Introducción a Python y librerias (Numpy, Matplotlib y Cartopy)

Lima, 1 de diciembre de 2021

¿Qué es python?

Python es un lenguaje de programación **interpretado, multiparadigma** (**programación orientado a objetos**, programación imperativa y programación funcional) que busca desarrollar un código legible.

Un lenguajes de programación pueden ser agrupado como:

- **Interpretados**: el código se va traduciendo a lenguaje máquina a medida que es necesario.
- Compilados: traduce por completo el código a un lenguaje máquina, mediante un proceso que es llamado de “Compilado”.

Paradigma: en programación hace referencia a la base y modelo para resolver problemas.

Conceptos básicos de programación

Tipos de datos

Por su formato:
int: 2, 45 , 3
float: 3.14, 20.15
complex: 4+3j, 5.4+3j
bool: True, False
str: ‘hola mundo’, ‘UNALM’

Operaciones básicas

```
>>> (4+5*2)/10.0
>>> (3**2+4**2)**0.5
>>> ‘Hola’ + ‘ ‘ + ‘alumnos’

>>> a = 25
>>> b = 30
>>> c = (a+b)**2

>>> nombre = ‘Joao’
>>> monedas = 5
>>> print(‘Mi nombre es {} y tengo {} monedas’.format(nombre, monedas))
```

Conceptos básicos de programación

Tipos de datos

Por su estructura:

tupla: (2, 3), (‘a’, ’b’) Es inmutable (no modificada, no adicionar nada)

list: [1, 2, 3.5], [‘manzana’, ‘naranja’, ‘piña’] Es mutable

array: array([4,2,5,10]), array([[4, 20.1, 3.0, 10],[1, 0.5, 2.0, 4.0]]) Es mutable

dict: {'frutas': ['naranja'], 'precio': [2.5, 3.0, 4]}, {‘nombre’: [José, Pedro, MIguel], ‘edad’: [10, 11, 10]} Es mutable

Datos indexados

0	1	2	3	
23	28	30	29	= a

	0	1	2	3	
0	23	28	30	29	
1	22	26	27	27	= b
2	22	24	24	25	

```
>>> print(a[2])
30

>>> print(b[2,3])
25
```

Conceptos básicos de programación

Ejercicio con **tupla**

```
>>> a = (2, 15, 10)
>>> print(a)
>>> print(a[0])
>>> a[0] = 4
```

Ejercicio con **lista**

```
>>> b = [2, 15, 10]
>>> print(b)
>>> print(b[0])
>>> b[0] = 4
>>> print(b)
>>> b/2.0
```

Ejercicio con **array**

```
>>> import numpy as np
>>> c = np.array([2, 15, 10])
>>> print(c)
>>> print(c[0])
>>> c[0] = 4
>>> print(c)
>>> c/2.0
```

Ejercicio con **dict**

```
>>> d = {}
>>> d[‘alumnos’] = [‘Jose’, ‘Luis’, ’Juan’]
>>> d['edad'] = [22, 23, 24]
>>> print(d)
>>> print(d['alumnos'][2])
>>> d[‘nota’] = [10.44, 11.0, 10.51]
>>> print(d)
>>> d.keys()
```

Uso de módulos, paquetes y librerías

- Módulo (Module): archivo que contiene un función, clase, variables, etc

```
def saludo(Nombre):
        print('Hola {}'.format(Nombre))
```

--> bienvenida.py

```
>>>import bienvenida as bv
>>>bv.saludo('Joao')
Hola Joao
```

- Paquete (Package): es el contenedor de múltiples módulos

```
habla/
    inicio/
        bienvenida.py
    final/
        despedida.py
```

- Bibliotecas (Library): es el contenedor de múltiples paquetes

¿Cómo utilizar módulos/paquetes/bibliotecas?

```
>>> import numpy
>>> numpy.ones([5])
array([1., 1., 1., 1., 1.])
```

```
>>> import numpy as np
>>> np.ones([5])
array([1., 1., 1., 1., 1.])
```

```
>>> from numpy import ones
>>> ones([5])
array([1., 1., 1., 1., 1.])
```

Programación orientado a objetos

La programación Orientada a objetos se define como un paradigma de la programación (una forma específica de programar), donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para lograr un determinado objetivo.


```
class Automovil:
    def __init__(self, marca, color):
        self.marca = marca
        self.color = color

    def __str__(self):
        return 'esta es la clase Automovil'

    def identificacion(self):
        return 'este objeto es automovil '+self.marca+' de color '+self.color
```

```
auto1 = Automovil('volkswagen', 'celeste')
auto2 = Automovil('peugeot', 'blanco')
auto3 = Automovil('audi', 'rojo')

print(auto1.identificacion())
print(auto1.marca)
```

Programación orientado a objetos

La programación Orientada a objetos se define como un paradigma de la programación (una forma específica de programar), donde se organiza el código en unidades denominadas clases, de las cuales se crean objetos que se relacionan entre sí para lograr un determinado objetivo.

¿Cómo esto me puede ayudar a lograr un objetivo?

Clase

Método

Objeto

Atributo

```
class Automovil:
    def __init__(self, m, c):
        self.marca = m
        self.color = c

    def __str__(self):
        return 'esta es la clase Automovil'

    def identificacion(self):
        return 'este objeto es automovil '+self.marca+' de color '+self.color
```

```
auto1 = Automovil('volkswagen', 'celeste')
auto2 = Automovil('peugeot', 'blanco')
auto3 = Automovil('audi', 'rojo')
```

Objeto

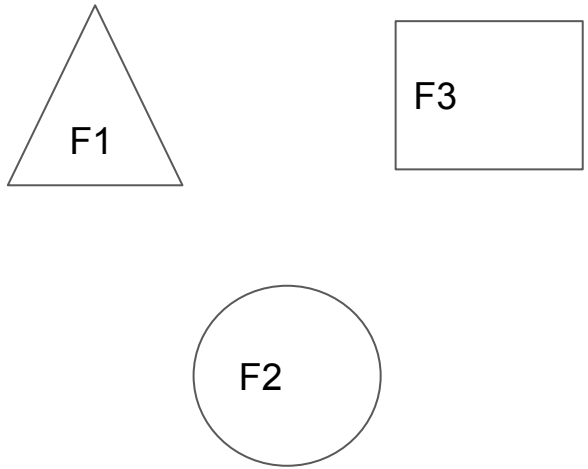
```
print(auto1.identificacion())
print(auto1.marca)
```

Método

Atributo

Ejercicio: identificar figuras

```
class Figura:
    def __init__(self, nombre, angulos):
        self.nombre = nombre
        self.angulos = angulos

    def tipo(self):
        if self.angulos == 0:
            self.tipo = 'circulo'
        elif self.angulos == 3:
            self.tipo = 'triangulo'
        elif self.angulos == 4:
            self.tipo = 'cuadrado'
        return self.nombre+' es un: '+self.tipo
```



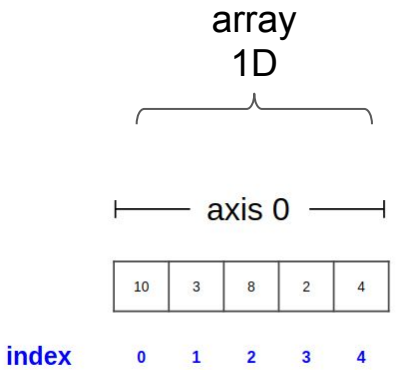
Ejercicio: monitorear el tiempo atmosférico

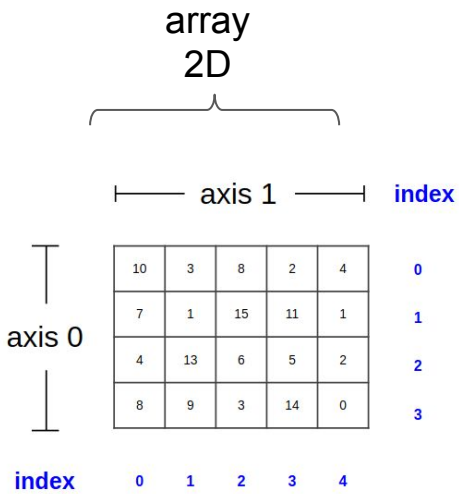
```
class Meteorologia:
    def __init__(self, nombre, temperatura, hr, presion):
        self.nombre = nombre
        self.temperatura = temperatura
        self.hr = hr
        self.presion = presion

    def tiempo(self):
        if self.temperatura >= 20.0:
            if self.hr>85.0:
                self.tiempo = 'caliente'
            else:
                self.tiempo = 'templado'
        elif self.temperatura < 20.0:
            self.tiempo = 'frio'
        return 'el tiempo esta: '+self.tiempo

    def pronostico(self):
        if self.presion < 1005.0:
            self.pronostico = 'lluvia'
        else:
            self.pronostico = 'cielo despejado'
        return self.pronostico
```

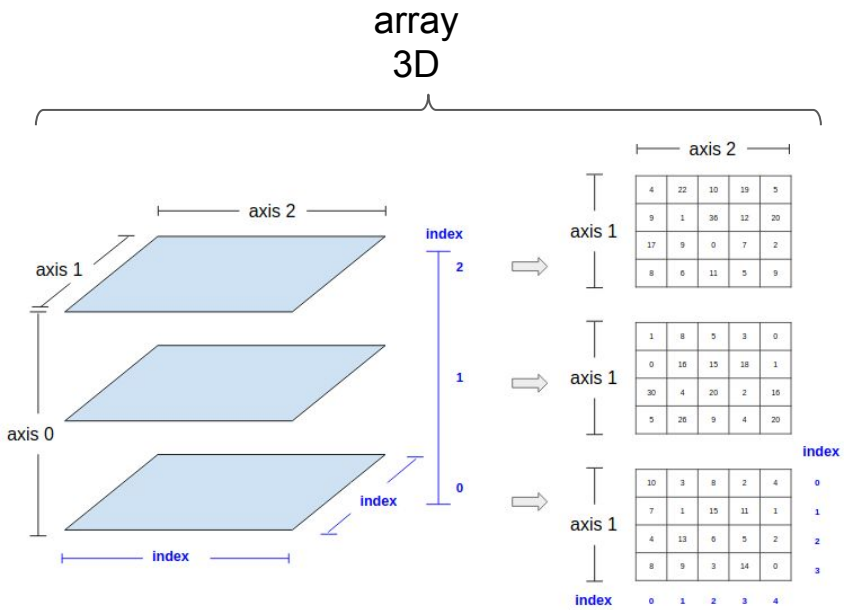
Nombre	Temperatura	HR	Presión atmosférica
La Molina	29	98	1000
Jesus Maria	23	80	1008
Miraflores	18	70	1010

Numpy

Es una librería que permite realizar operaciones con **arrays** de forma **eficiente** (rápido y sin mucho consumo de memoria).
Array es un conjunto de elementos organizados multidimensionalmente (similar al de una matriz).







Ejemplos:

import numpy **<----- importa la librería sin un alias**
import numpy as np **<----- importa la librería con un alias**
from numpy import array **<---- importa solo el modulo array**

Crear una matriz 1D:

```
a = np.array([10,3,8,2,4])
```

Crear una matriz 2D:

```
b = np.array([[10,3,8,2,4],[7,1,15,11,1],[4,13,6,5,2]])
```

Imprime algunos elementos (recordar que los índices comienzan en 0):

```
print(a[3])
print(a[1:3], a[1:4])
print(b[2,1])
print(b[1,:])
print(b[0:2,1:4])
```

Atributos de los arrays

.shape : retorna la forma del array
.size : retorna el tamaño del array
.ndim : retorna el número de dimensiones del array
.dtype : retorna el formato del array

Ejemplo:
a = np.random.randint(1,15,[4,5])

Ejecutar:
a.ndim
a.shape
a.size
a.dtype

Métodos de los arrays

.sum() : suma todos los elementos del array
.min() : retorna el mínimo valor del array
.max() : retorna el máximo valor del array
.mean() : retorna el promedio del array
.std() : retorna la desviación estándar del array
.sort() : ordena el array
.astype(formato) : cambia el formato del array
.reshape() : re-estructura la forma del array
.flatten() : re-estructura el array a 1D

Ejemplo:
a = np.random.randint(1,15,[4,5])

Ejecutar:
print(a)
a.astype('float')
print(a.sum(), a.min(), a.max(), a.mean())
a.reshape([2,10])

Tip:
Para conocer la descripción de funciones o métodos utiliza el comando **help()**

Módulos de Numpy

np.array() : crea un array
np.random.random() : crea un array con números decimales de forma aleatoria
np.random.randint() : crea un array con números enteros de forma aleatoria
np.linspace(Vmin, Vmax, N) : crea un array con N elementos, que van desde Vmin hasta Vmax
np.arange(Vmin, Vmax, Delta) : crea un array de Vmin a Vmax (sin considerarlo) cada Delta unidades
np.ones() : crea un array de unos
np.zeros() : crea un array de ceros
np.transpose() : transpone el array

Ejemplo:

```
a = np.array([1,2,3,4,5])
b = np.array([10,20,30,40,50])
```

¡Cuidado al momento de copiar arrays!

```
a = np.random.randint(1,10,[4,3])
b = a
c = b.copy()
b[0,0] = 100
print(a)
print(b)
print(c)
```

Operaciones con arrays

np.concatenate() : concatena arrays

Ejemplo:
a = np.array([1,2,3,4,5,6])
b = np.array([10,20,30,40,50,60])

print(a.ndim, b.ndim)
np.concatenate([a,b])

Re-estructura los arrays y concatenarlos:
a = a.reshape([2,3])
b = b.reshape([2,3])
np.concatenate([a,b], axis=0)
np.concatenate([a,b], axis=1)

Operaciones con Numpy

np.where() : realiza una operación donde se cumple una condición

Ejemplo 1:

```
a = np.random.randint(1,10,[8,6])
b = np.where(a<5,0,1)
print(b)
```

Ejemplo 1:

```
m = np.random.randint(1,10,[8,6])
n = np.random.randint(1,10,[8,6])
z = np.where(m>n,0,1)
print(z)
```

Operaciones con Numpy

np.meshgrid() : convierte un array 1D en 2D, utilizando el número de elementos de los arrays 1D.

