Файловый менеджер на питоне в 430 строк для начинающих и чайников

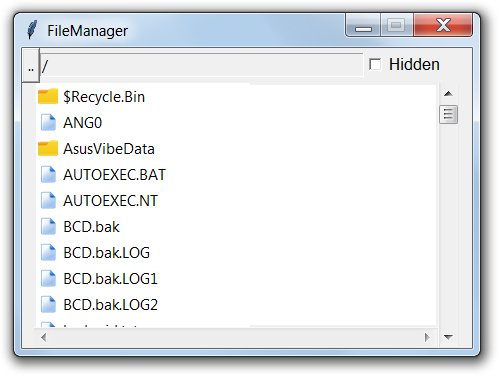
Python,

Программирование

Из песочницы

Всем привет!

Я захотел обобщить свои знания питона и решил написать файловый менеджер для пк.



# Внимание!

# Импорт библиотек:

import tkinter

import os

import subprocess

from tkinter import messagebox

from tkinter import simpledialog

# Главное меню:

class MainContextMenu(tkinter.Menu):

''' Контекстное меню для внутренней области директории'''

def \_\_init\_\_(self, main\_window, parent):

super(MainContextMenu, self).\_\_init\_\_(parent, tearoff = 0)

self.main\_window = main\_window

self.add\_command(label="Создать директорию", command = self.create\_dir)

self.add\_command(label="Создать файл", command = self.create\_file)

def popup\_menu(self, event):

''' функция для активации контекстного меню'''

#если активны другие меню - отменяем их

if self.main\_window.file\_context\_menu:

self.main\_window.file\_context\_menu.unpost()

if self.main\_window.dir\_context\_menu:

self.main\_window.dir\_context\_menu.unpost()

self.post(event.x\_root, event.y\_root)

def create\_dir(self):

''' функция для создания новой директории в текущей'''

dir\_name = simpledialog.askstring("Новая директория", "Введите название новой директории")

if dir\_name:

command = "mkdir {0}".format(dir\_name).split(' ')

#выполняем команду отдельным процессом

process = subprocess.Popen(command, cwd=self.main\_window.path\_text.get(), stdout = subprocess.PIPE, stderr = subprocess.PIPE)

out, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Операция невозможна!","Отказано в доступе.")

self.main\_window.refresh\_window()

def create\_file(self):

''' функция для создания нового файла в текущей директории'''

dir\_name = simpledialog.askstring("Новый файл", "Введите название нового файла")

if dir\_name:

command = "touch {0}".format(dir\_name).split(' ')

#выполняем команду отдельным процессом

process = subprocess.Popen(command, cwd=self.main\_window.path\_text.get(), stdout = subprocess.PIPE, stderr = subprocess.PIPE)

out, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Операция невозможна!","Отказано в доступе.")

self.main\_window.refresh\_window()

def insert\_to\_dir(self):

''' функция для копирования файла или директории в текущую директорию'''

copy\_obj = self.main\_window.buff

to\_dir = self.main\_window.path\_text.get()

if os.path.isdir(self.main\_window.buff):

#выполняем команду отдельным процессом

process = subprocess.Popen(['cp', '-r', copy\_obj, to\_dir], stdout = subprocess.PIPE, stderr = subprocess.PIPE)

out, err = process.communicate()

if err:

messagebox.showwarning("Операция невозможна!", err.decode("utf-8"))

else:

#выполняем команду отдельным процессом

process = subprocess.Popen(['cp', '-n', copy\_obj, to\_dir], stdout = subprocess.PIPE, stderr = subprocess.PIPE)

out, err = process.communicate()

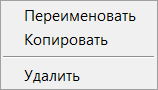
#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Операция невозможна!",err.decode("utf-8"))

self.main\_window.refresh\_window()

# При нажатии на файл должно выводиться контекстное меню:



class FileContextMenu(tkinter.Menu):

def \_\_init\_\_(self, main\_window, parent):

super(FileContextMenu, self).\_\_init\_\_(parent, tearoff = 0)

self.main\_window = main\_window

self.add\_command(label="Открыть файл", command = self.open\_file)

self.add\_separator()

self.add\_command(label="Копировать", command = self.copy\_file)

self.add\_command(label="Переименовать", command = self.rename\_file)

self.add\_separator()

self.add\_command(label="Удалить", command = self.delete\_file)

def open\_file(self):

