مبادئ الذكاء الاصطناعي

الجلسة الأولى

2021- 2020

❖ مقــدمة :

اشتُقت كلمة prolog من كلمتي "Programming in Logic " أي البرمجة حسب المنطق وهي لغة برمجة مخصصة لتوجيه الحواسيب للقيام بمهام متعددة.
البرمجة في prolog تصريحية، أي يتم التصريح عن المعطيات والحقائق المتعلقة بالمسألة في قاعدة المعرفة، ثم نصمم القواعد لاستخراج المعلومات من قاعدة المعرفة حسب الحاجة.

إذاً : **prolog** هي لغة برمجة للحاسب تعتمد مبادئ المنطق وهذا ما يفسر اسمها.

من الجيد أن تتعلم **Prolog** لأنها تستخدم بشكل واسع في الأبحاث وخاصة الذكاء الصنعي،

وفي النظم الخبيرة، وقد تزايد استخدامها في التطبيقات التجارية.

لا تتكون لغة Prolog من تعليمات لتشكيل البرنامج بل هي عبارة عن مجموعة عبارات، هذه العبارات هي حقائق facts وقواعد rules تشكل قاعدة المعرفة. يقوم المفسر بتنفيذ عمليات على هذه الحقائق والقواعد استجابة لأهداف goals تعطيها له أنت.

❖ البيئة المستخدمة :

وهي SWI-Prolog والتي سنقوم بتحميلها من الموقع الرسمي عبر الرابط الموضح هنا :

https://www.swi-prolog.org/download/stable

بعد أن نقوم بتثبيت البيئة على الجهاز ، سنلاحظ أن الخط فيها صغير للغاية ، يمكننا تعديل ذلك عبر الخطوات التالية :

Settings → Font →

اختر نوع الخط : Times New Roman ، و الحجم 18 مثلاً

ومن ثم اضغط على OK، والآن يمكننا أن نبدأ . . .

❖ التعامل مع SWI Prolog :

التعامل مع هذه البيئة سهل للغاية ، سنقوم بدراسة عدد من الأمثلة مع التنفيذ مباشرةً لنكتشف من خلالها لغة prolog شيئاً فشيئاً.

لنبدأ مع أبسط العبارات التي يمكننا كتابتها ؛ سنلاحظ أن التنفيذ (الخرج) سيكون إما true بحال كون العبارة صحيحة بالنسبة لبرولوغ ، أو false بعكس ذلك.

- **ضع في ذهنك أن كل عبارة سوف تكتبها يجب أن تنتهي بنقطة .**
- **لرؤية نتيجة التنفيذ نضغط على زر الإدخال Enter.**

جرب تنفيذ العبارة التالية : **" لا تنسى وضع النقطة في نهاية العبارة ومن ثم اضغط Enter "**

```
?- 2=2.
```

ستجد أن Prolog يجيب ب **true** لأن 2=2 حقاً.

- جرب:

```
?- apple=apple.
true.
```

أجاب Prolog مجدداً ب **true** لأنه من الواضح أن كلمة apple الأولى مشابهة تماماً لكلمة apple الثانية.

apple هنا في Prolog ندعوها بالذرة atom أي مجموعة من الرموز الأبجدية. فالذرة في Prolog هي بمثابة السلسلة النصية String في لغات البرمجة الأخرى.

تعريف الذرة Atom في برولوغ :

وهي أي سلسلة نصية تبدأ بحرف صغير أو أي نص يوضع بين إشارتي تنصيص ' '

أو سلسلة من المحارف الخاصة مثل:@= =< .

- جرب :

```
?- 2+2=2+2.
```

أيضاً الخرج **true** ، لكن ماذا لو جربنا العبارة التالية ؟

```
?- 4=2+2.
false.
```

هذه المرة يعيد Prolog الجواب **false** وهذه إجابة غير متوقعة !

والسبب أنَّ :

الإشارة = في Prolog : تستخدم لفحص **التشابه** من الناحية النظرية، **وليس المساواة** الحسابية التي نعرفها في الرياضيات.

