



# Alarconpy: a Python Package for Meteorologists

Albenis Pérez-Alarcón<sup>1</sup> , José C. Fernández-Alvarez<sup>1</sup> 

<sup>1</sup> Departamento de Meteorología, Instituto Superior de Tecnologías y Ciencias Aplicadas, Universidad de La Habana, 10400 La Habana, Cuba

**Abstract:** Python is a programming language in which libraries are continuously developed with application in atmospheric sciences. Alarconpy is a tools collection in Python programming language that offers many functions and algorithms for several meteorological applications as well as for quick processing of weather data. It includes, among others, functions for calculating some atmospheric variables such as saturation vapor pressure and relative humidity, units converting, maps creation, predefined color palettes for plotting meteorological variables and some tropical cyclones applications such as radial wind profiles. It also contains in the same function the facilities of other Python packages benefiting existing and future users.

**Keywords:** meteorologist, scientific programming, atmospheric sciences

\*\*\*

**Resumen:** Python es un lenguaje de programación en el que las bibliotecas se desarrollan continuamente con diversas aplicaciones en las ciencias atmosféricas. De hecho, Alarconpy es una colección de herramientas en lenguaje de programación Python que ofrece varias funciones y algoritmos para solucionar problemas de las ciencias

meteorológicas, así como para un procesamiento rápido de datos meteorológicos. Incluye, entre otras, funciones para el cálculo de algunas variables atmosféricas como presión de vapor de saturación y humedad relativa, conversión de unidades, creación de mapas, paletas de colores predefinidas para representar campos meteorológicos, y algunas aplicaciones específicas para el trabajo con ciclones tropicales, como perfiles radiales de viento. También contiene en una misma función las facilidades de otras bibliotecas de Python posibilitando su utilización de forma fácil en la solución de problemas

**Palabras clave:** meteorología, programación científica, ciencias atmosféricas

## 1 Introduction

IN atmospheric science, researchers generally must process a large volume of data obtained from the network of meteorological observations on a local and global scale, as well as from the output of numerical weather forecast models. This data collection is stored in netCDF (Network Common Data Form) and GRIB (GRIdded Binary or General Regularly-distributed Information in Binary form, WMO (2003) format. GRIB is a concise data format commonly used in meteorology to store forecast and historical meteorological data. It is standardized by the Commission for Basic Systems of the World

Meteorological Organization (WMO).

These formats for storing meteorological data have led to the development of several applications for their processing, such as the NetCDF Operators (NCO) and the Climate Data Operators (CDO). The former focuses on simple data curation (e.g. viewing the contents of a file, selecting a subset of the data or editing the metadata within a file), while the latter provides for simple statistical analysis (e.g. calculating the climatology, percentile, correlation or heatwave index). It has also been observed in the rapid development of libraries in different programming languages such as Fortran, C, C++, MATLAB (MATrix LABoratory), Python, NCL (National Center for Atmospheric Research (NCAR) Command Language) and IDL (Interactive Data Language) to manage them.

Python is a modern, open-source, interpreted computer language whose use in the atmospheric and oceanic sciences is growing by leaps and bounds (Irving, 2019). Moreover, Python offers the IPython interactive command line that allows scientists to view data, test new ideas, combine algorithmic approaches, and evaluate their results directly. This process could lead to final results, or it could clarify how to build broader and more static production code (Pérez and Granger, 2007). Therefore, Python is an excellent tool for such a workflow (Yang et al., 1998). The development in Python of several libraries such as Metpy (May et al., 2008 - 2020) and Siphon (May et al., 2014 - 2017) have made it possible to integrate the functionalities of various standard Python packages in the processing of weather and oceanic data.

The flexibility of Python as a programming language as well as the functions provided by various Python packages for the processing of weather data motivated the development of the

Alarconpy package. It integrates facilities that separately provide libraries such as Metpy (May et al., 2008 - 2020), Scipy (<https://www.scipy.org>) and Cartopy (<https://scitools.org.uk/cartopy/docs/latest/>), as well as the development of new applications. In this way, the user is provided with a set of built-in tools for rapid data processing.

