中 国 海 洋 大 学

Ocean University of China

项 目 报 告

项目名称: **ARGO 数据可视化**

小组名称: **2021 秋周一三节上机 ARGO 数据小组**

组　　长: **余汉学(20010006082)**

组　　员: **周修远(20010006094)**

周子涵(20010006100)

赵泽旭(20110031029)

中 国 海 洋 大 学

Ocean University of China

项 目 报 告

成绩	

项目报告

一、项目任务

1. 总体目标：

- 绘制 Argo 全球数据的可视化图像

2. 功能要求：

- 能够下载 Argo 数据
- 能够处理 NetCDF4 格式文件，并且从中提取所需数据
- 能够使用 Matplolib 可视化图像
- 拥有 GUI 界面
- 可选动态化结果

3. 程序运行方式：

- 已经使用 Pyinstaller 打包为 exe 程序

4. 程序资源：

- Python 3.9.7 及所需模块
- BOA_ARGO 的.nc(NetCDF4)文件

二、背景资料

1. ARGO

Argo 是一个海洋观测系统的名称，可为气候、天气、海洋学及渔业研究提供实时海洋观测数据。

该观测系统由大量布放在全球海洋中小型、自由漂移的自动探测设备（Argo 剖面浮标）组成。大部分浮标在 1000 米漂移（被称为停留深度），每隔 10 天下潜到 2000 米深度并上浮至海面，在这过程中进行海水温度和电导率等要素的测量，由此可计算获得海水盐度和密度。观测数据通过卫星传送到地面科研人员，并向所有人免费、无限制提供。Argo 计划的名字起源于希腊神话中勇士伊阿宋（Jason）和阿尔戈英雄（Argonauts）寻找金色羊毛时所乘的船。之所以选用该名字，意在强调 Argo 计划与杰森卫星高度计（Jason-2 (Ocean Surface Topography Mission)）的相互补充。Argo 计划通过全球 30 多个国家的合作来维持一个全球海洋观测网，使任何国家可以探测海洋环境要素。Argo 是 Global Ocean Observing System (GOOS)的重要组成部分。

2. NetCDF

网络通用数据格式（英语：Network Common Data Form，NetCDF）是一种自描述、与机器无关、基于数组的科学数据格式，同时也是支持创建、访问和共享这一数据格式的函数库。该项目主页位于美国大气科学研究大学联盟（UCAR）的 Unidata 规划网站。它也是 NetCDF 软件、标准开发、更新等的主要来源。NetCDF 格式是一种开放标准。NetCDF 的经典格式和 64 位偏移量格式是开放地理空间协会采用的国际标准。NetCDF 库支持 NetCDF 文件的多种不同的二进制格式，有格式都是“自描述的”。 这意味其中有一个头部，它描述文件余下部分的格局，特别是数

组数据，连同以名称/值特性形式的任意文件元数据。这个格式是跨平台的，涉及的问题如字节序在软件库中解决。数据以允许有效的构造子集的方式来存储。起始于版本 4.0，NetCDF API 允许使用 HDF5 数据格式。NetCDF 用户可以创建 HDF5 文件从而获得 NetCDF 格式不具备的利益，比如更大的文件和多重无限制的维度。完全后向兼容，可访问旧有 NetCDF 文件并支持以前版本的 C 和 Fortran API。该项目开始于 1989 年，UCAR 目前对其积极支持，在新发行版中改进性能、增加功能并修正缺陷，当前版本系列是 NetCDF-4

3. Matplolib

Matplotlib 是 Python 语言及其数值计算库 NumPy 的绘图库。它提供了一个面向对象的 API，用于使用通用 GUI 工具包（如 Tkinter、wxPython、Qt 或 GTK）将绘图嵌入到应用程序中。它还有一个基于状态机（如 OpenGL）的过程式编程“Pylab”接口，其设计与 MATLAB 非常类似，但不推荐使用。SciPy 使用 matplotlib 进行图形绘制。Pyplot 是 matplotlib 的一个模块，它提供了一个类似 MATLAB 的接口。Matplotlib 被设计成与 MATLAB 一样可用，能够使用 Python，并且具有免费和开源的优点。

