Elective Course
Course Code: CS4103
Autumn 2025-26

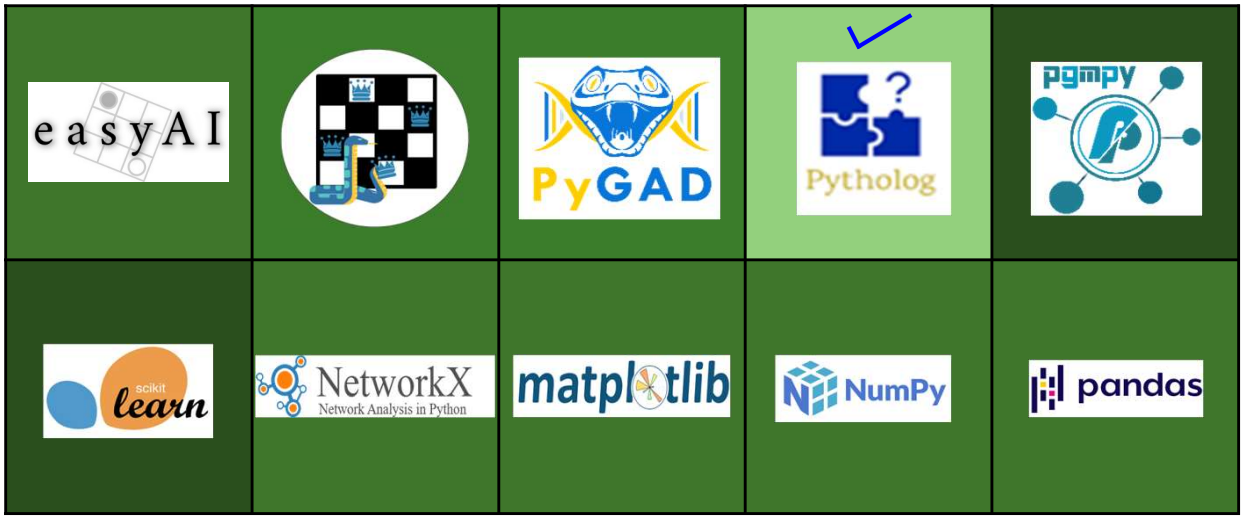
Lecture #27

Artificial Intelligence for Data Science

Week-8:
Exploring Logic Programming using Pytholog

Course Instructor:
Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

A Few Python Libraries for AI



Pytholog

- Developer: M. N. Fawi
- Open-source library to enable logic programming within Python
- Allows using **PROLOG** inside python environment

- **PROLOG:** *PROgrammation en LOGique*
  - declarative programming language
    - the program is a set of *facts* and *rules*, which define relations.
    - computation is initiated by running a query over the program
  - Based on first-order logic
  - uses backtracking search to answer the queries.
- Prolog uses lowercased variables to describe "constant values" and uppercased values to describe "variables" that need to be updated from the query.

Source: https://github.com/MNoorFawi/pytholog?tab=readme-ov-file; https://en.wikipedia.org/wiki/Prolog

Pytholog Example: Family Tree


```
import pytholog as pl

kb = pl.KnowledgeBase("family_tree")

# Facts about gender and parent, and rules regarding father, mother, sibling
kb([ "male(richard)",
     "male(alberto)",
     "female(lina)",
     "female(kate)",
     "female(karen)",

     "parent(richard, alberto)",
     "parent(richard, lina)",
     "parent(lina, kate)",
     "parent(lina, karen)",

     "father(X, Y):- male(X), parent(X, Y)",
     "mother(X, Y):-female(X), parent(X, Y)",
     "sibling(X, Y):-parent(Z, X), parent(Z, Y), neq(X, Y)"])
```

Pytholog Example: Family Tree


```
# Find all children of Richard
children_of_richard = kb.query(pl.Expr("parent(richard,Who)"))
print(f"Children of Richard: {children_of_richard}")

# Find the father of Alberto
father_of_alberto = kb.query(pl.Expr("father(Who,alberto)"))
print(f"Father of Alberto: {father_of_alberto}")

# Find siblings of Kate
siblings_of_kate = kb.query(pl.Expr("sibling(kate, Who)"))
print(f"Siblings of Kate: {siblings_of_kate}")
```

```
Children of Richard: [{'Who': 'alberto'}, {'Who': 'lina'}]
Father of Alberto: [{'Who': 'richard'}]
Siblings of Kate: [{'Who': 'karen'}]
```


Pytholog Example: Food/Flavor


```
import pytholog as pl

kb = pl.KnowledgeBase("flavor")
kb(["likes(alice, sausage)",
    "likes(michael, pasta)",
    "likes(diego, cookie)",
    "likes(kate, sausage)",
    "likes(george, limonade)",
    "food_type(cheddar, cheese)",
    "food_type(ritz, cracker)",
    "food_type(steak, meat)",
    "food_type(sausage, meat)",
    "food_type(limonade, juice)",
    "food_type(cookie, dessert)",
    "flavor(sweet, dessert)",
    "flavor(savory, meat)",
    "flavor(savory, cheese)",
    "flavor(sweet, juice)",
```

