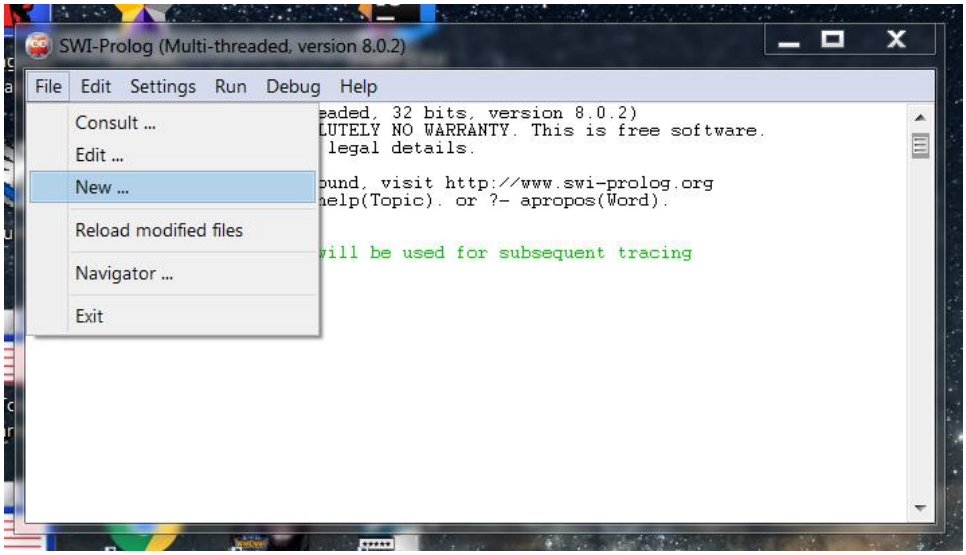
Laborator 14 - Paradigme de Programare

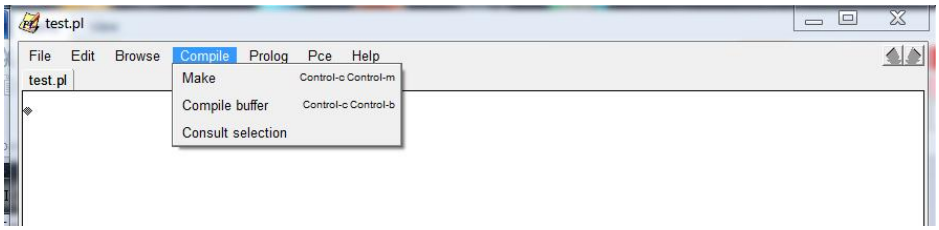
Programarea utilizând Prolog și interoperabilitatea cu Python-ul

Pentru a fi eficient în dezvoltarea rapidă a unei aplicații mixte între Prolog și Python este recomandat ca în primă fază programul să fie dezvoltat, testat și optimizat utilizând un interpretor de prolog (cum ar fi SWI Prolog). Abia apoi se poate trece la realizarea interoperabilității cu aplicațiile specifice Python-ului.

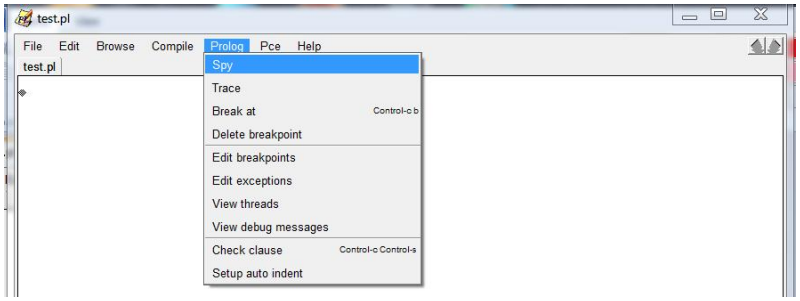
În utilizarea SWI Prolog sunt două moduri de dezvoltare. Executarea directă de la consolă care este utilă pentru teste rapide și editarea unui fișier care să conțină un modul prolog. În versiunea pentru Windows procesul este automatizat. Aici avem posibilitatea de a crea și edita un fișier direct prin intermediul Prolog SWI după cum se observă mai jos.



După ce am scris codul prolog în respectivul fișier avem la dispoziție comenzi pentru încărcarea acestuia în interpretor și nu numai.



În faza finală putem executa la acest nivel și echivalentul comenzilor directe pentru depanarea programului.