4. FTP

文件传输协议（英语：File Transfer Protocol，缩写：FTP）是一个用于在计算机网络上在客户端和服务器之间进行文件传输的应用层协议。文件传送（file transfer）和文件访问（file access）之间的区别在于：前者由 FTP 提供，后者由如 NFS 等应用系统提供。FTP 是一个 8 位的客户端-服务器协议，能操作任何类型的文件而不需要进一步处理，就像 MIME 或 Unicode 一样。但是，FTP 有着极高的延时，这意味着，从开始请求到第一次接收需求数据之间的时间，会非常长；并且不时的必须执行一些冗长的登录进程。运行 FTP 服务的许多站点都开放匿名服务，在这种设置下，用户不需要账号就可以登录服务器，默认情况下，匿名用户的用户名是：“anonymous”。这个账号不需要密码。

5. Cartopy

Cartopy 最初是由英国气象局开发的，旨在使科学家能够快速，轻松，最重要的是准确地在地图上可视化他们的数据。Cartopy 是一个 Python 包，专为地理空间数据处理而设计，以生成地图和其他地理空间数据分析。Cartopy 利用强大的 PROJ，NumPy 和 Shapely 库，并包括一个基于 Matplotlib 构建的程序化界面，用于创建出版质量的地图。Cartopy 的主要特征是其面向对象的投影定义，以及它能够在这些投影之间变换点，线，矢量，多边形和图像。

6. Numpy

NumPy 是 Python 语言的一个扩展程序库。支持高阶大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。NumPy 参考 CPython（一个使用字节码的解释器），而在这个 Python 实现解释器上所写的数学算法代码通常远比编译过的相同代码要来得慢。为了解决这个难题，NumPy 引入了多维数组以及可以直接有效率地操作多维数组的函数与运算符。因此在 NumPy 上只要能被表示为针对数组或矩阵运算的算法，其执行效率几乎都可以与编译过的等效 C 语言代码一样快。NumPy 提供了与 MATLAB 相似的功能与操作方式，因为两者皆为解释型语言，并且都可以让用户在针对数组或矩阵运算时提供较标量运算更快的性能。两者相较之下，MATLAB 提供了大量的扩展工具箱（例如 Simulink）；而 NumPy 则是根基于 Python 这个更现代、完整并且开放源代码的编程语言之上。此外 NumPy 也可以结合其它的 Python 扩展库。例如 SciPy，这个库提供了更多与 MATLAB 相似的功能；以及 Matplotlib，这是一个与 MATLAB 内置绘图功能类似的库。而从本质上来说，NumPy 与 MATLAB 同样是利用 BLAS 与 LAPACK 来提供高效率的线性代数运算。

三、项目设计

1. 技术方案

(一) Argo 数据下载

1、数据来源：ftp://data.argo.org.cn/pub/ARGO/BOA_Argo/。

2、采用 python 内置 Ftplib 来下载。

3、此 ftp 站点是公共资源，所以可以匿名登陆。

4、首先会爬取目录。

5、会获得用户需求，生成用户目录，并逐一下载。

(二) NetCDF4 文件读取

将.nc 文件信息写入一个结构体变量，在从里面提取所需的数据。

(三) 可视化的对象及过程

1、 采用 matplotlib 模块绘制。

2、 使用 Contour 函数绘制不同数值的等高线。

3、 Contourf 函数填充等高线。

4、 数据都以 Numpy 数组形式存储和读取。

(四) 海岸线绘制

采用 Cartopy 模块绘制在 Matplotlib 生成的 Figure 上。

(五) GIF 动图绘制

采用 animation 模块生成，其中单帧函数由静态图修改而成，添加帧函数控制变量，以 FPS=15 保存。

(六) GUI 设计

采用 Tkinter 设计，具体就是使用 button 调用函数，注销原来的界面，生成新界面。

(七) 打包 EXE

使用虚拟机重新配置 Python 环境使用 Pyinstaller 打包

2. 项目开发过程

阶段一、寻找数据

我们最先采用的是美国气象卫星 JASON 的原始数据（来自 NOAA），但随后我们就找到了问题：1:数据 ftp 网站不是匿名登陆而是实名登陆。2:ftp 国内网络访问异常。3:数据为散点，如果需要可视化处理需要自研算法。并且由于数据量太大而导致运算量巨大，如果要达到预期的时间还需要做并行计算处理。

