

الذكاء الاصطناعي والأنظمة الخبيرة



الدليل العملي
لغة البرمجة برولوج
SWI Prolog

أ. سامي زواهره

حقوق الطبع محفوظة

جامعة القدس المفتوحة



كلية التكنولوجيا
والعلوم التطبيقية

رقم المقرر 1484

الفناء الصناعى والأنظمة الخبيرة

الدليل العملى

SWI PROLOG

أ. سامى زواهره

فرع بىء لحم

مراجعة: د. بسام ترك، أ. رائء زحالقة، أ. جمال حمدان، أ. سيرىن ابو عىشة

المحتويات

الموضوع

الصفحة

5	1. روابط مفيدة
	1. 1. تحميل وتنصيب برنامج SWI Prolog
	1. 2. كتابة البرنامج الأول بلغة برولوج
	1. 3. دورة كاملة للبرمجة بلغة برولوج
	1. 4. الذكاء الصناعي بلغة Python
6	2. مقدمة لغة برولوج
6	3. كيفية الإعلان عن الحقائق والسؤال عنها
8	4. الثوابت والمتغيرات
9	5. أنواع البيانات data types التي توفرها لغة البرولوج
9	6. المعاملات operators
10	7. التراكيب structures
11	8. البحث الراجع backtracking
15	9. العمليات الرياضية arithmetic operations
16	10. القوائم lists
17	11. معامل القطع للبحث الراجع
18	12. أمثلة على لغة برولوج (Swi-Prolog)
28	13. مثال تطبيقي على مسيطر المنطق الغامض Fuzzy Logic Controller باستخدام لغة البرمجة بايثون

PROLOG | 

لغة البرمجة برولوج Prolog

1. روابط مفيدة:

في بداية هذا الدليل اليك عزيزي الطالب مجموعة من روابط اليوتيوب المفيدة والتي ستساعدك في البدء بكتابة كود Prolog باستخدام برنامج SWI Prolog:

1. 1. تحميل وتنصيب برنامج SWI Prolog على الوندوز تابع المقطع التالي:
https://youtu.be/3sIdEO_KUJg

1. 2. لكتابة البرنامج الأول بلغة برولوج باستخدام برنامج SWI Prolog تابع المقطع التالي:

<https://youtu.be/BCW8-ueg8LY>

<https://youtu.be/-v1K9AnkAeM>

<https://youtu.be/t6L7O7KiE-Q>

1. 3. دورة كاملة بعنوان Logical Programming Course للبرمجة بلغة برولوج باستخدام برنامج SWI Prolog، تحتوي الدورة على ثلاثة عشر فيديو للوصول لها ادخل على القناة وابحث عن الدورة، ستجد في هذه القناة العديد من الدورات الاخرى المفيدة، تابع المقطع التالي:

https://youtu.be/jOAR_pAPP90

بالطبع إذا بحثت في اليوتيوب عن البرمجة بلغة برولوج باستخدام برنامج SWI Prolog ستجد العديد من الروابط المفيدة التي يمكنك الاستفادة منها والخيار لك في النهاية.

1. 4. الذكاء الصناعي بلغة Python:

إذا كنت من المهتمين بلغة بايثون وكيفية التعامل مع مسائل الذكاء الصناعي بهذه اللغة فيمكنك الدخول الى الرابط التالي لمعرفة ذلك:

<https://www.edureka.co/blog/artificial-intelligence-with-python>

بالطبع يمكنك إيجاد الكثير من المقاطع على اليوتيوب التي تشرح هذا الامر كما يوجد عشرات الكتب التي توضحه أيضا ويمكنك عمل مشروع تخرج ذا قيمة عالية إذا تعلمت هذه المهارة.

2. مقدمة لغة برولوج:

لغة البرولوج (PROLOG) هي اختصار (PROGRAMMING in LOGIC) أي البرمجة المنطقية. وصمم هذه اللغة أستاذ بجامعة مرسيليا بفرنسا يدعى ألن كولميرير (Alan Colmeraur) سنة 1972 ولغة برولوج هي لغة تصريحية Declarative Language. تستخدم لغة البرولوج في العديد من برامج الذكاء الاصطناعي وبرامج معالجة اللغات الطبيعية. تختلف طريقة البرمجة في برولوج عن اللغات التقليدية. في برولوج يتم كتابة الحقائق (Facts) والقواعد (Rules) في قاعدة بيانات، ثم يتم كتابة استفسارات queries من قاعدة البيانات. الحقيقة عبارة عن سند (predicate) وهي الوحدة الأساسية للبرولوج.

ونسستخدم SWI-Prolog وهي بيئة لبرولوج مجانية شاملة. منذ بدايته في عام 1987 ، كان تطوير SWI-

Prolog مدفوعاً باحتياجات تطبيقات العالم الحقيقي. يستخدم SWI-Prolog على نطاق واسع في البحث

والتعليم بالإضافة إلى التطبيقات التجارية. يعمل SWI-Prolog على منصات Unix و Windows و

Linux و Macintosh.

تعتمد لغة البرولوج على مفهوم البرمجة المنطقية (Logic Programming) ، والتي تتعامل مع جمل (Statements) تحتوي على أشياء (Objects) والعلاقات (Relationships) التي تربط بينها وبين الجملة:

professor (ahmad, bisan).

في هذه الجملة تسمى كلمة (professor) بالمسند أو المحمول (Predicate) وتمثل العلاقة بين المعاملات (Ahmad) هو أستاذ (Bisan).

وعلى هذا فإن لغة البرولوج تسمح للمبرمج بتمثيل العلاقات بين الأشياء وتجميع وتنظيم هذه العلاقات حتى يمكن الوصول إلى استنتاج منطقي من الحقائق التي تمثلها تلك العلاقات . وذلك على عكس اللغات التقليدية مثل لغة سي (C) مثلا التي تطلب من المبرمج كتابة الخطوات التفصيلية التي يجب إتباعها.

والبرمجة بلغة البرولوج تنقسم إلى ثلاثة مراحل هي:

1- إعلان الحقائق عن الأشياء (Objects) والعلاقات التي تربط بينها.

2- تعريف القواعد (Rules) التي تحكم كلا من الأشياء والعلاقات التي تربط بينها .

3- السؤال عن الأشياء والعلاقات التي تربطها.

والمرحلة الثالثة يمكن أن تأتي بعد المرحلة الأولى مباشرة حيث يمكن السؤال عن الأشياء دون تطبيق أي قواعد.

3. كيفية الإعلان عن الحقائق والسؤال عنها :

الإعلان عن الحقائق في برنامج البرولوج يجب أولاً تحديد الأشياء (Objects) والعلاقات التي تمثل تلك الحقائق. فمثلاً إذا كانت هناك حقيقة تقول أن (Ahmad likes Sami) فالأشياء في هذه الحقيقة هي الأسماء (Ahmad, Sami) أما العلاقة بينهما فهي (likes) وتسمى بالمسند (Predicate) أي الصفة التي تتعلق بشيء

ما أو العلاقة التي تربط بين شيئين أو أكثر. ولتمثيل هذه الحقيقة في برنامج البرولوج تكتب كالاتي :

likes (ahmad , sami).