## 2 Alarconpy description

Alarconpy is a set of tools developed in Python for the processing of meteorological data. Its current version is 1.0.3 (Figure 1) and it includes functions to calculate some atmospheric variables such as relative humidity and the saturation vapor pressure, operations with dates and a module with specific algorithms for working with tropical cyclones (TCs). It also includes several predefined color palettes for plotting the wind field, the surface and mean sea level pressure, the radar reflectivity, the Gálvez-Davison Index (Gálvez and Davison, 2016), the precipitation as well as the color palettes available in Basemap (<https://basemaptutorial.readthedocs.io/en/latest/>).

```
In [2]: import alarconpy as al
In [3]: al.__name__
Out[3]: 'alarconpy'
In [4]: al.__version__
Out[4]: '1.0.4'
In [5]: al.__contact__
Out[5]: 'apalarcon1991@gmail.com'
```

Figure 1: AlarconPy information

Furthermore, it incorporates in the same function the mathematic interpolation methods available in Scipy and Metpy, facilitating to the user the implementation of any of them. Moreover, It contains a function for creating maps

with Cartopy and allows the calculation of the distance between two geographic points using the implemented Haversine function.

## 2.1 Alarconpy requirements

Alarconpy is developed for Python 3.x and there are some python libraries required for use it:

- **Numpy:** It is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object, sophisticated functions and useful linear algebra, Fourier transform, and random number capabilities. The NumPy array object is the common interface for working with typed arrays of data across a wide-variety of scientific Python packages. NumPy also features a C-API, which enables interfacing existing Fortran/C/C++ libraries with Python and NumPy (<https://www.scipy.org>) Oliphant (2006).
- **Metpy:** It is a modern meteorological open-source toolkit for Python. It is a maintained project of Unidata to serve the academic meteorological community. MetPy consists of three major areas of functionality: Plots, Calculations and File Input/Output (<https://unidata.github.io/MetPy/latest/index.html>) (May et al., 2008 - 2020).
- **Cartopy:** It is a Python package designed for geospatial data processing in order to produce maps and other geospatial data analyses. It makes use of the powerful PROJ.4, NumPy and Shapely libraries and includes a programmatic interface built on top of Matplotlib for the creation of publication quality maps.
- **Scipy:** It is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, optimization, linear algebra, and statistics.
- **NetCDF4:** It is the Python interface to the netCDF C library. This module can read and write files in both the new netCDF 4 and the old netCDF 3 format, and can create files that are readable by HDF5 clients (<https://unidata.github.io/netcdf4-python/netCDF4/index.html>).
- **Matplotlib:** It is a plotting library for the Python programming language and its numerical mathematics extension NumPy (<https://matplotlib.org>). Also, it is a sophisticated library capable of producing publication-quality graphics in a variety of formats, and with full LaTeX support (Barrett et al., 2004).
- **Time:** This module provides various time-related functions (<https://docs.python.org/3/library/time.html>).

## 3 Alarconpy usage examples

### 3.1 Plotting maps

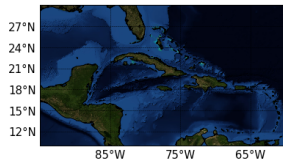
Figure 2 shows the use of the Alarconpy to create a geographic map with Cartopy. As can be seen, in practically one line of code (without count the lines to import required packages), a geographic map was created, while making this same map using a directly Cartopy package, it is required at least 10 lines of code.

The `get_map(arguments*)` function contains the possibility to use new backgrounds

```
In [2]: import alarconpy as al
In [3]: import matplotlib.pyplot as plt
In [4]: mapa=al.get_map((-95,10),(-60,30),bg="stock")

27°N
24°N
21°N
18°N
15°N
12°N
85°W 75°W 65°W
```

```
In [5]: mapa=al.get_map((-95,10),(-60,30),bg="BM",res="medium")
```



**Figure 2:** Alarconpy output from geographic map creation. This code was generate using IPython's physics profile. The map in the top was generated using the defaults Cartopy map background while the map in the bottom was used the Blue Marble background

for the maps, as shown in the Figure 2 on the bottom map, where the Blue Marble (<https://neo.sci.gsfc.nasa.gov/view.php?datasetId=BlueMarbleNG-TB>) background was used. It also allows to change the resolution of the map background, the font size and the interval to draw the parallels and meridians.

