Starting the program:-

main/0:-

a predicate that welcomes the user to the game, then calls **build_kb/0** and **play/0** predicates.

```
%------------------------main-------------------------

main:-
    write('Welcome to Pro-Wordle'),nl,
    write('----------------------'),nl,nl,
    build_kb,
    play.
```

Building the Database:-

build_kb/0:-

a predicate that prompts the player to build the database by entering words and their categories on separate lines, returns true when the user enters done.

```
%-----------------------Build-----------------------
build_kb:-
    write('Please enter a word and its category on seperate lines:'),nl,
    read(Word),
    ((Word\==done,read(Category),assert(word(Word,Category)),build_kb);
    (Word==done,nl,write('Done building the words database...'),nl)).
```

Choosing a category:-

The program asks the player to choose the category of the word they want to guess by using the following predicates.

category_choice(Category):-

Shows the player the available categories and asks the player to choose one of them, the program takes the input and passes it to the **is_category/1** predicate that checks whether or not it's a valid input, If it is valid the player moves on to choose the length of the word he wants to guess, if it's not the program then calls the **category_choice_again/1** predicate.

```
%-------------------------Category--------------------

category_choice(Category):-
    nl,write('The available categories are: '),
    categories(L),
    write(L),nl,
    write('Choose a category:'),nl,
    read(C),
    ((is_category(C),Category=C);(\+is_category(C),category_choice_again(Category))).
```

category_choice_again(Category):-

called incase category_choice/1 fails and asks the user to enter another input.

```
category_choice_again(Category):-
    write('This category does not exist'),nl,
    write('Choose a category:'),nl,
    read(C),
    ((is_category(C),Category=C);(\+is_category(C),category_choice_again(Category))).

%
```

is_category(Category):- & categories(L):-

is_category/1: Checks whether or not the input category is valid or not.

categories/1: puts all the categories entered by the in the building phase in a list.

```
is_category(C):-
    word(_,C).

%-------------------------------------------------------

categories(L):-
    setof(Category,is_category(Category),L).
```

Choosing a length:-

The program asks the player to choose the length of the word they want to guess by using the following predicates.

length_choice(Length, C):-

Asks the player to input the number of letters in the word he wants to guess, then checks in the database if there are any words that have that number of letters using the **available_length/1** predicate, if true, it checks if this length is available in the category chosen earlier using **available_length_in_category/2** if it's also true, the program initializes a number of guesses equals to the number of letters + 1 and asks the player to start guessing, if either of them is false, the program moves to the **length_choice_again/2** predicate.

```
%-------------------------length--------------------

length_choice(Length,C):-
    nl,write('Choose a length:'),nl,
    read(L),
    ((available_length(L),available_length_in_category(L,C),Length=L,G is L+1,write('Game started. You have '),write(G),write(' guesses.'),nl);
    ((\+available_length(L);\+available_length_in_category(L,C)),length_choice_again(Length,C))).
%---------------------------
```

length_choice_again(Length, C):-

Called incase **length_choice/2** fails, tells the user length entered was not valid and calls the **length_choice/2** again.

```
%---------------------------
length_choice_again(Length,C):-
    write('There are no words of this length.'),nl,
    length_choice(Length,C).

%-----------------------------------------------
```

available_length(L):-

Checks if there is a word with a number of letters equal to the input length.

```
available_length(L):-
    word(W,_),
    atom_length(W,L).
```

available_length_in_category(L, C):-

Checks if there is a word in the specified category with a number of letters equal to the input length.

```
%-----------------------------
available_length_in_category(L,C):-
    word(W,C),
    atom_length(W,L).
```

Guessing the word:-

pick_word(W,L,C): -

picks a word from the database to be guessed with the length and category that the player specified.

```
pick_word(W,L,C):-
    word(W,C),
    atom_length(W,L).
```

word_choice(Word,L,G):-

prompts the player to guess a word with the length they choose in the **length_choice/1** predicate and checks if the length of the guessed word is equal to the length of the actual word, if it is, the code proceeds to check the letters, if its not, **word_choice_again/3** is called.

```
word_choice(Word,L,G):-
    nl,write('Enter a word composed of '), write(L),write(' letters:'),nl,
    read(W),
    ((atom_length(W,L),Word=W);(\+atom_length(W,L),word_choice_again(Word,L,G))).
%-----------------------------
```

word_choice_again(Word,L,G): -

called incase **word_choice/3** fails, tells the player that length of the word they entered is incorrect, then displays the remaining guesses and prompts the player to enter a valid word by calling **word_choice/3** .

```
word_choice_again(Word,L,G):-
    write('Word is not composed of '), write(L),write(' letters. Try again.'),nl,
    write('Remaining Guesses are '),write(G),nl,
    word_choice(Word,L,G).
```

