Coding with and without Functions

Without Functions:

x=int(input("Enter month according to numbers[1 to 12]:"))  
if 1<=x<=3:  
 print("the given number is in the first quarter of the year.")  
elif 4<=x<=6:  
 print("the given number is in the second quarter of the year.")  
elif 7<=x<=9:  
 print("the given number is in the third quarter of the year.")  
elif 10<=x<=12:  
 print("the given number is in the fourth quarter of the year.")  
else:  
 print("the given number is not counted.")

With Functions:

def determine\_quarter(month):

if 1 <= month <= 3:

return "first"

elif 4 <= month <= 6:

return "second"

elif 7 <= month <= 9:

return "third"

elif 10 <= month <= 12:

return "fourth"

else:

return "not counted"

def main():

x = int(input("Enter month according to numbers [1 to 12]: "))

quarter = determine\_quarter(x)

if quarter != "not counted":

print("The given number is in the {quarter} quarter of the year.")

else:

print("The given number is not counted.")

main()

Summary:

The code snippets provided showcase two different approaches to achieving the same functionality, one without using functions and the other with functions.  
  
In the first approach without functions, the code is concise and straightforward. It takes user input, checks the input against predefined conditions, and prints a message based on the quarter of the year directly within the main script. While this method is simple and easy to understand for a small program like this, it lacks modularity. Modifying or extending the code in the future may become challenging, as all the logic is confined within a single block. Readability and maintainability could be compromised as the program grows.  
  
On the other hand, the second approach with functions introduces a higher level of organization and readability. By encapsulating the logic for determining the quarter within a separate function ('determine\_quarter'), the code becomes more modular. This modularity allows for better code reuse and easier maintenance. The main function takes care of user input and calls the quarter-determination function, providing a clear separation of concerns.  
  
In summary, while the first approach without functions is simpler for a small-scale program, the second approach with functions is preferable for larger and more complex projects. Functions enhance code modularity, promote reuse, and make the codebase more scalable and maintainable. While the second approach involves more lines of code, the benefits in terms of code organization and maintainability outweigh the additional complexity.

Navya Shah

11 – A.I [7]