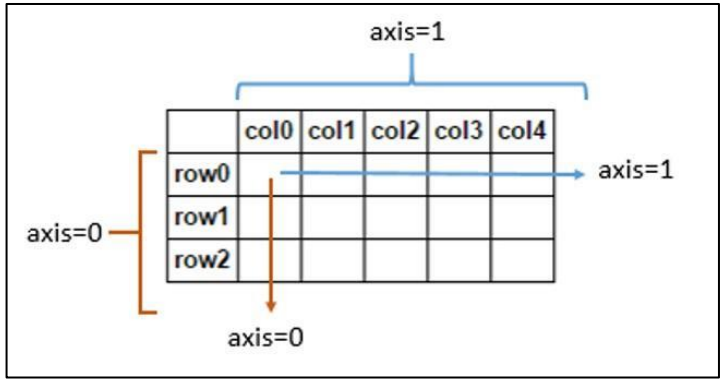
PYTHON APLICADO A LA CLIMATOLOGÍA

Bch. Javier Chiong Ravina

Xarray

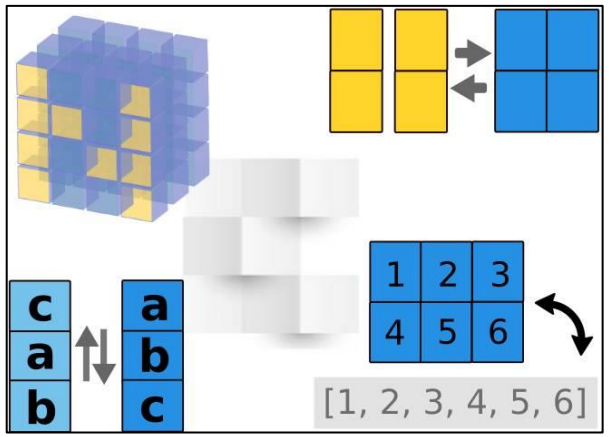
- Es una librería que trabaja con matrices etiquetadas multidimensionales(N-Dimensionales).
- Numpy provee la estructura fundamental de datos y el API para trabajar con matrices N-dimensionales.
- Inspirado en Pandas cuyo enfoque son los datos tabulares etiquetados.

Pandas



https://programmerclick.com/article/28481749963/

Numpy



https://python-para-impacientes.blogspot.com/2019/11/convertir-copiar-ordenar-unir-y-dividir.html