ولأن 4 لا تشبه 2+2 كانت النتيجة **false** ، بينما في العبارة السابقة 2+2 تشبه تماماً 2+2 لذلك كانت النتيجة **true**.

أمثلة إضافية :

ماهي نتيجة تنفيذ كل من العبارات التالية ؟ اختبر معلوماتك دون أن تنفذ ومن ثم تأكد من الخرج عبر التنفيذ :

```
?- one=1.
?-2+5=5+2.
?-false=false.
?-88=88.
?-9=3*3.
?-hello=heLLo.
?-10/2=5.
?-6-5=6-5.
```

❖ برولوغ والرياضيات :

وجدنا أن تنفيذ العبارة 4=2+2 سيعطي false ، حاول أن تستبدل إشارة = بالمعامل is أي جرب أن تنفذ العبارة التالية :

```
?- 4 is 2+2.
true.
```

هذه المرة سيكون الجواب **true** ، والسبب أن : **is تستخدم لحساب قيمة التعابير**.

و 4 هي بالفعل 2+2.

- لكن جرب أن تنفذ :

```
?- 2+2 is 4.
false.
```

هذا ما يجب أن تتنبه إليه هنا ، الكتابة بواسطة is ليست تبديلية إنما لها صياغة محددة وهي من اليسار إلى اليمين كالتالي :

العبارة التي تحوي العملية الحسابية >> is >> الناتج

بالطبع لا تقتصر العملية الحسابية على الجمع فقط ، بل إنّ جميع العمليات ممكنة (* / + -) ، فقط تذكر أنه ليكون تنفيذ العبارة **true** فإن العملية الحسابية تقع على **يمين** المعامل is ، ولا يمكن وضع عملية حسابية في الطرف الأيسر من is ، وبالطبع يجب أن تكون العبارة رياضياً صحيحة وإلا هي خاطئة بالأساس.

على سبيل المثال :

```
?- 8 is 2*4.
true.

?- 8 is 3+2.
false.

?- 2*4 is 8.
false.
```

حاول أن تفسر سبب التنفيذ لتتأكد من فهمك الجيد للفكرة

❖ الطباعة على الشاشة :

كل الأمثلة السابقة كانت نتيجة تنفيذها إما **true** أو **false** لكن الخرج في برولوغ لا ينحصر بهما فقط ، فمثلاً يمكننا استخدام المعلن write بالشكل : write(1atom) وذلك لطباعة ذرة واحدة فقط موجودة بين قوسين على الشاشة.
الخرج هنا سيكون : الذرة الموجودة بين قوسين تليها **true** والتي تفيد بإتمام التنفيذ بنجاح.
مثال على ذلك :

```
?- write(lilac).
lilac
true.
```

انتبه أنه ليس هنالك أي فراغ (space) ما بين write والقوسين بعدها ، وإن جربت أن تترك فراغاً سيكون ناتج التنفيذ : ERROR: Syntax error: Operator expected.

يمكن أن نقوم بكتابة عبارتي طباعة متتاليتين ولكن نضع بينهما فاصلة (,) وهي تعبر عن المعامل AND أي يجب تحقيق العبارتين على طرفي الفاصلة وبالتالي طباعة الذرتين بنفس الوقت ، وكما جرت العادة لا ننسى أن نضع نقطة في النهاية قبل الضغط على Enter.
مثال :

```
?- write(lilac),write(flowers).
lilacflowers
true.
```

بالرغم من أن برولوغ قد أنجز المهمة وأضاف **true** بعدها، إلا أنه يمكننا أن نلاحظ ما هي المشكلة بتنفيذ العبارة أعلاه ؛ الذرة الثانية flowers قد ظهرت ملاصقة للذرة الأولى lilac ....
كيف يمكننا معالجة ذلك ؟

بحال قد تبادر إلى ذهنك كتابة العبارة بالصيغة التالية:

write(lilac flowers).

