الجمهورية العربية السورية
وزارة التعليم العالي
جامعة تشرين
كلية الهندسة الميكانيكية والكهربائية _ الهمك
قسم هندسة الاتصالات والإلكترونيات

مشروع برمجي بعنوان:

TODO GUI Application Using TKinter

الباحثون : ديانا نضال عيسى 2491 يارا عيسى إبراهيم 2372

الباحثون من طلاب السنة الخامسة قسم هندسة الاتصالات والإلكترونيات

بإشراف الدكتور المهندس: مهند عيسى

الكلمات المفتاحية Key words :

Tkinter	الواجهة الرسومية المستخدمة في التصميم
Widget	أدوات واجهة المستخدم الرسومية
Geometry	المساحة
Button	الزر
Message box	كلاس مربع الرسائل
Task	المهمة
Label	خاصية إنشاء نص
Text	ادخال نص
Command	الامر
Entry	ادخال
Submit	ارسال
Def	خاصية تعريف التابع
Grid	شبكة
Root	جذر
Set	تعيين
Handling	معالجة
Font	نوع الخط
Ipedx,ipedy	البعد عن المحور x والبعد عن المحور y داخليا (ضمن الحقل)
Pady,padx	البعد عن المحور x والمحور y خارجيا
Sticky	خاصية ملئ الخلية في الشبكة
Tk	كلاس أساسي موجود في tkinter

ملخص:

إنشاء تطبيق To-do list بسيط يعتمد على واجهة المستخدم الرسومية حيث يمكنك إضافة مهمة وحذفها

مقدمة :

كثرة المهمات والواجبات اليومية المفروضة عليك قد تؤدي الى نسيان بعض المهام لذا تم انشاء تطبيق ال to-do الذي هو عبارة عن تطبيق يسمح لك بتنظيم وجدولة مهامك بصورة جيدة حيث تقوم بإضافة مهامك التي لاغنى عنها للإنجاز خلال فترة معينة ومن ثم كل مهمة تم تنفيذها تستطيع حذفها ومن ثم زيادة إنتاجيتك وتنظيم وقتك.

منهجية البحث:

في البداية بحثنا عن واجهة tkinter وكيفية تصميم واجهة رسومية باستخدامها من أزرار وعنوان وألوان وخلفيات وخطوط وأوامر وإضافة نصوص ثم رسمنا مخطط للواجهة التي أردنا تصميمها ومن ثم بدأنا بالتصميم العملي مستفيدين من فيديوهات على اليوتيوب شرحت بوضوح كل تعليمة نستطيع استخدامها وكيفية تطبيقها بالإضافة لمقالات عن واجهة tkinter وأكواد جاهزة مرفوعة على ال Github لتعلم آلية تنسيق الكود.

أهمية البحث:

1. التعرف على الواجهة الرسومية tkinter وكيفية استخدامها
2. بناء تطبيق يساعد في تنظيم المهام وجدولتها

مناقشة البحث:

تقدم python خيارات متعددة لتطوير واجهة المستخدم الرسومية (GUI) . من بين جميع طرق واجهة المستخدم الرسومية، TKinter هي الطريقة الأكثر استخداما. في هذه المقالة، سوف نتعلم كيفية إنشاء تطبيق To-do GUI باستخدام TKinter

دليل إنشاء TKinter خطوة بخطوة :

1.استيراد الوحدة –Tkinter

2.إنشاء النافذة الرئيسية (الحاوية container)

3. إضافة أي عدد من عناصر واجهة المستخدم إلى النافذة الرئيسية .

4. قم بتطبيق الحدث Trigger على عناصر واجهة المستخدم .