لاحظ كتابة أسماء الأشياء والعلاقات بالحروف الصغيرة (Small Letters) وذلك لأن الأسماء التي تبدأ بالحروف الكبيرة (Capital Letters) أو بالحرف (Underscore) (_) يعتبرها البرولوج متغيرات (Variables). وتختلف الحقيقة باختلاف ترتيب أسماء الأشياء بمعنى أن:

likes(ahmad ,sami)تختلف عن

likes (sami , ahmad)وينتهي الإعلان عن الحقيقة بوضع نقطة (.) في آخرها.

ويطلق على التعبير likes (sami , ahmad) في لغة البرولوج لفظ العبارة (Clause) ويمكن ترجمة اللغة المكتوبة بإحدى اللغات الطبيعية (الإنجليزية أو العربية) إلى عبارة أو أكثر من عبارات البرولوج. ويتم ذلك بتحديد الأشياء التي يدور حولها موضوع الجملة وتحديد الصفات أو العلاقات التي تميزها، أو محمول الجملة.

وبمجرد تخزين الحقائق في قاعدة بيانات يمكن بعد ذلك الاستفسار (Query) عن أي أشياء والعلاقات التي تربط بينها فعلى سبيل المثال إذا أخذنا قاعدة المعرفة التي تمثل العلاقة بين (Ahmad , Sami) في المثال الأول فيمكن السؤال عن العلاقة بينهما بإحدى الطريقتين التاليتين :

likes(ahmad , sami) – ?

وبناء على هذا السؤال يقوم البرولوج بالبحث في قاعدة المعرفة (Knowledgebase) عن عبارة (Clause) تطابق العبارة الموجودة في السؤال ويحدث الاتفاق عندما يتفق كل من المسند في عبارة السؤال مثل (like) مع المسند في الحقيقة الموجودة في قاعدة المعرفة وأيضا عندما تتفق معاملات السؤال مع مثيلاتها من المعاملات الحقيقية.

وبما أن السؤال يتفق مع الحقيقة يقوم البرولوج بالرد بالإيجاب (Yes) على ذلك السؤال ولكن إذا تغير ترتيب المعاملات، مثل:

likes (sami , ahmad).

يكون الرد بالنفي (No).

وهذا يعني أن الحاسوب (أو برنامج البرولوج) لا يعلم حقيقة تتفق مع هذا السؤال ، أو بمعنى أدق لا توجد عبارة تمثل هذه الحقيقة في قاعدة المعرفة.

4. الثوابت والمتغيرات:

يمكن أن تحتوي العبارة في لغة البرولوج على نوعين من البيانات ثابت (Constants) ومتغيرات (Variables). والثابت (Constant) هو أي أسم يقوم بوصف شيء محدد (Specific Object) ,مثل (sami) , (ahmad) , أو وصف أي علاقة محددة (Specific Relation) , مثل (likes) وهناك نوعان من الثوابت في لغة البرولوج وهما الأعداد الصحيحة (Integers) والعناصر (Atoms).

أ_ الأعداد الصحيحة (Integers)

وهي مجموعة الأعداد الصحيحة الموجبة أو السالبة .

ب_ العناصر (Atoms)

العنصر (Atom) هو عبارة عن سلسلة من الحروف أو الأعداد أو الحروف الخاصة والتي تصف اسم أي شيء (Object) أو علاقة (Relationship) كما يجب أن يتحقق فيه الشروط الثلاثة التالية :

1- ألا يبدأ بعدد صحيح أو حرف كبير (Capital) أو العلامة (underscore) (_).

2- ألا يحتوي على علامة (-) (hyphen).

3- إذا احتوى العنصر على أي علامة من العلامات السابقة يجب أن ينحصر بين علامتي التنصيص (").

ومن أمثلة العناصر ما يلي:

abcmahammad

chapter_10 ' This is an atom.'

و الأمثلة التالية ليست عناصر:

97 Vector

Street5 _Tax

Large_Number

أما المتغير (Variable) فهو أي سلسلة من الحروف تبدأ إما بحرف كبير (Capital) وأما بالعلامة (Underscore) (_)، وهو اسم خاص يمكن أن يتفق مع أي شيء (Object) موجود بقاعدة المعرفة ، والأسماء الآتية تعتبر متغيرات:

Abc _Ten Like

X Xaxis Move

Yz Ali X_Y_Z

وتستخدم المتغيرات عادة للتعبير عن أي شخص أو عن أي شيء، فعلى سبيل المثال إذا أردنا أن نعبر عن الجملة (Everyone Likes Ahmad) نكتب:

likes(X , ahmad).

وفي هذه الحالة إذا سألنا عن أي شخص هل هو يحب أحمد فسيكون رد البرولوج بالإيجاب، أي إذا سألنا:

likes(rami,ahmad).

سيكون الرد هو :

Yes

5. أنواع البيانات data types التي توفرها لغة البرولوج:

1. integer
2. real
3. character
4. string 'sami'
5. symbol sami

6. المعاملات (operators):

- 1- Arithmetic: + , - , * , div , mod
- 2- Relational: = , > , < , >= , <= , <>
- 3- Logical: 1. and ,
2. or ;
3. not

7. التراكيب (Structure):

التركيب (Structure) ، ويسمى أيضا بالمسند المركب (Compound Predicate) ، هو العبارة التي تأخذ الشكل:

predicate (argument1 , argument2 ,).

حيث المعامل الأول (Argument1) يمكن أن يكون ثابتا (constant) أو متغيرا (variable)
أو عبارة (clause) أو تركيبا (structure) . والأمثلة التالية تعتبر تراكيب صحيحة .

like (ahmad , sami).

point (x , y , z).

owned (ahmad, book (X , author ('Mc Craw Hill'))).

ويمكن أن يحتوي التركيب على المعاملات المنطقية (And)، (Or) . والمثال التالي يوضح كيفية استخدام المعامل (,) أي (and) .

likes (ahmad , sami) , like (rami , sami) .

وللإجابة على هذا السؤال يقوم البرولوج بالبحث في قاعدة المعرفة عن الشرط الأول من السؤال ، أي العبارة الأولى ، فإذا وجدها لا تتفق مع أي من الحقائق الموجودة رد بالنفي (no) وإذا وجدها تتفق مع حقيقة من الحقائق يبدأ بالبحث عن الشرط الثاني، أي العبارة الثانية ، فإذا وجدها لا تتفق مع أي من الحقائق رد بالنفي (NO) ذلك رد بالإيجاب (yes) .

أما المثالين التاليين فيوضحان كيفية استخدام المعامل (or) و (|) وكلاهما يؤدي نفس المعنى :

likes (ahmad , sami) ; likes (ahmad , rami) .

أو likes (ahmad , sami) | likes (ahmad , sami)

وفي هذه الحالة يرد البرولوج بالإيجاب (yes) إذا وجد اتفاق بين التعبير الأول وحقيقة من الحقائق وإذا لم يجد يقوم بالبحث عن التعبير الثاني في الحقائق فإذا تم الاتفاق بينه وبين أي حقيقة رد بالإيجاب (yes) وإذا لم يجد رد بالنفي (no).

8. البحث الراجع (Backtracking):

تتميز لغة البرولوج بإمكانية البحث الراجع (Backtracking) ، وهي طريقة من طرق البحث عن معلومات معينة داخل قاعدة المعرفة . والمثال التالي يوضح أهمية وجود هذه الخاصية .

إذا زدنا البرولوج بقاعدة المعرفة التالية :

likes (ahmad , sami) .

likes (ahmad , sami) .

likes (ahmad , sami) .

likes (ahmad , sami) .

likes (ahmad , sami) .