Instalación

conda install -c conda-forge xarray

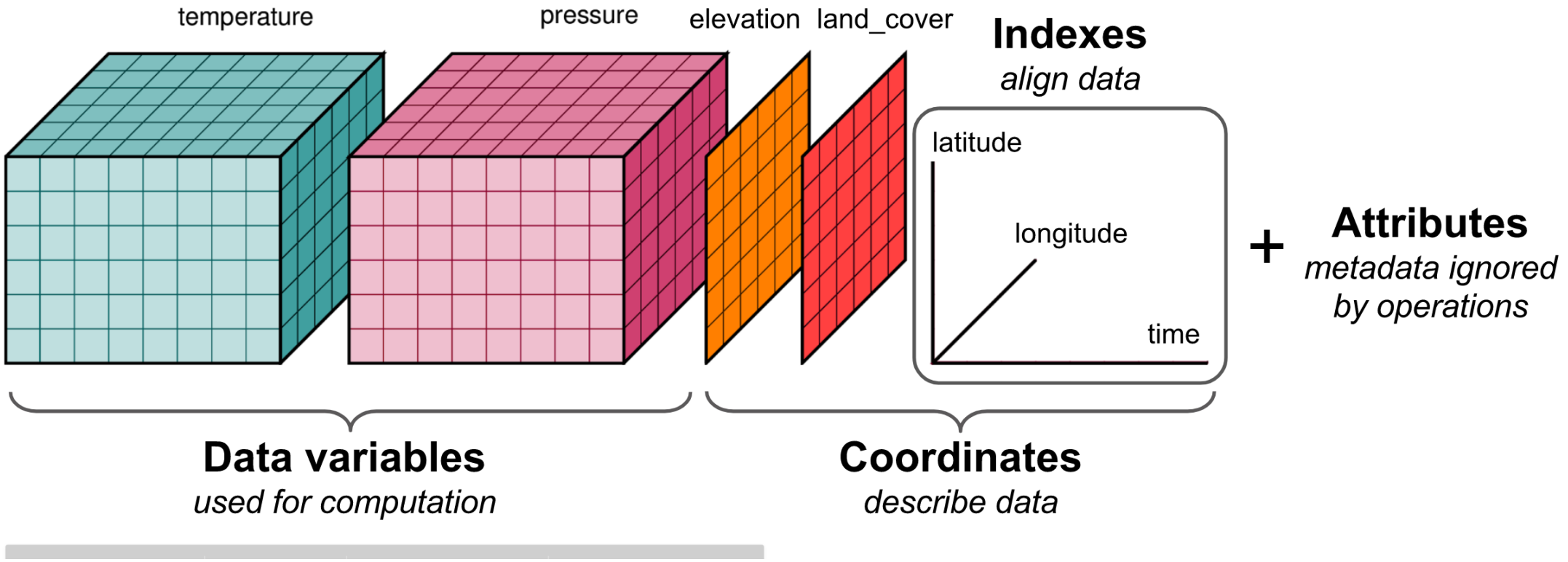
conda install xarray

Dependencias necesarias (http://xarray.pydata.org/en/stable/getting-started-guide/installing.html)

- Netcdf4
- Cfgrib
- Rasterio
- Dask
- Bottleneck

Características

- El mundo real no solo son matrices de números en cada punto, ellos poseen etiquetas que guardan(codifican) información sobre como esos números tienen asignados ubicaciones en el espacio, tiempos, etc.
- Es configurado particularmente para trabajar con datos NetCDF, puede trabajar con formato GRIB.



Estructura

- **DataArray**: Implementada de una matriz N-Dimensional etiquetada, es una generalización de pandas.series. Contiene variables multidimensionales individuales y sus coordenadas.
- **Dataset**: Es una base de datos de matrices. Es como un contenedor(diccionario) de objetos DataArray alineados a lo largo de cualquier numero de dimensiones compartidas, en Xarray tiene un propósito similar al pandas.Dataframe. Contiene múltiples variables que potencialmente comparten las mismas coordenadas.
- Es posible extraer matrices por nombre, es posible seleccionar o combinar datos de una dimensión en todas las matrices simultáneamente.

Estructura de un DataArray

Atributos escenciales:

- data: numpy.ndarray o dask.array conteniendo los valores del array.
- dims: nombres de dimensión para cada eje. Ej: (x,y,z)(lat,lon,time)
- coords: Contenedor de arrays(coordenadas) que etiqueta cada punto.
- attrs: Es un OrderedDict que contiene atributos/metadata arbitrarios(como las unidades)
- name: nombre arbitrario del array

Ejercicio 1

```
#ejercicio xarray
import xarray as xr
# cargar  el archivo
ds = xr.open_dataset("C:/Users/drago/Desktop/CURSO_PY/X190.236.8.122.323.19.59.51.nc", engine="netcdf4")
```

engine, opciones= "netcdf4", "scipy", "pydap", "h5netcdf", "pynio", "cfgrib", "pseudonetcdf", "zarr" .
Si no se proporciona, por defecto se elige en base a las dependencias disponibles, con preferencia por "netcdf4"

http://xarray.pydata.org/en/stable/generated/xarray.open_dataset.html

Explorar atributos esenciales de un DataArray

Ejercicio 1 (continuación)

```
##################################################
######## EXPLORAR DATASET ########################
ds.info() #resumen conciso de variables y atributos
# explorar las variables del archivo
ds.data_vars
# explorar dimensiones del dataset
ds.dims
# dataset coordenadas
ds.cords
# explorar la metadata del dataset.
ds.attrs
# Extraer la variable/datarray air
ds["air"]
# extrayendo un dataArray coordenada de .coords
ds.coords["level"] # coords: lat, lon, level, time
```

