ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΔΟΜΟΣΤΑΤΙΚΗΣ

Ακαδ. Έτος: 2016-2017

Μάθημα: Εφαρμογές Η/Υ

Δευτέρα, 14/11/2016

Διδάσκοντες:
Ν.Δ. Λαγαρός (Επικ. Καθηγητής), Αθ. Στάμος (ΕΔΙΠ), Χ. Φραγκουδάκης (ΕΔΙΠ)

Παραδείγματα για την 7η παράδοση – Γραφικό περιβάλλον διεπαφής με χρήστη

1. Υπολογισμός αντοχής δοκιμίων

Να συνταχθεί πρόγραμμα σε python το οποίο:
1. Να κάνει GUI για εισαγωγή αντοχών N (για παράδειγμα N=20) δοκιμίων.
2. Κάνοντας κλικ σε κουμπί να υπολογίζει το μ.ο. των αντοχών αγνοώντας τα M τελευταία δοκίμια αν είναι κενά (M<N).
3. Να δείχνει το αποτέλεσμα σε Entry readonly.

Λύση

Καταρχήν συντάσσεται αντικείμενο Entry με δυνατότητα εγγραφής ακόμα και αν είναι readonly, για λόγους ευκολίας.

```
class Entry2(tk.Entry):
    "An Entry widget with set capability."

    def set(self, v):
        "Set the v (numeric or text) to the Entry widget, even if it is readonly."
        state2 = self["state"]
        self["state"] = tk.NORMAL
        self.delete(0, tk.END)
        self.insert(0, "{}".format(v))
        self["state"] = state2
```

Το αντικείμενο κληρονομεί όλες τις μεθόδους του Entry, και προσθέτει μία καινούρια, η οποία αποθηκεύει την υφιστάμενη state, γράφει την τιμή v στο widget, και μετά επαναφέρει την αρχική state.
Στη συνέχεια συντάσσεται μέθοδος που δημιουργεί Entry και Label με κείμενο επεξήγησης του Entry, σε συγκεκριμένη γραμμή και στήλη του νοητού κανάβου. Επιστρέφει το αντικείμενο Entry.

```
class DokWin:
    "Παράθυρο για την εισαγωγή και υπολογισμό δοκιμίων."
    def addLabEntry(self, text2, width2, state2, ir2, ic2):
        "Add a label and and an Entry at the same row."
        temp = tk.Label(self.root, text=text2)
        temp.grid(row=ir2, column=ic2, sticky="e")
        temp = Entry2(self.root, width=width2, state=state2)
        temp.grid(row=ir2, column=ic2+1, sticky="we")
        return temp
```

Τα widgets Label και Entry των δοκιμίων τοποθετούνται στην πρώτη και δεύτερη στήλη (0 και 1). Το πλεονέκτημα της προγραμματιστικής δημιουργίας widgets, σε αντίθεση τη δημιουργία με το χέρι, είναι ότι μπορούμε να χρησιμοποιήσουμε την εντολή for για να κάνουμε μαζική δημιουργία 20 widgets. Ένα άλλο πλεονέκτημα είναι ο οδηγός γεωμετρίας που αυτόματα αλλάζει το μέγεθος των widgets καθώς αλλάζει το μέγεθος του παραθύρου.
Στην τρίτη στήλη (2) τοποθετείται ένα Label με μερικά κενά για να αφήσει λίγο χώρο κατά τον άξονα x. Στην τέταρτη και πέμπτη στήλη (3 και 4) τοποθετείται το κομβίο (κουμπί – button) που ξεκινά τον υπολογισμό και το Entry readonly για το αποτέλεσμα. Το κομβίο λειτουργεί περίπου ως μενού, δηλαδή