فإذا أردنا الاستفسار عن الأشخاص الذين يحبهم " أحمد " نكتب السؤال التالي :

likes (ahmad , People_Ahmad_Likes) .

في هذه الحالة يبدأ البرولوج بالبحث في قاعدة المعرفة لإيجاد قيم المتغير (People_Ahmad_Likes) وهذا المتغير يسمى غير محدد (uninstantiated) وعندما يجد حقيقة (أي عبارة) تتفق مع السؤال (وهي أول عبارة) يقوم البرولوج بإسناد الاسم (أي الشيء) المناظر لذلك المتغير إليها وضع (People_Ahmad_Likes=sami) . وفي هذه الحالة يقال أن المتغير تم تحديده (instantiated) بالعنصر (أو الشيء أو الثابت) (sami)

والآن دعنا نرى الإجابة على هذا السؤال وبعد ذلك نرى كيف توصل البرولوج إلى هذه الإجابة . فالإجابة على السؤال السابق ستكون كما يلي :

People_Ahmad_Like = sami

People_Ahmad_Like= rami

People_Ahmad_Like = omar

People_Ahmad_Like = reem

والذي حدث داخل البرولوج للوصول إلى هذه النتيجة هو كالتالي:
يبدأ البرولوج في البحث داخل قاعدة المعرفة من أول عبارة وعندما يجد أي توافق بين المسند والمعامل الأول في السؤال والمسند والمعامل الأول في الحقيقة يقوم بوضع علامة عند هذه الحقيقة ويحاول بعد ذلك البحث عن حقيقة أخرى تتفق مع السؤال وهكذا إلى أن يتم البحث في قاعدة المعرفة كلها .

وعند الانتهاء من البحث في قاعدة المعرفة يعود البرنامج إلى تلك المعاملات التي كان قد وضعها ، ومن ثم يقوم بإسناد المعاملات المقابلة للمتغير في تلك الحقائق إلى المتغير ، ومع كل إسناد يقوم بالرد بقيم المتغير الذي وجدها في كل حقيقة بالصورة السابق توضيحها .

إذا أردنا البحث عن شخص ما (person 1) والذي يكون أبا لشخص معروف (person 2) نكتب القاعدة التالية :

brother_of (Person1 , Person2):-

parent(X,Person 1),

parent(X,Person 2),

sex(Person1,male),

diff(Person 1, Person 2).

diff (X , Y):- X/==Y.

وهذا معناه أننا نحدد أن الشخص الأول (person1) يكون أبا للشخص الثاني (person2) إذا كان والد (parent) الشخص الأول هو نفسه والد الشخص الثاني ، وأن الشخص الأول ذكر (لأنه "أخ" الشخص الثاني) وكذلك الشخص الأول يختلف عن الشخص الثاني (وذلك لكي لا يرد علينا البرولوج بأن محمد أخو نفسه لأن لهم نفس الأب !) والقاعدة التي تفرق بين الشخصين تم كتابتها بعد القاعدة الأولى وهي تقول أن المتغير (X) لا يساوي (/==) المتغير (Y)

ولتوضيح هذه القاعدة نكتب قاعدة المعرفة الخاصة بالأسرة المكونة من (Ali) أب و (Fatima) أم و (Mona ,)

. أبناء (Ahmad, Mohammad , Khalid)

وهي تكتب كالاتي:

parent (ali,ahmad).

parent (ali , mohammad).

parent (ali , khalid).

parent (ali , mona).

parent (fatima , mona).

parent (fatima , mohammad).

parent (fatima , khalid).

parent (fatima , ahmad).

sex (ali , male).

sex (fatima , female).

sex (mona , female).

sex (ahmad , male).

sex (mohammad , male).

sex (khalid , male).

وهذه التعبيرات أو الحقائق التي تكون قاعدة المعرفة للأسرة تمثل العلاقات التي تربط الأسماء ببعضها وهي أن (Ali) هو والد كل من (Ahmad) و (Mohammad) و (Khalid) وهو ذكر (male) وهم ذكور وهو أيضا والد (Mona) وهي أنثى (female) وان (Fatima) هي أم كل من (Ahmad) و (Khalid) و (Mohammad) و (Mona) وهي أنثى (female) وبعد كتابة قاعدة المعرفة كما سبق الإيضاح يمكن إجراء الاستفسارات التالية:

1-هل (Khalid) أخو (Mohammad) ويكون السؤال كالتالي :

brother_of(khalid , mohammad).

وتكون الإجابة بالإيجاب: Yes

وذلك لان البرولوج تتبع الخطوات التي تحقق القاعدة (brother_of) وهي أن الشخص الأول والثاني لهما نفس الأبويين والأول ذكر ويختلف عن الثاني (أي أن Khalid لا يساوي Mohammad).

2- من هم أخوة (ahmad), ويكون السؤال كالتالي:

brother_of(X , ahmad).

وفي هذه الحالة يكون الرد بأسماء كل إخوة (Ahmad) الذين يتفقون مع القاعدة (brother_of) ويكون الرد كالتالي

X=mohammad

X=khalid

أي أن كل من (Mohammad) و (Khalid) يتفق مع المتغير (X) والذي يمثل اسم أخوة (Ahmad) ويلاحظ انه لم يذكر (Mona) وذلك لان نوعها (sex) يختلف عن النوع الموجود بالقاعدة (brother_of).

لنفرض الحقائق (facts) الآتية:

Sami likes reading

Ahmad likes swimming

Bisan likes playing

Dima likes what Bisan likes.

لتمثيل هذه الحقائق بلغة برولوج :

```
File Edit Browse Compile Prolog Pce Help
like.pl
likes(sami,reading) .
likes(ahmad,swimming) .
likes(bisan,playing) .
likes(dima,X):- likes(bisan,X) .
```

وعند إجراء عدد من الاستفسارات بعد تنفيذ البرنامج:

```
?-
| likes(X,Y).
X = sami,
Y = reading ;
X = ahmad,
Y = swimming ;
X = bisan,
Y = playing ;
X = dima,
Y = playing.
```

```
?- likes(ahmad,swimming).
true.
```

```
?- likes(bisan,dancing).
false.
```

```
?- likes(X,playing).
X = bisan ;
X = dima.
```

9. العمليات الرياضية (Arithmetic Operations):

تحتوي معظم إصدارات لغة البرولوج على المعاملات الرياضية (Arithmetic Operations) الموضحة في الجدول التالي , كما أن بعض الإصدارات تحتوي على المعامل (is) والذي يقوم بعمل المعامل (=), أي أن العبارة (x is 10+20) تكافئ (x = 10+20)

المعامل	المعنى	مثال	النتائج
+	الجمع	2+3	5
-	الطرح	3-1	2
*	الضرب	3*5	15
/	نتائج القسمة الصحيحة	7/4	1
mod	باقي القسمة الصحيحة	7 mod 4	3
=	اختبار التساوي	2=2	true
/=	اختبار عدم التساوي	3/=2	true
>	اكبر من	3>5	false
<	اقل من	3<5	true
<=	اقل من أو يساوي	x<=y	غير محدد
>=	اكبر من أو يساوي	X >=y	غير محدد

10. القوائم (Lists):

قد تكون القائمة فارغة ويرمز لها بالرمز :

[]

وإذا لم تكن فارغة فإن العنصر الأول يسمى بالرأس والقائمة المكونة من بقية العناصر تسمى بالذيل. والمثال التالي يمثل كيفية استخدام القوائم في البرولوج

وإذا اعتبرنا الحقيقة التالية:

```
friends([a , b , c , d , e, ])
```

فإذا أردنا معرفة رأس القائمة و ذيلها نكتب السؤال التالي :

```
friends([Head |Tail ]).
```

فيجب البرولوج بالآتي:

```
Head =a
```

```
Tail=[b,c,d,e]
```

أي أن ذيل القائمة هو أيضا قائمة فرعية

فعلى سبيل المثال يمكن كتابة الدالة (member) في لغة البرولوج ، بتعريفها بالقاعدتين التاليتين :

1- العنصر (A) عضو (member) في القائمة (P) إذا كان (A) هو أول عنصر في (P).