我们于是就寻找偏向公众已经处理好的数据，ARGO 计划完美符合了我们的要求，这是一个全球拥有海洋实力的国家共建的项目平台，数据从 2004 年收集，至今依旧在更新。并且提供已经处理好 1° x1° 的网格化数据，便于可视化处理。

阶段二、NC 文件数据读取

NC 文件是拥有特定格式的二进制文件，我们采用的是使用 Unidata 官方所推荐的 NetCDF4 模块，并且采用 Numpy 模块将数据转化为 Numpy 数组。

阶段三、绘制可视化图片

可视化图片采用 Matplotlib 绘制。使用 Contour 和 Contourf 函数绘制等高线图。

阶段四、绘制动图

采用 Animation 模块，并且设计了逐帧绘制的函数，并统一封装为一个函数模块

阶段五、加入海岸线绘图

最先采用使用 Basemap 模块，但是这个模块已经很长时间没更新了，与安装的其他包存在依赖冲突。于是放弃。随后我们考虑下载原始 shape 文件来绘制海岸线，事实上我们已经试验成功了，但这会导致绘制过程时间过长，原来在不绘制海岸线的情况下，一个 NC 文件处理为 58 个图片在 23 秒左右，加上 shape 文件加载这个时间将会延长两倍。动图的绘制时间达到了三倍。为了时间考虑我们还是放弃了这个方案。

我们最终选择了 Cartopy 模块，Cartopy 与其他的模块兼容的很好，绘制的时间也少。缺点是由于 Cartopy 开发的时间较晚，使用者很少，网上可以找到的例子也较少。

阶段六、模块整合添加引导部分

将数据下载（FTP）模块，可视化（JPG，GIF）模块打包并封装为函数模块，在主程序中设计好流程控制。

阶段七、制作 GUI

重新设置程序，使用 Python 内置 Tkinter 模块设计了 GUI。

阶段八、打包为 EXE

我们使用编写代码原环境使用 Pyinstaller 总是打包失败，所以考虑使用 Conda 重新建立 python 环境，但是依旧无法成功。前后花费了将近六小时，换了很多种不同的打包方案，但都打包失败，问题出在主要以下三个问题：

1. 打包的 NetCDF4 模块无法调用。
2. 无法调用 Matplotlib。
3. 缺少.dll 文件。

第一个问题在 Pyinstaller 的日志中可以翻到是缺少 Cftime 模块,只需要在程序中补上引用 Cftime Pyinstaller 就会将 Cftime 模块打包进去。第二个问题是出在 Matplotlib 的 Numpy 依赖上，需要重新安装附加 mkl(Intel® Math Kernel Library)的 Numpy 包。第三个问题没有找到根源，但是我们有解决方案：在微软 Windows 开发者页面下载 Windows11 开发者镜像安装进虚拟机，在虚拟机上配置 Python 环境（非 Conda）在进行打包，便无相关问题。

阶段九、编写 MARKDOWN 文件

考虑到我们是数据处理类项目,涉及了稍微复杂的模块依赖，打包的 EXE 文件也存在文件夹依赖，我们就对源程序和打包的 EXE 程序都编写了 README.MD 帮助说明，有使用方式，环境配置，已知 bug 及相关解决方案。