استيراد TKinter هو نفسه استيراد أي وحدة نمطية أخرى في كود python

نلاحظ أن اسم ال module في Python 2.x هو 'Tkinter' وفيPython 3.x كذلك 'Tkinter'

هناك طريقتان رئيسيتان مستخدمتان يحتاج المستخدم الى تذكرهما أثناء انشاء تطبيق python باستخدام واجهة المستخدم الرسومية هما:

1.لانشاء نافذة رئيسية ,يقدم tkinter طريقة
Tk(screenName=None, baseName=None, className='Tk', us eTk=1لتغيير اسم النافذة يمكنك تغيير ال className الى المطلوب

الكود الأساسي المستخدم لانشاء النافذة الرئيسية للكود هو `m=tkinter.Tk()` حيث m هو الاسم من كائن النافذة الرئيسية

2.mainloop() : هناك طريقة معروفة بالاسم mainloop() تستخدم عندما يكون التطبيق الخاص بك جاهزا للتشغيل نهائيا . ال mainloop() هي حلقة لانهائية تستخدم لتشغيل التطبيق .

انتظر حدثا طالما لم يتم اغلاق النافذة .

```
import tkinter
m = tkinter.Tk()
'''
widgets are added here
'''
m.mainloop()
```

يوفر tkinter أيضا الوصول الى التكوين الهندسي للودجات widget التي يمكنها تنظيم عناصر واجهة المستخدم في النوافذ الاصلية هناك ثلاث فئات رئيسية لمدير الهندسة :

1. pack() method : طريقة تنظم عناصر واجهة المستخدم في كتل قبل وضعها في عنصر واجهة المستخدم الأصلي
2. grid() method: طريقة الشبكة يتم فيها تنظيم عناصر واجهة المستخدم في الشبكة (بنية تشبه الجدول) قبل وضعها في عنصر واجهة المستخدم الأصلي .
3. place() method : طريقة المكان تنظم الحاجيات بوضعها في مواقع محددة يوجهها المبرمج .

هناك عدد من الأدوات التي يمكن وضعها في تطبيق tkinter الخاص بك .يتم شرح بعض عناصر واجهة المستخدم الرئيسية أدناه:

1. الزرButton : يتم استخدام هذه الأداة لاضافة زر في التطبيق الخاص بك

الصيغة العامة هي :

w=Button(master, option=value)

حيث ال master هو البارامتر الرئيسي المستخدم لتمثيل النافذة الاصلية .

هناك عدد من الخيارات التي تستخدم لتغيير تنسيق الازرار .يمكن تمرير عدد من الخيارات كبارامترات مفصولة بفواصل ,تم ذكر بعضها أدناه:

- **activebackground :** لضبط لون الخلفية عندما يكون الزر تحت المؤشر
- **bg :** لتعيين لون الخلفية العادي
- **Command:** لاستدعاء وظيفة
- **Font :** لضبط الخط على زر التسمية
- **Image :** لضبط الصورة على الزر
- **Width :** لضبط عرض الزر
- **Height :** لضبط ارتفاع الزر

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
r.mainloop()
```

2.Label: يشير إلى مربع العرض حيث يمكنك وضع أي نص أو صورة كما يمكن تحديثها في أي وقت حسب الكود .

الصيغة العامة:

```
w=Label(master, option=value)
```

هناك عدد من الخيارات التي تستخدم لتغيير تنسيق الأداة .

- **Bg:** لتعيين لون الخلفية
- **command :**لاستدعاء التابع
- **font:** لتعيين الخط على تسمية الزر.
- **image**: لضبط الصورة على الزر
- **width:** لضبط عرض الزر
- **height:** لضبط ارتفاع الزر.

3.الإدخال Entry:يتم استخدامه لإدخال نص (سطر واحد) من المستخدم...لإدخال نص متعدد الأسطر, يتم استخدام عنصر واجهة المستخدم للنص.

الصيغة العامة:

```
w=Entry(master, option=value)
```

هناك عدد من الخيارات التي تستخدم لتغيير تنسيق الأداة:

- **bd:**لتعيين عرض الحدود بالبيكسل
- **bg:** لتعيين لون الخلفية
- **cursor:** لضبط المؤشر المستخدم
- **command:** استدعاء وظيفة
- **highlightcolor:** لضبط اللون الموضح في تسليط الضوء على التركيز
- **width:** لضبط عرض الزر
- **height:** لضبط ارتفاع الزر

4. النص Text: لتحرير نص متعدد الأسطر وتنسيقه

الصيغة العامة:

w =Text(master, option=value)

هناك عدد من الخيارات التي تستخدم لتغيير تنسيق النص

- **highlightcolor:** لتعيين لون تسليط الضوء على التركيز عندما يكون عنصر واجهة المستخدم التركيز
- **insertbackground:**لتعيين خلفية الاداء.
- **bg:** لتعيين لون الخلفية
- **font:** لتعيين الخط على تسمية الزر
- **image:** لضبط الصورة على الأداة
- **Width:** لضبط عرض الأداة
- **Height:** لضبط ارتفاع الأداة

الجزء العملي:

الكود الموافق:

```
# import all functions from the tkinter
from tkinter import *

# import messagebox class from tkinter
from tkinter import messagebox

# global list is declare for storing all the task
tasks_list = []

# global variable is declare for counting the task
counter = 1


# Function for checking input error when empty input is given in task field
def inputError():
    # check for enter task field is empty or not
    if enterTaskField.get() == "":
        # show the error message
        messagebox.showerror("Input Error")

        return 0

    return 1


# Function for clearing the contents of task number text field
def clear_taskNumberField():
    # clear the content of task number text field
```

```
    taskNumberField.delete(0.0, END)


# Function for clearing the contents of task entry field
def clear_taskField():
    # clear the content of task field entry box
    enterTaskField.delete(0, END)


# Function for inserting the contents from the task entry field to the text area
def insertTask():
    global counter

    # check for error
    value = inputError()

    # if error occur then return
    if value == 0:
        return

    # get the task string concatenating
    # with new line character
    content = enterTaskField.get() + "\n"

    # store task in the list
    tasks_list.append(content)

    # insert content of task entry field to the text area
    # add task one by one in below one by one
    TextArea.insert('end -1 chars', "[ " + str(counter) + " ] " + content)

    # incremented
    counter += 1

    # function calling for deleting the content of task field
    clear_taskField()


# function for deleting the specified task
def delete():
    global counter

    # handling the empty task error
    if len(tasks_list) == 0:
        messagebox.showerror("No task")
        return

    # get the task number, which is required to delete
    number = taskNumberField.get(1.0, END)

    # checking for input error when empty input in task number field
    if number == "\n":
        messagebox.showerror("input error")
        return

    else:
        task_no = int(number)

    # function calling for deleting the  content of task number field
    clear_taskNumberField()

    # deleted specified task from the list
    tasks_list.pop(task_no - 1)
```

```
    # decremented
    counter -= 1

    # whole content of text area widget is deleted
    TextArea.delete(1.0, END)

    # rewriting the task after deleting one task at a time
    for i in range(len(tasks_list)):
        TextArea.insert('end -1 chars', "[ " + str(i + 1) + " ] " + tasks_list[i])


# Driver code
if __name__ == "__main__":
    # create a GUI window
    gui = Tk()

    # set the background colour of GUI window
    gui.configure(background="gray")

    # set the title of GUI window
    gui.title("ToDo App")

    # set the configuration of GUI window
    gui.geometry("250x300")

    # create a label : Enter Your Task
    enterTask = Label(gui, text="Enter Your Task", bg="light green")

    # create a text entry box for typing the task
    enterTaskField = Entry(gui)

    # create a Submit Button and place into the root window
    # when user press the button, the command or
    # function affiliated to that button is executed
    Submit = Button(gui, text="Submit", fg="Black", bg="light green",
command=insertTask)

    # create a text area for the root
    # with lunida 13 font
    # text area is for writing the content
    TextArea = Text(gui, height=5, width=25, font="lucida 13")

    # create a label : Delete Task Number
    taskNumber = Label(gui, text="Delete Task Number", bg="light green")

    taskNumberField = Text(gui, height=1, width=2, font="lucida 13")

    # create a Delete Button and place into the root window
    # when user press the button, the command or
    # function affiliated to that button is executed .
    delete = Button(gui, text="Delete", fg="Black", bg="light green", command=delete)

    # create a Exit Button and place into the root window
    # when user press the button, the command or
    # function affiliated to that button is executed .
    Exit = Button(gui, text="Exit", fg="white", bg="black", command=exit)


