

pyufunc: A Set of Utility Functions that Keep Python Sweet

Xiangyong Luo¹

1 Oak Ridge National Laboratory, United States

Summary

Welcome to pyufunc, your go-to Python package for a wide array of frequently used utility functions. Simplify your coding experience with this powerful toolkit, meticulously designed to enhance your productivity and streamline your development process. Whether you're a seasoned developer or just starting with Python, pyufunc provides a curated collection of utilities that cater to your everyday programming needs (Lott, 2018; Mertz, 2015).

Pyufunc aims to bring together the most commonly used utility functions from different libraries and provide them in a single, cohesive package. By consolidating utility functions from multiple sources, pyufunc simplifies the process of finding and integrating various utility libraries into your projects. It provides a centralized resource for accessing a diverse set of utility functions, ultimately saving time and effort.

Key Features




- Intuitive and Easy-to-Use:** Simplicity is at the core of pyufunc's design. Every utility function is thoughtfully documented, making it easy for developers of all skill levels to integrate them seamlessly into their projects. Whether you're working on a small script or a large-scale application, pyufunc enhances your code without adding complexity.
- Modularity and Extensibility:** pyufunc is structured with modularity in mind. Each utility function is a standalone entity, allowing you to cherry-pick the ones you need without introducing unnecessary dependencies. Furthermore, the package is designed to be extensible, making it effortless to contribute your own utility functions and enrich the community.
- Robust Collection of Utility Functions:** pyufunc offers a versatile assortment of utility functions, carefully crafted and thoroughly tested to meet industry standards. The package covers diverse domains, including data manipulation, file handling, string operations, mathematical functions, and much more.
- Regular Updates and Maintenance:** Our team is dedicated to providing regular updates, ensuring that pyufunc remains compatible with the latest Python releases and industry best practices. We actively welcome community feedback and continually refine the package to meet developers' evolving requirements.
- Time and Effort Savings:** You can avoid reinventing the wheel by leveraging pre-existing, widely used utility functions. This saves you time and effort in writing custom utility functions and allows you to focus on the core aspects of your project.

Let pyufunc take care of the repetitive tasks while you focus on building remarkable Python applications. Empower your projects with the efficiency and elegance that comes with pyufunc - your all-inclusive Python utility toolkit. Happy coding!

This manuscript has been authored in part by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the [DOE Public Access Plan](#)

DOI: [DOIunavailable](#)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Pending Editor](#) 

Reviewers:

- [@Pending Reviewers](#)

Submitted: N/A

Published: N/A

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

No dependencies will be installed in your coding environment unless you use functions that require specific dependencies. The function will automatically install the necessary packages when you use it.

If you discover useful functions that you believe should be included in the package for broader use, or if you have suggestions for additional utility functions, please share your comments here: [Issues](#) or pull the repository and commit functions.

Statement of need

In the Python development community, efficiency and productivity are often hindered by repetitive tasks and scattered utility functions across numerous libraries. Developers frequently encounter the inconvenience of integrating multiple packages, each providing different subsets of common utility functions, leading to complexity in dependency management and integration challenges.

`pyufunc` addresses this issue by consolidating the most frequently used utility functions into a single cohesive Python package. This centralized approach streamlines the coding experience, significantly reducing the overhead involved in identifying, installing, and managing various disparate utility libraries.

The necessity for a package like `pyufunc` arises from the following common scenarios faced by Python developers:

- **Fragmentation of utility functions:** Often, commonly used functions such as data manipulation, file handling, and mathematical operations are scattered across multiple libraries, each with its own dependencies, documentation standards, and installation processes.
- **Complexity in dependency management:** Integrating multiple small libraries often results in dependency conflicts or bloated virtual environments, making project setups cumbersome.
- **Repetitive development tasks:** Writing similar utility functions repeatedly in different projects consumes unnecessary time and resources.

By providing an intuitive, modular, and carefully maintained library, `pyufunc` simplifies these challenges. It ensures that Python developers have immediate access to a versatile set of robust, well-documented, and regularly updated utility functions.

The design of `pyufunc` emphasizes simplicity and modularity, enabling developers at all skill levels to integrate essential utilities effortlessly into their projects. Its modular structure facilitates easy addition or customization of individual functions, minimizing unnecessary dependencies and enhancing extensibility.

Utility functions are particularly beneficial in automated systems and workflows, where reliability and consistency are critical. In automation contexts, leveraging well-tested utility functions from `pyufunc` reduces the risk of errors and enhances operational stability, making automated pipelines more predictable, efficient, and maintainable.

Furthermore, `pyufunc` maintains a proactive approach towards compatibility and community engagement. Regular updates align the package with the latest Python standards and practices, while open community feedback mechanisms encourage continual improvement and expansion.

In conclusion, `pyufunc` fulfills a clear and practical need within the Python developer community, promoting efficiency, reducing redundant efforts, and enabling developers to focus on the core functionalities of their applications. By offering a streamlined, comprehensive toolkit

for common tasks, `pyufunc` significantly enhances the development process and fosters a productive coding environment.