تذكر أن المعلن write يقوم بطباعة **ذرة واحدة فقط** ضمنه ؛ لذلك لا يمكن أن نذكر بين قوسين ذرتين وبينهما فراغ !
جرب أن تنفذها ستلاحظ أن الخرج هو عبارة عن **ERROR**:

```
?- write(lilac flowers).
ERROR: Syntax error: Operator expected
ERROR: write(lilac
ERROR: ** here **
ERROR:  flowers) .
```

لدينا طريقتين لمعالجة ذلك :

- **إظهار الذرة الثانية بسطر جديد :** ويتم ذلك بأن نضع -قبل عبارة write الثانية- الحرفينnl؛ هو معلن قياسي آخر في Prolog ومعناه سطر جديد "new line".
  وبالطبع لا ننسى الفواصل بين العبارات، يكون ذلك كالتالي :

```
?- write(lilac),nl,write(flowers).
lilac
flowers
true.
```

- **إظهار الذرتين على سطر واحد وبوجود فراغ بينهما :** يتم ذلك عن طريق كتابة العبارة كما نريد أن تظهر **لكن ضمن إشارتي تنصيص " "** وليس هنالك فرق إن كانت إشارتي التنصيص 'مفردة' أم "مزدوجة" .

```
?- write('lilac flowers').
lilac flowers
true.

?- write("lilac flowers").
lilac flowers
true.
```

هل قمنا بذلك بطباعة ذرتين ضمن write ؟؟

بالتأكيد لا ؛ بحال عدنا لتعريف الذرة Atom لوجدنا أنها أيضاً تمثل **أي نص يوضع بين إشارتي تنصيص ' '** ، بالتالي مهما كان عدد الفراغات أو حالة المحارف لهذا النص **طالما أنه ضمن إشارتي تنصيص** فهو يمثل **ذرة وحيدة** وتطبع كما هي تماماً.

```
?- write(' Hello I am Prolog !!  ').
 Hello I am Prolog !!
true.

?- write(' now ... try to type your full name ').
 now ... try to type your full name
true.
```

كل من **write , nl** تعتبر كلمة من الكلمات الخاصة في **prolog** والتي تدعى **المعلنات**

وهي موجودة دائماً في قاعدة المعرفة

❖ إضافة حقائق إلى قاعدة المعرفة :

الحقيقة يتم التعبير عنها بالصيغة التالية :

اسم المعلن(وسيط).

تشبه عبارة الطباعة بواسطة المعلن write ، لكن الفرق هنا أن **ما بين قوسين** يمكن أن يكون :

- ✓ **ذرة** نريد إضافتها إلى قاعدة المعرفة أو حتى الاستعلام عن وجودها لدينا في قاعدة المعرفة.
- ✓ أو **متحول** نريد الحصول بواسطته على المعلومات الموجودة ضمن قاعدة المعرفة وتخص المعلن المذكور -سنتحدث عنه في فقرة لاحقة-

على سبيل المثال : .color(purple) تمثل حقيقة ؛ اسم المعلن فيها هو اللون color والوسيط هو ذرة تمثل شي ينتمي لهذا المعلن وهو اللون البنفسجي .

ويمكن أن يكون ما بين قوسين عدة وسطاء بينها فاصلة مثل .parent(alex, joe) أي أن alex هو أب لـ joe (سنكتفي في هذه الجلسة بوسيط واحد لنفهم الفكرة).

يمكننا **إضافة هذه الحقيقة إلى قاعدة المعرفة** عبر المعلن **assert( ).** ؛ حيث نكتب ضمن قوسيه ما نريد إضافته.

أي :

assert(color(purple)).

وعند الضغط على Enter سيرد البرنامج فقط ب **true** أي أنّ عملية الإضافة إلى قاعدة المعرفة تمت بنجاح.

يمكننا **الاستعلام عن وجود حقيقة ما في قاعدة المعرفة** من خلال كتابة اسم المعلن والوسيط الذي نريد أن نستعلم عن وجوده.