2- إذا لم يكن (A) هو أول عنصر في (P) فإن (A) يكون عنصرا في (P) فقط إذا كان عنصر في ذيل القائمة (P) .

وتكتب هاتان القاعدتان في لغة البرولوج كالتالي :

```
member(A ,[A|_ ]).
```


`member(A ,[_ |Y]):- member(A ,Y).`

ففي القاعدة (أو الحقيقة) الأولى نعرف أن المسند (member) يتحقق إذا كان العنصر (A) هو أول عنصر في القائمة بغض النظر عن ذيل القائمة (وتعرف هذه بوضع العلامة `_`) وتسمى (Underscore) مكان ذيل القائمة.

أما في القاعدة الثانية فإن المسند (member) يتحقق إذا كان عنصرا في ذيل القائمة بغض النظر عن رأس القائمة وهذه القاعدة تسمى بالقاعدة التكرارية (Recursive) .

11. معام القطع للبحث الراجع:

معام القطع (Cut Operator) هو من أهم المعاملات في لغة البرولوج والذي يتحكم في عملية البحث الراجع (Backtracking) ويمثل هذا المعامل بعلامة التعجب (!). و استخدام هذا المعامل يعني للبرولوج تخطي الاختبارات السابقة للاختبار الذي يوجد به معام القطع (!) وذلك عندما تبدأ البرولوج في عملية البحث الراجع وهذا المعامل يستخدم لزيادة سرعة البرنامج وتقليل المساحة المستخدمة من الذاكرة وكذلك منع البرولوج من إعطاء عدد كبير أو لا نهائي من الحلول.

ولتوضيح تأثير هذا المعامل نأخذ المثال التالي :

باستخدام القاعدتين السابقتين للدالة (member) وللاستفسار بالسؤال التالي :

`member(X ,[a ,b ,c , d, e]).`

يرد البرولوج بالتالي:

`X=a`

`X=b`

`X=c`

`X=d`

`X=e`

وللحد من هذا العدد الكبير من الحلول نستخدم معام القطع في قاعدتي تعريف الدالة (member) كالتالي:

```
member(X,[X|_]) :- !.
```

```
member(X,[_|Y]):- member(X,Y).
```

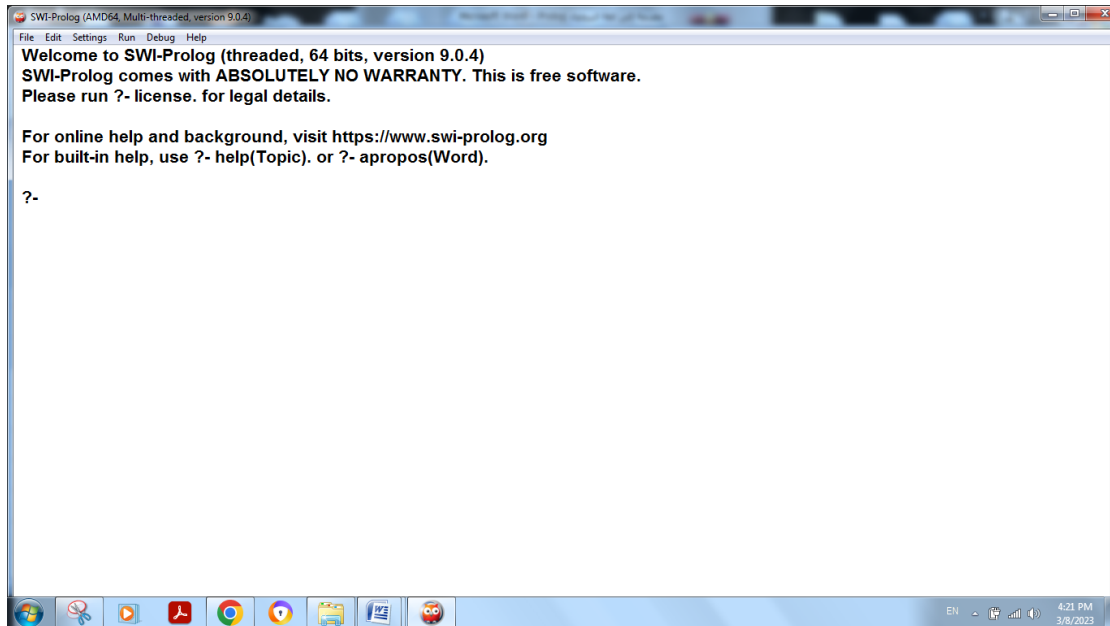
في هذه الحالة عند الاستفسار بالسؤال

```
member(X,[a,b,c,d,e,]).
```

يكون الرد كالآتي: $X=a$

ففي هذه الحالة عند تحقيق القاعدة الأولى يعرف البرولوج مكان معامل القطع ، وبالتالي عندما يبدأ في البحث الراجع أوتوماتيكيا يتوقف عند مكان معامل القطع ، وهكذا يعني أن القاعدة الثانية ستنفذ مرة واحدة فقط.

12. أمثلة على لغة برولوج (Swi-Prolog):



مثال 1: أكتب برنامج بلغة برولوج للحقائق الآتية:

Sami is the father of Ahmad

Abed is the father of Sami

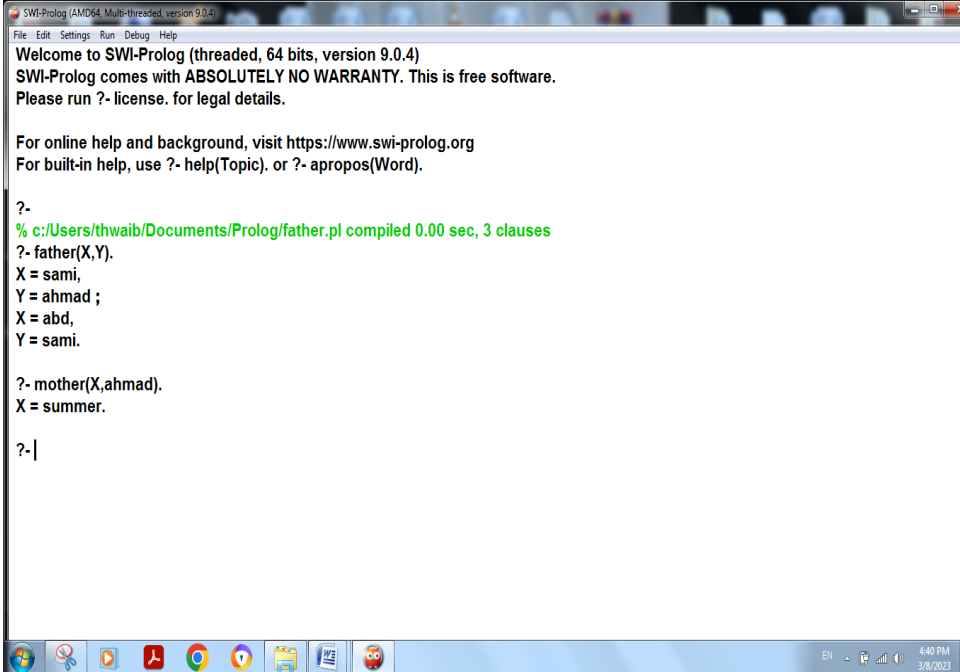
Summer is the mother of Ahmad

البرنامج بلغة برولوج مع تنفيذه كما يلي:

father(sami,ahmad).

father(abd,sami).

mother(summer,ahmad).