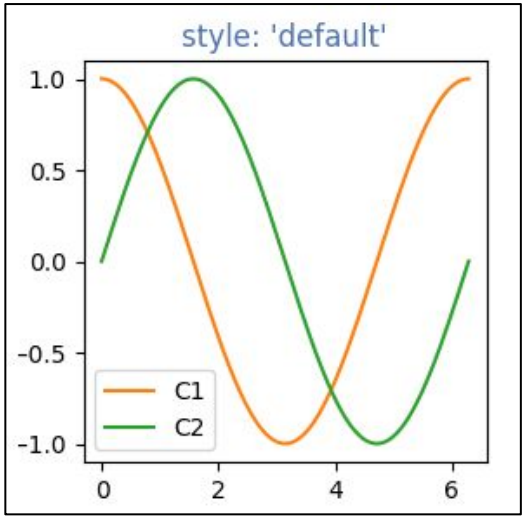
Ejemplo:

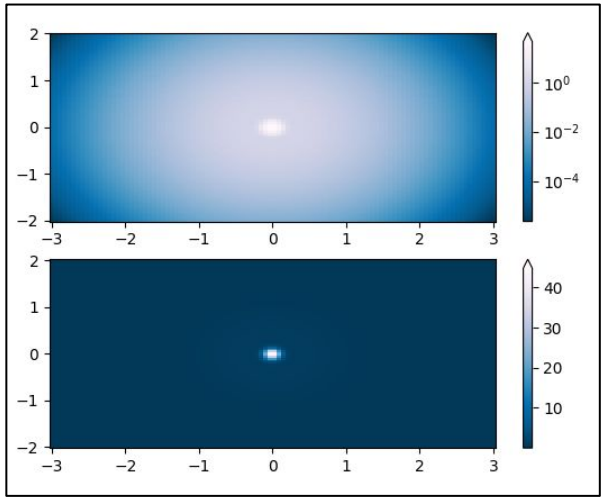
```
x = np.arange(-100, -90, 2)
x
x.shape
y = np.arange(0, -15, -1)
y
y.shape

x2d, y2d = np.meshgrid(x, y)
x2d
x2d.shape
y2d
y2d.shape
```

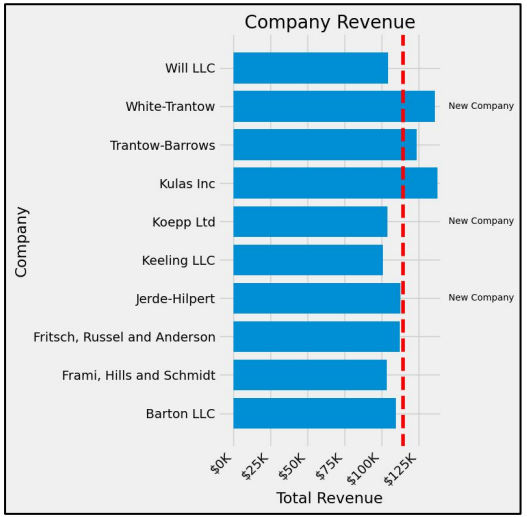
Matplotlib

Es una librería enfocada en la elaboración de estáticos, animados e interactivos gráficos en python.

import matplotlib.pyplot as plt

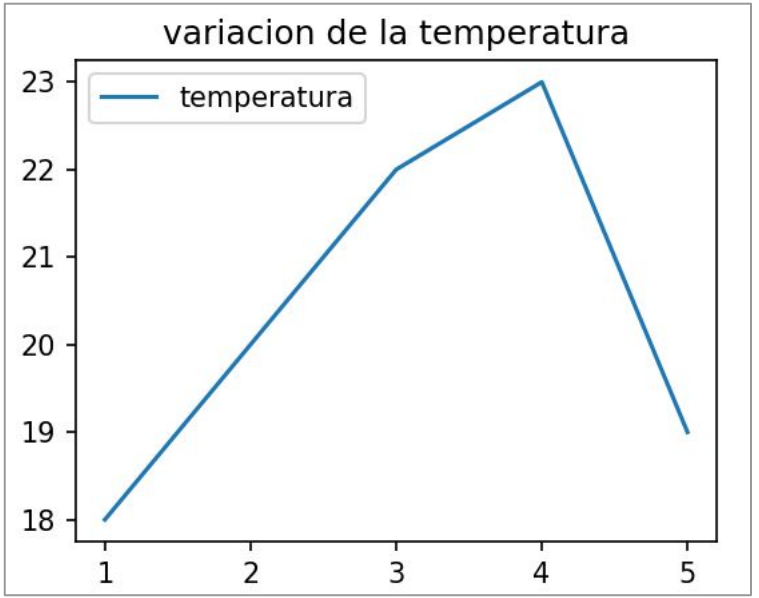






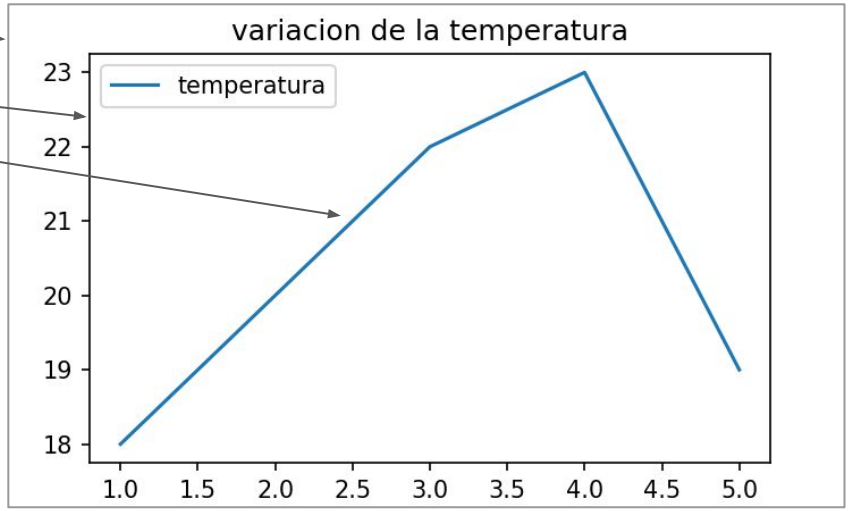
Cómo crear una imagen

```
fig = plt.figure('ejemplo', figsize=(4,3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8])

linea = plt.plot([1,2,3,4,5], [18,20,22,23,19], label='temperatura')
plt.legend()

plt.title('variacion de la temperatura')

plt.show()
```



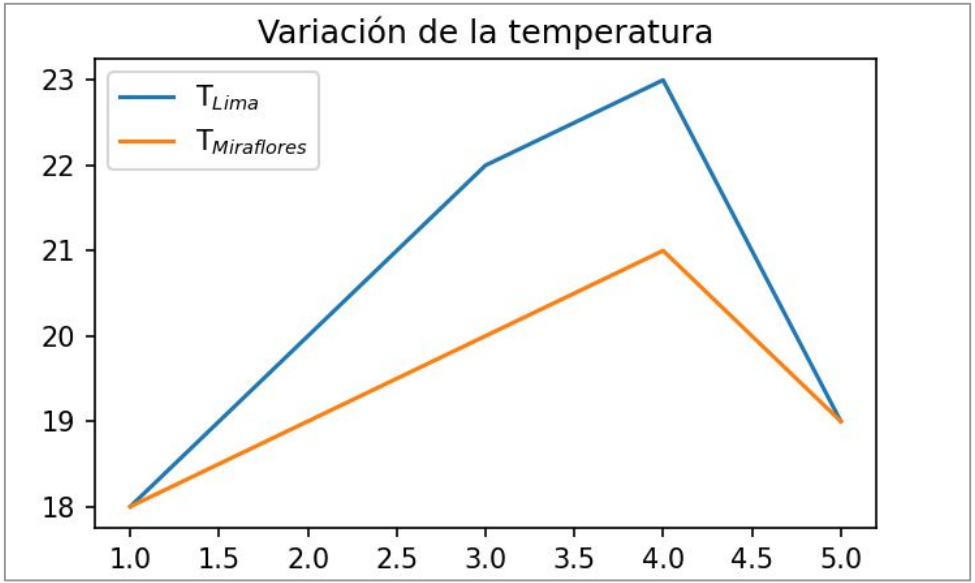
Ejercicio

```
fig = plt.figure('ejemplo', figsize=(5,3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8])

linea = plt.plot([1,2,3,4,5], [18,20,22,23,19], label='temperatura')
plt.legend()

plt.title('variación de la temperatura')

plt.show()
```



Ejercicio

```
fig = plt.figure('ejemplo2', figsize=(5,3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8])

linea1 = plt.plot([1,2,3,4,5], [18,20,22,23,19], label='T$_{Lima}$')
linea2 = plt.plot([1,2,3,4,5], [18,19,20,21,19], label='T$_{Miraflores}$')

plt.legend()

plt.title('Variación de la temperatura')

plt.show()
```

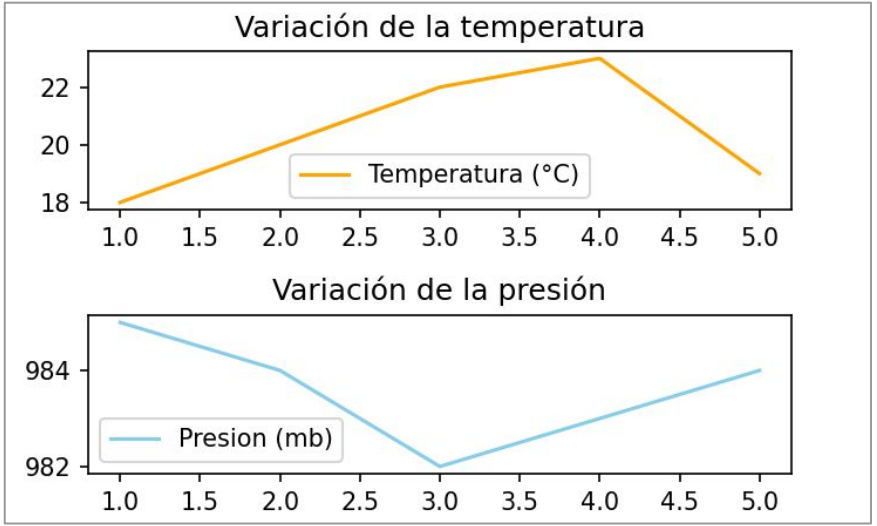


Lista de colores: https://matplotlib.org/examples/color/named_colors.html

Ejercicio

```
fig = plt.figure('metodo_1', figsize=(5,3), dpi=150)


ax1 = fig.add_axes([0.1,0.6,0.8,0.3])
linea = plt.plot([1,2,3,4,5], [18,20,22,23,19], color='orange', label='Temperatura (°C)')
plt.legend()
plt.title('Variación de la temperatura')


ax2 = fig.add_axes([0.1,0.1,0.8,0.3])
linea = plt.plot([1,2,3,4,5], [985,984,982,983,984], color='skyblue', label='Presion (mb)')
plt.legend()
plt.title('Variación de la presión')


plt.show()
```