Checking the correct letters:-

correct_letters_out(W2,W): -

Uses the pre-defined **atom_chars/2** predicate to get the letters of the guessed word and the actual word and places them in separate lists, those lists are then passed to the **correct_letters/3** predicate.

```
correct_letters_out(W2,W):-
    atom_chars(W2,W2List),
    atom_chars(W,WList),
    correct_letters(W2List,WList,CL),
    write('Correct letters are: '),write(CL),nl.
```

correct_letters(L1,L2,Cl):-

This is a recursive predicate that stops when first list is empty, The first list L1 represents the letters of the guessed word, The second list L2 represents the letters of the actual word, and the third list Cl represent the output list with letters common between the two words, it starts by checking if the first letter of the guessed word is a member of L2 using the pre-defined **member/3** predicate, if it succeeds, **remove_first/3** is used to remove the letter from L2 and letter added to Cl, Then **correct_leters/3** is called again to check the rest of L1,if **member/3** fails, **correct_letters/3** is then called skipping the first member in L1

```
correct_letters([],_,[]).
correct_letters([H|T],L2,[H|T1]):-
    member(H,L2),
    remove_first(H,L2,L2new),
    correct_letters(T,L2new,T1).
correct_letters([H|T],L2,CL):-
    \+member(H,L2),
    correct_letters(T,L2,CL).
```

remove_first(X,[H,T],[H,T1]):-

Removes the first occurrence of X in the list and outputs the rest of the list.

```
remove_first(X,[H|T],[H|T1]):-
    H\==X,
    remove_first(X,T,T1).
remove_first(X,[X|T],T).
```

Checking the correct positions:-

correct_positions_out(W2,W):-

Uses the pre-defined **atom_chars/2** predicate to get the letters of the guessed word and the actual word and places them in separate lists, those lists are then passed to the **correct_positions/3** predicate.

```
correct_positions_out(W2,W):-
    atom_chars(W2,W2List),
    atom_chars(W,WList),
    correct_positions(W2List,WList,CL),
    write('Correct letters in correct positions are: '),write(CL),nl.
```

correct_positions(L1,L2,L3):-

Takes the letters in same positions from the guessed word and actual word and places them in a list L3.

```
correct_positions([],_,[]).
correct_positions([H|T],[H|T1],[H|T2]):-
    correct_positions(T,T1,T2).
correct_positions([H|T],[H1|T1],CL):-
    H\==H1,
    correct_positions(T,T1,CL).
```

Playing the game:-

play/0:-

Lets the player choose the category using **category_choice/1**, then choose the length of the word within this category that the player wants to guess using **length_choice/2**, the program chooses a word to be guessed using within the given length and category using **pick_word/3** and initializes a guess variable with value equals length + 1, then proceeds with the **play_helper/3** predicate.

```
play:-
category_choice(C),
length_choice(L,C),
pick_word(W,L,C),
G is L+1,
play_helper(W,L,G).
%----------------------------
```

The play Helper Predicate:-

play_helper(W,L,G):-

Starts by calling the **word_choice/3** to get a guessed word from the player, if they guessed the correct word the program displays "You Won", if not, **correct_letters_out/3** and **correct_positions_out/3** are called to get which letters are correct and which positions are correct, Then the guesses are decremented and displayed and the predicate is called again with the new number of guesses. If the number of guesses is 1, The program calls word_choice/3 again compares the guess to the actual word, if its right, "You Won" is displayed, if it's wrong, "You Lost" is displayed.

```
play_helper(W,L,1):-
    word_choice(Word,L,1),
    ((Word==W,write('You Won!'));
    (Word\==W,write('You Lost!'))),!.
play_helper(W,L,G):-
    word_choice(Word,L,G),
    ((Word==W,write('You Won!'));
    (Word\==W,
    correct_letters_out(Word,W),
    correct_positions_out(Word,W),
    G1 is G-1,
    write('Remaining Guesses are '),write(G1),nl,
    play_helper(W,L,G1))).
```