四、总结和项目开发心得

1、学会了科学数据处理(Numpy)和科学绘图(Matplotlib)模块的使用。

2、项目是在 Jupyter 和 Jupyter Lab 的 IDE 下开发的，使我们加速开发。

3、学会了如何分析问题，并且从源头解决问题。

五、参考资料

资料来源：

1. Numpy 介绍：NumPy - 维基百科，自由的百科全书 (wikipedia.org)
2. Argo 计划介绍：Argo 计划 - 维基百科，自由的百科全书 (wikipedia.org)
3. FTP 介绍：文件传输协议 - 维基百科，自由的百科全书 (wikipedia.org)
4. Matplotlib 介绍：matplotlib - 维基百科，自由的百科全书 (wikipedia.org)
5. NetCDF 介绍：NetCDF - 维基百科，自由的百科全书 (wikipedia.org)
6. Numpy 开发案例参考来源：NumPy
7. Argo 数据来源（国内）：杭州全球海洋 Argo 系统野外科学观测研究站
8. Argo 数据来源（国际）：Argo (ucsd.edu)
9. Jason 数据,海岸线 shape 文件来源：National Oceanic and Atmospheric Administration (noaa.gov)
10. 开发时参考过的海洋类数据：Physical Oceanography Distributed Active Archive Center (PO.DAAC) | JPL / NASA
11. FTP 模块开发参考：ftplib --- FTP 协议客户端 — Python 3.10.1 文档
12. Matplotlib 使用参考：Matplotlib — Visualization with Python
13. Animation 模块开发参考：matplotlib.animation — Matplotlib 3.5.1 documentation
14. Netcdf4 格式文件数据提取参考：netCDF4 API documentation (unidata.github.io)
15. Cartopy 使用参考：Introduction — cartopy 0.20.0 documentation (scitools.org.uk)
16. Pyinstaller 打包参考：PyInstaller Quickstart — PyInstaller bundles Python applications
17. Jupyter Lab 使用参考：JupyterLab Documentation — JupyterLab 3.2.5 documentation
18. Anaconda 配置与使用：Anaconda package lists — Anaconda documentation
19. Intel 数学核心函数库: Math Kernel Library - Wikipedia
20. 打包环境来自：Download a Windows virtual machine - Windows app development (microsoft.com)