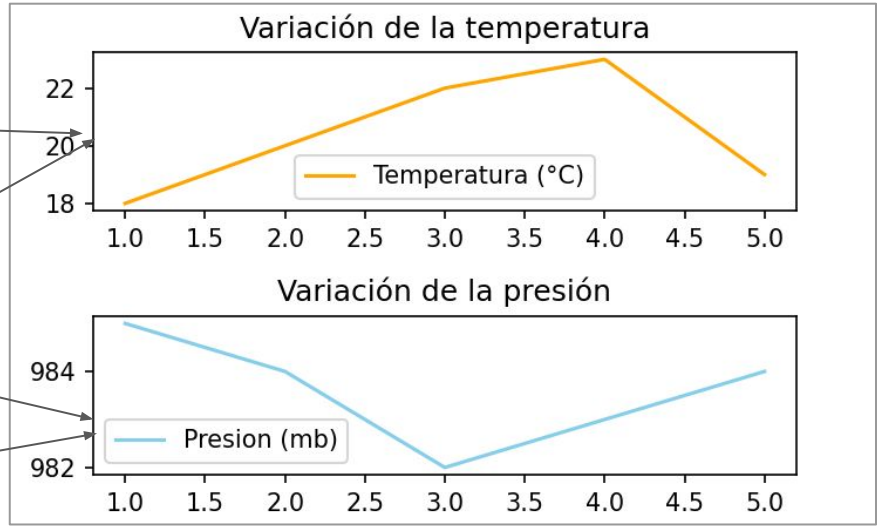
```
fig = plt.figure('metodo_2', figsize=(5,3), dpi=150)


ax1 = fig.add_axes([0.1,0.6,0.8,0.3])
ax2 = fig.add_axes([0.1,0.1,0.8,0.3])


linea = ax1.plot([1,2,3,4,5], [18,20,22,23,19], color='orange', label='Temperatura (°C)')
ax1.legend()
ax1.set_title('Variación de la temperatura')


linea = ax2.plot([1,2,3,4,5], [985,984,982,983,984], color='skyblue', label='Presion (mb)')
ax2.legend()
ax2.set_title('Variación de la presión')


plt.show()
```



Ejercicio

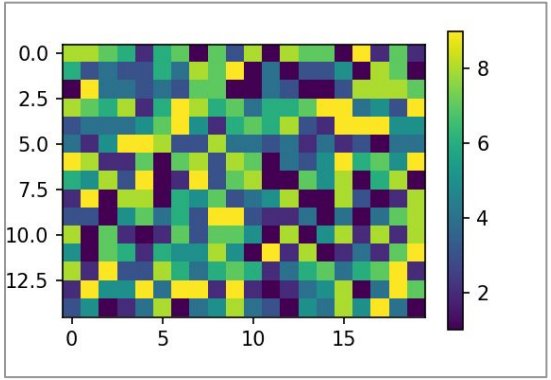
```
fig = plt.figure('metodo_1', figsize=(5,3), dpi=150)


ax1 = fig.add_axes([0.1,0.6,0.8,0.3])
linea = plt.plot([1,2,3,4,5], [18,20,22,23,19], color='orange', label='Temperatura (°C)')
plt.legend()
plt.title('Variación de la temperatura')


ax2 = fig.add_axes([0.1,0.1,0.8,0.3])
linea = plt.plot([1,2,3,4,5], [985,984,982,983,984], color='skyblue', label='Presion (mb)')
plt.legend()
plt.title('Variación de la presión')


plt.show()
```

```
fig = plt.figure('metodo_2', figsize=(5,3), dpi=150)


ax1 = fig.add_axes([0.1,0.6,0.8,0.3])
ax2 = fig.add_axes([0.1,0.1,0.8,0.3])


linea = ax1.plot([1,2,3,4,5], [18,20,22,23,19], color='orange', label='Temperatura (°C)')
ax1.legend()
ax1.set_title('Variación de la temperatura')


linea = ax2.plot([1,2,3,4,5], [985,984,982,983,984], color='skyblue', label='Presion (mb)')
ax2.legend()
ax2.set_title('Variación de la presión')


plt.show()
```



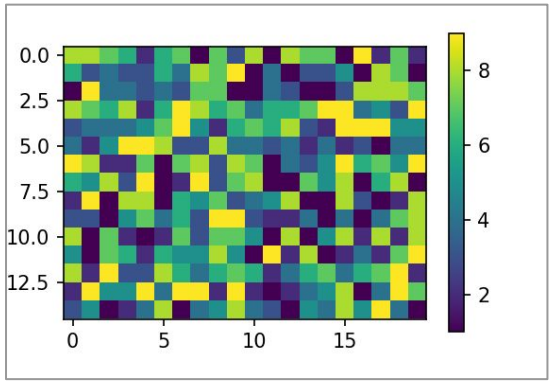
Graficando de arrays 2D

```
import numpy as np
import matplotlib.pyplot as plt

a = np.random.randint(1,10,[15,20])

fig = plt.figure('array 2d', figsize=(4,3), dpi=150)
ax = fig.add_axes([0.1,0.2,0.8,0.7])
img = plt.imshow(a)
plt.colorbar(img)
plt.show()
```

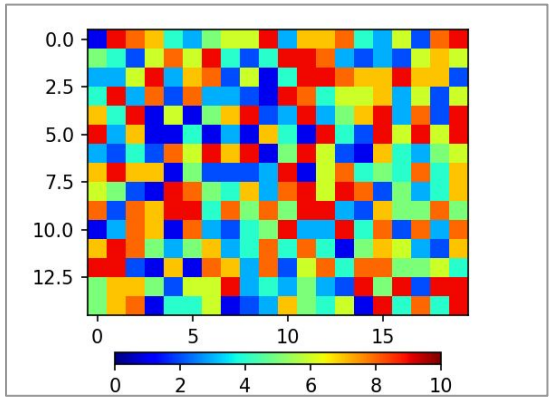


Graficando de arrays 2D

```
import numpy as np
import matplotlib.pyplot as plt

a = np.random.randint(1,10,[15,20])

fig = plt.figure('array 2d', figsize=(4,3), dpi=150)
ax = fig.add_axes([0.1,0.2,0.8,0.7])
img = plt.imshow(a)
plt.colorbar(img)
plt.show()
```

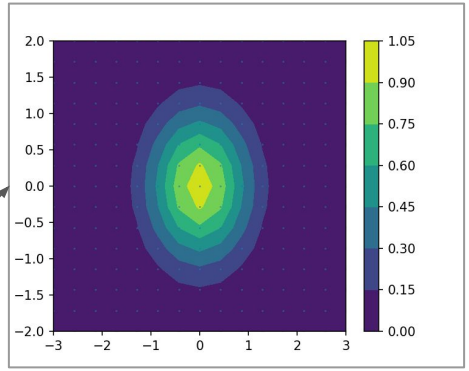


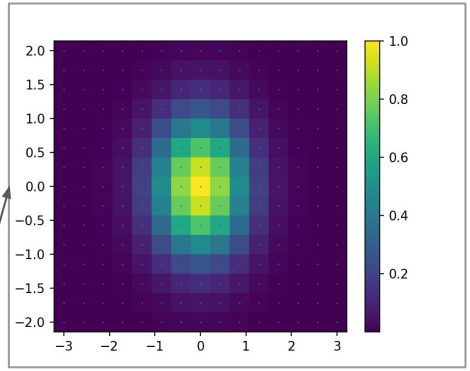

```
import numpy as np
import matplotlib.pyplot as plt

a = np.random.randint(1,10,[15,20])

fig = plt.figure('array 2d', figsize=(4,3), dpi=150)
ax = fig.add_axes([0.1,0.2,0.8,0.7])
img = plt.imshow(a, cmap='jet', vmin=0, vmax=10)
plt.colorbar(img, orientation='horizontal', cax=fig.add_axes([0.2,0.08,0.6,0.03]))
plt.show()
```

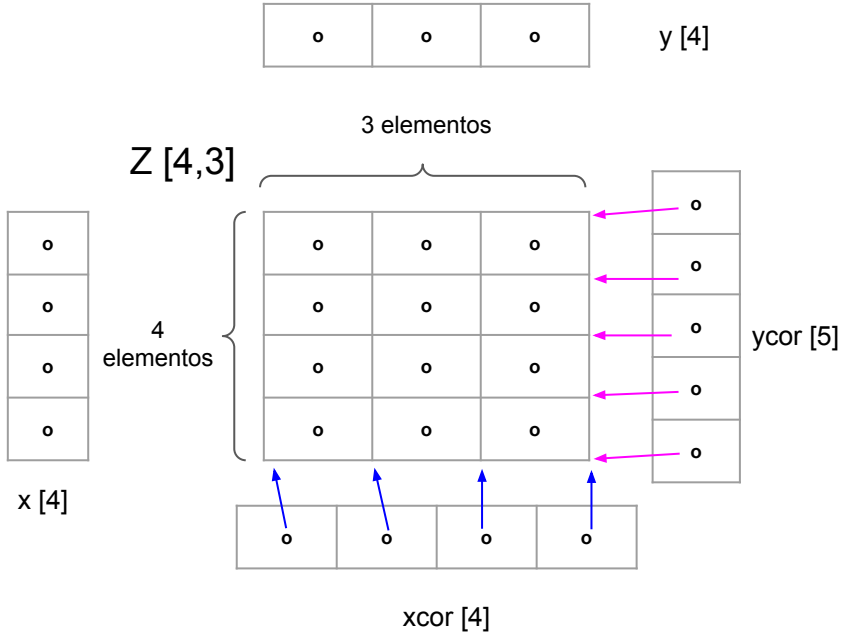


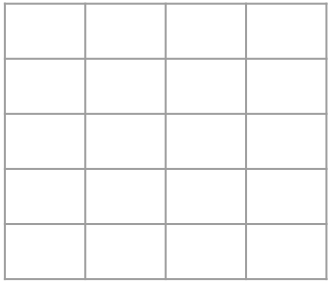
Lista de colores: https://matplotlib.org/tutorials/colors/colormaps.html

```
import numpy as np
import matplotlib.pyplot as plt

N = 15
x = np.linspace(-3.0, 3.0, N)
y = np.linspace(2.0, -2.0, N)
X, Y = np.meshgrid(x, y)
Z = np.exp(-X**2 - Y**2)

fig = plt.figure('contourf', figsize=(5,4), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8])
img = plt.contourf(X, Y, Z)
plt.colorbar(img)
plt.scatter(X,Y,s=0.1)

dx = x[1]-x[0]
dy = y[1]-y[0]

xf = x[-1]+dx
yf = y[-1]+dy

xcor = np.concatenate([x, [xf]])-dx/2
ycor = np.concatenate([y, [yf]])-dy/2
Xcor, Ycor = np.meshgrid(xcor, ycor)

fig = plt.figure('pcolormesh', figsize=(5,4), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8])
img = plt.pcolormesh(Xcor, Ycor, Z)
plt.colorbar(img)
plt.scatter(X,Y,s=0.1)

plt.show()
```