''' функция для открытия файла сторонними программами'''

ext = self.main\_window.take\_extention\_file(self.main\_window.selected\_file)

full\_path = self.main\_window.path\_text.get() + self.main\_window.selected\_file

if ext in ['txt', 'py', 'html', 'css', 'js']:

if 'mousepad' in self.main\_window.all\_program:

subprocess.Popen(["mousepad", full\_path], start\_new\_session = True)

else:

self.problem\_message()

elif ext == 'pdf':

if 'evince' in self.main\_window.all\_program:

subprocess.run(["evince", full\_path], start\_new\_session = True)

else:

self.problem\_message()

elif ext in ['png', 'jpeg', 'jpg', 'gif']:

if 'ristretto' in self.main\_window.all\_program:

subprocess.run(["ristretto", full\_path], start\_new\_session = True)

else:

self.problem\_message()

else:

self.problem\_message()

def problem\_message(self):

messagebox.showwarning("Проблема при открытии файла", 'Прости, но я не могу открыть этот файл')

def copy\_file(self):

''' функция для копирования файла'''

#заносим полный путь к файлу в буффер

self.main\_window.buff = self.main\_window.path\_text.get() + self.main\_window.selected\_file

self.main\_window.refresh\_window()

def delete\_file(self):

''' функция для удаления выбранного файла'''

full\_path = self.main\_window.path\_text.get() + self.main\_window.selected\_file

#выполняем команду отдельным процессом

process = subprocess.Popen(['rm', full\_path], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Проблема при удалении файла", 'У Вас нет прав для удаления данного файла')

self.main\_window.refresh\_window()

def rename\_file(self):

''' функция для переименования выбранного файла'''

new\_name = simpledialog.askstring("Переименование файла", "Введите новое название файла")

if new\_name:

old\_file = self.main\_window.path\_text.get() + self.main\_window.selected\_file

new\_file = self.main\_window.path\_text.get() + new\_name

#выполняем команду отдельным процессом

process = subprocess.Popen(['mv', old\_file, new\_file], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Проблема при переименовании файла", 'У Вас нет прав для переименования данного файла')

self.main\_window.refresh\_window()

def popup\_menu(self, event):

''' функция для активации контекстного меню'''

self.post(event.x\_root, event.y\_root)

#если активны другие меню - отменяем их

if self.main\_window.main\_context\_menu:

self.main\_window.main\_context\_menu.unpost()

if self.main\_window.dir\_context\_menu:

self.main\_window.dir\_context\_menu.unpost()

self.main\_window.selected\_file = event.widget["text"]

# То же самое для директории:

class DirContextMenu(tkinter.Menu):

def \_\_init\_\_(self, main\_window, parent):

super(DirContextMenu, self).\_\_init\_\_(parent, tearoff = 0)

self.main\_window = main\_window

self.add\_command(label="Переименовать", command = self.rename\_dir)

self.add\_command(label="Копировать", command = self.copy\_dir)

self.add\_separator()

self.add\_command(label="Удалить", command = self.delete\_dir)

def copy\_dir(self):

''' функция для копирования директории'''

self.main\_window.buff = self.main\_window.path\_text.get() + self.main\_window.selected\_file

self.main\_window.refresh\_window()

def delete\_dir(self):

''' функция для удаления выбранной директории'''

full\_path = self.main\_window.path\_text.get() + self.main\_window.selected\_file

if os.path.isdir(full\_path):

#выполняем команду отдельным процессом

process = subprocess.Popen(['rm', '-rf', full\_path], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Проблема при удалении директории", 'У Вас нет прав для удаления данной директории')

self.main\_window.refresh\_window()

def rename\_dir(self):

''' функция для переименования выбранной директории'''

new\_name = simpledialog.askstring("Переименование директории", "Введите новое название директории")

if new\_name:

old\_dir = self.main\_window.path\_text.get() + self.main\_window.selected\_file

new\_dir = self.main\_window.path\_text.get() + new\_name

#выполняем команду отдельным процессом

process = subprocess.Popen(['mv', old\_dir, new\_dir], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

output, err = process.communicate()

#при возникновении ошибки выводим сообщение

if err:

messagebox.showwarning("Проблема при переименовании директории", 'У Вас нет прав для переименования данной директории')

self.main\_window.refresh\_window()

def popup\_menu(self, event):

''' функция для активации контекстного меню'''

self.post(event.x\_root, event.y\_root)