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/thwaib/Documents/Prolog/father.pl compiled 0.00 sec, 3 clauses
?- father(X,Y).
X = sami,
Y = ahmad ;
X = abd,
Y = sami.

?- mother(X,ahmad).
X = summer.

?-|
```

مثال 2 : بعض الإضافات على البرنامج السابق:

father(sami,ahmad).

father(abed,sami).

mother(summer,ahmad).

/*father(ahmad,dima).*/

parent(P1,P2):- father(P1,P2).

parent(P1,P2) :- mother(P1,P2).

grandparent(P1,P2):- parent(P3,P2),parent(P1,P3).

ancestor(P1,P2):- parent(P1,P2).

ancestor(P1,P2) :- parent(P1,P3),ancestor(P3,P2).

```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
| father(X,Y).
X = sami,
Y = ahmad ;
X = abed,
Y = sami.

?- parent(X,ahmad).
X = sami ;
X = summer.

?- grandparent(X,Y).
X = abed,
Y = ahmad .

?- ancestor(X,Y).
X = sami,
Y = ahmad ;
X = abed,
Y = sami

```

مثال 3 : كيفية التعامل مع القيم العددية:

```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

?- 5 = 4+1.
false.

?- 5 is 4+1.
true.

?- X is 5 , Y is X+2.
X = 5,
Y = 7.

?- X is 5 , Y is X+3 , Result is X+Y.
X = 5,
Y = 8,
Result = 13.

```

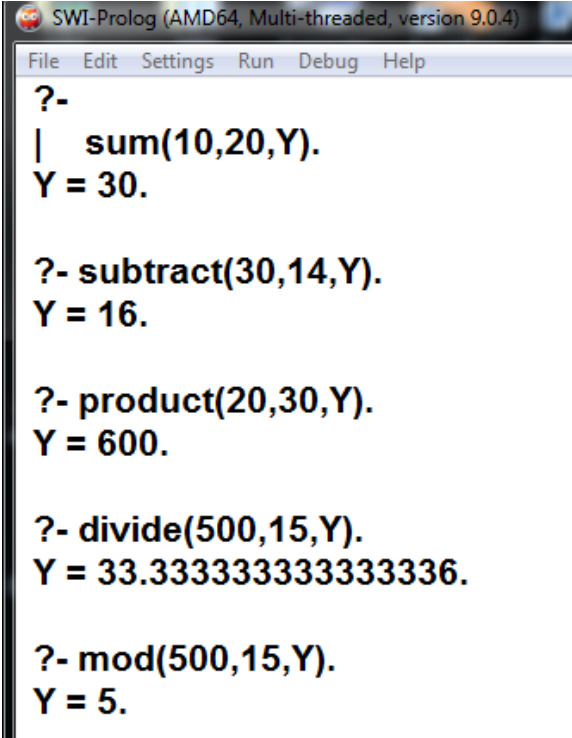
sum(X1,X2,Result):- Result is X1+X2.

subtract(X1,X2,Result) :- Result is X1-X2.

product(X1,X2,Result) :- Result is X1*X2.

divide(X1,X2,Result) :- Result is X1 / X2.

mod(X1,X2,Result) :- Result is X1 mod X2.



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?-
| sum(10,20,Y).
Y = 30.

?- subtract(30,14,Y).
Y = 16.

?- product(20,30,Y).
Y = 600.

?- divide(500,15,Y).
Y = 33.333333333333336.

?- mod(500,15,Y).
Y = 5.
```

مثال 5: برنامج بلغة برولوج لإيجاد مضروب العدد (factorial) N (N!) باستخدام الاستدعاء الذاتي (recursion).

البرنامج بلغة برولوج مع تنفيذه كما يلي:

fact(0,1).

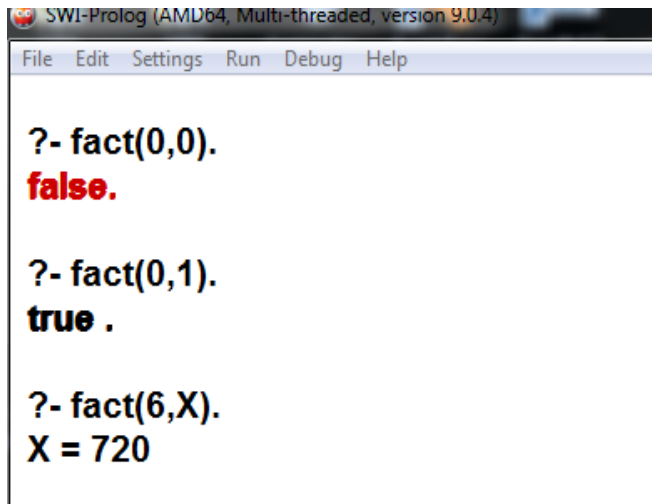
fact(N,X):- N>0,

 N1 is N-1,

 fact(N1,X1),

 X is X1*N.

تنفيذ البرنامج:



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

?- fact(0,0).
false.

?- fact(0,1).
true.

?- fact(6,X).
X = 720
```

مثال 6: برنامج بلغة برولوج لإيجاد مجموع الأعداد الصحيحة من 1 إلى N

segma(1,1).

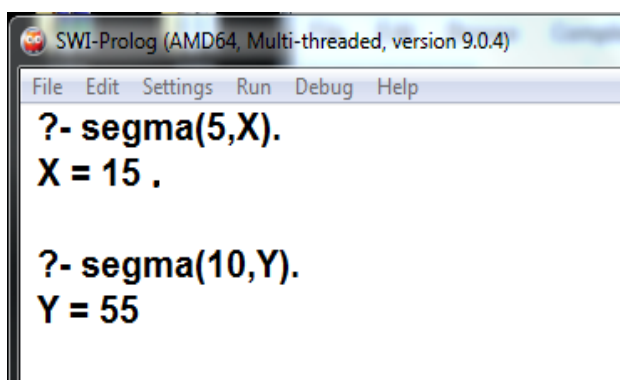
segma(N,Res):- N>1,

N1 is N-1,

segma(N1,Res1),

Res is Res1 + N.

تنفيذ البرنامج:



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

?- segma(5,X).
X = 15.

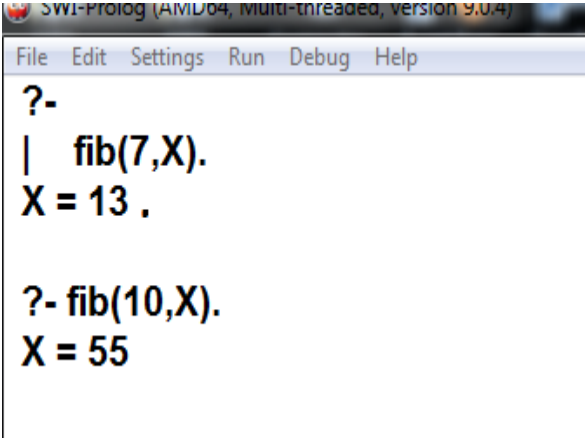
?- segma(10,Y).
Y = 55
```

مثال 7: برنامج بلغة برولوج لإيجاد العدد في متتالية فيبوناتشي (Fibonacci)

0 , 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , 144 ,

fib(0,0).
 fib(1,1).
 fib(N,Res):- N>1,
 N1 is N-1,
 N2 is N-2 ,
 fib(N1,Res1),
 fib(N2,Res2) ,
 Res is Res1 +Res2.