Pytholog Example: Food/Flavor

```
 "food_flavor(X, Y) :- food_type(X, Z), flavor(Y, Z)",
 "dish_to_like(X, Y) :- likes(X, L), food_type(L, T), flavor(F, T),
                        food_flavor(Y, F), neq(L, Y)"])


# Does Alice like Sausage?
print(kb.query(pl.Expr("likes(alice,sausage)")))

# Does Alice like Cookie?
print(kb.query(pl.Expr("likes(alice,cookie)")))

# Recommending dishes to Alice based on her taste preference
print(kb.query(pl.Expr("dish_to_like(alice,Reco)")))
```

```
['Yes']
['No']
[{'Reco': 'cheddar'}, {'Reco': 'steak'}]
```

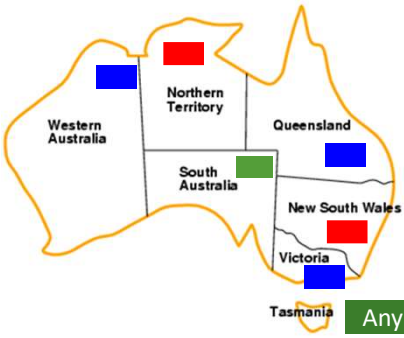

Pytholog Example: Map Coloring

```
import pytholog as pl

mcolor = pl.KnowledgeBase("Map_color")

mcolor([
    "diff(red, green)",
    "diff(red, blue)",
    "diff(green, red)",
    "diff(green, blue)",
    "diff(blue, red)",
    "diff(blue, green)",

    "coloring(WA,NT,SA,Q,NSW,V,T) :-
diff(WA,NT),diff(WA,SA),diff(NT,SA),diff(NT,Q),diff(SA,Q),diff(SA,NSW),
diff(SA,V),diff(Q,NSW),diff(V,NSW)"])
```

Pytholog Example: Map Coloring


```
#Query
print("Solution:",mcolor.query(pl.Expr("coloring(WesternAustralia,NorthernTerritory
,SouthAustralia,Queensland,NewSouthWales,Victoria,Tasmania)"),cut=True))
```

```
Solution: [{'WesternAustralia': 'blue', 'NorthernTerritory': 'red', 'SouthAustralia': 'green',
'Queensland': 'blue', 'NewSouthWales': 'red', 'Victoria': 'blue'}]
```



Any color for Tasmania

Pytholog Example: Chemical Reaction


```
#Logic programming to predict the products of basic chemical reactions
import pytholog as pl


#Creating knowledge base
kb = pl.KnowledgeBase("chemical_reactions")
kb(["element(carbon)",
    "element(hydrogen)",
    "element(oxygen)",
    "valency(carbon, 4)",
    "valency(hydrogen, 1)",
    "valency(oxygen, 2)",
    "fuel(hydrocarbon)",
    "reacts_with(oxygen, X):- fuel(X)",
    "combustion(X) :- reacts_with(oxygen, X)",
    "products(X, carbon_dioxide, water) :- combustion(X)"])
```

Pytholog Example: Chemical Reaction


```
# A few queries
#What are the products of hydrocarbon combustion?
print(kb.query(pl.Expr("products(hydrocarbon, X,Y)")))
#What are the products of hydrogen combustion?
print(kb.query(pl.Expr("products(hydrogen, X,Y)")))
#What does react with Oxygen?
print(kb.query(pl.Expr("reacts_with(oxygen, X)")))
#Is carbon a fuel?
print(kb.query(pl.Expr("fuel(carbon)")))
#What can be combustible?
print(kb.query(pl.Expr("combustion(What)")))
```

```
[{'X': 'carbon_dioxide', 'Y': 'water'}]
['No']
[{'X': 'hydrocarbon'}]
['No']
[{'What': 'hydrocarbon'}]
```


Using Pytholog in Other Applications

- Probabilistic Logic

```
battery_kb = pl.KnowledgeBase("battery")
battery_kb([
    "battery(dead,P) :- voltmeter(battery_terminals,abnormal,P2), P is P2 + 0.5",
    "battery(dead,P) :- electrical_problem(P), P >= 0.8",
    "battery(dead,P) :- electrical_problem(P2), age(battery,old,P3), P is P2 * P3 * 0.9",
    "electrical_problem(0.7)",
    "age(battery,old, 0.8)",
    "voltmeter(battery_terminals,abnormal,0.3)"])

battery_kb.query(pl.Expr("battery(dead, Probability)"))
```

- Rules from Machine Learning

```
iris_kb = pl.KnowledgeBase("iris")
iris_kb([## Rules
    "species(setosa, Truth) :- petal_width(W), Truth is W <= 0.80",
    "species(versicolor, Truth) :- petal_width(W), petal_length(L), Truth is W > 0.80 and L <= 4.95",
    "species(virginica, Truth) :- petal_width(W), petal_length(L), Truth is W > 0.80 and L > 4.95",
    ## New record
    "petal_length(5.1)",
    "petal_width(2.4)"])
```

- Graph Traversals with Pytholog

Using Pytholog in Other Applications

- Chemical Science
  - Determining all possible products from a given set of initial chemicals.
  - Simulating a chemical synthesis process
- Physical Science
  - Diagnosing hardware failures in complex scientific equipment
  - Identifying stellar category
- Earth/Environmental Science
  - Identifying rock type
  - Auto-interpreting earth surface image
- Biological/Medical Science
  - Developing diagnostic system
  - Pathway analysis

Questions?