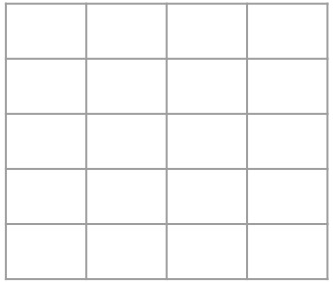




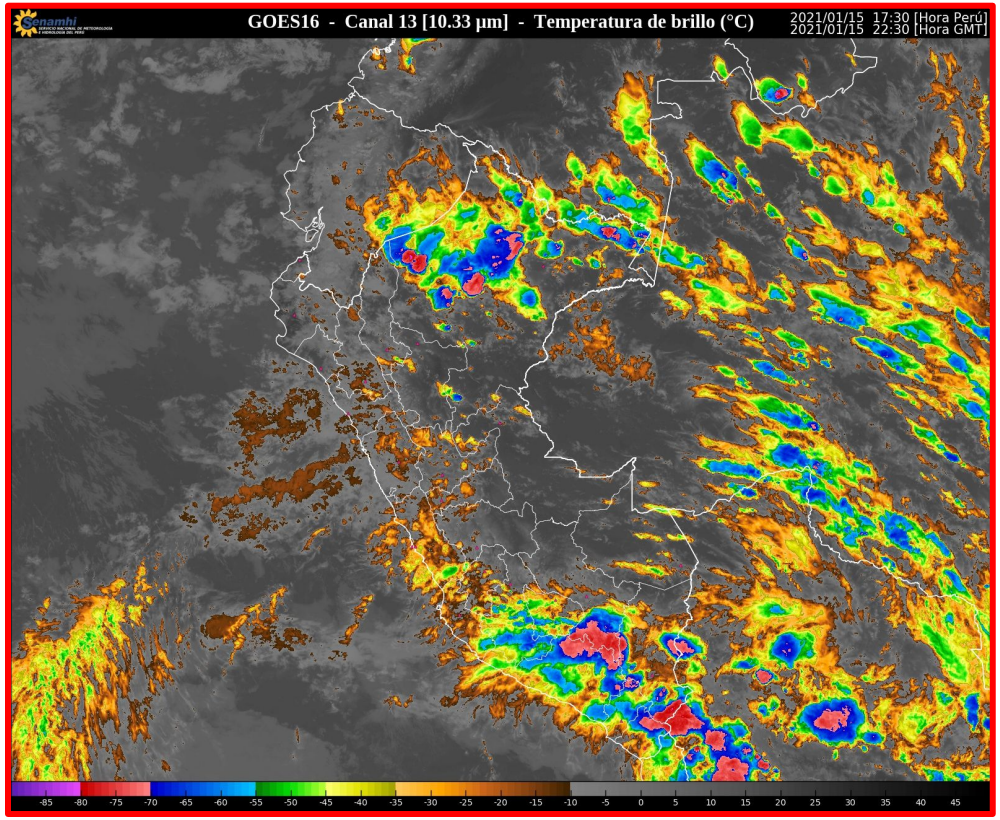


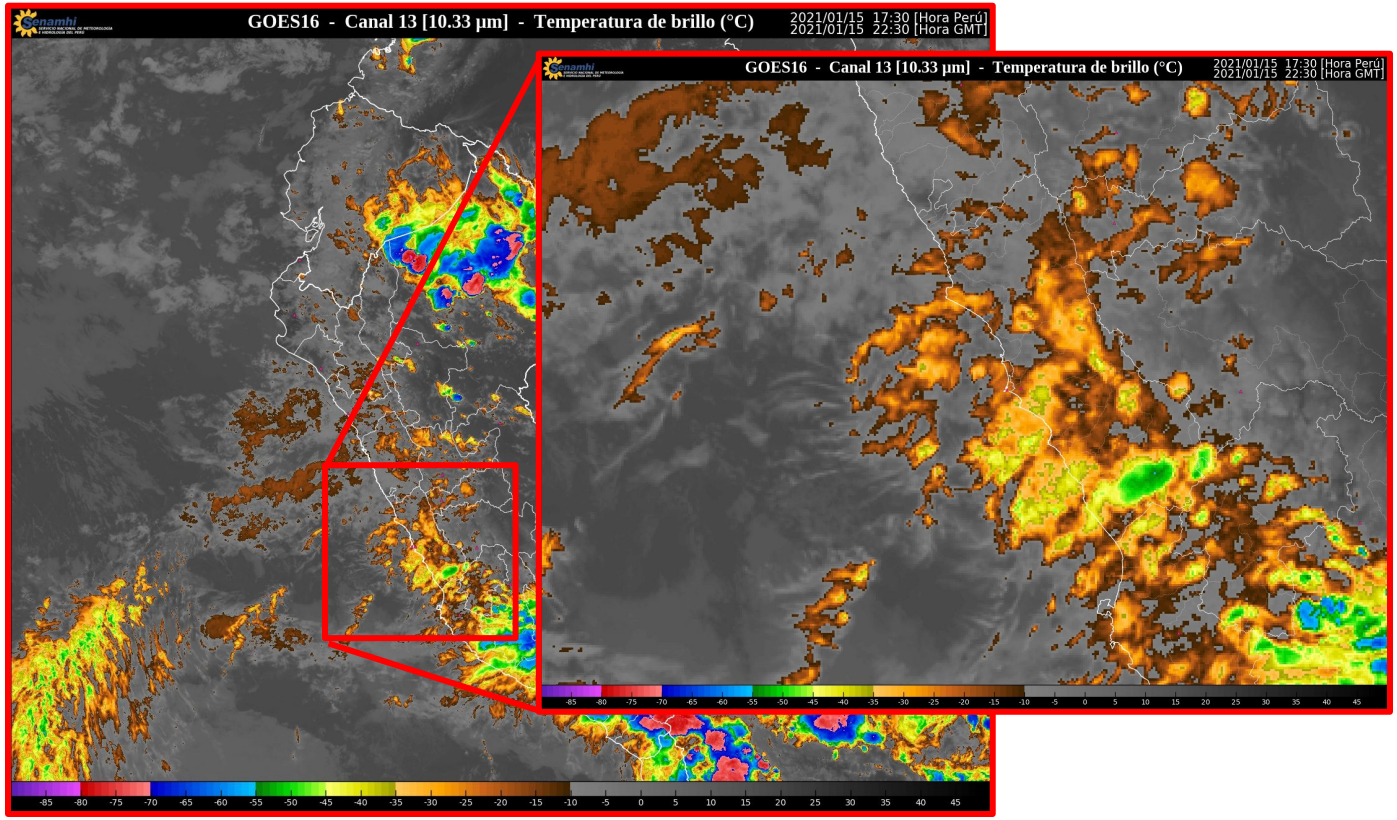


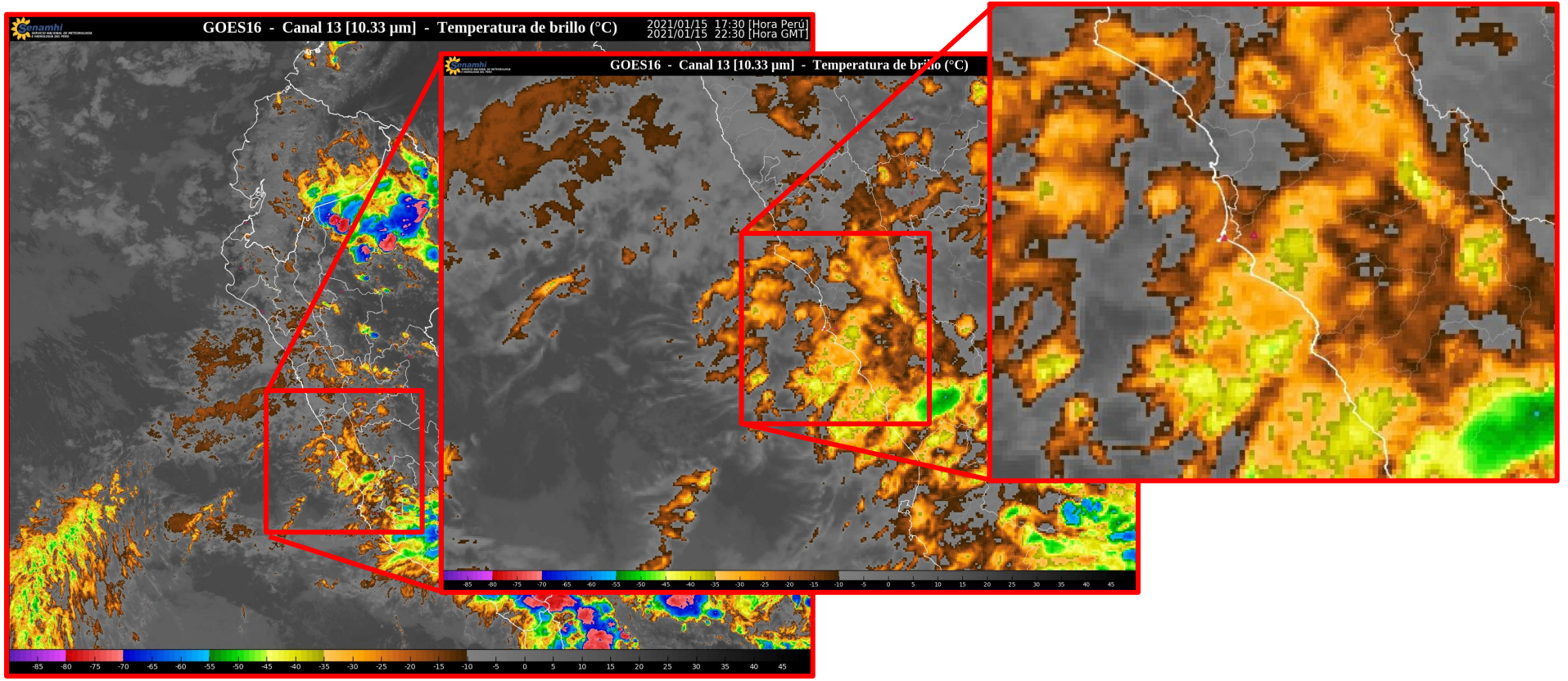
Xcor [5,4]



Ycor [5,4]


GOES16 - Canal 13 [10.33 µm] - Temperatura de brillo (°C)
2021/01/15 17:30 [Hora Perú]
2021/01/15 22:30 [Hora GMT]
-85 -80 -75 -70 -65 -60 -55 -50 -45 -40 -35 -30 -25 -20 -15 -10 -5 0 5 10 15 20 25 30 35 40 45

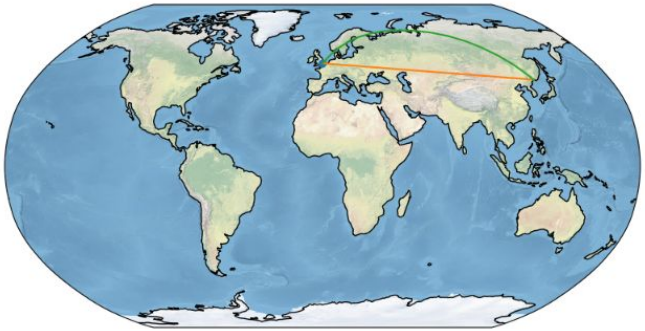

GOES16 - Canal 13 [10.33 µm] - Temperatura de brillo (°C)
2021/01/15 17:30 [Hora Perú]
2021/01/15 22:30 [Hora GMT]
-85 -80 -75 -70 -65 -60 -55 -50 -45 -40 -35 -30 -25 -20 -15 -10 -5 0 5 10 15 20 25 30 35 40 45


GOES16 - Canal 13 [10.33 µm] - Temperatura de brillo (°C)
2021/01/15 17:30 [Hora Perú]
2021/01/15 22:30 [Hora GMT]
GOES16 - Canal 13 [10.33 µm] - Temperatura de brillo (°C)
-85 -80 -75 -70 -65 -60 -55 -50 -45 -40 -35 -30 -25 -20 -15 -10 -5 0 5 10 15 20 25 30 35 40 45

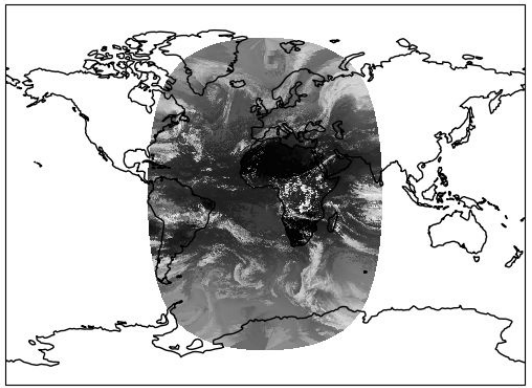
Cartopy

“Cartopy es un paquete de Python diseñado para el procesamiento de datos geoespaciales con el fin de producir mapas y otros análisis de datos geoespaciales.

Cartopy hace uso de las poderosas bibliotecas PROJ.4, NumPy y Shapely e incluye una interfaz programática construida sobre Matplotlib para la creación de mapas de calidad de publicación.”