附录源程序

```
import os
import shutil
import tkinter as tk
from ftplib import FTP

import cftime
import cartopy.crs as ccrs
import matplotlib.animation as animation
import matplotlib.pyplot as plt
import numpy as np
from NetCDF4 import Dataset


def getdata():
    global file_dir, file_list, window, var, forma
    '''
    说明：
    1、本模块数据来源：中国 Argo 实时资料中心网站（或自然资源部杭州全球海洋 Argo 系统野外科学观测研究站）
    2、版权所属：自然资源部第二海洋研究所中国 Argo 实时资料中心
    3、编写者对用户因使用此模块产生的损失和不良后果不负任何法律责任。
    4、本模块采用匿名登录 ftp 方式下载数据。
    5、已知 bug：受制于主机和服务器的带宽有一定概率下载失败，如果没有显示 '下载完毕。' 则一致认定为下载失败。
    WRITE BY YuHanxue in 2021.12.1 in OUC
    联系邮箱：hanxueyu555@gmail.com
    '''
    window.destroy()
    window = tk.Tk()
    window.geometry("500x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    window.title('FTP 模块')
    tk.Label(window, text='已启动 FTP 下载模块', font=('Arial', 12)).place(x=0, y=0)
    ftpserver = 'data.argo.org.cn'
    ftpath = '/pub/ARGO/BOA_Argo/NetCDF'
    localpath = './data/'
    ftp = FTP()
    try:
        ftp.connect(ftpserver, 21)
        ftp.login()
        ftp.cwd(ftpath)
    except:
        raise IOError('FTP 数据连接失败，请检查您的网络环境')
    else:
        tk.Label(window, text=f'{ftpserver}欢迎信息:{ftp.getwelcome()}', font=(
            'Arial', 12)).place(x=0, y=20)
        tk.Label(window, text='FTP 连接成功', font=('Arial', 12)).place(x=0, y=40)
        tk.Label(window, text=f'成功进入 FTP 服务器:{ftp.pwd()}', font=(
            'Arial', 12)).place(x=0, y=60)
    file_list = list(ftp.nlst())
    for i in range(13):
        file_list.pop()
    file_sta = (file_list[0]).split('_')
    file_end = (file_list[-1]).split('_')
    tk.Label(window, text=f'NC 文件记录时间从{file_sta[2]:4}年{file_sta[3][0:2]:2}月\
            到{file_end[2]:4}年{file_end[3][0:2]:2}月', font=('Arial', 12)).place(x=0, y=80)
    # 下载数据
    if not os.path.exists(localpath):
        os.makedirs(localpath)
    tk.Label(window, text='请问需要单个数据还是批量数据？', 
             font=('Arial', 12)).place(x=0, y=100)
    print()

    def single():
        global file_dir, file_list, window, var, forma

        def get():
            global file_dir, file_list, window, var, forma
            year = e_year.get()
            mon = e_mon.get()
            filename = 'BOA_Argo_'+str(year)+'_'+str(mon).zfill(2)+'.nc'
            bufsize = 1024
            path = os.path.join(localpath, filename)
            with open(path, 'wb') as fid:
                tk.Label(window, text='正在下载:').grid(row=3, column=1)
                window.update()
                ftp.retrbinary('RETR {0}'.format(filename), fid.write, bufsize)
                tk.Label(window, text='下载完毕。').grid(row=4, column=1)
                tk.Button(window, text='进入可视化', command=printfil).grid(
                    row=5, column=1)
        window.destroy()
        window = tk.Tk()
        window.geometry("300x500")
        window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
        window.title('ARGO 数据单项下载')
```

```
        l_year = tk.Label(window, text='年：(2004-2021)')
        l_year.grid(row=0)
        e_year = tk.Entry(window)
        e_year.grid(row=0, column=1)
        l_mon = tk.Label(window, text='月：(1-12)')
        l_mon.grid(row=1)
        e_mon = tk.Entry(window)
        e_mon.grid(row=1, column=1)
        b_sure = tk.Button(window, text='确定', command=get)
        b_sure.grid(row=2, column=1)

    def batch():
        global file_dir, file_list, window, var, forma

        def get():
            global file_dir, file_list, window, var, forma
            tk.Label(window, text='正在下载，请不要退出
').grid(row=6)
            window.update()
            year1 = e_year.get()
            mon1 = e_mon.get()
            year2 = e_year2.get()
            mon2 = e_mon2.get()
            file_down_start = 'BOA_Argo_' + \
                str(year1)+'_'+str(mon1).zfill(2)+'.nc'
            file_down_start_index =
file_list.index(file_down_start)
            file_down_end =
'BOA_Argo_'+str(year2)+'_'+str(mon2).zfill(2)+'.nc'
            file_down_end_index =
file_list.index(file_down_end)
            i = 7
            for filename in
file_list[file_down_start_index:file_down_end_index+1]:
                bufsize = 1024
                path = os.path.join(localpath, filename)
                with open(path, 'wb') as fid:
                    tk.Label(window, text=f'正在下载:
{filename}').grid(row=i)
                    window.update()
                    ftp.retrbinary('RETR {0}'.format(
