Eberswalde University for Sustainable Development – HNEE

University of Applied Sciences
Faculty of Forest and Environment

Adebayo Banjo
Martikel-nr: 18212141

Topic:

TREE LISTING RECORD

Teacher:
Prof. Dr Luis Miranda

Introduction

Trees are one of the most important assets that we have, not only are they the main source for oxygen which supports all the living organisms, but it also provides shelter , food , and supplies us with many products that we use daily. Today many types of trees face extinction and are getting significantly less by each day, and this puts on our shoulder the task to count them and keep track of different types of trees , forest scientists and researchers have repeatedly reported that there is a lack in the software's that can help them do this job , that is why a tree catalogue software is what I chose to develop to try and contribute to solving this problem. Understanding the composition and structure of the forest is critical for appropriately assessing its environmental impact. For the reason that gathering tree information and carefully storing little prints is incredibly important. Traditionally, the forest department completes forest structural operations and stores them as physical copies. Currently, the government offers various websites for this archiving, however finding good software may be a challenge. If organizations or businesses have strong software for gathering crucial tree data, it could be highly useful for storing and analyzing the world and vegetation in a proper way. With a data management system, the worth of the timber may be simply calculated for timber factories. Especially at this time.

Objectives and functionalities

Allow the user to make entries about any tree , specifying DBH , height category , ... Update tree table to contain all the tree entries Search through the previously tree entries and filter them based on certain category value Store trees under different officers' names Export the tree entries table to an external database file "excel".

Programming language

Python is a high-level, interpreted general-purpose programming language. Its design philosophy prioritizes code readability through significant indentation. Its linguistic structure and object-oriented approach are designed to enable programmers to write clear and logical code for both small and large projects.

Database

As for the database we represent our database as an excel table , which is a widely supported format for representing database due to it's similar to SQL tables , it is also intuitive and easy to handle and read for users , it can be viewed on any pc that uses office using Microsoft excel or using Google's online spreadsheet platform to provide users with more flexibility.

IDE

PyCharm was used to create this project. An IDE is a piece of software for developing apps that combines various developer tools into a single graphical interface (GUI). This makes it easier for programmers to combine the many parts of creating a computer virus. Programmers begin with a blank file and create a program with just a few lines of code. It provides a lot of features like syntax highlighting and autocomplete, IDEs make this process easier. Many IDEs are now available for coding in Python. PyCharm has code analysis, a graphical debugger, an integrated unit tester, version control integration, and web development with Django and Anaconda. Making software with multiple functions necessitates the use of libraries. Python libraries are a collection of helpful functions that remove the need to write code.

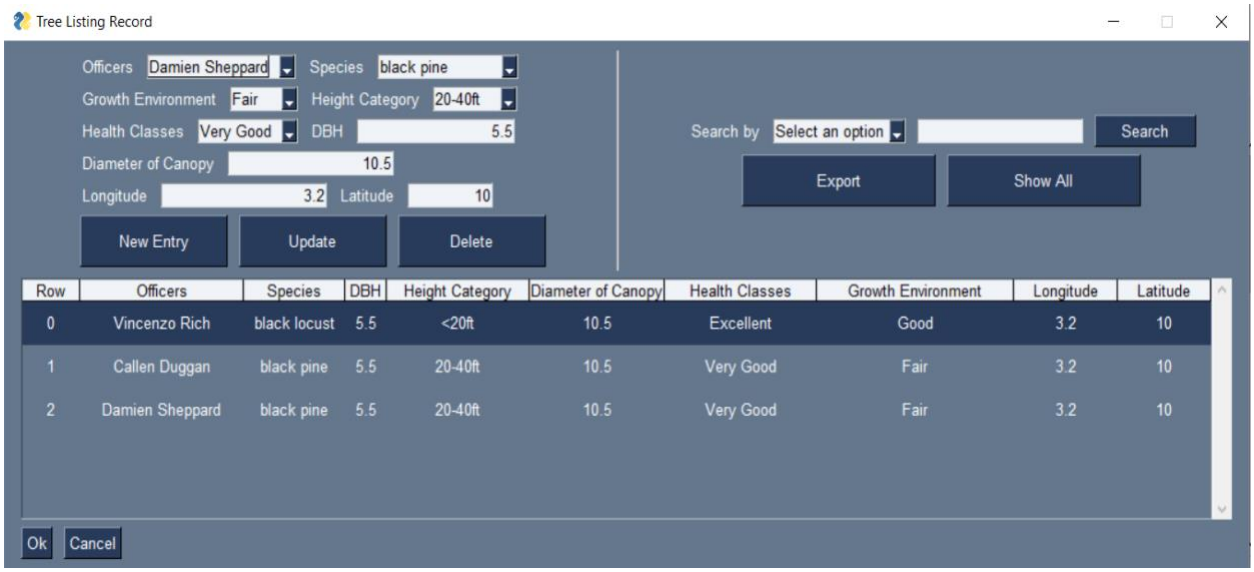
Development Environment

- Operating system: Win 10
- Python version: Python 3.10.1
- Database handler: openpyxl
- Dependencies and requirements: - PySimpleGUI~=4.56.0 - pandas~=1.4.0 -openpyxl~=3.0.9