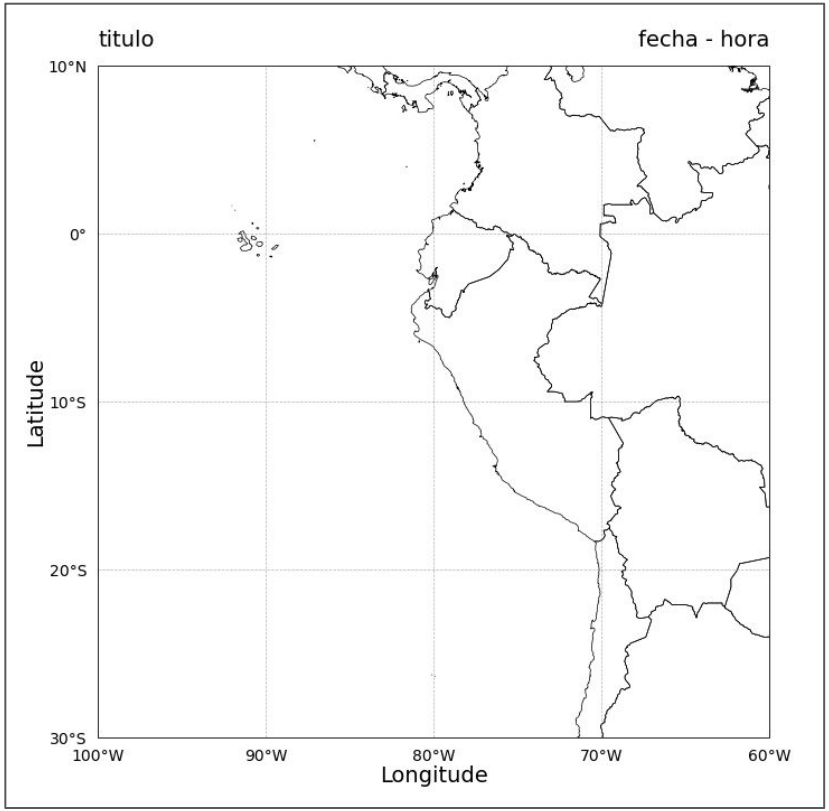
Proyección Robinson



Proyección Miller

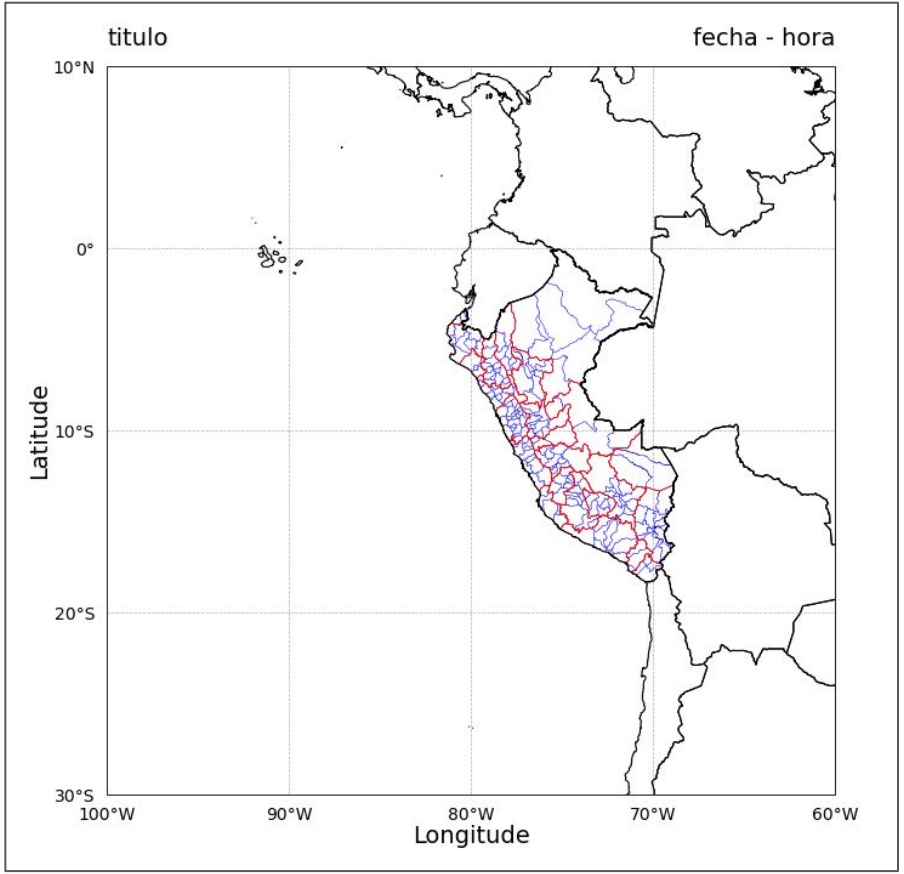
Plantilla para graficar un mapa geográfico

```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter

lonmin, lonmax, latmin, latmax = [-100.0, -60.0, -30.0, 10.0]

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='black', linewidth=0.25)

# Aqui adicionas los campos

ax.set_title('titulo', fontsize=7, loc='left')
ax.set_title('fecha - hora', fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.show()
```



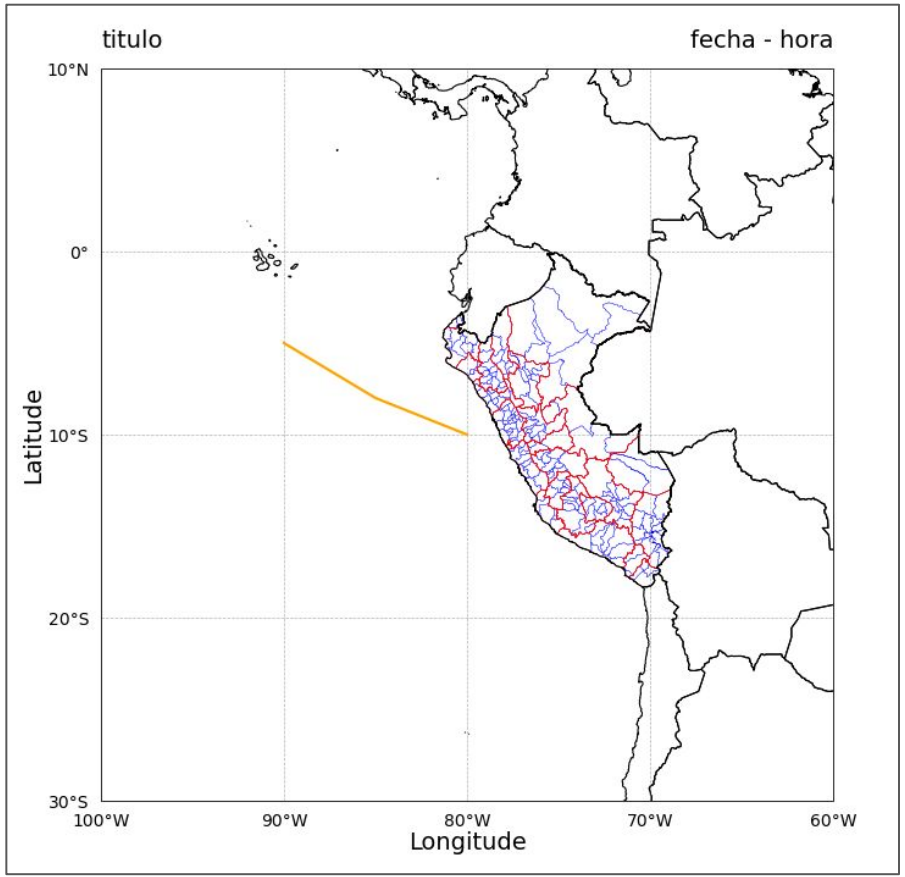
Plantilla para graficar un mapa geográfico, utilizando shapefiles

```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter
import cartopy.io.shapereader as shpreader

lonmin, lonmax, latmin, latmax = [-100.0, -60.0, -30.0, 10.0]

path_shp = '/home/joao/Dropbox/programas/extras/'  # carpeta de los shapefiles

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

reader = shpreader.Reader(path_shp+'prov_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='blue', linewidth=0.1)

reader = shpreader.Reader(path_shp+'depa_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='red', linewidth=0.2)

reader = shpreader.Reader(path_shp+'mundo_corregido.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='black', linewidth=0.4)

# Aqui adicionas los campos

ax.set_title('titulo', fontsize=7, loc='left')
ax.set_title('fecha - hora', fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.show()
```

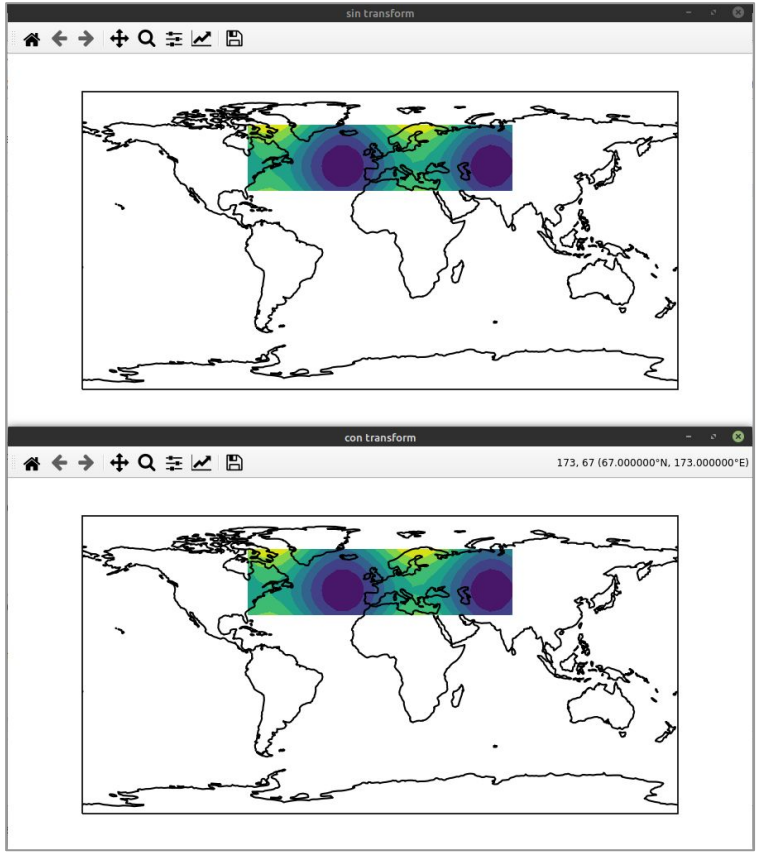


Ejemplo

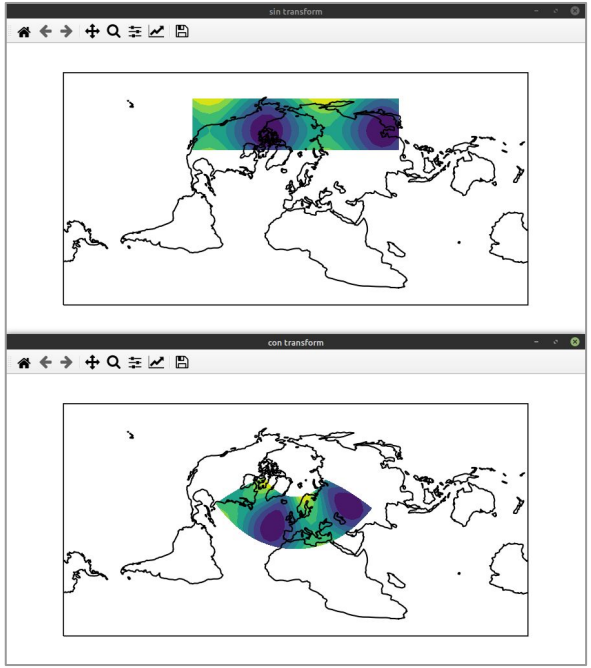
```
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter
import cartopy.io.shapereader as shpreader

lonmin, lonmax, latmin, latmax = [-100.0, -60.0, -30.0, 10.0]

path_shp = '/home/joao/Dropbox/programas/extras/'  # carpeta de los shapefiles

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

reader = shpreader.Reader(path_shp+'prov_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='blue', linewidth=0.1)

reader = shpreader.Reader(path_shp+'depa_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='red', linewidth=0.2)

reader = shpreader.Reader(path_shp+'mundo_corregido.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='black', linewidth=0.4)

ax.plot([-90.0,-85.0,-80.0],[-5.0,-8.0,-10.0], color='orange', linewidth=0.8, transform=ccrs.PlateCarree())

ax.set_title('titulo', fontsize=7, loc='left')
ax.set_title('fecha - hora', fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.show()
```



```
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt

lon = np.linspace(-80, 80, 25)
lat = np.linspace(30, 70, 25)
lon2d, lat2d = np.meshgrid(lon, lat)
data = np.cos(np.deg2rad(lat2d) * 4) + np.sin(np.deg2rad(lon2d) * 4)

fig = plt.figure('sin transform', figsize=(6, 3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8,], projection=ccrs.PlateCarree())
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)

fig = plt.figure('con transform', figsize=(6, 3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8,], projection=ccrs.PlateCarree())
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data, transform=ccrs.PlateCarree())

plt.show()
```



```
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt

lon = np.linspace(-80, 80, 25)
lat = np.linspace(30, 70, 25)
lon2d, lat2d = np.meshgrid(lon, lat)
data = np.cos(np.deg2rad(lat2d) * 4) + np.sin(np.deg2rad(lon2d) * 4)

fig = plt.figure('sin transform', figsize=(6, 3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8,], projection=ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5))
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data)

fig = plt.figure('con transform', figsize=(6, 3), dpi=150)
ax = fig.add_axes([0.1,0.1,0.8,0.8,], projection=ccrs.RotatedPole(pole_longitude=-177.5, pole_latitude=37.5))
ax.set_global()
ax.coastlines()
ax.contourf(lon, lat, data, transform=ccrs.PlateCarree())

plt.show()
```



Explicación

El parámetro **projection** indica la proyección del mapa.
El parámetro **transform** convierte las coordenadas en coordenadas geograficas.

ccrs.PlateCarree() es una proyección plana, donde las coordenadas decimales de latitud y longitud son considerados como unidades.
Ejemplo:
-90.5 longitud decimal es -90.5 unidades
10.25 latitud decimal es 10.25 unidades

Latitud sexagesimal: 40° 42' 51" N = **Latitud decimal**: 40.714
Longitud sexagesimal: 74° 0' 21" O = **Longitud decimal**: -74.006

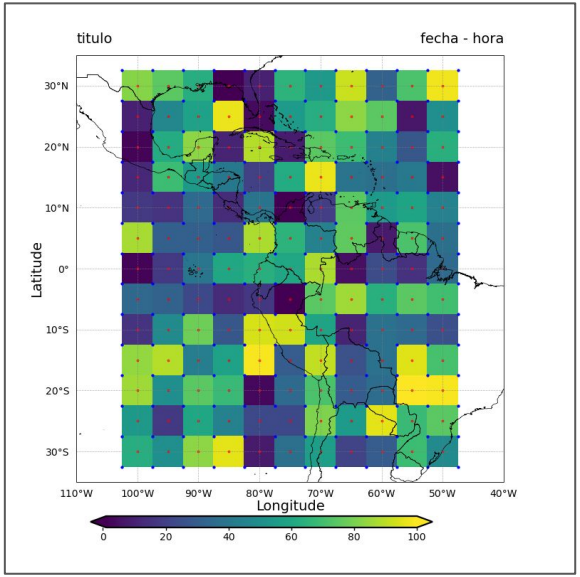
Si no se define **transform**, entonces matplotlib entiende que las coordenadas son unidades.
Ejemplo: 25.2, 89.3, 12.5

Por este motivo, si **projection** = ccrs.PlateCarree(), entonces **transform** no es necesario.

