Provide example code and necessary elaborations for demonstrating the advantages of Dynamic Scoping in using Perl to implement the Advanced Tournament Dual game as compared to the corresponding codes in Python:

Dynamic Scoping could get a temporary change the value of a variable in some scopes without modifying it permanently, and the value is based on the call stack instead of define stack. This makes temporary use of values more convenient, and the logic of searching variables more distinct. Although its effect can be imitated by other methods, but need more work and not concise.
Thus, dynamic scoping may be useful in cases when we need our program to temporary modify a variable from other scope, and call it with functions in different scopes, with values depend on different condition, but don't want them influence each other and the original value.
For example, in the "update_fighter_properties_and_award_coins" part of the project, we can use local variable to do temporary changes of data in AdvancedFighter.pm, their values remain original after used

```
sub update_fighter_properties_and_award_coins{
    my $self = shift;
    my $fighter = shift;
    my $flag_defeat = shift;
    my $flag_rest = shift;

    local $AdvancedFighter::delta_attack = $AdvancedFighter::delta_attack;
    local $AdvancedFighter::delta_defense = $AdvancedFighter::delta_defense;
    local $AdvancedFighter::delta_speed = $AdvancedFighter::delta_speed;
    local $AdvancedFighter::coins_to_obtain = $AdvancedFighter::coins_to_obtain;

    my $consecutive_win = 0;
    my $consecutive_lose = 0;
    #calculate rest
    if ($flag_rest){
        $AdvancedFighter::coins_to_obtain    /= 2;
        local $AdvancedFighter::delta_attack = 1;
        local $AdvancedFighter::delta_defense = 1;
        local $AdvancedFighter::delta_speed = 1;
    }
    ...
```

But in Python, We must reset the value after use them whenever we modify them.

```
if flag_rest:
            AdvancedFighterFile.coins_to_obtain /= 2
            AdvancedFighterFile.coins_to_obtain =
int(AdvancedFighterFile.coins_to_obtain)
```

```
          AdvancedFighterFile.delta_attack = 1
          AdvancedFighterFile.delta_defense = 1
          AdvancedFighterFile.delta_speed = 1
...
       AdvancedFighterFile.coins_to_obtain = 20
       AdvancedFighterFile.delta_attack = -1
       AdvancedFighterFile.delta_defense = -1
       AdvancedFighterFile.delta_speed = -1
```

We the amount of variables or call times is large, its will be very inconvenient, and we must store the original value somewhere to reset it correctly.

Discuss the keyword local in Perl (e.g. its origin, its role in Perl, and real practical applications of it) and give your own opinions:

Local variable can set a temporary value of a global variable, it comes from global value, but do not influence the original one when we modify it. It is the key component of dynamic scoping in perl, as we can set the value different in different scope by local variable without influence the original one and each other.
Besides examples mentioned above, As the value is based on call stack, Local variable is also helpful when a function needs to get values from higher scope in the define stack beside the input, by using local value to modify the original value to mimic this process.