## 3.2 Operations with time

Many meteorological data files contain the date in Unix Time format (number of seconds elapsed since the start of the Unix epoch at 1 January 1970 00:00:00 UTC). The Alarconpy package provides a function to convert the date in Unix Time to Julian date. The following code shows the use of the `time_calc(arguments*)` function:

```
import alarconpy as al

Out [2]: str(al.time_calc("1970-01-01
00:00:00",1592094920.964/3600.))
```

```
Out [3]: '2020-06-14 00:35:20.964000'
```

It also includes a function to calculate the total hours elapsed between two dates:

```
Out [3] al.time_dif("20200610000000",
"20200612000000")

Out [3]: 48
```

## 3.3 Interpolation methods

As mentioned above, Alarconpy includes the Scipy and Metpy interpolation methods in the same function, which facilitates the implementation of any of them easily and quickly. Available interpolation methods are "linear", "nearest", "cubic" and "rbf" from Scipy and "natural\_neighbor", "barnes" and "cressman" from Metpy". Below is an example of the use of some of them.

```
import alarconpy as al
```

```
In [2]: print(points)
[[-99.875  5.125]
 [-99.625  5.125]
 [-99.375  5.125]
 ...
 [-15.375 35.125]
 [-15.125 35.125]
 [-14.875 35.125]]
```

```
In [3]: print(ipoints)
[[-50.  22.]]
```

```
In [4]: print(sst)
[[28.27 28.31 28.40 ...
 28.85 28.70 28.63]
 [28.61 28.61 28.63 ...
 28.84 28.68 28.61]
 [28.77 28.76 28.76 ...
```

```

28.89 28.73 28.65]
...
[-- 19.60 19.61 19.60]
[-- 19.53 19.56 19.55]
[-- 19.52 19.55 19.56]]

In[5]: al.points_interpolation(points,
    sst.flatten(), ipoints,
    interp_type="linear")
Out[5]: array([26.3949995])

In[6]: al.points_interpolation(points,
    sst.flatten(), ipoints,
    interp_type="natural_neighbor")
Out[6]: array([26.39])

In[7]: al.points_interpolation(points,
    sst.flatten(), ipoints,
    interp_type="nearest")
Out[7]: array([26.44999885559082])

In[8]: al.points_interpolation(points,
    sst.flatten(),
    ipoints, interp_type="cubic")
Out[8]: array([26.37522029])

```

### 3.4 Units conversion

This python package presented in this article includes a function that allows unit conversion. Supports conversion of units, among others, in acceleration, angle, area, the moment of inertia, density, length, mass, temperature and velocity. Below is shown an example:

```

import alarconpy as al

In[2]: al.units_conversion( 1, "knots",
    "m/s" )
Out[2]: 0.514444

```

```

In[3]: al.units_conversion( 1, "km",
    "m" )

```

```

Out[3]: 1000

```

```

In[4]: al.units_conversion( 300,
    "degK", "degC" )

```

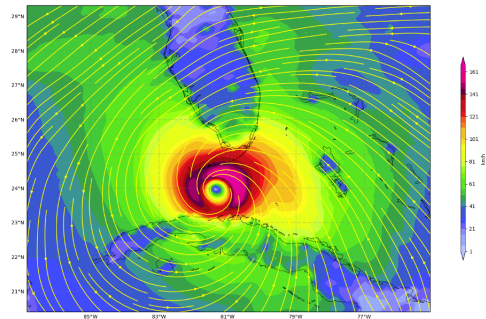
```

Out[4]: 26.850000000000023

```

### 3.5 Color palettes

Alarconpy includes several color palettes for representing different meteorological variables. Figure 3 shows an example of their use. The color palettes developed in Alarconpy are in agreement with the color palettes used for plotting the corresponding meteorological variables at the Cuban Institute of Meteorology.