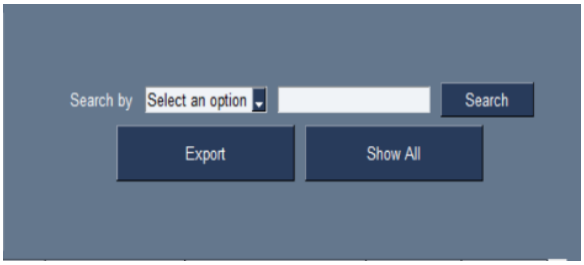
Functions

The tree catalogue GUI is made to register tree details, it is intended to help forest scientists and researchers keep track of different trees species, it allows them to submit an entry with fields (Officers , Species , DBH , Height Category , Diameter of Canopy, Health Classes , Growth Environment , Longitude , Latitude) , and it displays it in a table view , it also allows them to update any

previous entries by selecting that entry and altering its fields to the new values then pressing the update button. Once the user is done adding entries, he can then search through the entries for a certain value and filter it using a certain column, he can also choose to export the table as an excel file to be stored locally and distributed to other users and backup the entries data.



- The user should start filling the table with entries.



- He can use the search feature to choose a column and search for a value in it, returning the entries of a certain officer name or height category for example.
- The user can also export the table content to an excel file by clicking export.

- The user can update an entry by selecting it from the table, then choosing the values he wants from the add new entry section, then pressing update entry.

- Deleting an entry can be done by selecting it and pressing the delete button from the add new entry section

- If the user wishes to return from filtered table view to see the whole table entries, he can click the show all button from the filter entries section.

Row	Officers	Species	DBH	Height Category	Diameter of Canopy	Health Classes	Growth Environment	Longitude	Latitude
0	Vincenzo Rich	black locust	5.5	<20ft	10.5	Excellent	Good	3.2	10
1	Callen Duggan	black pine	5.5	20-40ft	10.5	Very Good	Fair	3.2	10
2	Damien Sheppard	black pine	5.5	20-40ft	10.5	Very Good	Fair	3.2	10

Ok Cancel

Main code parts

- Importing core libraries for the GUI and data handling.

```
import PySimpleGUI as sg
import pandas as pd

# Defining headers for the table and the data frame.
headers = ['Officers', 'Species', 'DBH', 'Height Category', 'Diameter of Canopy', 'Health Classes',
           'Growth Environment', 'Longitude', 'Latitude']
# Initializing Data frame.
```

- Initializing data frame and building fields , the main parts used here are simple gui's text for labels , combo box for carrying values that user can choose from and text inputs where user can type his own input.

```
#Initializing Data frame.
df = pd.DataFrame(columns=headers)

# Calculating column width based on the length of each header's string.
col_widths = list(map(lambda x: len(x) + 1, headers))

# Defining each entry as a list of text field and it's appropriate input.
# Giving keys is essential to retrieve the values of elements inside the event loop.
officers = [sg.Text('Officers'),
            sg.Combo(['Vincenzo Rich', 'Callen Duggan', 'Damien Sheppard', 'Cherry Morales', 'Jackson Blake'],
                     key='-officers-', default_value='Select an option')]
species = [sg.Text('Species'),
           sg.Combo(['black locust', 'black pine', 'common juniper', 'common yew', 'eurpean larch'], key='-species-',
                    default_value='Select an option')]
dbh = [sg.Text('DBH'), sg.InputText('0.0', size=(19, 20), justification='right', key='-dbh-')]
height = [sg.Text('Height Category'),
          sg.Combo(['<20ft', '20-40ft', '>40ft', '>50ft'], key='-height-', default_value='Select an option')]
diameter = [sg.Text('Diameter of Canopy'), sg.InputText('0.0', size=(20, 20), justification='right', key='-diameter-')]
health = [sg.Text('Health Classes'),
          sg.Combo(['Excellent', 'Very Good', 'Good', 'Fair', 'Poor', 'Dead'], key='-health-',
                   default_value='Select an option')]
growth = [sg.Text('Growth Environment'),
          sg.Combo(['Good', 'Fair', 'Poor'], size=(6, 20), key='-growth-', default_value='Select an option')]
longitude = [sg.Text('Longitude'), sg.InputText('0.0', size=(20, 20), justification='right', key='-longitude-')]
latitdude = [sg.Text('Latitude'), sg.InputText('0.0', size=(10, 20), justification='right', key='-latitdude-')]
```

