```
"""
Package installation utility that checks for package existence and installs if needed.
Supports both pip and conda package managers.
"""

import importlib.util
import subprocess
import sys
from typing import Literal, Optional, Union
from swarms.utils.loguru_logger import initialize_logger
import pkg_resources


logger = initialize_logger("autocheckpackages")


def check_and_install_package(
    package_name: str,
    package_manager: Literal["pip", "conda"] = "pip",
    version: Optional[str] = None,
    upgrade: bool = False,
) -> bool:
    """
    Check if a package is installed and install it if not found.

    Args:
```

```
    package_name: Name of the package to check/install
    package_manager: Package manager to use ('pip' or 'conda')
    version: Specific version to install (optional)
    upgrade: Whether to upgrade the package if it exists

Returns:
    bool: True if package is available after check/install, False if installation failed

Raises:
    ValueError: If invalid package manager is specified
"""
try:
    # Check if package exists
    if package_manager == "pip":
        try:
            pkg_resources.get_distribution(package_name)
            if not upgrade:
                logger.info(
                    f"Package {package_name} is already installed"
                )
                return True
        except pkg_resources.DistributionNotFound:
            pass

        # Construct installation command
        cmd = [sys.executable, "-m", "pip", "install"]
```

```
    if upgrade:
        cmd.append("--upgrade")

    if version:
        cmd.append(f"{package_name}=={version}")
    else:
        cmd.append(package_name)

elif package_manager == "conda":
    # Check if conda is available
    try:
        subprocess.run(
            ["conda", "--version"],
            check=True,
            capture_output=True,
        )
    except (subprocess.CalledProcessError, FileNotFoundError):
        logger.error(
            "Conda is not available. Please install conda first."
        )
        return False

    # Construct conda command
    cmd = ["conda", "install", "-y"]
    if version:
        cmd.append(f"{package_name}={version}")
```

```
            else:
                cmd.append(package_name)
        else:
            raise ValueError(
                f"Invalid package manager: {package_manager}"
            )

        # Run installation
        logger.info(f"Installing {package_name}...")
        subprocess.run(
            cmd, check=True, capture_output=True, text=True
        )

        # Verify installation
        try:
            importlib.import_module(package_name)
            logger.info(f"Successfully installed {package_name}")
            return True
        except ImportError:
            logger.error(
                f"Package {package_name} was installed but cannot be imported"
            )
            return False

    except subprocess.CalledProcessError as e:
        logger.error(f"Failed to install {package_name}: {e.stderr}")
```

```
            return False
        except Exception as e:
            logger.error(
                f"Unexpected error while installing {package_name}: {str(e)}"
            )
            return False


def auto_check_and_download_package(
    packages: Union[str, list[str]],
    package_manager: Literal["pip", "conda"] = "pip",
    upgrade: bool = False,
) -> bool:
    """
    Ensure multiple packages are installed.

    Args:
        packages: Single package name or list of package names
        package_manager: Package manager to use ('pip' or 'conda')
        upgrade: Whether to upgrade existing packages

    Returns:
        bool: True if all packages are available, False if any installation failed
    """
    if isinstance(packages, str):
        packages = [packages]
```

```
success = True
for package in packages:
    if ":" in package:
        name, version = package.split(":")
        if not check_and_install_package(
            name, package_manager, version, upgrade
        ):
            success = False
    else:
        if not check_and_install_package(
            package, package_manager, upgrade=upgrade
        ):
            success = False

return success
```