```
import platform
import subprocess


import pkg_resources
import psutil
import toml




def get_python_version():
    return platform.python_version()




def get_pip_version() -> str:
    """Get pip version


    Returns:
        str: The version of pip installed
    """
    try:
        pip_version = (
            subprocess.check_output(["pip", "--version"])
            .decode()
            .split()[1]
        )
    except Exception as e:
        pip_version = str(e)
    return pip_version



def get_swarms_verison() -> tuple[str, str]:
    """Get swarms version from both command line and package

    Returns:
        tuple[str, str]: A tuple containing (command line version, package version)
    """
    try:
        swarms_verison_cmd = (
            subprocess.check_output(["swarms", "--version"])
            .decode()
            .split()[1]
        )
    except Exception as e:
        swarms_verison_cmd = str(e)
    swarms_verison_pkg = pkg_resources.get_distribution(
        "swarms"
    ).version
    swarms_verison = swarms_verison_cmd, swarms_verison_pkg
    return swarms_verison



def get_os_version() -> str:
    """Get operating system version
    Returns:
        str: The operating system version and platform details
    """
    return platform.platform()


def get_cpu_info() -> str:
    """Get CPU information

    Returns:
        str: The processor information
    """
    return platform.processor()


def get_ram_info() -> str:
    """Get RAM information

    Returns:
        str: A formatted string containing total, used and free RAM in GB
    """
    vm = psutil.virtual_memory()
    used_ram_gb = vm.used / (1024**3)
    free_ram_gb = vm.free / (1024**3)
    total_ram_gb = vm.total / (1024**3)
    return (
        f"{total_ram_gb:.2f} GB, used: {used_ram_gb:.2f}, free:"
        f" {free_ram_gb:.2f}"
    )


def get_package_mismatches(file_path: str = "pyproject.toml") -> str:
    """Get package version mismatches between pyproject.toml and installed packages

    Args:
        file_path (str, optional): Path to pyproject.toml file. Defaults to "pyproject.toml".

    Returns:
        str: A formatted string containing package version mismatches
    """
    with open(file_path) as file:
        pyproject = toml.load(file)
    dependencies = pyproject["tool"]["poetry"]["dependencies"]
    dev_dependencies = pyproject["tool"]["poetry"]["group"]["dev"][
        "dependencies"
    ]
    dependencies.update(dev_dependencies)

    installed_packages = {
        pkg.key: pkg.version for pkg in pkg_resources.working_set
    }
    mismatches = []
    for package, version_info in dependencies.items():
        if isinstance(version_info, dict):
            version_info = version_info["version"]
        installed_version = installed_packages.get(package)
        if installed_version and version_info.startswith("^"):
            expected_version = version_info[1:]
            if not installed_version.startswith(expected_version):
                mismatches.append(
                    f"\t  {package}: Mismatch,"
                    f" pyproject.toml={expected_version},"
                    f" pip={installed_version}"
                )
        else:
            mismatches.append(f"\t  {package}: Not found in pip list")

    return "\n" + "\n".join(mismatches)


def system_info() -> dict[str, str]:
    """Get system information including Python, pip, OS, CPU and RAM details

    Returns:
        dict[str, str]: A dictionary containing system information
    """
```

```
return {
    "Python Version": get_python_version(),
    "Pip Version": get_pip_version(),
    # "Swarms Version": swarms_verison,
    "OS Version and Architecture": get_os_version(),
    "CPU Info": get_cpu_info(),
    "RAM Info": get_ram_info(),
}
```