Download ERA5 data with python

Code from Diego Jiménez de la Cuesta-Otero for python, using the Climate Data Storage API (cdsapi)

getERA5.py

```
import cdsapi   # Climate data storage API
import calendar # Calendar module

# I. Section that the user can change

# You can give it a fancy interface to use from command line, if you
have time
# to program this.

# I.0. Pressure levels or one-level fields
# Possible values are "prl" for pressure levels or "sfc" for one-level
fields.

levs="prl"

# I.1. Format of the output
# Possible values are "grib" or "netcdf".

formato="grib"

# I.2. Lat-Lon box (The extreme lats and lons of your box)

N_lat=27.5
S_lat=11.5
W_lon=-112
E_lon=-88

# I.3. Resolution (In both directions and not below 0.25 degrees [~ 30
km])

grid_lat=0.25
grid_lon=0.25

# I.4. Pressure levels to download
# If levs="sfc" selected, this is ignored.

prls=[
"1","2","3","5","7","10","20","30","50","70","100","125","150","175","2
00",
"225","250","300","350","400","450","500","550","600","650","700","750"
,
 "775","800","825","850","875","900","925","950","975","1000"
]
```

```
# Note: Here I download the data in all pressure levels.
# I.5. Date (gives the limits of your request in years, months and
days)

year_ini=2015   # Initial year
year_fin=2019   # Final year
month_ini=3     # Initial month
month_fin=2     # Final month
day_ini=1       # Initial day
day_fin=calendar.monthlen(year_fin,month_fin) # Final day

# Note: I used monthlen to calculate the last day of a month in a given
year.
# I.6. Time (which timesteps you want to download)

times=[ "{0:02d}:00".format(hh) for hh in range(24) ]

# Note: Here I download all the 24 hours of each day.
# I.7. Variables in pressure levels by long name (Check ERA5
documentation)
# If levs="sfc" selected, this is ignored.

prl_vars=[
 "u_component_of_wind","v_component_of_wind",
 "geopotential",
 "relative_humidity","specific_humidity","temperature"
]

# I.8. One-level Variables by long name (Check ERA5 documentation)
# If levs="prl" selected, this is ignored.

sfc_vars=[
 '10m_u_component_of_wind','10m_v_component_of_wind',
 'sea_surface_temperature','skin_temperature',
 '2m_dewpoint_temperature','2m_temperature','surface_pressure'
 'soil_temperature_level_1','soil_temperature_level_2',
 'soil_temperature_level_3','soil_temperature_level_4',
 'volumetric_soil_water_layer_1','volumetric_soil_water_layer_2',
 'volumetric_soil_water_layer_3','volumetric_soil_water_layer_4',
 'sea_ice_cover','snow_depth',
 'land_sea_mask','mean_sea_level_pressure'
]

# I.9. Path and prefix
# Here you should give the path where you will store files.
# Also, you should give the file's prefix. Always the files have the
date as
# suffix. Depending on your selection of levs, the program will take
# prefix_prl or prefix_sfc to form file names.
```

```
path="./"
prefix_prl="era5_regio_prl"
prefix_sfc="era5_regio_sfc"

# HERE ENDS THE SECTIONS THAT A NORMAL USER SHOULD MODIFY.
# II. Construction of the file names
# It uses the user defined prefixes and uses substitution fields to let
the
# program fill the spaces with the date information.

if formato == "grib":
 extension=".grb"
elif formato == "netcdf":
 extension=".nc"

name_sfc=path+prefix_sfc+"_{0:04d}{1:02d}{2:02d}"+extension
name_prl=path+prefix_prl+"_{0:04d}{1:02d}{2:02d}"+extension

# III. Construction of date dictionaries
# The program will download the data in daily files. Thus, it needs to
know
# how many days each month has. I am sure there are more direct
solutions but
# this script was made for didactical purpouses. You can change this if
you
# like.

years=list(range(year_ini,year_fin+1))
months={}
days={}
for year in years:
 sizey=12
 if year == years[0]:
  months[year]=list(range(month_ini,sizey+1))
 elif year == years[-1]:
  months[year]=list(range(1,month_fin+1))
 else:
  months[year]=list(range(1,sizey+1))
 days[year]={}
 for month in months[year]:
  sizem=calendar.monthlen(year,month)
  if year == years[0]:
   if month == months[year][0]:
    days[year][month]=list(range(day_ini,sizem+1))
   else:
    days[year][month]=list(range(1,sizem+1))
  elif year == years[-1]:
   if month == months[year][-1]:
    days[year][month]=list(range(1,day_fin+1))
   else:
```

```
    days[year][month]=list(range(1,sizem+1))
  else:
   days[year][month]=list(range(1,sizem+1))

# IV. Dictionary for the CDS API.
# It constructs the options for both levs options.

options={}
options["prl"]={
 "product_type"   : "reanalysis",
 "variable"       : prl_vars,
 "pressure_level" : prls,
 "year"           : None,
 "month"          : None,
 "day"            : None,
 "time"           : times,
 "area"           : [N_lat,W_lon,S_lat,E_lon],
 "grid"           : [grid_lat,grid_lon],
 "format"         : formato
}

options["sfc"]={
 "product_type"    : "reanalysis",
 "variable"        : sfc_vars,
 "year"            : None,
 "month"           : None,
 "day"             : None,
 "time"            : times,
 "area"            : [N_lat,W_lon,S_lat,E_lon],
 "grid"            : [grid_lat,grid_lon],
 "format"          : formato
}

# V. Execution
# The program now makes a request to the CDS for each day in the range
with
# the provided information. Note: CDS only accepts one request from a
given
# user. That means you cannot download in parallel, unless you have an
account
# with special access.

for year in days.keys():                          # For each year...
 for month in days[year].keys():                  # For each month...
  for day in days[year][month]:                   # For each day...
   options[levs]["year"]="{0:04d}".format(year)   # Set date in dict
   options[levs]["month"]="{0:02d}".format(month)
   options[levs]["day"]="{0:02d}".format(day)
   c=cdsapi.Client()                              # Initialise CDS API
client
```

```
   if levs == "sfc":                                  # Set download file
name
    filname=name_sfc.format(year,month,day)           # and dataset
depending on
    dataset="reanalysis-era5-single-levels"           # levs value
   elif levs == "prl":
    filname=name_prl.format(year,month,day)
    dataset="reanalysis-era5-pressure-levels"
   c.retrieve(dataset,options[levs],filname)          # Make the actual
request
```