Question 1: Find out what the following terms mean: Function, Divide and Conquer, Code Reuse, Function, Header, Block, Mainline Logic, Abstraction, Local Variable, and Scope.

- Function → Reusable block of code that can be designed to perform specific tasks
- Divide and Conquer → This is a strategy, where bigger problems are broken up into smaller, less complex problems to tackle with ease
- Code Reuse → Reusing already existing code for new programs or problems to save time
- Header → The first part of a function defining its name and parameters
- Block → Multiple statements that are treated as a single unit together
- Mainline Logic → The main part of the execution that controls the overall sequence of the program
- Abstraction → This is when you hide small details to make it simpler for the user
- Local Variable → This is a variable that is defined in a function and only exists when that function is being called
- Scope → This is the area in the program where the variable is understood and accessed

Question 2: Briefly explain the four main benefits of designing a program with functions.

The first benefit is code reusability, as instead of having to type all of the code out again, you can just define a function and have it taken in multiple values. Functions also allow for simpler code, as people can understand detailed and well-named functions rather than a jumble of code (spaghetti-code) that isn't easy to read. If the code is well written and clear to understand, this leads to the third benefit of functions, easier debugging. This means that it is much easier to detect and remove errors from your code when there is much less to read. Finally, functions also support teamwork, as when the program is split into more functions, it is easier for people to individually focus on one task or function and then implement everything together rather than have all the code in one big file.

Question 3: Where is a local variable defined, and what is its scope?

Local variables are defined within a function, loop, or module, and are only accessible within the block of code it is created in. This means if I were to create a function named start() and in the function I defined my_var = 32, if I attempted to run print(my_var), it would result in an error as my_far is a local variable only defined within the start() function.

Question 4: Is it permissible to use the same local variable name in two different functions? Explain why or why not. Write an example program.

It is permissible to use the same local variable name in two different functions, as the variable's scope is limited to only its own function. This makes the computer see it as two completely different variables. Example:

```
def player_one_stats():
    score = 10
    print(f"Player One Score: {score}")

def player_two_stats():
    score = 25
    print(f"Player Two Score: {score}")

player_one_stats()
player_two_stats()
```

Question 5: Modify this program so that it uses mainline logic (i.e., add and call a main() function) to drive the program:

```
def greetings(name):
    print("Greetings, ", name)

def main():
    user_name = input("Please enter your name: ")
    greetings(user_name)

main()
```

Question 6: What happens if you try to change the value of name inside the greetings() function? Why do you think this happens?

If I were to change the value of name inside of the greetings() function, it would only change the value of name within that specific function. The name value in the main() function would still be the same as what the user typed. This happens because when creating a new value for name inside of greetings(), it has a local scope instead of a global scope.