Discussion-7 Functions

Discussion Topic:

Working with Python, you could either use a predefined function/method or write a user-defined function/method. What are three criteria that you would use to determine whether to use predefined or user-defined functions/methods? Discuss your rationale in your post. Lastly, provide an example of your method declaration and return type. Actively participate in this discussion by providing constructive feedback and/or justifications on the criteria, rationales, and examples posted by your peers.

My Post:

Hello class.

In the context of Python programming, a function is a reusable block of code that is defined by a Function definition object that wraps around the block of code and can be called (executed) using a function name.
Functions can be standalone, just referred to as functions, or methods, which are functions that belong to a class. They can also be user-defined functions or Python built-in functions.

An example of a built-in function is: `print()`
An example of a built-in method is: `List.pop()`

Code sample:

```
print("Hello, world!") # -->  Hello, world!

my_list = [1, 5, 9]
scores = [10, 20, 30] # Creates a List object
last_item = scores.pop() # pop() is a method
print(last_item)  # --> 30
print(scores)     # --> [10, 20]
```

As the name indicates, user-defined functions are defined by programmers.
Python does separate function “declaration” from “definition”. For example, in C++, you can declare functions before you define them. Python does not do runtime method overloading by signature like C++ does. Below is an example of C++ code using a function declaration. Note that the function declaration is coded before the main method, and the function definitions are coded after the main function. The reason the functions are declared before the main function is to inform the compiler of the functions’ names, parameter types, and return types before it’s called.

Below is an example of a function definition, call, and declaration in C++:

```cpp
// Function declaration
int square(int x);

// Main Function
int main() {
    std::cout << square(7) << "\n";  // call before definition
    return 0;
}

// Function definition
int square(int x) {
    return x * x;
}
```

Same functionality in Python:

```python
# Declaration and Definition
def square(x: int) -> int:
    return x * x

# Main Function
def main() -> None:
    print(square(7))

# Main Function Call
if __name__ == "__main__":
    main()
```

or

```python
# Main Function Call
def main() -> None:
    print(square(7))

# Declaration and Definition
# Although this will not result in an error
```

```python
# It is not good practice to define a function after the main function definition
# However, defining a function after the main function is called (__if __name__ == "__main__")
# will generate an error
def square(x: int) -> int:
    return x * x

# Main Function Call
if __name__ == "__main__":
    main()
```

User-defined methods are part of user-defined classes; they work pretty much the same way as standalone functions. With the difference that their scope is bound to the class they belong to (instances of the classes).

On a side note: all functions are methods in Java - Java does not have standalone functions, meaning all functions need to be part of a class.

To determine whether to use predefined or user-defined functions/methods. If we defined predefined functions/methods as functions/methods from the standard library, built-in, or a third-party library. I use the following criteria:

1. I will check if a function/method that implements the exact functionality that I need already exists. For example, functions like `len()` or `zip()` are part of Python's built-in functions, and methods like `math.sqrt()` or `json.load()` are part of the standard library.
   Using these functions/methods allows me to save development time, as I do not need to reinvent the wheel. Moreover, these functions have been optimized and tested, reducing the risk of bugs from using my own functions, which need to be tested and probably optimized.
   On the other hand, I write a user-defined function if the functionality that it implements is unique, and it is not implemented by the standard library, built-in, or a tested/optimized third-party library.

2. For performance and known algorithms, I try to use predefined functions/methods, as many of these functions/methods are optimized algorithms and coded in C. For example, methods from modules like `statistics`, `heapq`, `bisect`, and `numpy`.
   I create my user-defines functions/methods, again if the exact functionality that I am looking for is not implemented by the standard library, built-in, or a tested/optimized third-party library. I also find myself writing my own functions/methods when I encounter performance issues with the pre-existing functions/methods, or when pre-existing functions/methods do not expose their algorithms -> they are black boxes that may create security issues.

3. I write my own user-define functions/methods when I am prioritizing code maintainability and clarity in a project, as using predefined functions/methods from a

third-party may inject unnecessary complexity, as many of them are not made for a specific project and are generalists. For example, functions and class names from third-party libraries are not domain-specific (project-specific).
On the other hand, well-known and widely used third-party libraries can reduce code lines and enhance code readability. Third-party libraries like `numpy` (numeric vectors), `pandas` (tabular data), `requests` (HTTP), `SQLAlchemy` (ORM), `Pydantic` (data validation), and `FastAPI` (web APIs) are widely used and known in the Python community, making projects using them easier to understand, read, and maintain for Python developers new to a project.
For my own functions/methods declaration and return type, I use PEP 8Links to an external site., 100-char lines, double quotes, trailing commas in multi-line, and type hintsLinks to an external site. (-> int return type), as well as Google-style docstringsLinks to an external site.. Below is an example of one of my function declarations and return types.

```
 ------------------------------------------------------------ validate_prompt_yes_or_no()
def validate_prompt_yes_or_no(prompt: str) -> bool: # Return type bool
    """Prompt the user until a valid yes/no answer is provided.

    Args:
        prompt: text to display before the "[Y/N]".

    Returns:
        True if the user selects yes ("y"/"yes"); False if no ("n"/"no").

    Examples:
        user input shown after [prompts]:

            >>> result = validate_prompt_yes_or_no("Continue?")
            Continue? [Y/N]: maybe
            Invalid input. Please enter 'Y' or 'N'.
            Continue? [Y/N]: y
            >>> result
            True
    """
    # Loop until a yes/no input is provided.
    while True:
        choice = input(f"{prompt} [Y/N]: ").strip().lower()
        # confirmation
        if choice in ("y", "yes"):
            return True
        # confirmation
        if choice in ("n", "no"):
```

```
            return False
        # invalid input message
        print(Fore.LIGHTRED_EX + "Invalid input. Please enter 'Y' or 'N'.")
# ------------------------------------------------------------------------------
```

-Alex