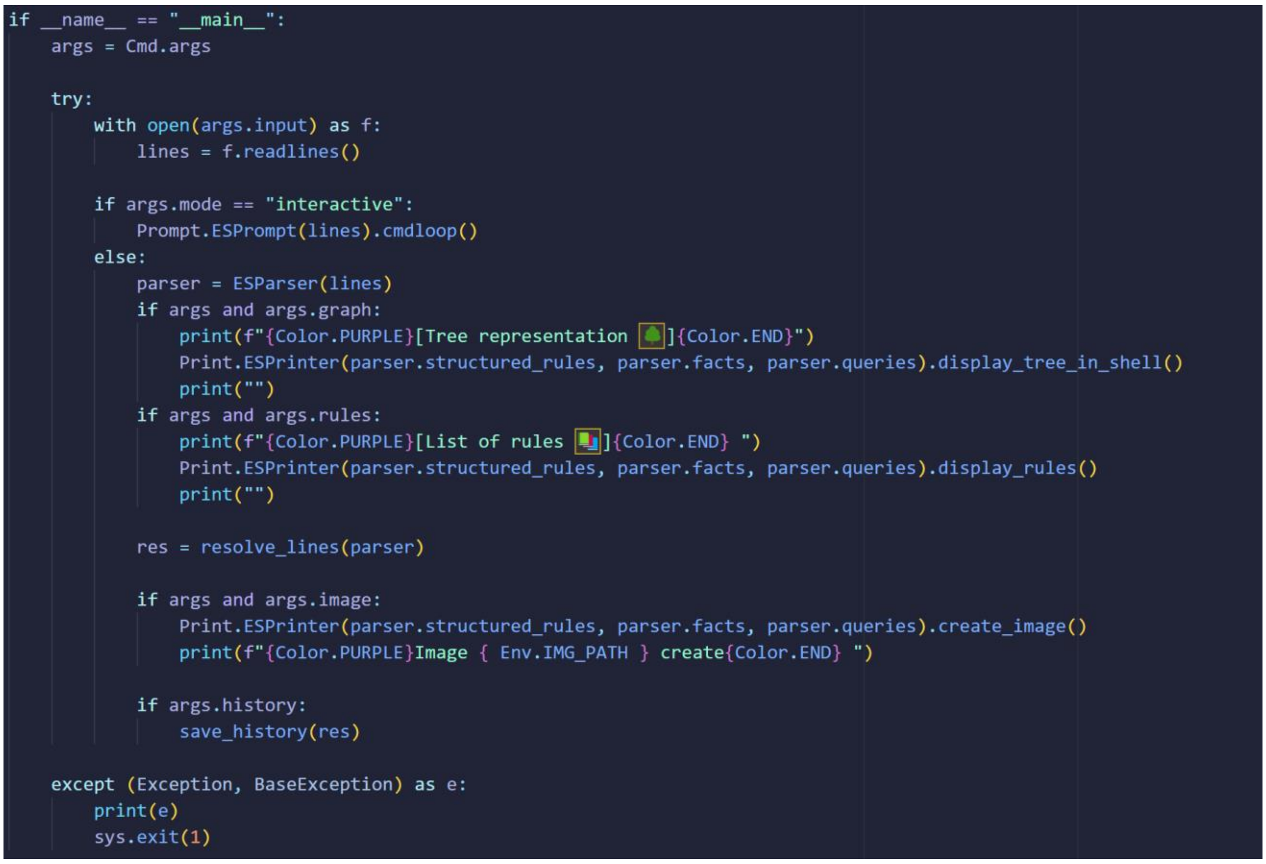
```
class ConnectorNode(Node):
    """
    A connector node represents a relation in the set: AND, OR, XOR, IMPLY.
    You must differentiate the node operands from children.
    """

    def __init__(self, connector_type, tree):
        super(ConnectorNode, self).__init__(tree)
        self.type = connector_type
        self.operands = []
        self.state = None
```

4. Aturan-aturan dan fakta-fakta yang diberikan akan dievaluasi dan hasil resolusi untuk setiap query disimpan dalam **results**.



```python
def resolve_lines(parser):
    tree = Tree.NPITree(parser.structured_rules, parser.facts, parser.queries)
    results = {}
    for query in parser.queries:
        results[query] = tree.resolve_query(query)
        color = Color.GREEN if results[query] is True else Color.FAIL
        print(f"{ query } resolved as { color }{ results[query] }{ Color.END }")
    return results
```

5. Kemudian untuk mengetahui serta membaca argumen dari baris perintah, membaca aturan-aturan dari file input, memeriksa mode, mengeksekusi aturan secara lebih rinci dapat dilihat menggunkan mode interactive.



```python
if __name__ == "__main__":
    args = Cmd.args

    try:
        with open(args.input) as f:
            lines = f.readlines()

        if args.mode == "interactive":
            Prompt.ESPrompt(lines).cmdloop()
        else:
            parser = ESParser(lines)
            if args and args.graph:
                print(f"{Color.PURPLE}[Tree representation 🌳]{Color.END}")
                Print.ESPrinter(parser.structured_rules, parser.facts, parser.queries).display_tree_in_shell()
                print("")
            if args and args.rules:
                print(f"{Color.PURPLE}[List of rules 📊]{Color.END} ")
                Print.ESPrinter(parser.structured_rules, parser.facts, parser.queries).display_rules()
                print("")

            res = resolve_lines(parser)

            if args and args.image:
                Print.ESPrinter(parser.structured_rules, parser.facts, parser.queries).create_image()
                print(f"{Color.PURPLE}Image { Env.IMG_PATH } create{Color.END} ")

            if args.history:
                save_history(res)

    except (Exception, BaseException) as e:
        print(e)
        sys.exit(1)
```