**Bài 7: Logic Programming - Lập trình AI bằng Python**

Trong bài này, chúng ta sẽ tập trung vào lập trình logic và nó giúp ích như thế nào trong Trí tuệ nhân tạo.

Chúng ta đã biết rằng logic là nghiên cứu các nguyên tắc của suy luận đúng đắn hay nói một cách đơn giản nó là nghiên cứu về cái gì xảy ra sau cái gì. Ví dụ, nếu hai câu lệnh đúng thì chúng ta có thể suy ra bất kỳ câu lệnh thứ ba nào từ nó.

**1. Khái niệm :**

Lập trình Logic là sự kết hợp của hai từ, logic và lập trình. Lập trình Logic là một mô hình lập trình trong đó các vấn đề được thể hiện dưới dạng các dữ kiện và quy tắc bằng các câu lệnh của chương trình nhưng trong một hệ thống logic chính thức. Cũng giống như các mô hình lập trình khác như hướng đối tượng, chức năng, khai báo và thủ tục, v.v., nó cũng là một cách cụ thể để tiếp cận lập trình.

**2. Xử lý vấn đề với Logic Programming:**

Lập trình Logic sử dụng các dữ kiện và quy tắc để giải quyết vấn đề. Đó là lý do tại sao chúng được gọi là khối xây dựng của Lập trình Logic. Một mục tiêu cần được xác định cho mọi chương trình trong lập trình logic. Để hiểu cách một vấn đề có thể được giải quyết trong lập trình logic, chúng ta cần biết về các nguyên tắc xây dựng - Sự kiện và Quy tắc -

**a . Sự kiện :**

Trên thực tế, mọi chương trình logic cần các dữ kiện để làm việc để nó có thể đạt được mục tiêu đã cho. Sự thật về cơ bản là những tuyên bố đúng về chương trình và dữ liệu. Ví dụ, Delhi là thủ đô của Ấn Độ.

**b. Quy tắc :**

Trên thực tế, các quy tắc là những ràng buộc cho phép chúng ta đưa ra kết luận về miền vấn đề. Các quy tắc về cơ bản được viết dưới dạng mệnh đề logic để diễn đạt các sự kiện khác nhau. Ví dụ, nếu chúng ta đang xây dựng bất kỳ trò chơi nào thì tất cả các quy tắc phải được xác định.

Các quy tắc rất quan trọng để giải quyết bất kỳ vấn đề nào trong Lập trình Logic. Các quy tắc về cơ bản là kết luận hợp lý có thể thể hiện các sự kiện. Sau đây là cú pháp của quy tắc

A∶− B1,B2,...,Bn.

Ở đây A là head và B1,B2,..Bn là body

**3. Cài đặt Packages:**

Để bắt đầu với logic programming trong python, ta cần phải install 2 packages sau :

**a. Kanren:**Nó cung cấp cho ta một cách để đơn giản hóa mà chúng ta thường sử dụng. Nó cho phép  thể hiện logic về các quy tắc và sự kiện. Cú pháp cài đặt kanren:

**pip** install kanren

**b. Sympy :**

SymPy là một thư viện Python hỗ trợ biểu tượng toán học. Nó nhằm mục đích trở thành một hệ thống đại số máy tính (CAS) đầy đủ tính năng trong khi vẫn giữ mã đơn giản nhất có thể để dễ hiểu và dễ mở rộng. Cú pháp cài đặt SymPy:

**pip** install sympy

**4. Ví dụ về Logic Programming :**

Dưới đây là một số ví dụ có thể được giải quyết bằng lập trình logic -

**a. Biểu thức toán học :**

Trên thực tế, chúng ta có thể tìm thấy các giá trị chưa biết bằng cách sử dụng lập trình logic một cách rất hiệu quả. Đoạn mã Python sau sẽ giúp bạn so khớp một biểu thức toán học:

Trước tiên, hãy xem xét việc nhập các gói sau:

**from** kanren **import** run, **var**, fact

**from** kanren.assoccomm **import** eq\_assoccomm **as** eq

**from** kanren.assoccomm **import** commutative, associative

Chúng ta cần xác định các phép toán mà chúng ta sẽ sử dụng -

add = 'add'

mul = 'mul'

**fact**(commutative, mul)

**fact**(commutative, add)

**fact**(associative, mul)

**fact**(associative, add)

Bắt buộc phải xác định các biến như sau :

a, b = **var**('a'), **var**('b')

Ta cần kết hợp biểu thức với mẫu ban đầu. Mẫu ban đầu là (5 + a) \* b -

Original\_pattern = (mul, (add, 5, a), b)

Sử dụng 2 biểu thức sau để khớp với mẫu ban đầu :

exp1 = (mul, 2, (add, 3, 1))

exp2 = (add,5,(mul,8,1))

**print**(run(0, (a,b), eq(original\_pattern, exp1)))

**print**(run(0, (a,b), eq(original\_pattern, exp2)))

Output :

((3,2))

()

Đầu ra đầu tiên đại diện cho các giá trị của a và b. Biểu thức đầu tiên khớp với mẫu ban đầu và trả về giá trị cho a và b nhưng biểu thức thứ hai không khớp với mẫu ban đầu do đó không có gì được trả về.

**5. Kiểm tra số nguyên tố :**

Với sự trợ giúp của lập trình logic, chúng ta có thể tìm các số nguyên tố từ một danh sách các số và cũng có thể tạo ra các số nguyên tố. Đoạn code sau sẽ tìm số nguyên tố từ danh sách các số và cũng sẽ tạo ra 10 số nguyên tố đầu tiên.

**from** kanren **import** isvar, run, membero

**from** kanren.core **import** success, fail, goaleval, condeseq, eq, **var**

**from** sympy.ntheory.generate **import** prime, isprime

**import** itertools **as** it

Bây giờ, chúng ta sẽ định nghĩa một hàm có tên là prime\_check sẽ kiểm tra các số nguyên tố dựa trên các số đã cho dưới dạng dữ liệu.

def prime\_check(**x**):

**if** isvar(**x**):

**return** condeseq([(eq,x,p)] **for** p in **map**(prime, it.count(1)))

**else**:

**return** success **if** isprime(**x**) **else** fail

Khai báo biến :

x = **var**()

print((**set**(run(0,x,(membero,x,(12,14,15,19,20,21,22,23,29,30,41,44,52,62,65,85)),

(prime\_check,x)))))

print((run(10,x,prime\_check(x))))

Output :

{19, 23, 29, 41}

(2, 3, 5, 7, 11, 13, 17, 19, 23, 29)