سنقوم بالاستعلام عن **اللون الأحمر** نلاحظ أن الخرج هو **false** ، المعلن أي color موجود لكن لم نضف هذا الوسيط red لقاعدة المعرفة.

وإن قمنا بالاستعلام عن **اللون البنفسجي** نلاحظ أن الخرج سيكون **true** أي أن هذا اللون موجود لدينا.

```
?- assert(color(purple)).
true.

?- color(red).
false.

?- color(purple).
true.
```

المعلومات التي أضفناها حتى الآن لقاعدة المعرفة :

color(purple).

فيما سبق قمنا بالاستعلام عن وجود وسيط **لمعلن تمت إضافته لقاعدة المعرفة** ، فكان الخرج إما true أو false ونقصد إما وسيط موجود أو وسيط غير موجود .

بحال استعلمنا عن وجود وسيط **لمعلن غير مضاف إلى قاعدة المعرفة** ، سيكون الخرج **ERROR**

على سبيل المثال :

```
?- student(hadi).
ERROR: Unknown procedure: student/1 (DWIM could not correct goal)
```

هنا بغض النظر عن الوسيط بين قوسين، فإن برولوغ يخبرنا أن المعلن أساساً لم يتم التعرف عليه من قبل !

- قم بإضافة الحقائق التالية لقاعدة المعرفة عبر assert كما تعلمنا :

```
color(blue).
student(ritta).
student(bill).
color(white).
student(alex).
fruit(apple).
```

المعلومات التي أضفناها حتى الآن لقاعدة المعرفة :

```
color(purple).
color(blue).
student(ritta).
student(bill).
color(white).
student(alex).
fruit(apple).
```

انتبه :

بحال قمت بإغلاق البرنامج وإعادة فتحته ...

سيتعين عليك إضافة الحقائق مجدداً لقاعدة المعرفة

لأنها لن تحتفظ بالمعلومات المضافة بعد إغلاق البرنامج.

❖ المعاملات and, or, not:

المعامل AND : يمكننا ربط عدة أشياء باستخدام and يتم التعبير عنه باستخدام (**الفاصلة ,**) بين العبارات.

عند الاستعلام عن عبارتين بينهما and يجب أن تكون كلتاهما true كي يظهر الخرج true ، بحال كانت واحدة منها ترد false فإن الخرج سيكون false حتماً.

سنقوم بالاستعلام عن وجود الحقائق التالية في قاعدة المعرفة لدينا :

```
?- color(purple),color(blue).
true.

?- color(purple),color(red).
false.

?- color(purple),student(ritta).
true.
```

هل لدينا في قاعدة المعرفة اللون البنفسجي **و** الأزرق ؟ **true**

هل لدينا في قاعدة المعرفة اللون البنفسجي **و** الأحمر ؟ لدينا البنفسجي ، الأحمر لا لذلك **false**

المعامل OR : يتم التعبير عنه باستخدام (**الفاصلة المنقوطة ;**) بين العبارات.

عند الاستعلام عن عبارتين بينهما or يكفي أن تكون إحداهما true ليكون الخرج هو true .

```
?- color(purple);color(blue).
true .

?- color(purple);color(red).
true .
```

هل لدينا في قاعدة المعرفة اللون البنفسجي **أو** الأحمر ؟ **true** لأن البنفسجي موجود.

المعامل NOT : يمكن عكس أي نتيجة باستخدام not ، يعبر عن نفسه بنفسه ف نكتب not قبل العبارة المراد نفي نتيجتها.

على سبيل المثال :

```
?- not(student(alex)).
false.

?- not(student(ahmad)).
true.
```

الطالب alex موجود في قاعدة المعرفة ، نتيجة الاستعلام عنه **بدون استخدام not** ستكون true ، وبعد استخدامها نقوم بنفي true وبالتالي يكون الخرج **false**.