In the Python development community, efficiency and productivity are often hindered by repetitive tasks and scattered utility functions across numerous libraries. Developers frequently encounter the inconvenience of integrating multiple packages, each providing different subsets of common utility functions, leading to complexity in dependency management and integration challenges (Lott, 2018; Mertz, 2015).

`pyufunc` addresses this issue by consolidating the most frequently used utility functions into a single cohesive Python package. This centralized approach streamlines the coding experience, significantly reducing the overhead involved in identifying, installing, and managing various disparate utility libraries.

The necessity for a package like `pyufunc` arises from the following common scenarios faced by Python developers:

- **Fragmentation of utility functions:** Often, commonly used functions such as data manipulation, file handling, and mathematical operations are scattered across multiple libraries, each with its own dependencies, documentation standards, and installation processes.
- **Complexity in dependency management:** Integrating multiple small libraries often results in dependency conflicts or bloated virtual environments, making project setups cumbersome.
- **Repetitive development tasks:** Writing similar utility functions repeatedly in different projects consumes unnecessary time and resources.

By providing an intuitive, modular, and carefully maintained library, `pyufunc` simplifies these challenges. It ensures that Python developers have immediate access to a versatile set of robust, well-documented, and regularly updated utility functions.

The design of `pyufunc` emphasizes simplicity and modularity, enabling developers at all skill levels to integrate essential utilities effortlessly into their projects. Its modular structure facilitates easy addition or customization of individual functions, minimizing unnecessary dependencies and enhancing extensibility (DeGrandis & Valetto, 2009; Walsh et al., 2004).

Furthermore, `pyufunc` maintains a proactive approach towards compatibility and community engagement. Regular updates align the package with the latest Python standards and practices, while open community feedback mechanisms encourage continual improvement and expansion.

In conclusion, `pyufunc` fulfills a clear and practical need within the Python developer community, promoting efficiency, reducing redundant efforts, and enabling developers to focus on the core functionalities of their applications. By offering a streamlined, comprehensive toolkit for common tasks, `pyufunc` significantly enhances the development process and fosters a productive coding environment.

Existing Utility Functions Categorized by Functionality

This document serves as a curated compendium of existing utility functions, meticulously organized by keywords to facilitate ease of navigation and application for developers across various disciplines. By categorizing these functions, we aim to provide a structured overview that not only simplifies the discovery process but also encourages the exploration of new methods and techniques that may have been previously overlooked. This categorization is intended to serve as a bridge, connecting developers with the tools they need to optimize their code, improve functionality, and innovate within their projects.

The categories outlined in this document span a wide range of functionalities, each category is accompanied by a brief description, followed by a list of utility functions that fall under its

umbrella, this comprehensive approach aims to arm developers with a robust toolkit, enabling them to select the most appropriate utility functions for their specific needs. Whether you are working on a complex application requiring advanced data manipulation or a simple project needing basic string operations, this guide endeavors to provide a valuable resource that enhances your development process and leads to more efficient, effective, and elegant coding solutions.

- [utility_function_by_category.md](#)

Existing Utility Functions Categorized by Keywords

This document is designed to serve as an invaluable resource for developers, offering an extensive list of existing utility functions organized by keywords. The use of keywords for organization purposes aims to streamline the search process, enabling developers to quickly and efficiently find the specific functions they need to enhance their projects. Utility functions play a crucial role in software development, providing pre-built solutions to common problems and tasks, thereby saving time and reducing the complexity of coding from scratch. By presenting these functions in a keyword-centric format, we facilitate a more intuitive and user-friendly approach to accessing a vast repository of tools, ensuring that developers can leverage the full potential of utility functions to optimize their code, improve performance, and innovate within their applications.

This methodical approach empowers developers to efficiently identify the functions that best match their current requirements, thereby enhancing their coding workflow and productivity. Whether tackling complex algorithmic challenges or implementing basic functionality, this guide aims to be an essential companion, fostering a deeper understanding and more effective use of utility functions in software development projects.

- [utility_function_by_keyword.md](#)

Comprehensive Review of Existing Python Utility Function Packages

In this section, you will find a comprehensive review of existing utility function packages. Before delving into the evaluations and insights, the author wishes to express sincere gratitude to all the developers behind these packages. Your contributions to the open-source community are invaluable, and it is with great appreciation that we acknowledge your efforts and dedication.

As part of our review, we have carefully selected and compiled useful utility functions from these packages into our own offering: `pyufunc` (Python Utility Functions), aimed at broadening their usage. Our goal with `pyufunc` is to collect all sorts of useful utility functions together to boost the efficiency of developers. If you need to use utility functions in your project, all you need is `pyufunc`.

This initiative is designed to streamline your development process, ensuring that you have access to a comprehensive toolkit that addresses a wide range of needs and scenarios. Through `pyufunc`, we aspire to provide a one-stop solution that encapsulates the best practices and functionalities from the open-source community, making it easier for developers to achieve their objectives with greater speed and efficiency.