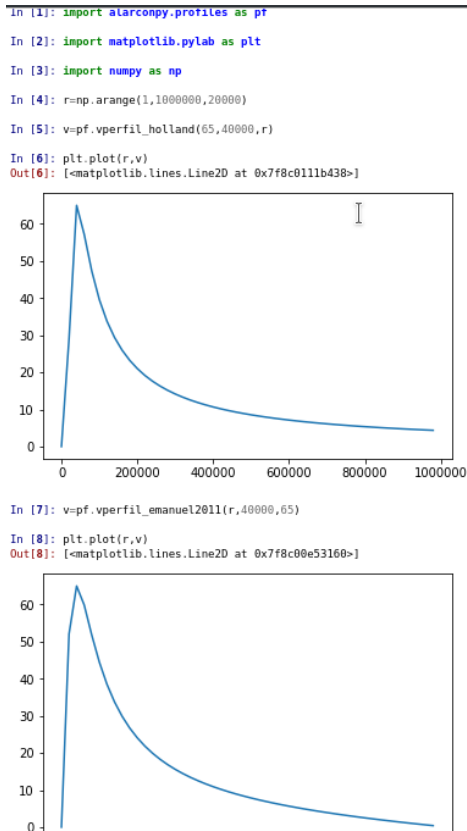


**Figure 3:** Stream flow for Hurricane Irma on September 10, 2017 at 0600 UTC. This figure was created using the `plot_wind_h(arguments*)` function and the color palette is obtained through the `colorbarwind()` function. These functions are available in Alarconpy package.

### 3.6 Tropical cyclones modules

The radial wind profiles of tropical cyclones are a powerful tool for describing the radial wind structure of these storms. Alarconpy incorporates a module with the implementation of several radial wind profiles: [Holland \(1980\)](#), [DeMaria \(1987\)](#), [Emanuel \(2004\)](#), [Willoughby et al.](#)

(2006), Emanuel and Rotunno (2011), Frisius and Scgönemann (2013) and Chavas et al. (2015). Figure 4 shows an example of using the **profiles module** to obtain the TC radial wind structure from Holland (1980) and Emanuel and Rotunno (2011) radial wind profiles.



**Figure 4:** Alarconpy output from tropical cyclones radial wind profiles implementation.

For the study of the risk associated with the impact of TCs on a geographic area, it is important to make a mask of the area enclosed by the TC that allows it to be distinguished from the environment undisturbed by cyclonic circulation. Therefore, the function **create\_TCmask(arguments\*)** was incorporated into the Alarconpy package, which allows it to make quickly this mask, as shown in Figure 5.

On several occasions, intending to provide

```
In [18]: import alarconpy.TCmodule as tc
In [19]: import matplotlib.pyplot as plt
In [20]: lat,lon,mask=tc.create_TCmask(18.5,-80.1,650)
In [21]: plt.imshow(mask)
Out[21]: <matplotlib.image.AxesImage at 0x7f8bfcaef668>
```

```
In [22]: |
```

**Figure 5:** Alarconpy output from TCmodule mask creation.

an evaluation of the TCs track forecast, it is necessary to calculate the distance between two geographic locations. The python package presented in this article includes the implementation of the Haversine function for calculating the distance between two geographic coordinates. The code segment below shows the simplicity of its use.

```
import alarconpy as al

lat1=22.5

lon1=-74.3

lat2=23.8

lon2=-83.2

al.haversine((lat1,lon1),(lat2,lon2),
             units='km')
Out[7]: 919.627811821197
```