Por otro lado, si **projection** ≠ ccrs.PlateCarree(), entonces es necesario adicionar **transform** al momento graficar el dato.

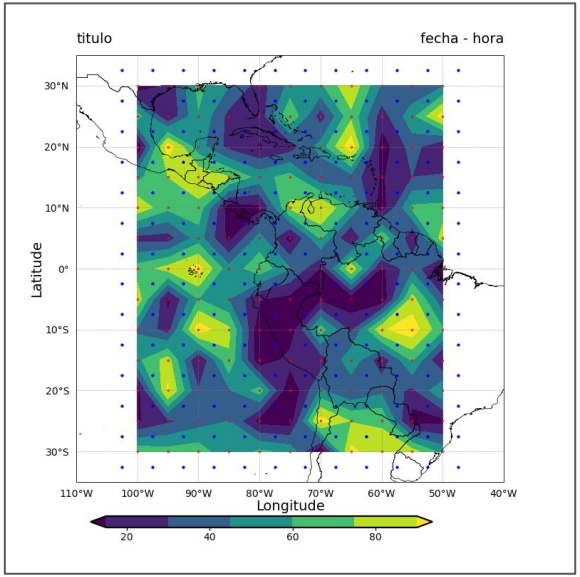
```
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter

lon = np.arange(-100.0, -45.0, 5.0)
lat = np.arange(30.0, -35.0, -5.0)
loncen2d, latcen2d = np.meshgrid(lon, lat)

loncor = np.arange(-100.0, -40.0, 5.0)-2.5
latcor = np.arange(30.0, -40.0, -5.0)+2.5
loncor2d, latcor2d = np.meshgrid(loncor, latcor)

data = np.random.randint(0,101,loncen2d.shape)


lonmin, lonmax, latmin, latmax = [-110.0, -40.0, -35.0, 35.0]


fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='black', linewidth=0.25)

img = ax.pcolormesh(loncor2d, latcor2d, data, transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=[0,20,40,60,80,100], extend='both', orientation='horizontal', cax=fig.add_axes([0.15,0.08,0.6,0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)

pts_cen = ax.scatter(loncen2d, latcen2d, s=0.1, color='red')
pts_cor = ax.scatter(loncor2d, latcor2d, s=0.3, color='blue')

ax.set_title('titulo', fontsize=7, loc='left')
ax.set_title('fecha - hora', fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False, labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.show()
```

Ejercicio:



```
import numpy as np
import cartopy.crs as ccrs
import matplotlib.pyplot as plt
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter

lon = np.arange(-100.0, -45.0, 5.0)
lat = np.arange(30.0, -35.0, -5.0)
loncen2d, latcen2d = np.meshgrid(lon, lat)

loncor = np.arange(-100.0, -40.0, 5.0)-2.5
latcor = np.arange(30.0, -40.0, -5.0)+2.5
loncor2d, latcor2d = np.meshgrid(loncor, latcor)

data = np.random.randint(0,101,loncen2d.shape)


lonmin, lonmax, latmin, latmax = [-110.0, -40.0, -35.0, 35.0]


fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='black', linewidth=0.25)

img = ax.contourf(loncen2d, latcen2d, data, extend='both', transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=[0,20,40,60,80,100], orientation='horizontal', cax=fig.add_axes([0.15,0.08,0.6,0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)

pts_cen = ax.scatter(loncen2d, latcen2d, s=0.1, color='red')
pts_cor = ax.scatter(loncor2d, latcor2d, s=0.3, color='blue')

ax.set_title('titulo', fontsize=7, loc='left')
ax.set_title('fecha - hora', fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False, labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.show()
```

Ejercicio:



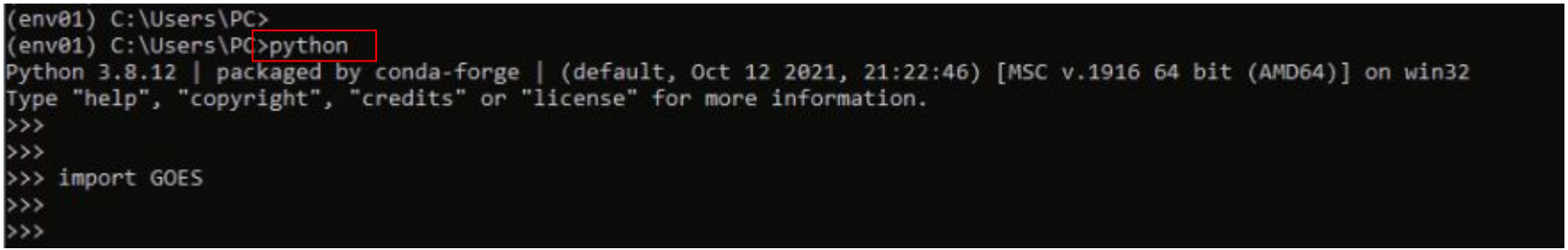
Librería GOES

Librería creada para descargar y manipular datos del GOES-16/17.

Página en github:
https://github.com/joaohenry23/GOES

Dentro de tu ambiente de trabajo, escribe lo siguiente en el terminal:
pip install GOES

Entrar a Python e importar la librería GOES:



Descarga de datos del GOES-16/17

```
import GOES
help(GOES.download)     # muestra la descripción de la función download. Utilizar la tecla “q” para salir.
GOES.show_products()    # muestra el listado de productos disponibles en el amazon.
```

Descarga de un canal del ABI entre un intervalo de tiempo:

```
GOES.download('goes16', 'ABI-L2-CMIPF', DateTimeIni='20200320-203000', DateTimeFin='20200320-210000', channel=['13'], path_out='/home/joao/Downloads/python/') # linux
GOES.download('goes16', 'ABI-L2-CMIPF', DateTimeIni='20200320-203000', DateTimeFin='20200320-210000', channel=['13'], path_out='C:\\Users\\Desktop\\python\\')    # windows
```

Descarga de 4 canales del ABI entre un intervalo de tiempo:

```
GOES.download('goes16', 'ABI-L2-CMIPF', DateTimeIni='20200320-203000', DateTimeFin='20200320-210000', channel=['08-10','13'], path_out='C:\\Users\\Desktop\\python\\')
```

Descarga de datos del GLM entre un intervalo de tiempo:

```
files = GOES.download('goes16', 'GLM-L2-LCFA', DateTimeIni = '20200320-203000', DateTimeFin = '20200320-203200', path_out='C:\\Users\\Desktop\\python\\')
```

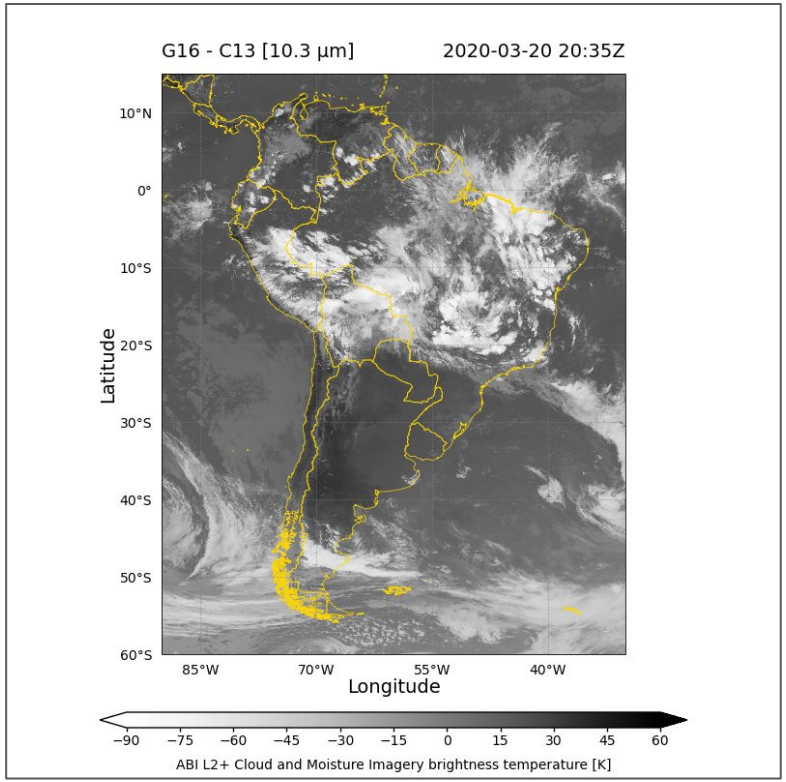
Descarga el producto RRQPEF (tasa de precipitación estimada) entre un intervalo de tiempo:

```
files = GOES.download('goes16', 'ABI-L2-RRQPEF', DateTimeIni = '20200320-203000', DateTimeFin = '20200320-205000', path_out='C:\\Users\\Desktop\\python\\')
```

Aplicaciones de la librería GOES - Imagen del C13

```
import GOES
import numpy as np
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter


domain = [-90.0, -30.0, -60.0, 15.0]
path = '/home/joao/Downloads/python/' #'C:\\Users\\Desktop\\python\\'
ds = GOES.open_dataset(path+'OR_ABI-L2-CMIPF-M6C13_G16_s20200802030177_e20200802039497_c20200802039590.nc')

print(ds) # muestra el contenido del archivo

CMI, Lons, Lats = ds.image('CMI', lonlat='corner', domain=domain)

sat = ds.attribute('platform_ID')
band_id = ds.variable('band_id').data[0]
wl = ds.variable('band_wavelength').data[0]
time = ds.variable('t').data.strftime('%Y-%m-%d %H:%MZ')

cbticks = np.arange(-90,75,15)
```

```
lonmin, lonmax, latmin, latmax = domain

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='gold', linewidth=0.25)

img = ax.pcolormesh(Lons.data, Lats.data, CMI.data-273.15, cmap='Greys', vmin=-90.0, vmax=60.0, transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=cbticks, extend='both', orientation='horizontal', cax=fig.add_axes([0.12, 0.065, 0.76, 0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)
cb.set_label(label='{} [{}]'.format(CMI.long_name, CMI.units), size=5, color='black', weight='normal')
cb.outline.set_linewidth(0.5)

ax.set_title('{} - C{} [{:.1f} µm]'.format(sat, band_id, wl), fontsize=7, loc='left')
ax.set_title(time, fontsize=7, loc='right')

xticks = np.arange(-160, 30, 15)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')

ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)
plt.savefig('/home/joao/Downloads/python/ch13_gris.png')
plt.show()
```



Librería custom_color_palette

Librería creada para crear paletas de colores.