Furthermore, we recognize the importance of proper usage and attribution of the utility functions we have integrated into `pyufunc`. If any package developer finds their utility function has been used improperly, we encourage you to reach out to the `pyufunc` developers for further discussion. We are committed to maintaining a respectful and collaborative relationship with the original developers, ensuring that all contributions are appropriately acknowledged and utilized within the bounds of open-source licenses and community norms. Your feedback and insights are crucial to us, as they help in refining `pyufunc` to better serve the open-source community.

Package Name	Description	Cite
pyutil	a library of useful Python functions and classes	tpltnt (2014)
PyHelpers	An open-source toolkit for facilitating Pythonusers' data manipulation tasks	Fu (2020)
psutil	Cross-platform lib for process and system monitoring in Python	Rodola (2009)
pyutilator	open source python package comprising of decorators that can be used for utility operations	Prince (2023)
pyutils	Python utilities	Gasch (2022)
common-pyutil	Bunch of common utility functions I've used in various projects.This package provides a uniform interface to them.	Badola (2019)
pyutl	functions and utilities to recycle code	Gómez (2019)
pyutilities	Useful utilities for python 3.10+	Dmitrii (2021)
dry-pyutils	This package's goal is to offer a set of utility methodsI end up using in a lot of projects.	Monteiro (2021)
pripy-utils	Python utilities	jaysen.lin (2025)
imutils	A series of convenience functions to make basic imageprocessing operations such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and Python.	PylImageSear (2015)
dateutil	Useful extensions to the standard Python datetime features	dateutil (2015)
nb_utils	python utility functions	Boyane (2020)
Python-Charmers	A collection of useful python programs.	iwasaki (2020)
python-in-action	python crawler in action	PrinceCheng (2021)
tbm13-utils	Python utils made for personal use on my projects.	Como (2023)
...

Acknowledgements

This open-source package is supported by National Science Foundation under grant no. TIP-2303748 titled, “[POSE: Phase II: CONNECT: Consortium of Open-source Planning Models for Next-generation Equitable and Efficient Communities and Transportation](#)”

References

- Badola, A. (2019). *Bunch of common utility functions i've used in various projects. This package provides a uniform interface to them.* GitHub. <https://github.com/akshaybadola/common-pyutil>
- Boyane, N. (2020). *Python utility functions.* GitHub. https://github.com/Nivratti/nb_utils
- Como, M. T. D. (2023). *Python utils made for personal use on my projects.* GitHub. <https://github.com/TBM13/tbm13-utils>

- dateutil. (2015). *Useful extensions to the standard python datetime features*. GitHub. <https://github.com/dateutil/dateutil>
- DeGrandis, P., & Valetto, G. (2009). Elicitation and utilization of application-level utility functions. *Proceedings of the 6th International Conference on Autonomic Computing*, 107–116.
- Dmitrii. (2021). *Useful utilities for python 3.10+*. GitHub. <https://github.com/dmitry-ed-gusev/pyutilities>
- Fu, Q. (2020). *PyHelpers: An open-source toolkit for facilitating python users' data manipulation tasks*. Zenodo. <https://doi.org/10.5281/zenodo.4017438>
- Gasch, S. (2022). *Python utilities*. GitHub. <https://github.com/scottgasch/pyutils>
- Gómez, J. (2019). *Functions and utilities to recycle code*. GitHub. <https://github.com/Jesrat/pyutl>
- iwasaki, shuto. (2020). *A collection of useful python programs*. GitHub. <https://github.com/iwasakishuto/Python-Charmers>
- jaysen.lin. (2025). *Python utilities*. GitHub. <https://github.com/linjonh/pripy-utils>
- Lott, S. F. (2018). *Functional python programming: Discover the power of functional programming, generator functions, lazy evaluation, the built-in itertools library, and monads*. Packt Publishing Ltd.
- Mertz, D. (2015). *Functional programming in python*. O'Reilly Media.
- Monteiro, V. (2021). *A set of utility methods end up using in a lot of projects*. GitHub. <https://github.com/monthero/dry-pyutils>
- Prince, A. (2023). *Open source python package comprising of decorators that can be used for utility operations*. GitHub. <https://github.com/antoprince001/pyutilator>
- PrinceCheng. (2021). *Python crawler in action*. GitHub. <https://github.com/Nevergiveupp/python-in-action>
- PyImageSearch. (2015). *A series of convenience functions to make basic image processing operations such as translation, rotation, resizing, skeletonization, and displaying matplotlib images easier with OpenCV and python*. GitHub. <https://github.com/PyImageSearch/imutils>
- Rodola, G. (2009). *Cross-platform lib for process and system monitoring in python*. GitHub. <https://github.com/giampaolo/psutil>
- tplnt. (2014). *A library of useful python functions and classes*. GitHub. <https://github.com/tplnt/pyutil>
- Walsh, W. E., Tesauro, G., Kephart, J. O., & Das, R. (2004). Utility functions in autonomic systems. *International Conference on Autonomic Computing, 2004. Proceedings.*, 70–77.