καθορίζουμε τη συνάρτηση/μέθοδο που θα κληθεί όταν ο χρήστης το πατήσει.

```
    def init1(self):
        "Create the widgets."
        N = 20
        ir = 0
        temp = tk.Label(self.root, text="Σχολή Πολιτικών Μηχανικών, ΕΜΠ",
            background="yellow")
        temp.grid(row=ir, column=0, columnspan=2, sticky="w", pady=20)

        self.wids = []
        for i in range(N):
            ir += 1
            temp = self.addLabEntry(text2="Δοκίμιο {} (MPa)".format(i+1), width2=15,
                state2=tk.NORMAL, ir2=ir, ic2=0)
            self.wids.append(temp)

        ir = 1
        temp = tk.Label(self.root, text="   ")   #Leave some space
        temp.grid(row=ir, column=2)
        temp = tk.Button(self.root, text="Υπολογισμός", command=self.ypol,
            bg="lightcyan", activebackground="cyan")
        temp.grid(row=ir, column=3)

        ir += 1
        self.entAver = self.addLabEntry(text2="Μέση Αντοχή (MPa)", width2=15,
            state2="readonly", ir2=ir, ic2=3)

        self.root.columnconfigure(1, weight=1)
```

Επιτρέπεται τα τελευταία M widgets Entry να είναι κενά, όπου M<N, δηλαδή τουλάχιστον το πρώτο widget δεν πρέπει να είναι κενό. Έτσι ελέγχουμε τις τιμές όλων των widgets από το Νοστό (N-1) έως το δεύτερο (1) για να προσδιορίσουμε το M. Στη συνέχεια ελέγχουμε τα πρώτα M widgets αν έχουν αριθμητική τιμή και μεγαλύτερη του μηδενός. Αν ναι επιστρέφουμε τις τιμές, αν όχι επιστρέφουμε None. Ας σημειωθεί ότι το M είναι το πλήθος των μη κενών widgets.

```
    def getVals(self):
        "Validate and return the values of the widgets."
        M = len(self.wids)
        #Determine blank entries at the end
        while M > 1:    #At least 1 entry must be non-empty
            if self.wids[M-1].get().strip() != "": break
            M -= 1

        vals = []
        for i in range(M):
            wid = self.wids[i]
            t = wid.get().strip().replace(",", ".")
            try: temp = float(t)
            except ValueError: temp = None
            if temp is None or temp <= 0:
                tk.messagebox.showinfo("Error", "A positive number was expected",
                    parent=self.root, icon=tk.messagebox.ERROR)
                wid.focus_set()
                return None
            vals.append(temp)
        return vals
```

Στη μέθοδο ypol() διαβάζουμε τις τιμές των widgets και αν αυτές έχουν λάθος δεν προχωρούμε στους υπολογισμούς. Αλλιώς υπολογίζουμε το μέσο όρο και τον τοποθετούμε στο widget του αποτελέσματος.

```
    def ypol(self):
        "Compute the average."
        vals = self.getVals()
        if vals is None: return   #An error has occured
        av = sum(vals)/len(vals)
        self.entAver.set("{:.1f}".format(av))
```

Για να τερματίσουμε την εφαρμογή, διαγράφουμε τις μεταβλητές στιγμιότυπου που αποθηκεύουν widgets και καλούμε την μέθοδο destroy() του παραθύρου.

```
    def destroy(self):
        "Delete all references to widgets and the destroy the root window (close the
window)."
        del self.wids, self.entAver
        self.root.destroy()
```

Το υπόλοιπο πρόγραμμα φαίνεται παρακάτω:

```
"Create multiple widgets programmaticaly."
import tkinter as tk
import tkinter.font, tkinter.messagebox

class DokWin:
    "Παράθυρο για την εισαγωγή και υπολογισμό δοκιμίων."
    def __init__(self):
        "Make the window."
        self.root = tk.Tk()
        self.root.title("Δοκίμια - Κύριο")
        self.setFonts()
        self.init1()

    def setFonts(self):
        "Set the fonts a little bigger."
        for t in ("TkDefaultFont", "TkFixedFont", "TkMenuFont", "TkCaptionFont",
            "TkTextFont"):
            font1 = tk.font.Font(name=t, exists=True)
            font1.config(size=14)   # Negative size means size in pixels
        font1.config(family="Arial")

    def mainloop(self):
        "Start the execution of the GUI."
        self.root.mainloop()

win = DokWin()
win.mainloop()
```

2. Υπολογισμός αντοχής δοκιμίων – δυναμική προσθήκη/αφαίρεση widgets

Στο προηγούμενο παράδειγμα το GUI να ξεκινά με ένα Entry και να δίνεται δυνατότητα στο χρήστη να προσθέτει νέα Entry για να δίνει και άλλα δοκίμια, και να αφαιρεί Entry (κάθε φορά το τελευταίο) αν δεν χρειάζονται.

