

Project Description

Subject: Concurrent Programming Languages (Paralleelprogrammeerimise keeled)

Work: Project

Student: Aleksander Ontin

Project assignment

Instructions: This requires writing a Python program. Submit your program via Courses. If you have several files, submit them as zip file. Submitting the source code is enough. Please provide a list of libraries needed to run the application.

Write a Python program which uses a simple GUI for downloading files using file transfer protocol (ftp). The GUI should have three elements: A field to enter a URL, a button to send of the URL, and progress bars to display the progress of the downloads.

You can use a GUI library of your choice. Also, for the actual download, you can use a library as you like. You can use <https://bmrp.io/ftp/pub/pdb/holdings/> for some large files to download via ftp. Important: The ftp server is called <ftp.bmrp.io> and the directory to switch to is pdb/holdings. This has the files you can see in the web interface. The web interface is not using ftp.

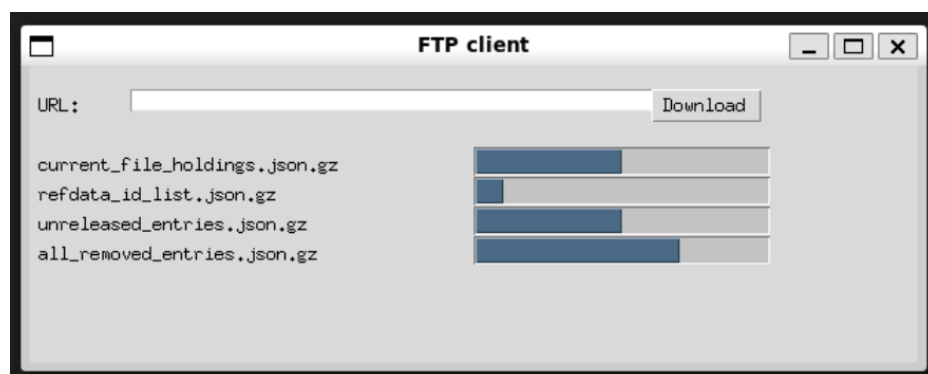
If the button is hit, the download of the file starts. It should be possible to start a new download immediately after that. For each download, a progress bar should be shown. The progress bar should update in reasonable intervals. The progress can be roughly estimated, finding a good estimation given the network speed etc. is not essential.

Consider error handling for at least the cases of a non-existing URL and a lost connection.

We will look for the following for marking:

- Working application
- Proper use of concurrency libraries
- Fluidity of user interface
- Error handling
- Proper use of comments to clarify the working of the program. You can also include a separate document, if you wish, but we do not expect an essay

Below is an idea for the UI, but the details are up to you.



Acronyms and the means

What is GUI?

GUI is an acronym for the designation “Graphical User Interface”, where UI is a “User Interface”. This is a digital interface in which the user interacts with graphical components such as icons, buttons, and menus. The visual elements displayed in the user interface (UI) convey information important to the user as well as actions the user can take.

It is very important in a project to take care of the graphical interface so that any user can understand what needs to be done there. The graphical interface should not be complicated and should contain as few details as possible for convenience and simplicity.

What is URL?

URL is an acronym for the designation “Uniform Resource Locator”. This is an address that provides a path to a certain file on a server. This source can be accessed via a network protocol such as http, https or ftp.

A URL consists of nine parts: scheme or protocol (*http://, https:// or ftp://*), subdomain (*www.*), second-level domain (*example.*), top-level domain (*com*), port (*:443 or :80*), subdirectory or path, query, parameter and fragment. While a URL doesn't have to contain all of these parts at once, it will always have at least three of them. Domain name consists of subdomain, second-level and top-level domain.

Example is <https://bmr.io/ftp/pub/pdb/holdings/>, where

https:// is a scheme or protocol,

bmr.io is a second-level domain,

io is a top-level domain,

/ftp/pub/pdb/holdings/ is a path,

and domain name is **bmr.io**.

