Discussion-5: Module-5 Function and Strings

Discussion Topic:

Module 5: Discussion Forum
In Python, you have the opportunity either to use a predefined function/method or to write a user-defined function/method. In this discussion, provide at least three criteria that would be used to develop an appropriate method of your choice and the rationale behind the selection of these criteria. Then provide an example of your method declaration and return type. Actively participate in this discussion by providing constructive feedback on the criteria, rationales, and examples posted by your peers. Provide at least one reference to support your findings.

Build on something your classmate said
Explain why and how you see things differently
Ask a probing or clarifying question
Share an insight from having read your classmates' postings
Offer and support an opinion
Validate an idea with your own experience
Expand on your classmates' postings
Ask for evidence that supports the post.

PLEASE USE THIS QUESTION FOR PARTICIPATION, WHICH EXTENDS THE STANDARD ONE ABOVE:

Select only one of the following questions and offer a good, substantive reply. Some repetition is inevitable, but please aim to reply to a question that has not already been answered. Please include the question number you respond to, so we all can better understand your post, such as “#1”, “#2”, etc.:

1. In Python, you can either use a predefined function/method or write a user-defined function/method. In this discussion, provide at least three criteria that would be used to develop an appropriate method of your choice and the rationale behind selecting these criteria. Then give an example of your method declaration and return type.
2. What is a modular program? What are the advantages of writing modular programs? What are the disadvantages?
3. How does a Python Function that returns values to the main program differ from a Python Function that doesn't return any values to the main program? Give a Python example of each. When would you use one vs. the other?
4. Can functions play a role in code reuse? Why would code reuse be significant in computer programming? What do we gain from it if anything?
5. Modularization is an essential topic in computer programming, and correctly using this technique will lead to readable, efficient, and smaller programs. This technique has been used since the earliest days of computer programming, so it has a long history. In our Python incarnation of the technique, we deal with functions, such as those we have been using since day one: int(), input(), float(), etc., except that it is now WE the ones creating the functions. Let's analyze this topic in good depth and get our creativity going!

My Post:

Hello class,

For this discussion, I chose question #1:

> In Python, you can either use a predefined function/method or write a user-defined function/method. In this discussion, provide at least three criteria that would be used to develop an appropriate method of your choice and the rationale behind selecting these criteria. Then give an example of your method declaration and return type.

The three criteria that I consider when developing my functions/methods are as follows:

1. **Code reusability**: When a block of code needs to be repeated multiple times in a program, the block of code is a good candidate to be modularized into a reusable function. This promotes DRY (Don't Repeat Yourself) code, a principal in software development whose goal is to reduce repetition (Schafer, 2015). DRY.
   The rationale behind Reusable functions is to make the code more modular, readable, and maintainable. Changes only need to be made in one place.
2. **Task complexity**: If a task is made of many steps (code blocks), wrapping the task's steps in well-named functions makes the task more modular and hides complexity.
   The rationale is to reduce the tasks' readable complexity by hiding the details. In other words, the goal is to separate the "what" from the "how" making the code more understandable, readable, and maintainable. This is very useful for tasks with many steps and very complex logic.
3. **Code testability**: Encapsuling code blocks in functions with clear inputs and outputs makes testing large programs easier to test.
   The rationale is that isolating code into functions with clear inputs and outputs enhances the program's robustness and maintainability by facilitating easier testing of the codebase.

Considering these three criteria is key to creating a professional and high-quality codebase which, in the long run, saves time and reduces frustration not only for you but also for your 'coding teammates.'
This is particularly true when it comes to refactoring the code. "Refactoring, or code refactoring in full, is a systematic process of amending previously built source code, without introducing new functionalities or altering the fundamental workings of the subject software." (Slingerland, 2023)

Some of the actions that software development teams commonly take during refactoring include:

- Reducing the size of the code
- Restructuring confusing code into simpler code
- Cleaning up code to make it tidier
- Removing redundant, unused code and comments
- Doing away with unnecessary repetitions
- Combining similar code
- Creating reusable code
- Breaking up long functions into simpler, manageable bits

(Slingerland, 2023)

Here is an example a code without functions and a code with functions. The following program filters prime numbers from a list, and then checks if these squares are less than 50.

No functions:

```python
num_lst = [2, 3, 4, 5, 6, 7, 8, 9, 10]
squared_primes_less_than_50 = []

for n in num_lst:
    # Check if n is prime
    if n > 1:
        is_prime = True
        for i in range(2, int(n**0.5) + 1):
            if n % i == 0:
                is_prime = False
                break
        if is_prime:
            # Calculate square
            square = n**2
            # Check if square is less than 50
            if square < 50:
                squared_primes_less_than_50.append(square)

print(squared_primes_less_than_50)

```

With function

```python
1 usage
def is_prime(n: int) -> bool:
    '''
        Check if a number is prime.
        :param: n, integer to check for primality
        :return: Boolean indicating if the number is prime
    '''
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

1 usage
def filter_primes(num_lst: list[int]) -> list[int]:
    '''
        Filter prime numbers from a list.
        :param: num_lst, list of integers to filter
        :return: List of prime numbers from the input list
    '''
    return [n for n in num_lst if is_prime(n)]

1 usage
def square_numbers(num_lst: list[int]) -> list[int]:
    '''
        Square each number in a list.
        :param: num_lst, List of integers to square
        :return: List of squared integers
    '''
    return [n**2 for n in num_lst]

1 usage
def squares_less_than(num_lst: list[int], threshold: int) -> list[int]:
    '''
        Filter numbers less than a specified threshold from a list.
        :param: num_lst, list of integers to filter
        :param: threshold threshold value for filtering
        :return: List of numbers from the input list that are less than the threshold
    '''
    return [n for n in num_lst if n < threshold]

#--- Execute the program
1 usage
def main() -> None:
    '''
        Main function to execute the program logic.
        :param: None
        :return: None
    '''
    numbers = [2, 3, 4, 5, 6, 7, 8, 9, 10]
    primes = filter_primes(numbers)
    squared_primes = square_numbers(primes)
    result = squares_less_than(squared_primes, 50)
    print(result)

#--- Execute the program
if __name__ == "__main__": main()
```

Please see the function's docstrings and hints for information about the return types.

-Alex

References:

Schafer, C. (2015, June 16). *Programming terms: Dry (don't repeat yourself)*. YouTube. https://www.youtube.com/watch?v=IGH4-ZhfVDk&t=7s

Slingerland, C. (2023, December 5). What is refactoring? 5 techniques you can use to improve your software. CloudZero. https://www.cloudzero.com/blog/refactoring-techniques/