❖ المتحولات :

يعرف المتحول في أية لغة برمجة وكذلك في الرياضيات بأنه كمية قد تأخذ أي قيمة. ويعبر عن المتحولات في Prolog بسلسلة من الرموز تبدأ بحرف كبير. أي سلسلة لا تبدأ بحرف كبير سيتم اعتبارها ثابتاً أو ذرة.

إذا المتحول هو سلسلة من المحارف تبدأ إما بحرف كبير أو بـ (_)، وفي حالة خاصة -سنأتي على ذكرها- لدينا متحول اسمه المتحول العشوائي يتم التعبير عنه باستخدام (_) فقط .

تمثيل المتحولات الفوري (دون الاستفادة من قاعدة المعرفة) :

هي العملية التي يتم فيها جعل المتحول يساوي قيمة ثابتة أي إعطائه قيمة ، ونتيجة التنفيذ (الخرج) تكون قيمة المتحولات المذكورة في الاستعلام (ناتج عملية حسابية ، سلسلة نصية ، ذرة ...).

مثال :

```
?- Y = 2+2.
Y = 2+2.

?- Z is 18/3.
Z = 6.
```

```
?- Fruite = orange.
Fruite = orange.

?- Message = " welcom to prolog ".
Message = " welcom to prolog ".
```

❖ الجبر مع لغة Prolog:

يمكن أن نكتب أكثر من متحول في العبارة، والخرج كما ذكرنا سابقاً سيكون قيمة هذه المتحولات على الترتيب.

```
?- Z=8 ,Y=4 , X is Z+Y.
Z = 8,
Y = 4,
X = 12.

?- Z=8 ,Y=4 , X=Z+Y.
Z = 8,
Y = 4,
X = 8+4.
```

وبحال جربنا أن نطبع قيمة متحول لم نسند له قيمة سيكون الخرج قيمة عشوائية (قيمة تعطى للمتحول وتختلف من تنفيذ إلى آخر).

```
?- write(X).
_9498
true.
```

❖ استخدام المتحولات في قاعدة المعرفة:

يمكن استخدام المتحولات للحصول على معلومات من قاعدة المعرفة ، مثلاً نريد الحصول على أسماء الطلاب لدينا في قاعدة المعرفة، فنكتب :

student(Z).

```
?- student(Z).
Z = ritta
```

تأخذ prolog وسيط الاستعلام وهو المتحول Z مثلاً وتفحص تطابقه مع الوسطاء التي تنتمي لهذا المعلن وموجودة في قاعدة المعرفة بالترتيب ، تقوم بإسناد أول قيمة مطابقة تم العثور عليها لذلك من خلال المطابقة أعطي المتحول القيمة ritta ، لكن هنالك طلاب آخرون غير ريتا لدينا .. !

لذلك تسأل Prolog إذا ما كنا نريد الاكتفاء بهذه النتيجة ؟ أو نريد البحث عن نتائج مطابقة أخرى؟

يمكن البحث عن نتائج مطابقة أخرى عبر الضغط على الفاصلة المنقوطة ، عندها يتم استبعاد القيمة المعطاة للمتحول Z وتسند إليه القيمة الجديدة المطابقة للاستعلام ، يمكن الاستمرار بالبحث عن نتائج مطابقة أخرى حتى الوصول إلى آخر نتيجة مطابقة من قاعدة المعرفة.

```
?- student(Z).
Z = ritta ;
Z = bill ;
Z = alex.

?-
```

نلاحظ أن نتيجة البحث قد كانت جميع أسماء الطلاب الموجودة لدينا في قاعدة المعرفة ، لو أردنا الاكتفاء عند الوصول لقيمة معينة فقط نضغط على Enter .

وبحال لم يكن في قاعدة المعرفة سوى قيمة واحدة ، فإن برولوغ لن ينتظر أو يسألك إن كنت تريد الاكتفاء أم لا ، فليس لديه قيم مطابقة أخرى.