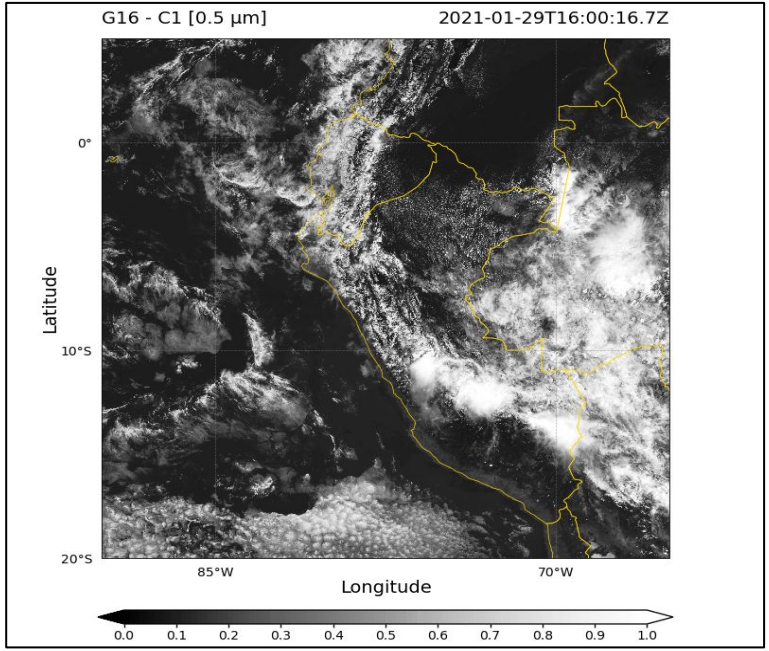
Página en github:
https://github.com/joaohenry23/custom_color_palette

Dentro de tu ambiente de trabajo, escribe lo siguiente en el terminal:
pip install custom-color-palette

Adición de una paleta personalizada

Paleta para canales visibles

```
import custom_color_palette as ccp

paleta = [ plt.cm.Greys_r,  ccp.range(0.0,1.0,0.01) ]

cmap, ticks, norm, bounds = ccp.creates_palette([paleta], extend='both')
cbticks = ccp.range(0.0,1.0,0.1)
```



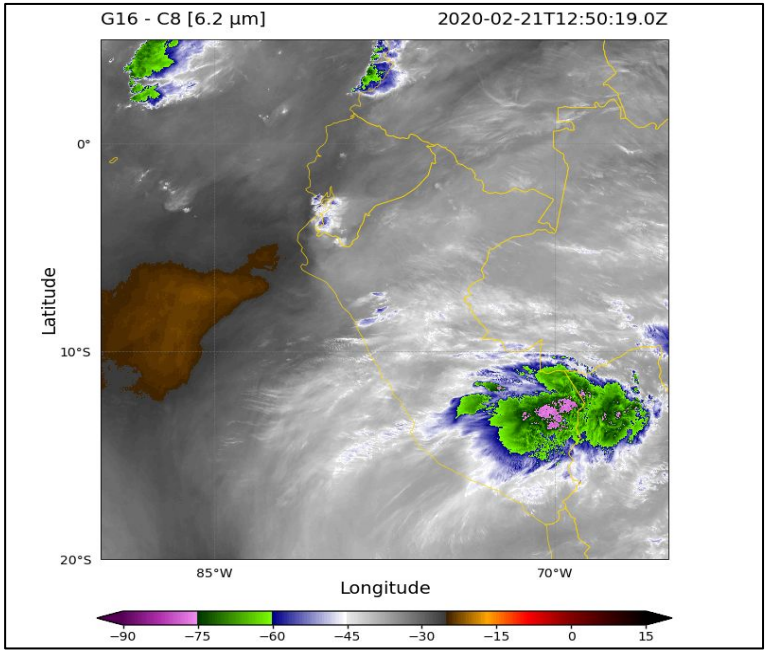
Nombre de las paletas:
https://matplotlib.org/stable/tutorials/colors/colormaps.html

Nombre de los colores:
https://matplotlib.org/stable/gallery/color/named_colors.html

Adición de una paleta personalizada

```
import custom_color_palette as ccp

paleta_1 = [ [(85/255,0/255,84/255),(174/255,46/255,172/255),(239/255,139/255,238/255)],  ccp.range(-90.0,-75.0,0.5) ]
paleta_2 = [ [(0/255,54/255,0/255), 'lawngreen'],  ccp.range(-75.0,-60.0,0.5) ]
paleta_3 = [ ['darkblue', 'white'],  ccp.range(-60.0,-45.0,0.5) ]
paleta_4 = [ [(240/255,240/255,240/255), (60/255,60/255,60/255)],  ccp.range(-45.0,-25.0,0.5),  [ccp.range(-46.0,-25.0,0.5),-45.0,-25.0] ]
paleta_5 = [ [(65/255,36/255,2/255), 'orange', 'red', 'darkred', (63/255,0/255,0/255), 'black'],  ccp.range(-25.0,15.0,0.5) ]

cmap, ticks, norm, bounds = ccp.creates_palette([paleta_1, paleta_2, paleta_3, paleta_4, paleta_5], extend='both')
cbticks = ccp.range(-90,15,15)
```

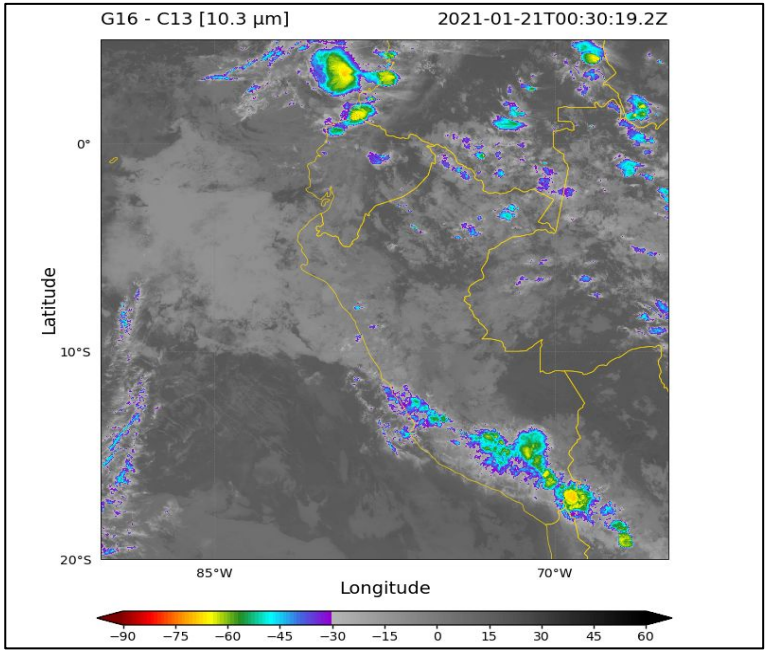
Paleta para canal de vapor de agua



Adición de una paleta personalizada

```
import custom_color_palette as ccp

paleta_1  =  [ ['maroon','red','darkorange','#ffff00','forestgreen','cyan','royalblue',(148/255,0/255,211/255)],  ccp.range(-90.0,-30.0,1.0) ]
paleta_2 = [ plt.cm.Greys,  ccp.range(-30.0,60.0,1.0),  [ccp.range(-90.0,60.0,1.0),-30.0,60.0] ]

cmap, ticks, norm, bounds = ccp.creates_palette([paleta_1,  paleta_2], extend='both')
cbticks = ccp.range(-90,60,15)
```

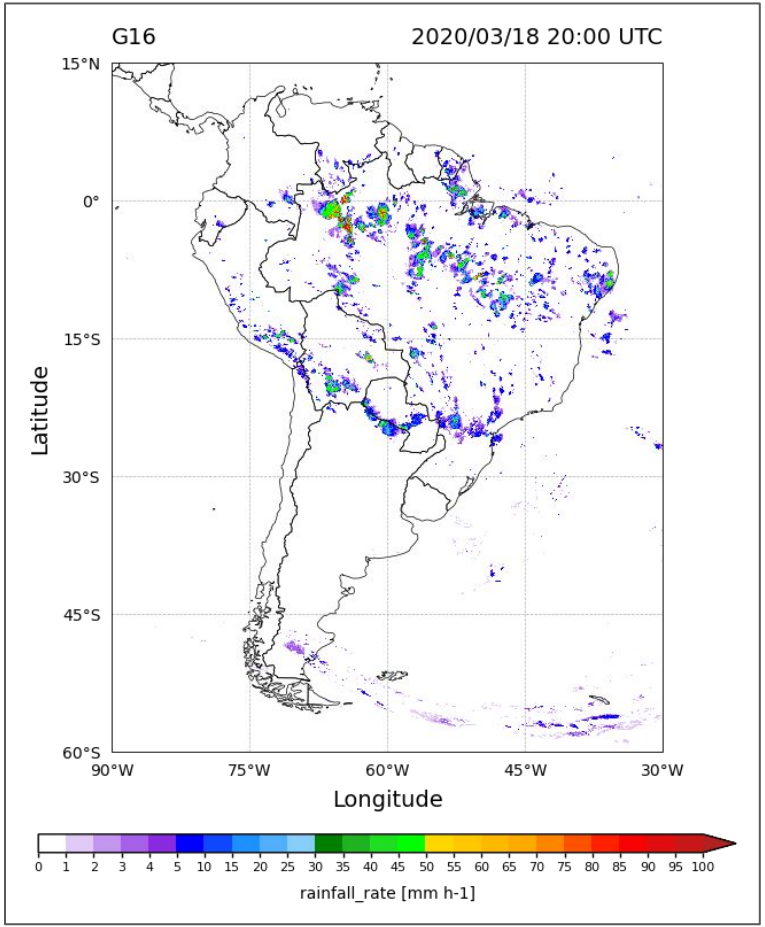
Paleta para canal infrarrojo en ventanas atmosféricas



Adición de una paleta personalizada

```
import custom_color_palette as ccp

paleta_1 = [['white','blueviolet'], [0,1,2,3,4,5]]
paleta_2 = [['blue','dodgerblue','lightskyblue'], [5,10,15,20,25,30]]
paleta_3 = [['green','limegreen','lime'], [30,35,40,45,50]]
paleta_4 = [['gold','orange','red','firebrick'], [50,55,60,65,70,75,80,85,90,95,100]]

cmap, ticks, norm, bounds = ccp.creates_palette([paleta_1, paleta_2, paleta_3, paleta_4], extend='max')
cbticks = ticks
```

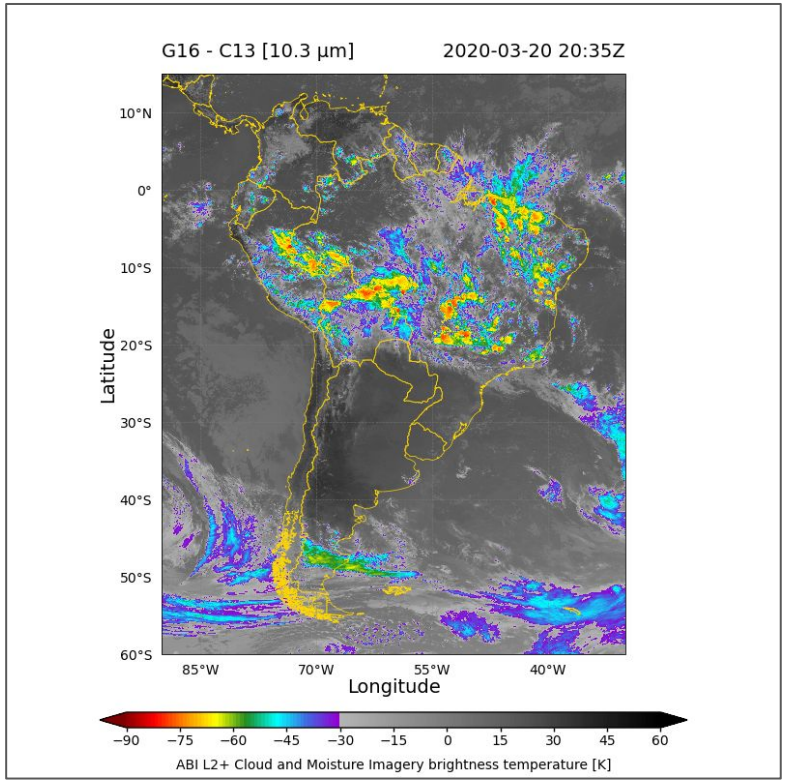
Paleta para precipitación estimada



Aplicaciones de la librería GOES - Imagen del C13

```
import GOES
import numpy as np
import matplotlib.pyplot as plt
import custom_color_palette as ccp
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter


domain = [-90.0, -30.0, -60.0, 15.0]
path = '/home/joao/Downloads/python/'
ds = GOES.open_dataset(path+'OR_ABI-L2-CMIPF-M6C13_G16_s20200802030177_e20200802039497_c20200802039590.nc')

CMI, Lons, Lats = ds.image('CMI', lonlat='corner', domain=domain)

sat = ds.attribute('platform_ID')
band_id = ds.variable('band_id').data[0]
wl = ds.variable('band_wavelength').data[0]
time = ds.variable('t').data.strftime('%Y-%m-%d %H:%MZ')

paleta_1  =  [ ['maroon','red','darkorange','#ffff00','forestgreen','cyan','royalblue',(148/255,0/255,211/255)],  ccp.range(-90.0,-30.0,1.0) ]
paleta_2 = [ plt.cm.Greys,  ccp.range(-30.0,60.0,1.0),  [ccp.range(-90.0,60.0,1.0),-30.0,60.0] ]
cmap, ticks, norm, bounds = ccp.creates_palette([paleta_1,  paleta_2], extend='both')
cbticks = ccp.range(-90,60,15)
```