                        filename), fid.write, bufsize)
                    tk.Label(window, text=f'{filename}文件下载
结束').grid(row=i+1)
                    window.update()
                    i += 2
                if filename ==
file_list[file_down_end_index]:
                    window.destroy()
                    window = tk.Tk()
                    window.geometry("300x500")
                    window.iconbitmap(r".\lib\ico\IDisk HD
ALT.ico")
                    window.title('ARGO 数据结束下载')
                    tk.Button(window, text='进入可视化',
command=printfil).pack()
        window.destroy()
        window = tk.Tk()
        window.geometry("300x500")
        window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
        window.title('ARGO 数据批量下载')
        l_year = tk.Label(window, text='起始年份：(2004-
2021)')
        l_year.grid(row=0)
        e_year = tk.Entry(window)
        e_year.grid(row=0, column=1)
        l_mon = tk.Label(window, text='起始月份：(1-12)')
        l_mon.grid(row=1)
        e_mon = tk.Entry(window)
        e_mon.grid(row=1, column=1)
        l_year2 = tk.Label(window, text='起始年份：(2004-
2021)')
        l_year2.grid(row=2)
        e_year2 = tk.Entry(window)
        e_year2.grid(row=2, column=1)
        l_mon2 = tk.Label(window, text='起始月份：(1-12)')
        l_mon2.grid(row=3)
        e_mon2 = tk.Entry(window)
        e_mon2.grid(row=3, column=1)
        b_sure = tk.Button(window, text='确定', command=get)
        b_sure.grid(row=4, column=1)

    b = tk.Button(window, text='单个数据', command=single,
width=35)
    c = tk.Button(window, text='批量数据', command=batch,
width=35)
    b.place(x=0, y=120)
    c.place(x=250, y=120)
    window.mainloop()
def ftp():
    global file_dir, file_list, window, var, forma
    file_dir = './data'
    if os.path.exists(file_dir):
        shutil.rmtree('./data')
        os.makedirs('./data')
    getdata()
def demo():
    global file_dir, file_list, window, var, forma
```

```
    file_dir = './demo_data'
    printfil()
def selfdata():
    global file_dir, file_list, window, var, forma
    file_dir = './self_data'
    if not os.path.exists(file_dir):
        os.makedirs(file_dir)
    window.destroy()
    window = tk.Tk()
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    window.title('自定义文件')
    tk.Label(window, text='请将文件放入 self_data 文件夹内',
font=('Arial', 12)).pack()
    tk.Label(window, text='文件必须是 BOA_Argo_yyyy_mm.nc 格
式，y 指年份，m 指月份',
             font=('Arial', 12)).pack()
    tk.Label(window, text='放置结束请按结束键', font=('Arial',
12)).pack()
    tk.Button(window, text='结束放置',
command=printfil).pack()
def printfil():
    global file_dir, file_list, window, var, forma
    file_list = list(os.listdir(file_dir))
    for i in range(len(file_list)):
        file_list[i] = file_dir+'/'+file_list[i]
    window.destroy()
    window = tk.Tk()
    window.geometry("300x500")
    window.title('文件确认')
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    tk.Label(window, text='您将处理以下文件：', font=('Arial',
12)).pack()
    for i in file_list:
        tk.Label(window, text=i, font=('Arial', 12)).pack()
    tk.Button(window, text='进入可视化',
command=choose).pack()
def choose():
    global file_dir, file_list, window, var, forma
    window.destroy()
    window = tk.Tk()
    window.title('请选择温度或盐度')
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    la1 = tk.Label(window, text='请选择可视化对象',
font=('Arial', 12))
    la1.place(y=0)
    b = tk.Button(window, text='温度', command=temp,
width=20)
    c = tk.Button(window, text='盐度', command=salt,
width=20)
    b.place(x=0, y=20)
    c.place(x=150, y=20)
def salt():
    global file_dir, file_list, window, var, forma
    window.destroy()
    window = tk.Tk()
    window.title('已选择盐度')
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    var = 'salt'
    la1 = tk.Label(window, text='请选择可视化格式',
font=('Arial', 12))
    la1.place(y=0)
    b = tk.Button(window, text='GIF 动图', command=forma_gif,
width=20)
    c = tk.Button(window, text='JPG 图片', command=forma_jpg,
width=20)
    b.place(x=0, y=20)
    c.place(x=150, y=20)
def temp():
    global file_dir, file_list, window, var, forma
    window.destroy()
    window = tk.Tk()
    window.title('已选择温度')
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    var = 'temp'
    la1 = tk.Label(window, text='请选择可视化格式',
font=('Arial', 12))
    la1.place(y=0)