    # grid method is used for placing
    # the widgets at respective positions
    # in table like structure.
    enterTask.grid(row=0, column=2)
```

```
    # ipadx attributed set the entry box horizontal size
    enterTaskField.grid(row=1, column=2, ipadx=50)

    Submit.grid(row=2, column=2)

    # padx attributed provide x-axis margin
    # from the root window to the widget.
    TextArea.grid(row=3, column=2, padx=10, sticky=W)

    taskNumber.grid(row=4, column=2, pady=5)

    taskNumberField.grid(row=5, column=2)

    # pady attributed provide y-axis
    # margin from the widget.
    delete.grid(row=6, column=2, pady=5)

    Exit.grid(row=7, column=2)

    # start the GUI
    gui.mainloop()
```

شرح الكود:

```
from tkinter import *
```
استيراد جميع الوظائف من tkinter//

```
from tkinter import messagebox
```
استيراد الكلاس messagebox يستخدم لأظهار مربعات الرسائل //

تم الإعلان عن القائمة العمومية لتخزين كافة المهام//

```
ks_list = []
counter = 1
```
عداد لحساب عدد المهمات المدخلة //

تابع للتحقق من خطأ الادخال حيث في حال تم ادخال حقل فارغ سوف تظهر رسالة خطأ

```
def inputError():
    if enterTaskField.get() == "":
        messagebox.showerror("Input Error")
```

```
        return 0

    return 1
def clear_taskNumberField():  // تابع مسح حقل نص رقم المهمة
    taskNumberField.delete(0.0, END)

def clear_taskField():// تابع لمسح المحتويات من حقل إدخال المهمة

                                    //إلغاء تحديد مربع إدخال حقل محتوى المهمة
 enterTaskField.delete(0, END)

// تابع لإدراج المحتويات من حقل إدخال المهمة إلى مربع منطقة النص
def insertTask():
    global counter

value = inputError()        // التحقق من وجود خطأ
if value == 0:
        return

                                    //في حال لم يحدث خطأ يتم إضافة مهمة بسطر جديد
 content = enterTaskField.get() + "\n"


                                    //تخزين المهمة في القائمة
tasks_list.append(content)
```

ادراج محتوى حقل إدخال المهمة بحيث تضاف الهمات الواحدة تلو الأخرى //

```
TextArea.insert('end -1 chars', "[ " +
str(counter) + " ] " + content)
```

```
counter += 1    //زيادة العداد بعد إضافة كل مهمة
```

```
  clear_taskField() // تابع يتم استدعائه لحذف محتوى حقل المهمة
```

```
def delete():    //تابع لحذف المهمة المحددة
    global counter
```

```
if len(tasks_list) == 0: //معالجة خطأ المهمة الفارغة
        messagebox.showerror("No task")
        return
 counter -= 1     //انقاص عداد المهام بمقدار1
```

```
/                                                    // جلب رقم المهمة المراد حذفها
number = taskNumberField.get(1.0, END)
```

```
                                    // في حال عدم ادخال رقم في حقل الرقم ستظهر رسالة خطأ
```

```
if number == "\n":
        messagebox.showerror("input error")
                return

    else:
        task_no = int(number)
```

```
clear_taskNumberField()
```
//تابع لحذف محتوى حقل رقم المهمة

```
tasks_list.pop(task_no - 1)
```
//حذف المهمة المحددة من القائمة

```
TextArea.delete(1.0, END)
```
//حذف كل المهام الموجودة في القائمة

// حلقة تمكننا من أعادة ادخال مهمة جديدة بعد حذف مهمة واحدة في كل مرة

```
for i in range(len(tasks_list)):
        TextArea.insert('end -1 chars', "[ " +
str(i + 1) + " ] " + tasks_list[i])
```

```
if __name__ == "__main__":
```
//تشغيل الكود

```
gui = Tk()
```
// إنشاء نافذة واجهة المستخدم الرسومية

//إنشاء زر للخروج من الواجهة

