1. Difference between static scoping and dynamic scoping in application

Static scoping is commonly used in python, C, etc. . Here is a example of static scoping in *AdvancedTournament.py*.

```python
    def update_fighter_properties_and_award_coins(self, fighter, flag_defeat=False, flag_rest=False):
        original_coin = AdvancedFighterFile.coins_to_obtain
        if (flag_rest == True):
            AdvancedFighterFile.coins_to_obtain /= 2
            AdvancedFighterFile.delta_attack = 1
            AdvancedFighterFile.delta_defense = 1
            AdvancedFighterFile.delta_speed = 1
        else:
            if (len(fighter.history_record) == 3 and fighter.history_record[0] == fighter.history_record[1] and
                if (fighter.history_record[0] == "W"):
                    AdvancedFighterFile.coins_to_obtain *= 1.1
                    AdvancedFighterFile.delta_attack = 1
                    AdvancedFighterFile.delta_defense = -2
                    AdvancedFighterFile.delta_speed = 1
                    fighter.history_record.clear()
                else:
                    if (fighter.history_record[0] == "L"):
                        AdvancedFighterFile.coins_to_obtain *= 1.1
                        AdvancedFighterFile.delta_attack = -2
                        AdvancedFighterFile.delta_defense = 2
                        AdvancedFighterFile.delta_speed = 2
                        fighter.history_record.clear()
            if (flag_defeat == True):
                AdvancedFighterFile.delta_attack += 1
                AdvancedFighterFile.coins_to_obtain *= 2
        fighter.obtain_coins()
        fighter.update_properties()
        AdvancedFighterFile.coins_to_obtain = original_coin
        AdvancedFighterFile.delta_attack = -1
        AdvancedFighterFile.delta_defense = -1
        AdvancedFighterFile.delta_speed = -1
```

In `update_fighter_properties_and_award_coins()`, we can change the variable in the outer scope by importing. However, the effect of the assignment is permanent through out the entire program execution. In other words, if we want to reset the variable to the default value defined to before the execution of the function, we **must** assign it at the end (See line 58 to 61)

However, there is a totally different case in perl which supports dynamic scoping.

```perl
sub update_fighter_properties_and_award_coins{
    my ($self) = shift;
    my $fighter = $_[0];
    my $flag_defeat = $_[1];
    my $flag_rest = $_[2];
    local $AdvancedFighter::coins_to_obtain = $AdvancedFighter::coins_to_obtain;
    local $AdvancedFighter::delta_attack = -1;
    local $AdvancedFighter::delta_defense = -1;
    local $AdvancedFighter::delta_speed = -1;
    if ($flag_rest == 1)
    {
        $AdvancedFighter::coins_to_obtain /= 2;
        $AdvancedFighter::delta_attack = 1;
        $AdvancedFighter::delta_defense = 1;
        $AdvancedFighter::delta_speed = 1;
    }
    else
    {
        if ($#{$fighter->{"history_record"}} + 1 == 3 && @{$fighter->{"history_record"}}[0] eq @{$figh
        {
            if (@{$fighter->{"history_record"}}[0] eq "W")
            {
                $AdvancedFighter::coins_to_obtain *= 1.1;
                $AdvancedFighter::delta_attack = 1;
                $AdvancedFighter::delta_defense = -2;
                $AdvancedFighter::delta_speed = 1;
                $fighter->{"history_record"} = [];
            }
            else
            {
                if (@{$fighter->{"history_record"}}[0] eq "L")
                {
                    $AdvancedFighter::coins_to_obtain *= 1.1;
                    $AdvancedFighter::delta_attack = -2;
                    $AdvancedFighter::delta_defense = 2;
                    $AdvancedFighter::delta_speed = 2;
                    $fighter->{"history_record"} = [];
                }
            }
        }
        if ($flag_defeat == 1)
        {
            $AdvancedFighter::delta_attack += 1;
            $AdvancedFighter::coins_to_obtain *= 2;
        }
    }
    $fighter->obtain_coins();
    $fighter->update_properties();
}
```

Captured from *AnvancedTournament.pm*. There are two main points in the function. The first one is the `local` declaration (line 158 to 161) we need initiate the variable by `local`, then we can assign the variable, and use it outside the function. After that, we **don't need** to reset it to the original value. At short, here is the difference:
perl treat it as the variable assigned run locally, and expires when it ends. When we call the function, the function will use "its own provided value" first, so if we are in the inside function those global variable value could be modified by local variable. And python is another story, when we change the variable outside, it really changes it, thats why we need to reset it at the end. If we want to declare the same variable name in local scope and call the function that uses global scope variable with same name, the one declared in local scope will not be used.
The advantage of dynamic scoping is the program will looks clearer, as we do not need to reset it. And also, some complicated operation like passing parameters are not necessary, it makes programming more convenient.

2. Discussion the usage of `local`

In Perl, there is 3 types of variable declaration, `my, our, local`. `my` is the most common one used in Perl. `my` is usually used inside function, it cannot change the variable outside, it is similar to the variable declared in C++. `local` is also used inside function usually, but the major difference is `local` can change the variable outside the declaring function temperately, when the function ends, this will be reset, even we run that function again, or call the variable outside the function. Which is the major method to provide dynamic scoping in perl.
Here is a example: `my` is a mirror, which blocks the vision of outside, we do not know anything or output anything to outside of the room; `local` is like a one-way glass, which can process or even change the concepts outside the room, but when we leave the room, the changes will not be applied.