    b = tk.Button(window, text='GIF 动图', command=forma_gif,
width=20)
    c = tk.Button(window, text='JPG 图片', command=forma_jpg,
width=20)
    b.place(x=0, y=20)
    c.place(x=150, y=20)
def forma_gif():
    global file_dir, file_list, window, var, forma
    forma = gifmake
    window.destroy()
    window = tk.Tk()
    window.title('GIF 制作')
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    for fil in file_list:
        file_name = var+'_of_'+fil.split('/')[-1]
        data = Dataset(fil)
```

```
        lon = data.variables['lon']
        lat = data.variables['lat']
        data1 = data.variables[var]
        lat = slice(np.min(lat), np.max(lat)+lat[1]-lat[0],
lat[1]-lat[0])
        lon = slice(np.min(lon), np.max(lon)+lon[1]-lon[0],
lon[1]-lon[0])
        Lat, Lon = np.mgrid[lat, lon]
        tk.Label(
            window, text=f'正在可视化{file_name[:-3]}',
font=('Arial', 12)).pack()
        window.update()
        forma(Lon, Lat, data1, file_name[:-3])
    end()
def forma_jpg():
    global file_dir, file_list, window, var, forma
    forma = picmake
    window.destroy()
    window = tk.Tk()
    window.title('JPG 制作')
    window.geometry("300x500")
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    for fil in file_list:
        file_name = var+'_of_'+fil.split('/')[-1]
        data = Dataset(fil)
        lon = data.variables['lon']
        lat = data.variables['lat']
        data1 = data.variables[var]
        lat = slice(np.min(lat), np.max(lat)+lat[1]-lat[0],
lat[1]-lat[0])
        lon = slice(np.min(lon), np.max(lon)+lon[1]-lon[0],
lon[1]-lon[0])
        Lat, Lon = np.mgrid[lat, lon]
        tk.Label(window, text=f'正在可视化{file_name}',
font=('Arial', 12)).pack()
        window.update()
        forma(Lon, Lat, data1, file_name[:-3])
    end()
def picmake(Lon, Lat, data1, name):
    global file_dir, file_list, window, var, forma
    for i in range(np.shape(data1)[1]):
        plt.cla()
        plt.figure(figsize=(20, 10))
        ax =
plt.axes(projection=ccrs.PlateCarree(central_longitude=180))
        ax.coastlines()
        data_drew = data1[0, i, :, :]
        plt.contour(Lon, Lat, data_drew, 16, alpha=0.75,
linewidths=0.5,
                    colors='black',
transform=ccrs.PlateCarree(central_longitude=0))
        c = plt.contourf(Lon, Lat, data_drew, 16,
                         transform=ccrs.PlateCarree(central_l
ongitude=0))
        plt.colorbar(c)
        plt.title(f'depth={i}', fontsize='xx-large')
        name1 = './pic/'+name+f'_depth={i}.jpg'
        plt.savefig(name1, dpi=200)
        plt.close()
def gifmake(Lon, Lat, data1, name):
    global file_dir, file_list, window, var, forma
    name = './gif/'+name+'.gif'
    fig = plt.figure(figsize=(20, 10))
    def updatefig(num):
        plt.cla()
        ax =
plt.axes(projection=ccrs.PlateCarree(central_longitude=180))
        ax.coastlines()
        data_drew = data1[0, num, :, :]
        ax.contourf(Lon, Lat, data_drew, 16,
                    transform=ccrs.PlateCarree(central_longit
ude=0))
        ax.contour(Lon, Lat, data_drew, 16, linewidths=0.5,
alpha=0.75,
                   colors='black',
transform=ccrs.PlateCarree(central_longitude=0))
        plt.title(f'depth={num}', fontsize='xx-large')
        return ax
    ani = animation.FuncAnimation(
        fig, updatefig, frames=range(np.shape(data1)[1]))
    ani.save(name, fps=15)


def end():
    global file_dir, file_list, window, var, forma
    window.destroy()
    window = tk.Tk()
    window.title('结束')
    window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
    if forma == picmake:
        tk.Label(window, text=f'制作完毕，请于 pic 文件夹内查看
', font=('Arial', 12)).pack()
    elif forma == gifmake:
        tk.Label(window, text=f'制作完毕，请于 gif 文件夹内查看
', font=('Arial', 12)).pack()
    tk.Button(window, text='结束程序', command=endgui).pack()
def endgui():
    window.destroy()
```

```
global file_dir, file_list, ishit, window
window = tk.Tk()
window.title('ARGO 数据可视化')
window.geometry("300x500")
window.iconbitmap(r".\lib\ico\IDisk HD ALT.ico")
la1 = tk.Label(window, text='请选择数据来源', font=('Arial',
12))
la1.place(y=0)
ishit = 0
b = tk.Button(window, text='ftp', command=ftp, width=13)
c = tk.Button(window, text='demo', command=demo, width=13)
d = tk.Button(window, text='self', command=selfdata,
width=13)
#b.grid(column=10, row=10)
b.place(x=0, y=20)
c.place(x=100, y=20)
d.place(x=200, y=20)
window.mainloop()
```