```
gui.configure(background="gray")
```

```
gui.title("ToDo App")
```
// إضافة عنوان للواجهة

```
gui.geometry("250x300")
```
//تعيين مساحة الواجهة(عرض*طول)

//إنشاء نص"إدخال المهمة بلون خلفية أخضر فاتح مع لون خط افتراضي أسود"

```
enterTask = Label(gui, text="Enter Your Task",
bg="light green")
```

إنشاء مربع إنشاء مربع إدخال نص لكتابة المهمة

```
enterTaskField = Entry(gui)
```

إنشاء زر ارسال ووضعه ضمن النافذة عندما يضغط المستخدم عليه يتم تنفيذ الوظيفة التابعة لهذا الزر والتي هي هنا ادخال مهمة//

```
Submit = Button(gui, text="Submit", fg="Black",
bg="light green", command=insertTask)
```

إنشاء المنطقة المخصصة لكتابة المحتوى بإرتفاع وعرض ونمط خط معين//

```
TextArea = Text(gui, height=5, width=25,
font="lucida 13")
```

إضافة نص "حذف رقم المهمة" بلون خلفية أخضر فاتح //

```
 taskNumber = Label(gui, text="Delete Task
Number", bg="light green")
```

انشاء حقل إضافة رقم المهمة المراد حذفها //

```
taskNumberField = Text(gui, height=1, width=2,
font="lucida 13")
```

إنشاء زر حذف ووضعه في النافذة عند الضغط عليه ينفذ أمر حذف //

```
delete = Button(gui, text="Delete", fg="Black",
bg="light green", command=delete)
```

إنشاء زر للخروج من الواجهة//

```
Exit = Button(gui, text="Exit", fg="white",
bg="black", command=exit)
```

لتحديد مواضع الودجات ضمن الواجهة أي تحديد الحجم والموقع باستخدام الميثود //grid

```
enterTask.grid(row=0, column=2)
```

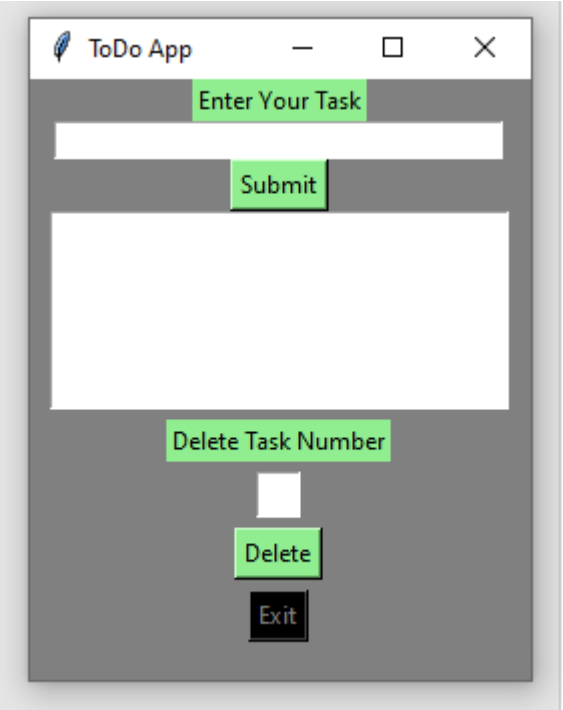
```
enterTaskField.grid(row=1, column=2, ipadx=50)

Submit.grid(row=2, column=2)
delete.grid(row=6, column=2, pady=5)

Exit.grid(row=7, column=2)
gui.mainloop()        تابع تشغيل الواجهة//
```

مناقشة النتائج:

واجهة التطبيق:



فيديو توضيحي لالية عمل التطبيق كيفية إضافة المهام وحذفها ومناقشة حالات الادخال الخاطئة:

bandicam 2022-06-10 15-43-39-500.mp4

تم رفع المشروع على Github على الرابط التالي:

https://github.com/dianaissa/the-finally_project/upload

المراجع:

https://www.geeksforgeeks.org

https://www.youtube.com/watch?v=NzEh3Dfa4Vg&list=PLSiLeKadTQ7nLJxpQo1-944miQKlheu-v

https://www.youtube.com/watch?v=YXPyB4XeYLA