```
import json
from typing import Any, List

import inspect
from typing import Callable
from swarms.utils.formatter import formatter


def scrape_tool_func_docs(fn: Callable) -> str:
    """
    Scrape the docstrings and parameters of a function decorated with `tool` and return a formatted string.

    Args:
        fn (Callable): The function to scrape.

    Returns:
        str: A string containing the function's name, documentation string, and a list of its parameters. Each parameter is represented as a line containing the parameter's name, default value, and annotation.
    """
    try:
        # If the function is a tool, get the original function
        if hasattr(fn, "func"):
            fn = fn.func
```

```
    signature = inspect.signature(fn)
    parameters = []
    for name, param in signature.parameters.items():
        parameters.append(
            f"Name: {name}, Type:"
            f" {param.default if param.default is not param.empty else 'None'},"
            " Annotation:"
            f" {param.annotation if param.annotation is not param.empty else 'None'}"
        )
    parameters_str = "\n".join(parameters)
    return (
        f"Function: {fn.__name__}\nDocstring:"
        f" {inspect.getdoc(fn)}\nParameters:\n{parameters_str}"
    )
except Exception as error:
    (
        formatter.print_panel(
            f"Error scraping tool function docs {error} try"
            " optimizing your inputs with different"
            " variables and attempt once more."
        ),
    )

    raise error
```

```
def tool_find_by_name(tool_name: str, tools: List[Any]):
    """Find the tool by name"""
    for tool in tools:
        if tool.name == tool_name:
            return tool
    return None



def is_str_valid_func_output(
    output: str = None, function_map: callable = None
):
    """
     Check if the output is a valid JSON string, and if the function name in the JSON matches any name in the function map.

    Args:
        output (str): The output to check.
        function_map (dict): A dictionary mapping function names to functions.

    Returns:
        bool: True if the output is valid and the function name matches, False otherwise.
    """
    try:
        # Parse the output as JSON
        data = json.loads(output)
```

```
    # Check if the output matches the schema
    if (
        data.get("type") == "function"
        and "function" in data
        and "name" in data["function"]
    ):

        # Check if the function name matches any name in the function map
        function_name = data["function"]["name"]
        if function_name in function_map:
            return True

except json.JSONDecodeError:
    pass

return False
```