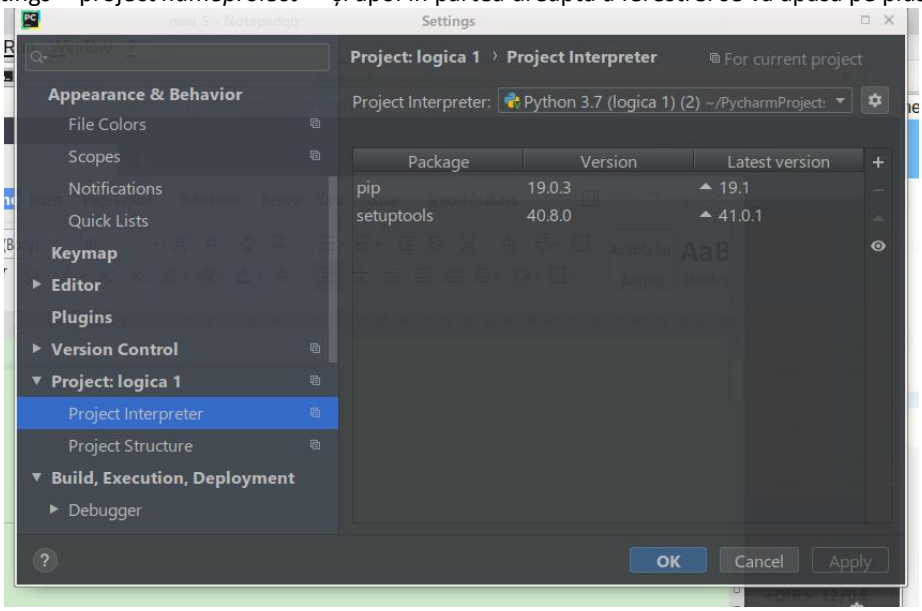
Pentru cazul SWI Prolog din linux acesta se lansează din consolă utilizator cu ajutorul comenzii swipl și astfel se obține prompterul interpretorului. Programul se editează utilizând un editor favorit și se

salvează în directorul unde ne aflam în momentul lansării în execuție a interpretorului. Pentru încărcarea programului se va da comanda [‘nume_program.pl’]. Dacă opțiunea de trace a fost activată (cu ajutorul comenzii trace.) atunci se va putea observa și maniera în care interpretorul realizează analiza programului în timpul încărcării. După aceasta se pot pune întrebări direct din prompter și apoi vizualiza rezultatul.
Informații ajutătoare despre limbajul prolog și interpretorul se pot consulta următoarele surse

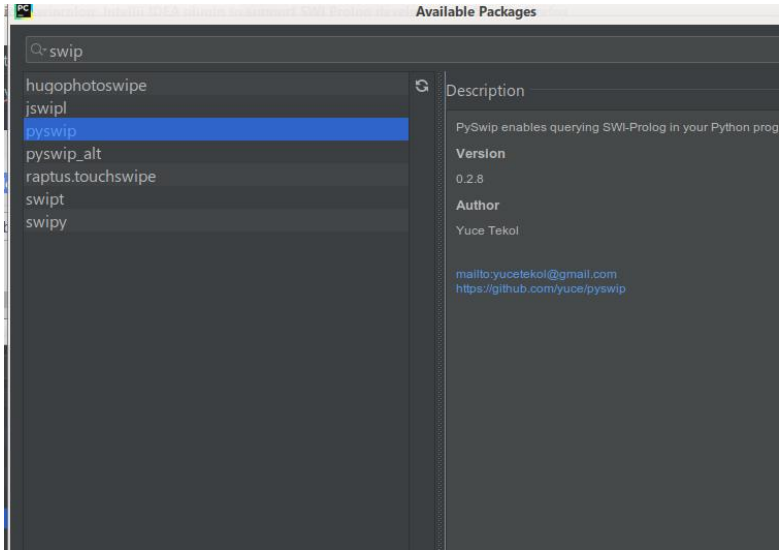
1. http://www.swi-prolog.org/pldoc/index.html
2. http://rigaux.org/language-study/syntax-across-languages-per-language/Prolog.html

În momentul în care avem partea de prolog funcțională putem trece la realizarea proiectului mixt în Pycharm. Pentru aceasta avem nevoie de următorii pași:

1. Creăm un nou proiect Pycharm
2. Creăm / Adăugam un fișier cu extensia .pl care va conține sursa programului testat în prima fază.
3. Creăm un fișier cu extensia .py.
4. Pentru a evita încărcarea nejustificată a sistemului de operare Pycharm ca și Intelij ne furnizează un mediu local virtual unde pot instala local orice am nevoie pentru o anumită aplicație. În acest mediu trebuie să instalam pyswip. Pentru aceasta vom naviga prin File -> Settings-> project *numeproiect* -> și apoi în partea dreaptă a ferestrei se va apăsa pe plus.



Acum se va deschide o fereastră de căutare în stânga unde vom scrie pyswip. Acum acesta se va adăuga (vezi în stânga jos - install packadge)



Acum se poate trece la testarea exemplelor din laborator.