What is FTP?

FTP is an acronym for the designation “File Transfer Protocol”. This is an application layer protocol that moves files between local and remote file systems. It runs on top of TCP, like HTTP. To transfer a file, FTP uses 2 TCP connections in parallel: a control connection and a data connection.

You can find information about FTP [here](#).

What is *BMRB*?

BMRB is an acronym for the designation “Biological Magnetic Resonance Data Bank” (*BioMagResBank*). This is the publicly-accessible depository for NMR results from peptides, proteins, and nucleic acids recognized by the International Society of Magnetic Resonance. BMRB makes biological NMR data FAIR: Findable, Accessible, Interoperable, Re-usable.

You can find information about BMRB [here](#) and [here](#).

Access to data in BMRB is free directly from its web site (URL <http://bmr.io>) and ftp site (URL <https://ftp.bmr.io>) and will remain so as public funding permits. Web site domain name is **bmr.io** and FTP site domain name is **ftp.bmr.io**.

FTP links (ftp://)

The entered URL must be correct and not a pseudo ftp https URL. One possible example of an FTP link was shown by lecturer Stefan Kuhn. User must enter ftp.bmr.io/pdb/holdings/file.gz or something similar.

Example: ftp://ftp.bmr.io/pdb/holdings/file.gz

Where the ftp server is called **ftp.bmr.io** and the directory to switch to is **pdb/holdings**.

For downloading files, it would be easier to create FTP links for the various files in advance and test them when the program is ready.

Below are links to various files from the **BMRB** website:

ftp://ftp.bmr.io/pdb/holdings/all_removed_entries.json.gz

ftp://ftp.bmr.io/pdb/holdings/current_file_holdings.json.gz

ftp://ftp.bmr.io/pdb/holdings/obsolete_experimental_data_last_modified_dates.json.gz

ftp://ftp.bmr.io/pdb/holdings/obsolete_structures_last_modified_dates.json.gz

ftp://ftp.bmr.io/pdb/holdings/refdata_id_list.json.gz

ftp://ftp.bmr.io/pdb/holdings/released_experimental_data_last_modified_dates.json.gz

ftp://ftp.bmr.io/pdb/holdings/released_structures_last_modified_dates.json.gz

ftp://ftp.bmr.io/pdb/holdings/unreleased_entries.json.gz

Python Modules in project

The program uses the following modules such as **tkinter** (as **tk**), **tkinter.ttk**, **ftplib** (FTP), **urllib.parse** (urlparse), **os**, **threading** and **time**.

In code form, their imports are as follows:

```
import tkinter as tk
from tkinter import ttk
from ftplib import FTP
from urllib.parse import urlparse
import os
import threading
import time
```

The **tkinter** module is used to create a graphical user interface and is one of the most important modules in the program. Thanks to it you can create a screen and inside it buttons, input fields, etc.

Information about tkinter can be found on the official Python pages [here](#) and [here](#).

The **ftplib** (FTP) module defines the class FTP that implements the client side of the FTP protocol. This allows you to create programs that perform a variety of automated FTP jobs. In project I used it to download a file via the FTP protocol. It is also one of the important modules in the program.

Information about ftplib can be found on the official Python pages [here](#).

The **urllib** module collects several modules for working with URLs. For example, urllib.parse that used for parsing URLs. We use **urllib.parse** module in the small functions "is_valid_url" and "is_ftp_url".

Information about urllib can be found on the official Python pages [here](#) and [here](#).

The **os** module is used to upload files to a specific folder where the program resides. We use **os** module in the "download_file_ftp" function.

Information about os can be found on the official Python pages [here](#).

The **threading** module is used to create threads, which in turn is used for parallelism. It is also one of the most important modules in the program.