#если активны другие меню - отменяем их

if self.main\_window.main\_context\_menu:

self.main\_window.main\_context\_menu.unpost()

if self.main\_window.file\_context\_menu:

self.main\_window.file\_context\_menu.unpost()

self.main\_window.selected\_file = event.widget["text"]

# Класс основного окна:

class MainWindow():

''' Класс основного окна'''

def \_\_init\_\_(self):

self.root = tkinter.Tk()

self.root.title("FileManager")

self.root.resizable(width = False, height = False)

self.root.geometry('450x300')

self.hidden\_dir = tkinter.IntVar()

self.buff = None

self.all\_program = os.listdir('C:/')

self.root.bind('<Button-1>', self.root\_click)

self.root.bind('<FocusOut>', self.root\_click)

#top frame

self.title\_frame = tkinter.Frame(self.root)

self.title\_frame.pack(fill = 'both', expand = True)

#back button

self.back\_button = tkinter.Button(self.title\_frame, text = "..", command = self.parent\_dir, width = 1, height = 1)

self.back\_button.pack(side = 'left')

#path entry

self.path\_text = tkinter.StringVar()

self.path\_text.set('/')

self.current\_path = tkinter.Entry(self.title\_frame, textvariable = self.path\_text, width = 40, state='readonly')

self.current\_path.pack(side = 'left')

#button show/hidde hidden dir/file

self.check\_button = tkinter.Checkbutton(self.title\_frame, text = "Hidden", font = ("Helvetica", 10), padx = 1, pady = 1, variable = self.hidden\_dir, command = self.refresh\_window)

self.check\_button.pack(side = 'left')

#main frame

self.main\_frame = tkinter.Frame(self.root)

self.main\_frame.pack()

# scroll bar

self.scrollbar\_vert = tkinter.Scrollbar(self.main\_frame, orient="vertical")

self.scrollbar\_vert.pack(side = 'right', fill = 'y')

self.scrollbar\_hor = tkinter.Scrollbar(self.main\_frame, orient="horizontal")

self.scrollbar\_hor.pack(side = 'bottom', fill = 'x')

#canvas

self.canvas = tkinter.Canvas(self.main\_frame, borderwidth=0, bg = 'white')

self.inner\_frame = tkinter.Frame(self.canvas, bg = 'white')

#команды для прокрутки

self.scrollbar\_vert["command"] = self.canvas.yview

self.scrollbar\_hor["command"] = self.canvas.xview

#настройки для canvas

self.canvas.configure(yscrollcommand=self.scrollbar\_vert.set, xscrollcommand = self.scrollbar\_hor.set, width=400, heigh=250)

self.canvas.pack(side='left', fill='both', expand=True)

self.canvas.create\_window((0,0), window=self.inner\_frame, anchor="nw")

#отрисовываем содержимое лиректории

self.dir\_content()

def root\_click(self, event):

''' функция для обработки события клика в root'''

#если есть контекстные меню - отменяем

if self.file\_context\_menu:

self.file\_context\_menu.unpost()

if self.main\_context\_menu:

self.main\_context\_menu.unpost()

if self.dir\_context\_menu:

self.dir\_context\_menu.unpost()

def dir\_content(self):

''' функция для определения содержимого текущей директории'''

#содержимое в текущей директории

dir\_list = os.listdir(self.path\_text.get())

path = self.path\_text.get()

if not dir\_list:

#общее контекстное меню

self.main\_context\_menu = MainContextMenu(self, self.canvas)

self.canvas.bind('<Button-3>', self.main\_context\_menu.popup\_menu)

if self.buff:

self.main\_context\_menu.add\_command(label="Вставить", command = self.main\_context\_menu.insert\_to\_dir)

self.inner\_frame.bind('<Button-3>', self.main\_context\_menu.popup\_menu)

#контекстное меню для файлов

self.file\_context\_menu = None

#контекстное меню для директории

self.dir\_context\_menu = None

return None

#общее контекстное меню

self.main\_context\_menu = MainContextMenu(self, self.canvas)

self.canvas.bind('<Button-3>', self.main\_context\_menu.popup\_menu)

if self.buff:

self.main\_context\_menu.add\_command(label="Вставить", command = self.main\_context\_menu.insert\_to\_dir)

#контекстное меню для файлов

self.file\_context\_menu = FileContextMenu(self, self.inner\_frame)

#контекстное меню для директории

self.dir\_context\_menu = DirContextMenu(self, self.inner\_frame)

i = 0

for item in dir\_list:

if os.path.isdir(str(path) + item):