### 3.7 Text files

One of the most important facilities of Alarconpy is the incorporation of the **index\_row (arguments\*)** function, which allows to quickly determine the indexes in which a string is found in a list of strings. The code presented below shows an example of the use of this function to determine the position of Hurricane Irma (AL112017) entries in the HURDAT2 database of the National Hurricane Center (NHC). The HURDAT2 database has a text format containing information every six hours about the location, maximum winds, minimum central pressure of all known tropical and subtropical cyclones.

```
import alarconpy as al

hurdat=open("hurdat2-1851-2019-
            052520.txt","r")

hurdat=hurdat.readlines()

index=al.index_row(hurdat,"AL112017")

print(index)
[52180]

index=index[0]

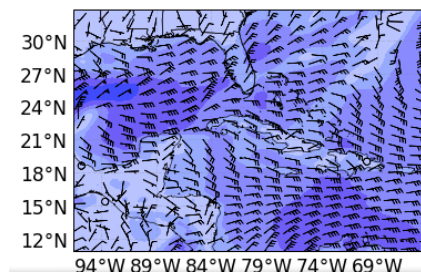
hurdat[index]
Out [9]: 'AL112017,          IRMA,          66,\n'
```

### 3.8 Plotting of meteorological variables

Finally, Alarconpy has several functions for plotting some meteorological fields. These functions were developed to mapping the Numerical Tools for Weather Forecast system

(NTWF) and the Numerical Tools for Hurricane Forecast system (NTHF) outputs. The NTWF and NTHF used the NMM (Nonhydrostatic Mesoscale Model)) the dynamic core of the WRF (Weather Research and Forecasting) model. Both forecast systems are operative in the Department of Meteorology of the Higher of Technologies and Applied Sciences, University of Havana. NTWF and NTHF outputs are available at <https://www.instec.cu/model/models.php>. Figure 6 shows an example of the use of these plotting functions.

```
In [93]: from netCDF4 import Dataset
In [94]: import alarconpy as al
In [95]: nc=Dataset("wrfout_d01_2019-01-27_06:00:00")
In [96]: lat=nc.variables["GLAT"][0,:]
In [97]: lon=nc.variables["GLON"][0,:]
In [98]: lat=lat*180/np.pi
In [99]: lon=lon*180/np.pi
In [100]: u10=nc.variables["U10"][0,:]*3.6
In [101]: v10=nc.variables["V10"][0,:]*3.6
In [102]: fig=plt.figure(figsize=(18,12))
<Figure size 1296x864 with 0 Axes>
In [103]: al.plot_wind(fig,lon,lat,u10,v10)
```



**Figure 6:** Use of Alarconpy package to plot the surface wind field obtained from the output of the NTWF system.

The Alarconpy's plotting functions could be used to plot the outputs of any numerical weather forecast model or any meteorological data matrix, which contains information of some meteorological variable included in this package.



## 4 Alarconpy user's guide

All functions implemented in the Alarconpy package have a description of each input parameters, as well as the type of return. This description can be revised using the IPython interactive command line.

## 5 Package availability

The Alarconpy package is freely available on Github (<https://github.com/apalarcon/alarconpy>) or Zenodo (<https://doi.org/10.5281/zenodo.4493257>)

### 5.1 Alarconpy installation

We suggest using the Alarconpy package in the Anaconda environment. Just clone the Alarconpy repository from Github or download it from Zenodo and copy the alarconpy folder into anaconda installation. Note that all dependencies needed for Alarconpy must be previously installed.

## 6 Conclusions

Python has become an extremely popular data analysis and mapping tool in the atmospheric sciences. In fact, Alarconpy ?? is a powerful Python tools collection for working with atmospheric and ocean data. This package integrates functionalities of other Python packages which benefiting existing and future users. Ongoing work will therefore include new functionalities and applications.

## References

Barrett, P., Hunter, J., and Greenfield, P.: Matplotlib: A Portable Python Plotting

Package, *Astronomical Data Analysis Software & Systems*, 14, 2004.

Chavas, D. R., Lin, N., and Emanuel, K. A.: A model for the complete radial structure of the tropical cyclone wind field. Part I: Comparison with observed structure., *Journal of the Atmospheric Sciences*, 72, 3647–3662, <https://doi.org/10.1175/JAS-D-15-0014.1>, 2015.