تنفيذ البرنامج:



```

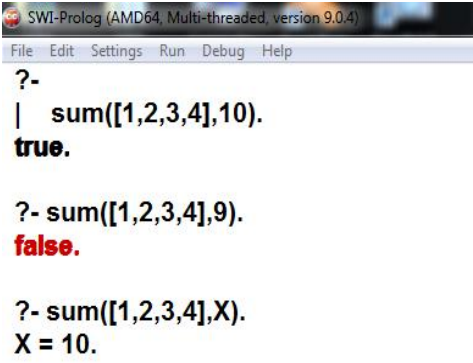
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?-
| fib(7,X).
X = 13 .

?- fib(10,X).
X = 55
  
```

مثال 8: برنامج بلغة برولوج لإيجاد مجموع القيم الرقمية في قائمة ما.

sum([],0).
 sum([X|T],S) :-
 sum(T,S1),
 S is S1 + X.

تنفيذ البرنامج:



```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?-
| sum([1,2,3,4],10).
true.

?- sum([1,2,3,4],9).
false.

?- sum([1,2,3,4],X).
X = 10.
  
```

مثال 9: برنامج بلغة برولوج لترتيب العناصر في قائمة بشكل عكسي وكذلك لدمج قائمتين في قائمة واحدة

reverse([],[]).

reverse([H|T],R):-

reverse(T,R1),

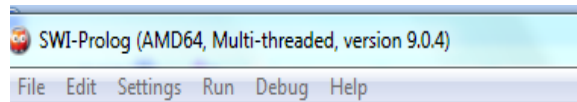
append(R1,[H],R).

append([],L,L).

append([H|T],L,[H|T1]):-

append(T,L,T1).

تنفيذ البرنامج:



?-

| reverse([1,2,3,4],X).

X = [4, 3, 2, 1].

?- append([1,2,3,4],[2,3,5,6]).

false.

?- append([1,2,3],[4,5,6],X).

X = [1, 2, 3, 4, 5, 6].

مثال 10: برنامج بلغة برولوج لإيجاد عدد العناصر في قائمة ما.

البرنامج بلغة برولوج مع تنفيذه كما يلي:

count([],0).

count([_|T],C):-

count(T,C1),

C is C1+1.


```

File Edit Settings Run Debug Help
?- count([2,5,7,9],X).
X = 4.

?- count([2,8,5],3).
true.

?- count([2,6,9],5).
false.

```

مثال 11: برنامج بلغة برولوج لتحديد إذا العنصر موجود في قائمة ما.

البرنامج بلغة برولوج مع تنفيذه كما يلي:

`member(X,[X|_]) .`

`member(X,[_H|T]) :- member(X,T).`

```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?- member(3,[4,7,8,9,3,1]).
true .

?- member(6,[7,4,9,3]).
false.

?- member(f,[r,t,s,f]).
true

```

مثال 12: برنامج بلغة برولوج لإيجاد مكعب العدد يوضح فيه عملية المدخلات والمخرجات

البرنامج بلغة برولوج مع تنفيذه كما يلي:

```

cube :-
    write('Write a number: '),
    read(Number),
    process(Number).

```

```

process(stop) :- !.
process(Number) :-
    C is Number * Number * Number,
    write('Cube of '),write(Number),write(': '),write(C),nl, cube.

```

```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?- cube.
Write a number: 3.
Cube of 3: 27
Write a number: |: 4.
Cube of 4: 64
Write a number: |:

```

مثال 13: برنامج بلغة برولوج لإيجاد مساحة دائرة ما.

circle :-

```

write('enter a radius of a circle:'),
    read(R),
    process(R).
process(stop) :- !.

```

process(R):-

```

    A is 3.14*R*R,
    write('the area of the circle of raduis = '),write(R),write(' : '),write(A),nl, circle.

```

```

SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?- circl.
enter a radius of a circle:5.
the area of the circle of raduis = 5 : 78.5
enter a radius of a circle:|: 10.
the area of the circle of raduis = 10 : 314.0
enter a radius of a circle:|: 8.
the area of the circle of raduis = 8 : 200.96
enter a radius of a circle:|: |

```

مثال 14: لكتابة برنامج بلغة برولوج يقرأ مجموعة أنصاف دوائر ويحسب مساحة كل دائرة ثم دائرة أخرى وهكذا ويتوقف فقط عندما يدخل المستخدم الأمر .Stop.

البرنامج بلغة برولوج مع تنفيذه كما يلي:

circle:-

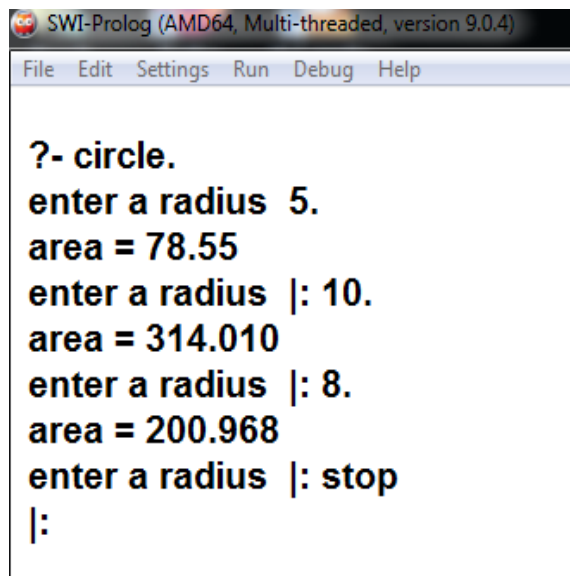
```
write('enter a radius '),
read(R),
compute(R).
```

circle.

compute(stop):- !.

compute(R):-

```
A is 3.14*R*R,
write('area = '),
write(A),write(R),nl,
circle.
```



```
SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help
?- circle.
enter a radius 5.
area = 78.55
enter a radius |: 10.
area = 314.010
enter a radius |: 8.
area = 200.968
enter a radius |: stop
|:
```

13. مثال تطبيقي على مسيطر المنطق الغامض Fuzzy Logic Controller باستخدام لغة البرمجة بايثون. -إعداد: د. بسام ترك-

سنقوم، عزيزي الطالب، في هذا المثال بإنشاء نظام تحكم (مسيطر) غامض Fuzzy Logic Controller يقوم بإيجاد سعر سيارة بناء على عمرها والمسافة المقطوعة مستخدماً القوانين التي سنزوده بها.