#обрабатываем директории

if os.access(str(path) + item, os.R\_OK):

if (not self.hidden\_dir.get() and not item.startswith('.')) or self.hidden\_dir.get():

photo = tkinter.PhotoImage(file ="img/folder.png")

icon = tkinter.Label(self.inner\_frame, image=photo, bg = 'white')

icon.image = photo

icon.grid(row=i+1, column=0)

folder\_name = tkinter.Label(self.inner\_frame, text=item, bg = 'white', cursor = 'hand1')

folder\_name.bind("<Button-1>", self.move\_to\_dir)

folder\_name.bind("<Button-3>", self.dir\_context\_menu.popup\_menu)

folder\_name.grid(row=i+1, column=1, sticky='w')

else:

if (not self.hidden\_dir.get() and not item.startswith('.')) or self.hidden\_dir.get():

photo = tkinter.PhotoImage(file ="img/folder\_access.png")

icon = tkinter.Label(self.inner\_frame, image=photo, bg = 'white')

icon.image = photo

icon.grid(row=i+1, column=0)

folder\_name = tkinter.Label(self.inner\_frame, text=item, bg = 'white')

folder\_name.bind("<Button-1>", self.move\_to\_dir)

folder\_name.grid(row=i+1, column=1, sticky='w')

else:

#обрабатываем файлы

if (not self.hidden\_dir.get() and not item.startswith('.')) or self.hidden\_dir.get():

ext = self.take\_extention\_file(item)

#фото, картинки

if ext in ['jpeg', 'jpg', 'png', 'gif']:

photo = tkinter.PhotoImage(file ="img/photo.png")

icon = tkinter.Label(self.inner\_frame, image=photo, bg = 'white')

icon.image = photo

icon.grid(row=i+1, column=0)

file\_name = tkinter.Label(self.inner\_frame, text=item, bg = 'white')

file\_name.grid(row=i+1, column=1, sticky='w')

file\_name.bind("<Button-3>", self.file\_context\_menu.popup\_menu)

else:

#другие файлы

if os.access(str(path) + item, os.R\_OK):

photo = tkinter.PhotoImage(file ="img/file.png")

icon = tkinter.Label(self.inner\_frame, image=photo, bg = 'white')

icon.image = photo

icon.grid(row=i+1, column=0)

folder\_name = tkinter.Label(self.inner\_frame, text=item, bg = 'white')

folder\_name.grid(row=i+1, column=1, sticky='w')

folder\_name.bind("<Button-3>", self.file\_context\_menu.popup\_menu)

else:

photo = tkinter.PhotoImage(file ="img/file\_access.png")

icon = tkinter.Label(self.inner\_frame, image=photo, bg = 'white')

icon.image = photo

icon.grid(row=i+1, column=0)

folder\_name = tkinter.Label(self.inner\_frame, text=item, bg = 'white')

folder\_name.grid(row=i+1, column=1, sticky='w')

i += 1

#обновляем inner\_frame и устанавливаем прокрутку для нового содержимого

self.inner\_frame.update()

self.canvas.configure(scrollregion=self.canvas.bbox("all"))

def move\_to\_dir(self, event):

''' функция для перехода в выбранную директорию'''

elem = event.widget

dir\_name = elem["text"]

fool\_path = self.path\_text.get() + dir\_name

if os.path.isdir(fool\_path) and os.access(fool\_path, os.R\_OK):

old\_path = self.path\_text.get()

self.path\_text.set(old\_path + dir\_name + '/')

self.root\_click('<Button-1>')

self.refresh\_window()

def parent\_dir(self):

''' функция для перемещения в родительскую директорию'''

old\_path = [i for i in self.path\_text.get().split('/') if i]

new\_path = '/'+'/'.join(old\_path[:-1])

if not new\_path:

new\_path = '/'

if os.path.isdir(new\_path):

if new\_path == '/':

self.path\_text.set(new\_path)

else:

self.path\_text.set(new\_path + '/')

self.refresh\_window()

def take\_extention\_file(self, file\_name):

''' функция для получения расширения файла'''

ls = file\_name.split('.')

if len(ls)>1:

return ls[-1]

else:

return None

def refresh\_window(self):

''' функция для обновления текущего отображения директорий/файлов'''

for widget in self.inner\_frame.winfo\_children():

widget.destroy()

self.dir\_content()

self.canvas.yview\_moveto(0)

# И наконец, создание окна и запаковка виджетов:

win = MainWindow()

win.root.mainloop()