DeMaria, M.: Tropical cyclone track prediction with a barotropic spectral model, *Monthly Weather Review*, 115, 2346–2357, [https://doi.org/10.1175/1520-0493\(1987\)115<2346:TCTPWA.2.0.CO;2](https://doi.org/10.1175/1520-0493(1987)115<2346:TCTPWA.2.0.CO;2), 1987.

Emanuel, K. and Rotunno, R.: Self-stratification of tropical cyclone outflow. Part I: Implications for storm structure., *Journal of the Atmospheric Sciences*, 68, 82 236–2249, <https://doi.org/10.1175/JAS-D-10-05024.1>, 2011.

Emanuel, K. A.: Tropical cyclones energetics and structure, *Atmospheric Turbulence and Mesoscale Meteorology*. E. Fedorovich, R. Rotunno, and B. Stevens, Eds., Cambridge University Press, p. 165–192, 2004.

Frisius, T. and Scgönemann, D.: The Impact of Gradient Wind Imbalance on Potential Intensity of Tropical Cyclones in an Unbalanced Slab Boundary Layer Model, *Journal of the Atmospheric Sciences*, 70, 1874–1890, <https://doi.org/10.1175/JAS-D-12-0160.1>, 2013.

Gálvez, J. M. and Davison, M.: The Gálvez-Davison Index for Tropical Convection, URL [https://www.wpc.ncep.noaa.gov/international/gdi/GDI\\_Manuscript\\_V20161021.pdf](https://www.wpc.ncep.noaa.gov/international/gdi/GDI_Manuscript_V20161021.pdf), [accessed Jun 19, 2020], 2016.



- Holland, G. J.: An analytic model of the wind and pressure profiles in hurricanes, *Monthly Weather Review*, 1008, 1212–1218, [https://doi.org/10.1175/1520-0493\(1980\)108,1212:AAMOTW.2.0.CO;2](https://doi.org/10.1175/1520-0493(1980)108,1212:AAMOTW.2.0.CO;2), 1980.
- Irving, D.: Python for atmosphere and ocean scientists, *Journal of Open Source Education*, 1, 37, <https://doi.org/10.21105/jose.00037>, 2019.
- May, R., Arms, S., Leeman, J., and Chastang, J.: Siphon: A collection of Python Utilities for Accessing Remote Atmospheric and Oceanic Datasets, <https://doi.org/10.5065/D6CN72NW>, URL <https://github.com/Unidata/siphon>, 2014 - 2017.
- May, R. M., Arms, S. C., Marsh, P., Bruning, E., Leeman, J. R., Goebbert, K., Thielen, J. E., and Bruick, Z. S.: MetPy: A Python Package for Meteorological Data, <https://doi.org/10.5065/D6WW7G29>, URL <https://github.com/Unidata/MetPy>, 2008 - 2020.
- Oliphant, T. E.: A guide to NumPy, vol. 1, Trelgol Publishing USA, 2006.
- Pérez, F. and Granger, B. E.: IPython: A System for Interactive Scientific Computing, *Computing in Science & Engineering*, 9, 21–29, <https://doi.org/10.1109/MCSE.2007.53>, 2007.
- Willoughby, H. E., Darling, R. W. R., and Rahn, M.: Parametric Representation of the Primary Hurricane Vortex. Part II: A New Family of Sectionally Continuous Profiles, *Monthly Weather Review*, 134, 1102–1120, <https://doi.org/10.1175/MWR3106.1>, 2006.
- WMO: General Regularly-distributed Information in Binary form, URL <http://www.wmo.int/pages/prog/www/DPS/FM92-GRIB2-11-2003.pdf>, [accessed Jun 13, 2020], 2003.
- Yang, T.-Y., Furnish, G., and Dubois, P.: *Steering Object-Oriented Scientific Computations*, *Proc. Technology of Object-Oriented Languages and Systems (TOOLS)*, IEEE CS Press, pp. 112–119, 1998.



© 2021 by the authors.  
Creative Commons  
Attribution 4.0 International