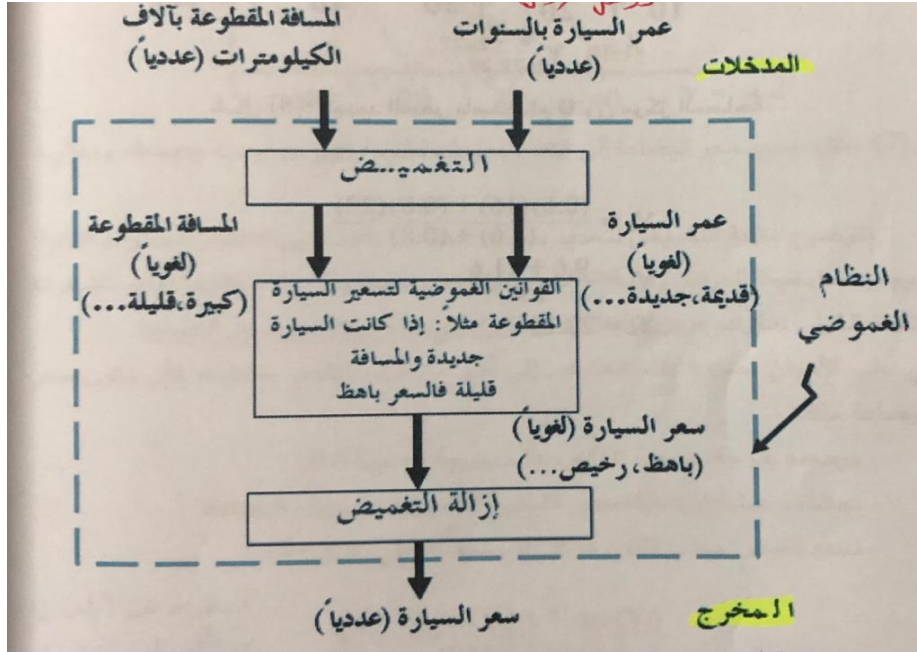
تطبيق المثال يتطلب مستلزمات بايثون التالية:

- تركيب برنامج بايثون (مثال python 3.8) من الموقع <https://www.python.org/downloads>
- تنزيل المكتبة skfuzzy باستخدام الامر `c:\>pip install -U scikit-fuzzy` حيث نقوم بعد ذلك باستدعائها من داخل برنامج بايثون باستخدام الامر `import skfuzzy as fuzz`
- ستكون بحاجة الى تنزيل مكتبة matplotlib الخاص بإظهار الرسومات من خلال الأمر `c:\>pip install -U matplotlib`

سنقوم بصياغة المشكلة كالتالي:

- المدخلات: وتسمى في مكتبة بايثون Antecednets
 - عمر السيارة
 - المجموعة الغامضة: قديمة، متوسطة العمر، جديدة أو poor, average, good
 - المسافة المقطوعة
 - المجموعة الغامضة: صغيرة، متوسطة، كبيرة أو poor, average, good
- المخرجات: وتسمى في مكتبة بايثون Consequents
 - السعر
 - المجموعة الغامضة: رخيص، متوسط، باهظ أو low, medium, high
- القوانين:
 - إذا كانت السيارة جديدة أو المسافة المقطوعة صغيرة فإن السعر باهظ
 - إذا كانت المسافة المقطوعة متوسطة فإن السعر متوسط
 - إذا كانت المسافة المقطوعة كبيرة أو ان السيارة قديمة فإن السعر رخيص
- تجربة النظام أو اختباره
 - إذا سألت المسيطر Controller ما الثمن المقدر لسيارة بالمدخلات التالية
 - العمر: 5 سنوات
 - المسافة المقطوعة: 10 (وتعني 10 الاف كيلومترات)
 - سوف يحسب النظام سعر السيارة حسب المدخلات السابقة ويعطيك النتيجة
 - 46.2 (أي 46.2 ألف دولار)

الشكل التالي مأخوذ من كتاب الذكاء الاصطناعي والأنظمة الخبيرة 1484 الوحدة السادسة وعنوانها المنطق الغامض Fuzzy Logic وهو يمثل المثال أعلاه.



أدناه كود البايثون:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

age = ctrl.Antecedent(np.arange(0, 25, 5), 'age')
distance = ctrl.Antecedent(np.arange(0, 250, 50), 'distance')
price = ctrl.Consequent(np.arange(0, 100, 25), 'price')

# Auto-membership function population is possible with .automf(3, 5, or 7)

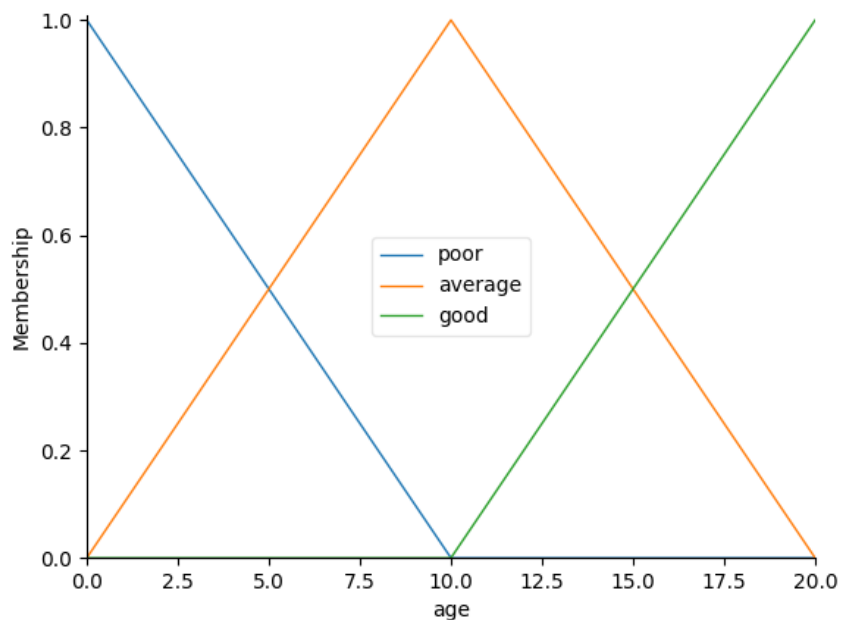
age.automf(3)
distance.automf(3)