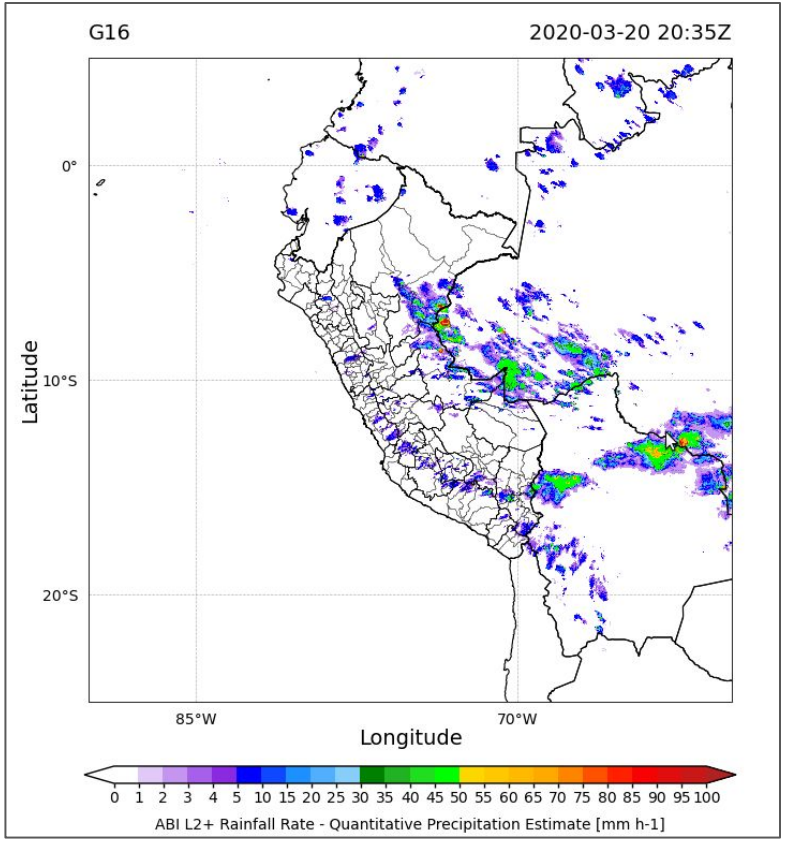
```
lonmin, lonmax, latmin, latmax = domain

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='gold', linewidth=0.25)

img = ax.pcolormesh(Lons.data, Lats.data, CMI.data-273.15, cmap=cmap, norm=norm, transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=cbticks, extend='both', orientation='horizontal', cax=fig.add_axes([0.12, 0.065, 0.76, 0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)
cb.set_label(label='{} [{}]'.format(CMI.long_name, CMI.units), size=5, color='black', weight='normal')
cb.outline.set_linewidth(0.5)

ax.set_title('{} - C{} [{:.1f} µm]'.format(sat, band_id, wl), fontsize=7, loc='left')
ax.set_title(time, fontsize=7, loc='right')

xticks = np.arange(-160, 30, 15)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')

ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)
plt.savefig('/home/joao/Downloads/python/ch13_color.png')
plt.show()
```



Aplicaciones de la librería GOES - Precipitación estimada

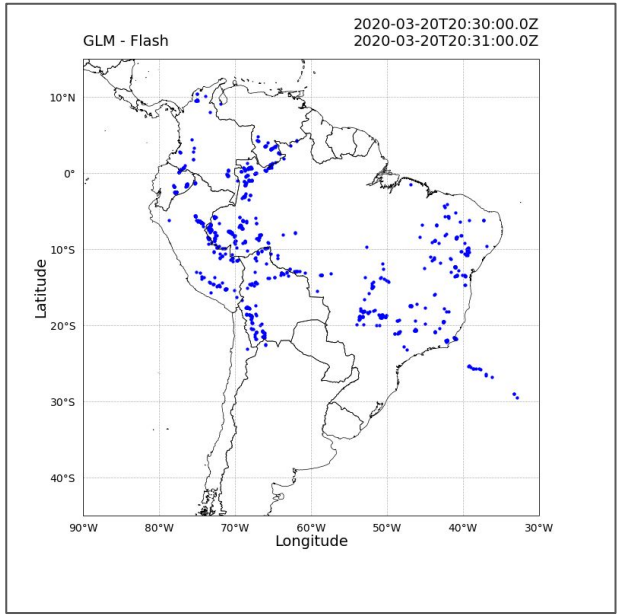
```
import GOES
import numpy as np
import matplotlib.pyplot as plt
import custom_color_palette as ccp
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter
import cartopy.io.shapereader as shpreader

path_shp = '/home/joao/Dropbox/programas/extras/'  # carpeta de los shapefiles
domain = [-90.0, -60.0, -25.0, 5.0]
path = '/home/joao/Downloads/python/'
ds = GOES.open_dataset(path+'OR_ABI-L2-RRQPEF-M6_G16_s20200802030177_e20200802039485_c20200802040007.nc')

RRQPE, Lons, Lats = ds.image('RRQPE', lonlat='corner', domain=domain)

sat = ds.attribute('platform_ID')
time = ds.variable('t').data.strftime('%Y-%m-%d %H:%MZ')

paleta_1 = [['white','blueviolet'], [0,1,2,3,4,5]]
paleta_2 = [['blue','dodgerblue','lightskyblue'], [5,10,15,20,25,30]]
paleta_3 = [['green','limegreen','lime'], [30,35,40,45,50]]
paleta_4 = [['gold','orange','red','firebrick'], [50,55,60,65,70,75,80,85,90,95,100]]

cmap, ticks, norm, bounds = ccp.creates_palette([paleta_1, paleta_2, paleta_3, paleta_4], extend='max')
cbticks = ticks
```

```
lonmin, lonmax, latmin, latmax = domain

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

reader = shpreader.Reader(path_shp+'prov_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='black', linewidth=0.1)

reader = shpreader.Reader(path_shp+'depa_18_ign.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='black', linewidth=0.2)

reader = shpreader.Reader(path_shp+'mundo_corregido.shp')
ax.add_geometries(reader.geometries(), ccrs.PlateCarree(), facecolor='none', edgecolor='black', linewidth=0.4)

img = ax.pcolormesh(Lons.data, Lats.data, RRQPE.data, cmap=cmap, norm=norm, transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=cbticks, extend='both', orientation='horizontal', cax=fig.add_axes([0.12, 0.065, 0.76, 0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)
cb.set_label(label='{} [{}]'.format(RRQPE.long_name, RRQPE.units), size=5, color='black', weight='normal')
cb.outline.set_linewidth(0.5)

ax.set_title('{}'.format(sat), fontsize=7, loc='left')
ax.set_title(time, fontsize=7, loc='right')

xticks = np.arange(-160, 30, 15)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False,
               labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')

ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')

ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)
plt.savefig('/home/joao/Downloads/python/RRQPE.png')
plt.show()
```



Aplicaciones de la librería GOES - ubicación de los flashes

```
import numpy as np
import GOES as GOES
import matplotlib.pyplot as plt
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter

path = '/home/joao/Downloads/python/'

flist = [path+'OR_GLM-L2-LCFA_G16_s20200802030000_e20200802030200_c20200802030227.nc',
         path+'OR_GLM-L2-LCFA_G16_s20200802030200_e20200802030400_c20200802030430.nc',
         path+'OR_GLM-L2-LCFA_G16_s20200802030400_e20200802031000_c20200802031031.nc']

ds = GOES.open_mfdataset(flist)

time_start = ds.attribute('time_coverage_start')[0]
time_end = ds.attribute('time_coverage_end')[-1]

flash_lon = ds.variable('flash_lon')
flash_lat = ds.variable('flash_lat')

lonmin, lonmax, latmin, latmax = [-90.0, -30.0, -45.0, 15.0]

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='black', linewidth=0.25)

pts = ax.scatter(flash_lon.data, flash_lat.data, s=0.3, color='blue', transform=ccrs.PlateCarree())

ax.set_title('GLM - Flash', fontsize=7, loc='left')
ax.set_title('{}\n{}'.format(time_start,time_end), fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False, labelbottom=True, labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')
ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)
plt.savefig('/home/joao/Downloads/python/flash_coordenadas.png')
plt.show()
```



Aplicaciones de la librería GOES - densidad de flash

```
import numpy as np
import GOES as GOES
import custom_color_palette as ccp
import matplotlib.pyplot as plt
import matplotlib as mpl
import cartopy.crs as ccrs
from cartopy.feature import NaturalEarthFeature
from cartopy.mpl.ticker import LatitudeFormatter, LongitudeFormatter

domain = [-100.0,-20.0,-40.0,40.0]

path = '/home/joao/Downloads/python/'

flist = [path+'OR_GLM-L2-LCFA_G16_s20200802030000_e20200802030200_c20200802030227.nc',
         path+'OR_GLM-L2-LCFA_G16_s20200802030200_e20200802030400_c20200802030430.nc',
         path+'OR_GLM-L2-LCFA_G16_s20200802030400_e20200802031000_c20200802031031.nc']


ds = GOES.open_mfdataset(flist)

time_start = ds.attribute('time_coverage_start')[0]
time_end = ds.attribute('time_coverage_end')[-1]

flash_lon = ds.variable('flash_lon')
flash_lat = ds.variable('flash_lat')

gridmap_lon, gridmap_lat = GOES.create_gridmap(domain, PixResol=8.0)

dens = GOES.accumulate_in_gridmap(gridmap_lon, gridmap_lat, flash_lon, flash_lat)

paleta1 = [ ['black'], [0,1] ]
paleta2 = [ ['yellow','darkorange','red','firebrick'], [1,5,10,15,20,25,30,40,50,60,70] ]
cmap, ticks, norm, bounds = ccp.creates_palette([paleta1, paleta2], extend='max', lower_color='black', upper_color='maroon')
cbticks = ticks
```

```
lonmin, lonmax, latmin, latmax = domain

fig = plt.figure('map', figsize=(4,4), dpi=200)
ax = fig.add_axes([0.1, 0.16, 0.80, 0.75], projection=ccrs.PlateCarree())
ax.outline_patch.set_linewidth(0.3)

l = NaturalEarthFeature(category='cultural', name='admin_0_countries', scale='10m', facecolor='none')
ax.add_feature(l, edgecolor='white', linewidth=0.25)

img = ax.pcolormesh(gridmap_lon.data, gridmap_lat.data, dens.data, cmap=cmap, norm=norm, transform=ccrs.PlateCarree())

cb = plt.colorbar(img, ticks=cbticks, orientation='horizontal', extend='max', cax=fig.add_axes([0.12, 0.08, 0.76, 0.02]))
cb.ax.tick_params(labelsize=5, labelcolor='black', width=0.5, length=1.5, direction='out', pad=1.0)
cb.set_label(label='{}'.format('número de flash'), size=5, color='black', weight='normal')
cb.outline.set_linewidth(0.5)

ax.set_title('Densidad de flash', fontsize=7, loc='left')
ax.set_title('{}\n{}'.format(time_start,time_end), fontsize=7, loc='right')

xticks = np.arange(-180, 0, 10)
ax.set_xticks(xticks, crs=ccrs.PlateCarree())
ax.xaxis.set_major_formatter(LongitudeFormatter(dateline_direction_label=True))
ax.set_xlabel('Longitude', color='black', fontsize=7, labelpad=0.5)

yticks = np.arange(90, -100, -10)
ax.set_yticks(yticks, crs=ccrs.PlateCarree())
ax.yaxis.set_major_formatter(LatitudeFormatter())
ax.set_ylabel('Latitude', color='black', fontsize=7, labelpad=0.5)

ax.tick_params(left=True, right=True, bottom=True, top=True, labelleft=True, labelright=False, labelbottom=True,
labeltop=False, length=0.0, width=0.05, labelsize=5.0, labelcolor='black')
ax.gridlines(xlocs=xticks, ylocs=yticks, alpha=0.6, color='gray', draw_labels=False, linewidth=0.25, linestyle='--')
ax.set_xlim(lonmin, lonmax)
ax.set_ylim(latmin, latmax)

plt.savefig('/home/joao/Downloads/python/flash_dens.png')
plt.show()
```