Estructura de un Dataset

- data_dims : OrderedDict de objetos DataArray correspondiendo a las variables del dato.
- dims : diccionario que asigna los nombres de las dimensiones a la longitud fija de cada dimensión. (e.g., {lat: 6, lon: 6, time: 8}).
- coords : contenedor tipo Dict de arrays (coordenadas) que etiqueta cada punto
- attrs: metadata arbitraria perteneciente al Dataset

Explorar atributos esenciales de un Dataset

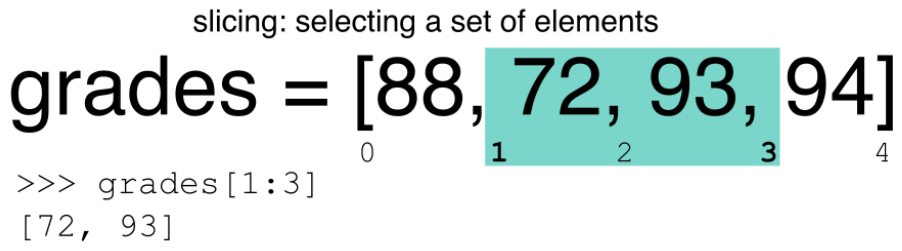
Ejercicio 1 (continuación)

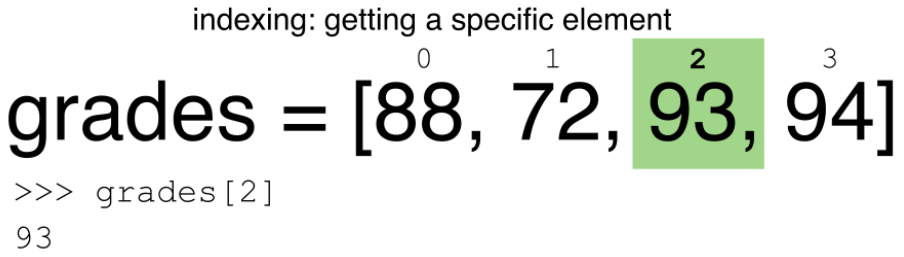
```
##############################################
####### EXPLORAR DATARRAY ####################

# el actual (numpy) array

ds.air.data

# dimensiones del dataarray/variable

ds.air.dims

# coordenadas del datarray/variable

ds.air.coords

# atributos del dataarray/variable

ds.air.attrs

# extrayendo un variable coordenada

ds.air.lon

# Agregar algun atributo arbitrario sobre el datarray
ds.air.attrs["atributo extra"] = "datos experimentales"

ds.air.attrs
```

Indexación y Segmentación

Basado en Pandas

- .isel: Devuelve un nuevo conjunto de datos en base a las dimensiones especificadas.
- .sel: Devuelve un nuevo conjunto de datos en base a las etiquetas especificadas