Problema 1.
După cum am discutat la curs vom începe cu o problemă specifică definirii relațiilor prin fapte și reguli pentru ca mai apoi să aflăm utilizând prologul răspunsul la diverse întrebări specifice problemei descrise. Pentru aceasta vom alege un nou exemplu din cadrul problemei relațiilor din familie prezentat mai jos.

```
tatal(ilie,vasale).
tatal(popa,vasale).
tatal(george,ilie).
tatal(maria,ilie).
tatal(petru,popa).
tatal(vasilica,cobelea).
tatal(dana,pavel).

mama(george,vasilica).
mama(maria,vasilica).
mama(vasilica,diana).
mama(petru,dana).
mama(matcu,dana).
mama(dana,elena).
mama(ilie,elena).
mama(popa,elena).

barbat(george).
barbat(petru).
barbat(ilie).
barbat(popa).
barbat(vasale).
barbat(cobelea).
barbat(pavel).

femeie(maria).
femeie(matcu).
femeie(vasilica).
femeie(ileana).
femeie(dana).
femeie(diana).
femeie(elena).

sot(vasilica,ilie).
sot(dana,popa).
sot(ileana,petru).

unchi(george,popa).
unchi(petru,ilie).

frate(X,Y) :- tatal(X,A),tatal(Y,A),barbat(X),barbat(Y).
%   frate(X,Y) :- frate(Y,X),barbat(X),barbat(Y).

sora(X,Y) :- tatal(X,A),tatal(Y,A),femeie(Y).

matusa(X,Y) :- sot(Y,Z),frate(Z,W), Z \= W, tatal(X,W).

bunicul(X,Y) :-
   barbat(Y),
   tatal(X,W),
   tatal(W,Y).

bunicul(X,Y) :-
   barbat(Y),
   mama(X,W),
   tatal(W,Y).
```

```
bunica(X,Y) :-
   femeie(Y),
   tatal(X,W),
   mama(W,Y).

bunica(X,Y) :-
   femeie(Y),
   mama(X,W),
   mama(W,Y).
```

Pentru acest caz se cere ca la laborator să se afle răspunsul la următoarele întrebări:

1. **ce pereche de membri au o mătusă?**
2. **ce pereche de membri au un bunic?**
3. **Cine este sora lui George?**

Mai întâi se va dezvolta și testa programul utilizând SWI Prolog și abia apoi se va crea proiectul mixt și reexecuta testele utilizând Pycharm. Executați problema activând abilitățile de trace ale interpretorului pentru a putea urmări evoluția analizei.

Problema 2

Aici vom verifica funcționarea problemei clasice a maimuței care trebuie să urce pe o cutie (care poate fi deplasată) pentru a ajunge la banană.

```
muta( stare( inmijloc, pecutie, inmijloc, nuarebanana),
iabanana,
stare( inmijloc, pecutie, inmijloc, arebanana) ).
muta( stare( P, pepodea, P, H),
catarapecutie,
stare( P, pecutie, P, H) ).
muta( stare( P1, pepodea, P1, H),
impingeCutia( P1, P2),
stare( P2, pepodea, P2, H) ).
muta( stare( P1, pepodea, B, H),
mergedinP1inP2( P1, P2),
stare( P2, pepodea, B, H) ).
poatelua( Stare1)  :-
muta( Stare1, _, Stare2),
poatelua( Stare2),write(Stare1),write(" ->  "), write(Stare2),nl.
poatelua( stare( _, _, _, arebanana) ):-
write("Am luat banana"),nl.
```

O posibilă întrebare este dacă poate lua banana pornind din următoarea stare (lausa, pepodea, lafereastra, nuarebanana)

Se va dezvolta și testa programul utilizând SWI Prolog și abia apoi se va crea proiectul mixt și reexecuta testele utilizând Pycharm. Încercați să propuneți și alte poziții/situații de plecare pentru analiză. Executați problema activând abilitățile de trace ale interpretorului pentru a putea urmări evoluția analizei.

Tema pe acasă

1. La problema 1 extindeți modelul și propuneți noi întrebări.
2. Scrieți programul care va rezolva automat problema lui Einstein

Problema lui Einstein

Albert Einstein once posed a brain teaser that he predicted only 2% of the worlds population would be able to solve.

FACTS:

1. There are 5 houses in 5 different colours.
2. In each house lives a person with a different nationality.

3. These 5 owners drink a certain beverage, smoke a certain brand of cigarette and keep a certain pet.
4. No owners have the same pet, brand of cigaratte, or drink.

CLUES:
1. The Brit lives in a red house
2. The Swede keeps a dog
3. The Dane drinks tea
4. The green house is on the left of the white house.
5. The green house owner drinks coffee.
6. The person who smokes Pall Mall keeps birds.
7. The owner of the yellow house smokes Dunhill.
8. The man living in the house right in the center drinks milk
9. The Norwegian lives in the first house.
10. The man who smokes Blend lives next to the one who keeps cats
11. The man who keeps horses lives next to the man who smokes Dunhill
12. The owner who smokes Camel drinks beer
13. The German smokes Marlborough.
14. The Norwegian lives next to the blue house
15. The man who smokes Blend has a neighbour who drinks water.

The question is, who keeps the fish?