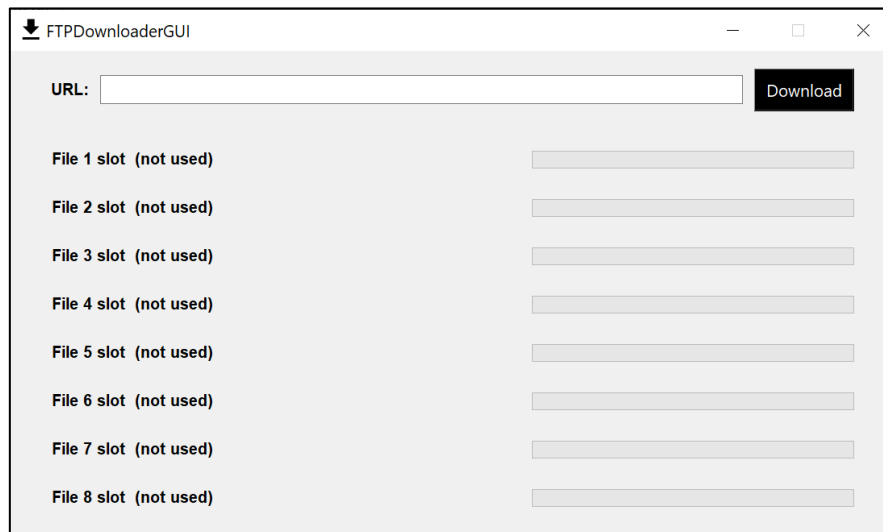
Information about threading can be found on the official Python pages [here](#).

I use the **time** module in the "download_file_ftp" function to slow down the file download. This made it possible to see more clearly how the GUI works.

Information about threading can be found on the official Python pages [here](#).

Garaphical User Interface in project

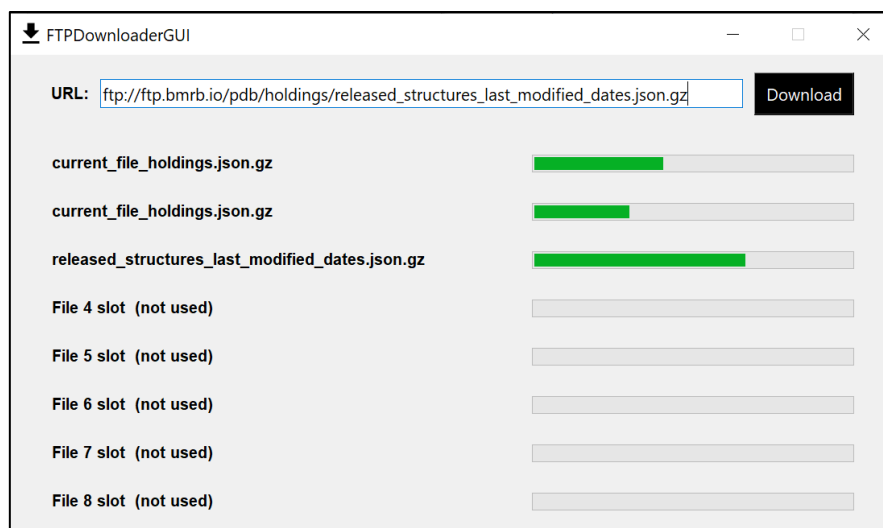
The GUI in my project looks like this:




At the very top left you can see the text “URL:”, in the middle there is an input field and on the right there is a button with the text “Download”.

Below are the rows that are slots for downloading files. Initially, they are not used, so the progress bars on the right are empty, and the names of the slots on the left are almost the same. You can download up to 8 files in total in parallel.

When the first file is downloaded, it will show on the first row, and if more files are downloaded, they will show lower in the second row and so on. The example is on the second picture:



You can download identical files at the same time and in the folder, they will look like this:

 current_file_holdings.json (1).gz	09.01.2024 15:39
 current_file_holdings.json.gz	09.01.2024 15:39
 released_structures_last_modified_dates.j...	09.01.2024 15:38

The files will be downloaded in the "downloads" folder, which is located in the folder where the program is located. If for some reason the folder is missing, the program will automatically create it before downloading the file.