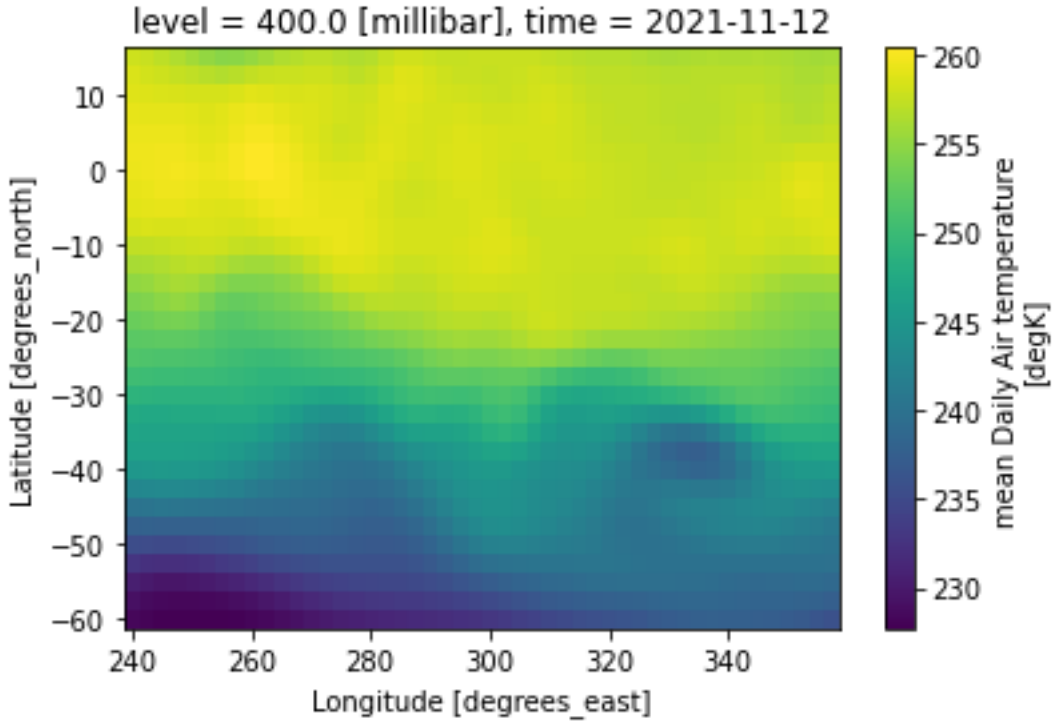


https://datacarpentry.org/python-ecology-lesson-es/03-index-slice-subset/

Plot básico con Indexación y Segmentación

Ejercicio 1 (continuación)

```
#########################################################
####### PLOT VARIABLE ###################################
# seleccionar una variable
ds2 = ds.air[:]
#recoge la informacion del dataset
ds2
# Plot
# indexacion
# usando .isel = devuelve un conjunto de datos en funcion
# de dimensiones especificadas
ds2.isel(time=4, level=2).plot()
# usando .sel = devuelve un conjunto datos en funcion
# de etiquetas especificadas
ds2.sel(time='2021-11-12', level='400').plot()
```



Operaciones y Cálculos

Xarray soporta muchos de los métodos de agregación que tiene numpy. Una lista parcial incluye: all, any, argmax, argmin, max, mean, median, min, prod, sum, std, var.

Mientras que la sintaxis de numpy requeriría ejes escalares, xarray puede utilizar nombres de dimensiones:lat, lon y tiempo.

Operaciones y Cálculos

- groupby : Juntar los datos en grupos y reducirlos
- resample : Groupby especializado en ejes temporales. Puede reducir o aumentar el tamaño de los datos.
- rolling : Operar en ventanas moviles de sus datos, por ejemplo, ejecutar la media móvil.
- coarsen : Reduce la muestra de sus datos
- weighted : Pondera los datos antes de aplicar las reducciones

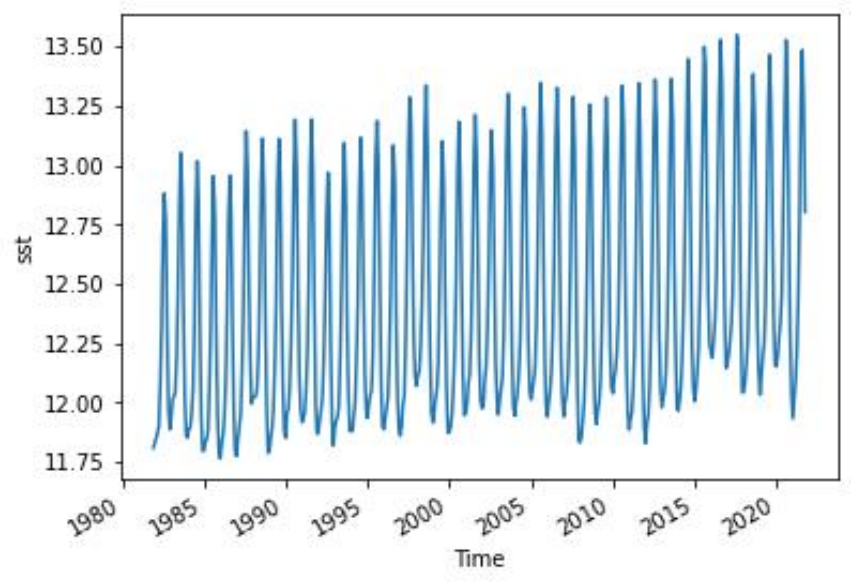
Ejercicio 2

```
#EJERCICIO 2
import xarray as xr
# CARGAR ARCHIVO
ds = xr.open_dataset("C:/Users/drago/Desktop/CURSO_PY/sst.mnmean.nc", engine="netcdf4")
da = ds["sst"]
da
#OPERACIONES ARITMETICAS
da + 273.15
#ESTADISTICA BASICA
da_mean = da.mean(dim="time")
da_mean= da.std(dim=["lat", "lon"]).plot()
# OPERACIONES ARITMETICAS
x = da - da_mean
```

