```
-- Beginner Prolog Cheat Sheet -- nj 2022 --
```

```
FORMAT
writef('... %w %w %w', [X, Y, Z])
```

```
EXECUTION EN MODE NON-INTERACTIF
f(X):-Y is X+1, writef('(%w %w)\n', [X,Y]).
go:-f(10).
```

```
$> swipl -s exemple1.pl -g go -t halt
(10 11)
```

```
OBTENIR PREMIERE REPONSE (MODE NON-INTERACTIF)
f([X]):-writef('(%w %w %w)\n', [X,X,X]).
go:-current_prolog_flag(argv, Argv),f(Argv).
```

```
$> swipl -s exemple2.pl -g go -t halt 10
(10 10 10)
```

```
OBTENIR PLUSIEURS REPONSES (MODE NON-INTERACTIF)
f([X]):-writef('(%w %w)\n', [X,X]),fail.
f([X]):-writef('(%w %w %w)\n', [X,X,X]).
go:-current_prolog_flag(argv, Argv),f(Argv).
```

```
$> swipl -s exemple3.pl -g go -t halt 10
(10 10)
(10 10 10)
```

```
RECURSIVITE
f(...):-a(...),f(...).        % sous-buts avant
f(...):-a(...),f(...),b(...). % sous-buts avant/après
f(...):-f(...),a(...).        % sous-buts après
f(...):-a(...);f(...).        % avec un OU avant
f(...):-f(...);a(...).        % avec un OU après
f(X):-f(X,0).                 % paramètres par défaut
```

```
ARITHMETIQUE
-Number is +Expr
+   addition
-   soustraction
*   multiplication
/   division flottante
//  division à l'entier inférieur
mod    modulo
rem    reste de la division
**     puissance
sqrt   racine-carrée
min    minimum
max    maximum
sign   signe
cos    cosinus
sin    sinus
tan    tangente
log    logarithme
exp    exponentielle
```

```
CASTS
integer(X)     en valeur entiere
float(X)       en valeur flottante
```

```
OPERATEURS BIT A BIT
/\     et bit à bit
\/     ou bit à bit
<<     décalage à gauche
>>     décalage à droite
```

```
OPERATEURS DE COMPARAISON
=     unification
==    égalité
\=    différent de
<     inférieur strict
>     supérieur strict
=<    inférieur ou égal
>=    supérieur ou égal
```

```
OPERATEURS DE COMPARAISON AVEC EVAL
=:=   égalité
=\=   différent de
```

```
PREDICATS DE TEST DE TYPE D'UN TERME
var(?term)       variable
nonvar(?term)    terme instancié
atom(?term)      atome
integer(?term)   entier
float(?term)     flottant
number(?term)    nombre
atomic(?term)    nombre ou atome
compound(?term)  terme complexe
callable(?term)  atome ou terme complexe
list(?term)      list
```

```
LIST
[a]     =  [a | []]
[a,b]   =  [a | [b]]
[a,b,c] =  [a | [b|[c]]]

length(?List, ?Length)
member(?Elem, ?List)
is_list(+Term)
append(+ListOfLists, ?List)
prefix(?Part, ?Whole)
nextto(?X, ?Y, ?List)
select(?Elem, ?List1, ?List2)
delete(+List1, @Elem, -List2)
nth0(?Index, ?List, ?Elem)
nth1(?Index, ?List, ?Elem)
last(?List, ?Last)
nth0(?N, ?List, ?Elem, ?Rest)
nth1(?N, ?List, ?Elem, ?Rest)
same_length(?List1, ?List2)
reverse(?List1, ?List2)
flatten(+NestedList, -FlatList)
max_member(-Max, +List)
min_member(-Min, +List)
sum_list(+List, -Sum)
max_list(+List:list(number), -Max:number)
min_list(+List:list(number), -Min:number)
numlist(+Low, +High, -List)
permutation(?Xs, ?Ys)
random_permutation(+List, -Permutation)
list_to_set(+List, ?Set)
is_set(@Set)
intersection(+Set1, +Set2, -Set3)
union(+Set1, +Set2, -Set3)
subset(+SubSet, +Set)
sort(+List, -Sorted)
msort(+List, -Sorted)
keysort(+List, -Sorted)
random(-R:float)
random(+L:int, +U:int, -R:int)
random(+L:float, +U:float, -R:float)
```