- I then added the table element after the grid to achieve this layout , in pysimple gui , the objects take parameters to help justify and algin them , headers , values to view , keys to help retrieve the data they carry , size which sets the width and height of different elements , every element's parameters can be found in pysimple gui's official documentation.

```
table = [sg.Table(values=df.values.tolist(),
                  key='-LIST-',
                  headings=headers,
                  display_row_numbers=True,
                  auto_size_columns=False,
                  max_col_width=40,
                  def_col_width=14,
                  justification='middle',
                  row_height=35,
                  col_widths=col_widths, enable_click_events=True,
                  num_rows=min(25, len(df.values.tolist()) + 5))]
# Designing the gui , in PySimple GUI , the gui is divided to rows and columns.
row1c2 = [sg.Text('Search by'),
          sg.Combo(values=headers, key="-searchbycombo-", default_value='Select an option', size=(14, 20)),
          sg.InputText(size=(20, 20), justification='right', key='-searchbytext-'),
          sg.Button('Search', size=(10, 1))]

row2c2 = [sg.Text('        '), sg.Button('Export', size=(20, 2)), sg.Button('Show All', size=(20, 2))]
```

- Elements are linked together in rows “python list = row”, and then column are formed from multiple rows .
- Layous is formed by adding columns and elements into a list , layout is here a list of lists of items , and each list in it is treated as a row element.

```
row2c2 = [sg.Text('        '), sg.Button('Export', size=(20, 2)), sg.Button('Show All', size=(20, 2))]

row1 = officers + species
row2 = growth + height
row3 = health + dbh
row4 = diameter
row5 = longitude + latitdude
row6 = [sg.Button('New Entry', size=(15, 2)), sg.Button('Update', size=(15, 2)), sg.Button('Delete', size=(15, 2))]
row7 = table
row8 = [sg.Button('Ok'), sg.Button('Cancel')]
col1 = [row1, row2, row3, row4, row5, row6]
col2 = [row1c2, row2c2]

layout = [[sg.Text('        '), sg.Column(col1), sg.Text('        '), sg.VerticalSeparator(), sg.Text('        '),
           sg.Col(col2)],
          row7,
          row8]
```

- For searching inside a dataframe , a function is implemented that returns a list of the found elements to be shown in the table object , if no entry was found that matches the search filer and value , the function returns an empty list which will throw a window to tell the user no entries were found that matches your search.

```python
# This is a search function that searches for certain value in a data frame given its key.
def search_function(df, column_name, search_value):
    values_to_search_in = list(df[column_name].values)
    print(values_to_search_in)
    found = []
    for index, value in enumerate(values_to_search_in):
        if search_value == value:
            found.append(df.values.tolist()[index])
        else:
            continue
    return found
```

- The main logic here is implemented inside the while true loop, the window layout is scanned for events and values , and if an event “buttons trigger events with their names” if triggered , it’s condition is satisfied and the code access the values it wants from the values of the window to carry on doing it’s functionality.
- If no event was triggered , the loop will continue and the tool will not update , if the cancel or close button are clicked then the tool break from the event loop closing the window.

```
while True:
    event, values = window.read()
    if event == sg.WIN_CLOSED or event == 'Cancel':  # if user closes window or clicks cancel
        break
    elif event == 'New Entry':
        try:
            current_window_values = [values['-officers-'], values['-species-'], values['-dbh-'], values['-height-'],
                                     values['-diameter-'],
                                     values['-health-'], values['-growth-'], values['-longitude-'],
                                     values['-latitdude-']]
            temp_df = pd.DataFrame([current_window_values], columns=headers)
            df = pd.concat([df, temp_df], ignore_index=True)
            window['-LIST-'].update(df.values.tolist())
        except:
            continue

    elif event == 'Export':
        foldername = sg.PopupGetFolder('Select folder', no_window=True)
        df.to_excel(foldername + '/Tree_inventory.xlsx')

    elif event == 'Search':
        column_name = values['-searchbycombo-']
        if column_name not in headers:
            continue
        value_to_look_for = values['-searchbytext-']
        found_values = search_function(df, column_name, value_to_look_for)
        if len(found_values) > 0:
            window['-LIST-'].update(found_values)
        else:
```

Runtime environment

The software is developed to run on any operating system and any machine that supports python given that it meets the requirements in the development section.

Installation

Make sure you have the development section’s python version or later

Clone the repo , then Run this command:

- Pip install -r dependencies.txt

Then run the software using this command

- python main.py

Result

This project was a very good way to understand and get familiar with python , it allowed me to search for libraries and install them to achieve a software that is usable and practical.

References

- PySimple Gui Official Documentation

  https://pysimplegui.readthedocs.io/en/latest/

- Demos

  https://github.com/PySimpleGUI/PySimpleGUI/tree/master/DemoPrograms

- Geeks For Geeks forum for Q&As

  https://www.geeksforgeeks.org/