كمثال على ذلك ، جرب أن تنفذ :

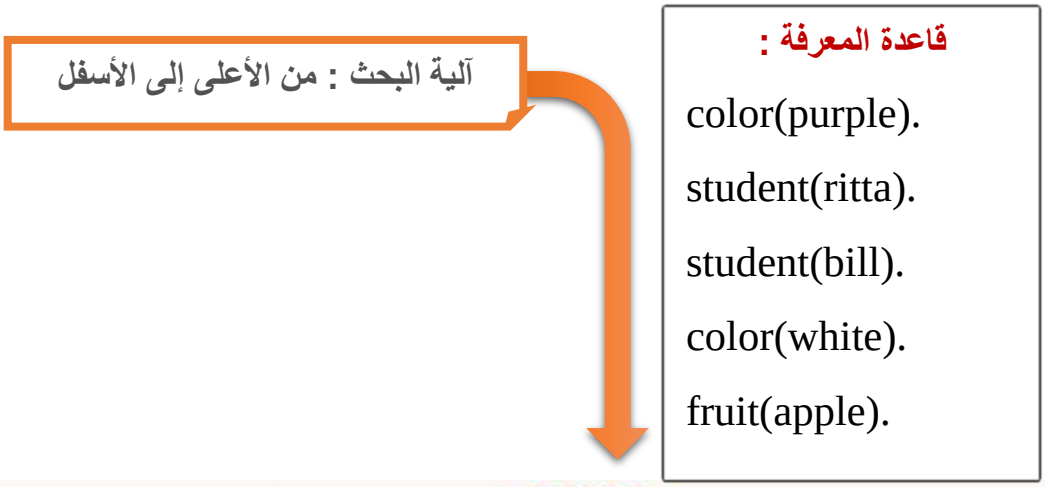
```
fruit(S).
```

❖ آلية عمل Prolog :

1. تتألف Prolog من قاعدة معرفة وآلية خاصة للبحث في هذه القاعدة.
2. عندما نقوم بالسؤال أو الاستعلام عن هدف ما ، فإن prolog تبحث في قاعدة المعرفة من الأعلى الى الأسفل FROM TOP TO BOTTOM وتطابق الهدف مع عناصر قاعدة المعرفة.
3. يمكن أن تعطى المتحولات قيماً ثابتة لتحقيق التطابق وإنجاح البحث.
4. إذا حصل التطابق، ينجح البحث أو يكون الهدف صحيحاً وتعاد قيم المتحولات التي تم العثور عليها.

تبقي Prolog مؤشراً حيث نجح البحث ومن الممكن أن يسأل عن المزيد من الحلول بالضغط على الفاصلة المنقوطة ، فتكمل البحث وتتابع حركة المؤشر نحو الأسفل بحثاً عن الحل التالي.

عندما لا تجد Prolog المزيد من الحلول ينتهي البحث.



❖ المتحولات العشوائية:

وهي المتحولات التي لا تهمنا قيمتها، مثلاً في قاعدة المعرفة لدينا التي تحوي معلومات عن الألوان قد نرغب بالسؤال ما اذا كان هناك أي ألوان في قاعدة المعرفة، لكن لايهمني ما هي الألوان إن وجدت.

في هذه الحالة يمكننا استخدام رمز underscore (_) والذي يدعى بالمتحول العشوائي.

ستبحث برولوغ عن قيمة للمتحول العشوائي وستجدها وسينجح الاستفسار لكنها لن ترد هذه القيمة، وبذلك نعرف أن هنالك ألواناً في قاعدة المعرفة ولكن دون تفاصيل عن ماهية هذه الألوان.

وكذلك الأمر بالنسبة للطلاب . .

```
?- color(_).
true .

?- student(_).
true .

?- boy(_).
ERROR: Unknown procedure: boy/1 (DWIM could not correct goal)
```

⏳ اختبر معلوماتك :

بناء على قاعدة المعرفة الموجودة لديك في هذه الجلسة ، ما الفرق بالتنفيذ بين العبارات التالية ؟ وما السبب ؟

color(x).
color(X).
color(_).

... انتهت الجلسة الأولى ...