The winning run screenshots:-

```
% /Users/omarahmed/Desktop/Prolog files/14.pl compiled 0.00 sec, 26 clauses
?- main.
Welcome to Pro-Wordle
---------------------

Please enter a word and its category on seperate lines:
|: rose.
|: flowers.
Please enter a word and its category on seperate lines:
|: sunflower.
|: flowers.
Please enter a word and its category on seperate lines:
|: jasmine.
|: flowers.
Please enter a word and its category on seperate lines:
|: math.
|: courses.
Please enter a word and its category on seperate lines:
|: cs.
|: courses.
Please enter a word and its category on seperate lines:
|: physics.
|: courses.
Please enter a word and its category on seperate lines:
|: spain.
|: countries.
Please enter a word and its category on seperate lines:
|: germany.
|: countries.
Please enter a word and its category on seperate lines:
|: brazil.
|: countries.
Please enter a word and its category on seperate lines:
|: done.

Done building the words database...

The available categories are: [countries,courses,flowers]
Choose a category:
|: countries.

Choose a length:
|: 2.
There are no words of this length.
```

```
|: 2.
There are no words of this length.

Choose a length:
|: 9.
There are no words of this length.

Choose a length:
|: 6.
Game started. You have 7 guesses.

Enter a word composed of 6 letters:
|: egypt.
Word is not composed of 6 letters. Try again.
Remaining Guesses are 7

Enter a word composed of 6 letters:
|: france.
Correct letters are: [r,a]
Correct letters in correct positions are: [r,a]
Remaining Guesses are 6

Enter a word composed of 6 letters:
|: greece.
Correct letters are: [r]
Correct letters in correct positions are: [r]
Remaining Guesses are 5

Enter a word composed of 6 letters:
|: canada.
Correct letters are: [a]
Correct letters in correct positions are: []
Remaining Guesses are 4

Enter a word composed of 6 letters:
|: colombia.
Word is not composed of 6 letters. Try again.
Remaining Guesses are 4

Enter a word composed of 6 letters:
|: brazil.
You Won!
true .

?-
```

The losing run screenshots:-

```
% /Users/omarahmed/Desktop/Prolog files/14.pl compiled 0.00 sec, 26 clauses
?- main.
Welcome to Pro-Wordle
---------------------

Please enter a word and its category on seperate lines:
|: math.
|: courses.
Please enter a word and its category on seperate lines:
|: physics.
|: courses.
Please enter a word and its category on seperate lines:
|: cs.
|: courses.
Please enter a word and its category on seperate lines:
|: arabic.
|: languages.
Please enter a word and its category on seperate lines:
|: english.
|: languages.
Please enter a word and its category on seperate lines:
|: french.
|: languages.
Please enter a word and its category on seperate lines:
|: deutsch.
|: languages.
Please enter a word and its category on seperate lines:
|: spain.
|: countries.
Please enter a word and its category on seperate lines:
|: france.
|: countries.
Please enter a word and its category on seperate lines:
|: belgium.
|: countries.
Please enter a word and its category on seperate lines:
|: done.

Done building the words database...

The available categories are: [countries,courses,languages]
Choose a category:
|: flowers.
This category does not exist
Choose a category:
```

```
|: flowers.
This category does not exist
Choose a category:
|: countries.

Choose a length:
|: 5.
Game started. You have 6 guesses.

Enter a word composed of 5 letters:
|: egypt.
Correct letters are: [p]
Correct letters in correct positions are: []
Remaining Guesses are 5

Enter a word composed of 5 letters:
|: japan.
Correct letters are: [a,p,n]
Correct letters in correct positions are: [n]
Remaining Guesses are 4

Enter a word composed of 5 letters:
|: sudan.
Correct letters are: [s,a,n]
Correct letters in correct positions are: [s,n]
Remaining Guesses are 3

Enter a word composed of 5 letters:
|: syria.
Correct letters are: [s,i,a]
Correct letters in correct positions are: [s,i]
Remaining Guesses are 2

Enter a word composed of 5 letters:
|: italy.
Correct letters are: [i,a]
Correct letters in correct positions are: [a]
Remaining Guesses are 1

Enter a word composed of 5 letters:
|: china.
You Lost!
true .

?- |
```