Ejercicio 2 (continuación)

```
#CALCULO DE ALTO NIVEL
ds
#GROUPBY # AGRUPAR POR ESTACIONES
ds.groupby("time.season")
ts=ds.groupby("time.season")
# POR DIAS DE LA SEMANA
ds.groupby("time.dayofweek")
# APLICAR ESTADISTICA EN EL AGRUPADO
seasonal_mean = ds.groupby("time.season").mean()
seasonal_mean
```

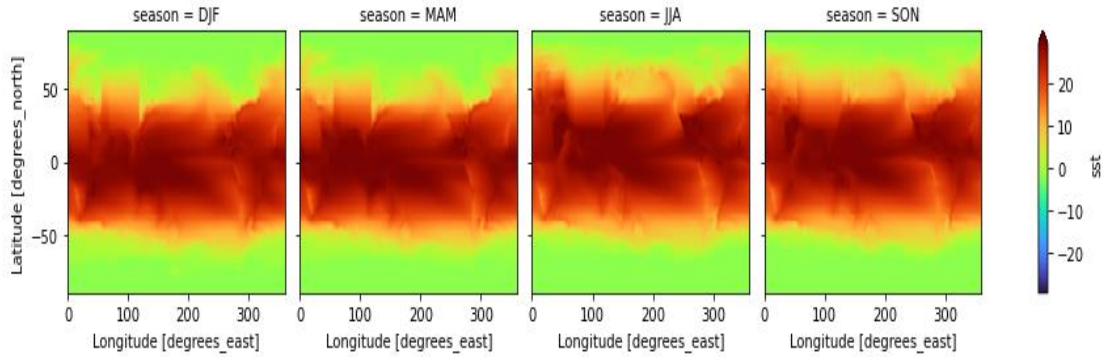
Ejercicio 3 (plot)




```
import xarray as xr

from matplotlib import pyplot as plt

# cargar el archivo

ds =
xr.open_dataset("C:/Users/drago/Desktop/CURSO_PY/sst.m
nmean.nc", engine="netcdf4")

ds

da = ds["sst"]

da

# .isel = para seleccionar data por posicion o dimensiones
ds.sst.isel(time=0).plot()

# .sel = para seleccionar data por etiquetas (lat,lon,time)
ds.sst.sel(time='1982-01-01').plot()

#para una coordenada puntual

ds.sst.sel(lat=-40.5, lon=300.5).plot()

#aplicando estadisticas basicas mean, std, var
ds.sst.mean(dim=('lon', 'lat')).plot()
```

Ejercicio 3 (continuación)

```
# LAS ESTACIONES ESTAN DESORDENADAS (ORDENADOS ALFABETICAMENTE).
# PARA ORDENAR SE SOLUCIONA USANDO . reindex
seasonal_mean = seasonal_mean.reindex(season=["DJF", "MAM", "JJA", "SON"])
seasonal_mean
#PLOTEAR
seasonal_mean.sst.plot(col="season", robust=True, cmap="turbo")
#METODOS DE AGREGACION/REDUCCION #RESAMPLE
# POR FRECUENCIA BI-MENSUAL
ds.sst.resample(time="2MS").mean()
#ROLLING
#UNA MEDIA MOVIL CON  UNA VENTANA DE TAMAÑO 7
ds.sst.rolling(time=7).mean()
```



Ejercicio 4 Calcular el índice ONI

Pasos para calcular el indice ONI

- (a) Calcular el promedio aritmético mensual del área total de la region 3.4
- (b) Calcular la climatología (ej., 1986-2015) del área total de la region 3.4.
- (c) Calcular la anomalía restando los valores del promedio aritmético mensual con la climatología del área total de la region 3.4.
- (d) Suavizar las anomalías con una media movil de orden 3