# Custom membership functions can be built interactively with a familiar,
# Pythonic API

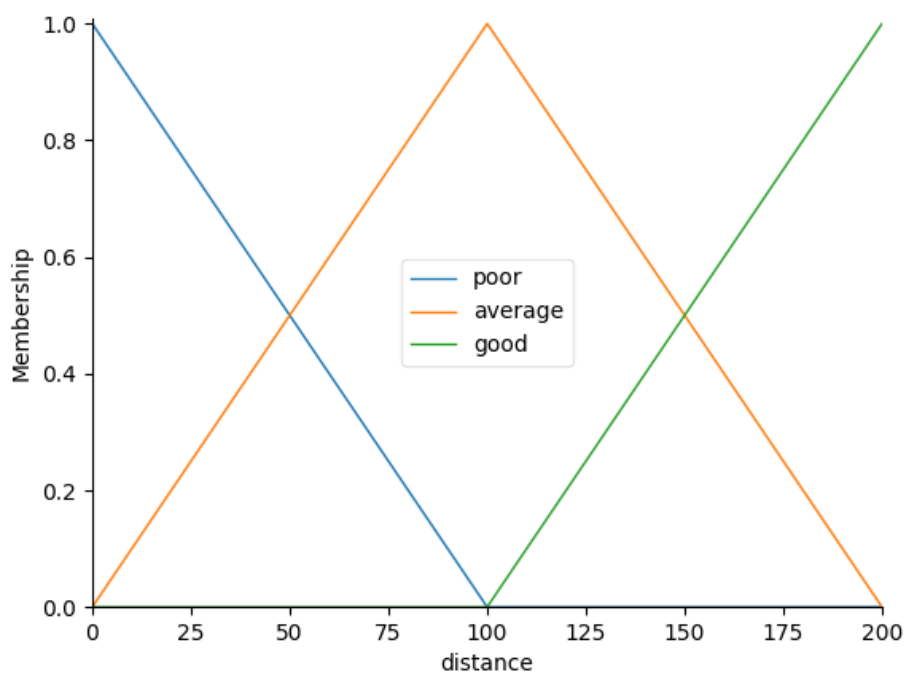
price['low'] = fuzz.trimf(price.universe, [0, 0, 25])
price['medium'] = fuzz.trimf(price.universe, [0, 25, 50])
price['high'] = fuzz.trimf(price.universe, [25, 50, 75])
```

بإمكانك الاطلاع على شكل الدالة المنتمية membership function من خلال الدالة view()

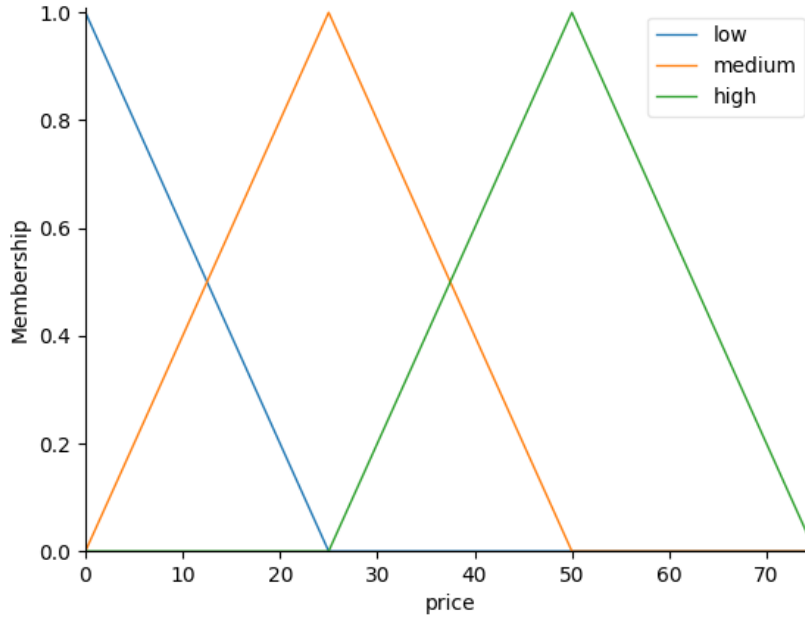
age.view()



distance.view()



price.view()



ثم نزود المسيطر Controller القوانين Fuzzy rules التالية:

1. إذا كانت السيارة جديدة أو المسافة المقطوعة صغيرة فإن السعر باهظ
2. إذا كانت المسافة المقطوعة متوسطة فإن السعر متوسط
3. إذا كانتا لمسافة المقطوعة كبيرة أو ان السيارة قديمة فإن السعر رخيص

من خلال الكود التالي:

```
#Fuzzy rules
rule1 = ctrl.Rule(age['poor'] | distance['poor'], price['high'])
rule2 = ctrl.Rule(distance['average'], price['medium'])
rule3 = ctrl.Rule(distance['good'] | age['good'], price['low'])
```

ثم نقوم بإنشاء/بناء المسيطر من خلال الأوامر التالية:

```
pricing_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
pricing = ctrl.ControlSystemSimulation(pricing_ctrl)
```

ثم نقوم بعمل محاكاة simulation لنظام التحكم (مسيطر المنطق الغامض) من خلال تزويد النظام بالمدخلات واستدعاء الدالة compute:

- العمر: 5 سنوات
- المسافة المقطوعة: 10 الاف كيلومترات

```
pricing.input['age'] = 5
pricing.input['distance'] = 10
pricing.compute()
```

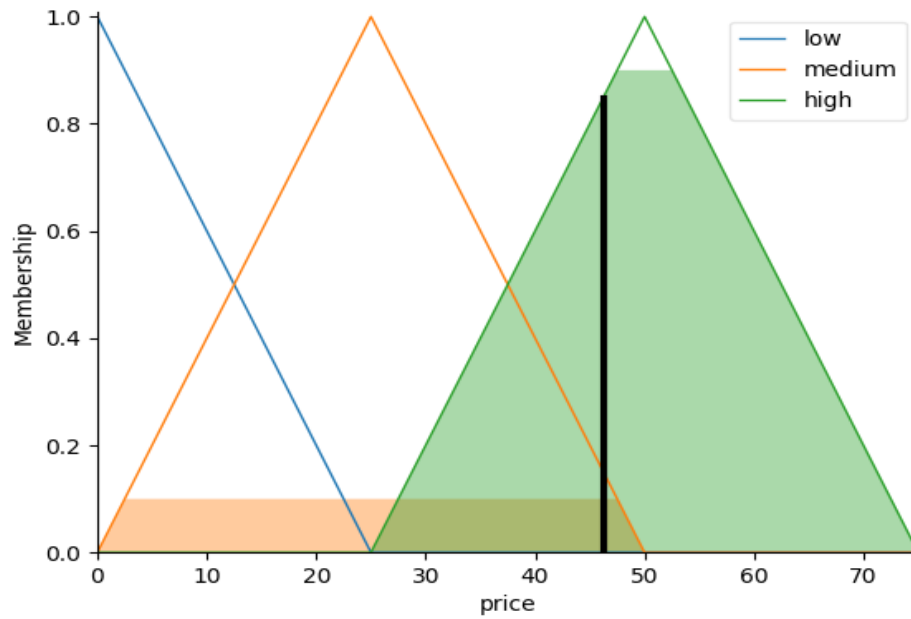
يمكنك طباعة النتيجة من خلال الامر:

```
print (pricing.output['price'])
```

يعطي النتيجة: 46.2

بإمكانك اظهار الرسمه الخاصة بالنتيجة من خلال الامر التالي:

```
price.view(sim=pricing)
```



أدناه الكود كاملا:

```
import numpy as np
```



```

import skfuzzy as fuzz
from skfuzzy import control as ctrl

age = ctrl.Antecedent(np.arange(0, 25, 5), 'age')
distance = ctrl.Antecedent(np.arange(0, 250, 50), 'distance')
price = ctrl.Consequent(np.arange(0, 100, 25), 'price')

# Auto-membership function population is possible with .automf(3, 5, or 7)
age.automf(3)
distance.automf(3)

price['low'] = fuzz.trimf(price.universe, [0, 0, 25])
price['medium'] = fuzz.trimf(price.universe, [0, 25, 50])
price['high'] = fuzz.trimf(price.universe, [25, 50, 75])

#To help understand what the membership looks like, use the view methods.

age.view()
distance.view()
price.view()

#Fuzzy rules
rule1 = ctrl.Rule(age['poor'] | distance['poor'], price['high'])
rule2 = ctrl.Rule(distance['average'], price['medium'])
rule3 = ctrl.Rule(distance['good'] | age['good'], price['low'])

#Control System Creation and Simulation
pricing_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
pricing = ctrl.ControlSystemSimulation(pricing_ctrl)

#suppose we rated the age 5 and the distance 10
pricing.input['age'] = 5
pricing.input['distance'] = 10
pricing.compute()

#we can view the computed result
print (pricing.output['price'])
#visualize the computed result
price.view(sim=pricing)

```

-انتهی-