Λύση

Εφόσον η δημιουργία των widgets γίνεται προγραμματιστικά, τυπικά στην αρχή της εκτέλεσης του προγράμματος, δεν υπάρχει λόγος να μην γίνεται, με το ίδιο ακριβώς τρόπο, και στη μετέπειτα εκτέλεση του προγράμματος, δηλαδή κατά την εκτέλεση του GUI. Απλά θα δημιουργηθεί ένα μενού ή ένα κομβίο, το οποίο όταν πατηθεί θα εκτελεί μία μέθοδο που δημιουργεί νέα widgets. Ομοίως για τη διαγραφή widgets. Κατ' αντιστοιχία με το παράθυρο, σε όλα τα widgets υπάρχει η μέθοδος destroy() που διαγράφει το widget από την εσωτερική λίστα του παραθύρου και από την οθόνη.

Αρχικά θέτουμε τη μεταβλητή στιγμιότυπου nDok=1, και κατά την αρχή της εκτέλεσης του προγράμματος δημιουργούμε ένα Label και ένα Entry και τα αποθηκεύουμε στις λίστες labs και wids αντίστοιχα. Επίσης δημιουργούμε το Entry για το αποτέλεσμα, το κομβίο για τον υπολογισμό, 2 ακόμα κομβία για την προσθήκη και διαγραφή widgets, και 2 Label για να αφήσουμε λίγο χώρο κατά x και κατά y. Στο Label που αφήνει χώρο κατά x δεν χρειάζεται να βάλουμε κάποιους κενούς χαρακτήρες, διότι το ύψος του Label δεν μπορεί να είναι μηδέν (είναι περίπου όσο το μέγεθος της γραμματοσειράς).

```
class DokWin:
    "Παράθυρο για την εισαγωγή και υπολογισμό δοκιμίων."
    def __init__(self):
        "Make the window."
        self.nDok = 1       #Πλήθος δοκιμίων
        self.root = tk.Tk()
```

```
        self.root.title("Δοκίμια - Κύριο")
        self.setFonts()
        self.init1()

    def init1(self):
        "Create the widgets."
        ir = 0
        temp = tk.Label(self.root, text="Σχολή Πολιτικών Μηχανικών, ΕΜΠ",
            background="yellow")
        temp.grid(row=ir, column=0, columnspan=2, sticky="w", pady=20)

        self.wids = []
        self.labs = []
        for i in range(self.nDok):
            ir += 1
            lab, ent = self.addLabEntry(text2="Δοκίμιο {} (MPa)".format(i+1), width2=15,
                state2=tk.NORMAL, ir2=ir, ic2=0)
            self.wids.append(ent)
            self.labs.append(lab)

        ir = 1
        temp = tk.Label(self.root, text="   ")   #Leave some horizontal space
        temp.grid(row=ir, column=2)

        temp = tk.Button(self.root, text="Προσθήκη δοκιμίου", command=self.add1,
            bg="lightcyan", activebackground="cyan")
        temp.grid(row=ir, column=3)
        temp = tk.Button(self.root, text="Αφαίρεση δοκιμίου", command=self.delete1,
            bg="lightcyan", activebackground="cyan")
        temp.grid(row=ir, column=4)

        ir += 1
        temp = tk.Label(self.root, text="")   #Κενή σειρά
        temp.grid(row=ir, column=3)

        ir += 1
        temp = tk.Button(self.root, text="Υπολογισμός", command=self.ypol, fg="white",
            bg="green", activebackground="lightgreen")
        temp.grid(row=ir, column=3)

        ir += 1
        temp, self.entAver = self.addLabEntry(text2="Μέση Αντοχή (MPa)", width2=15,
            state2="readonly", ir2=ir, ic2=3)

        self.root.columnconfigure(1, weight=1)
```

Η μέθοδος που δημιουργεί νέο προσθέτει 1 στο nDok. Η γραμμή στο νοητό κάναβο των νέων widgets θα έπρεπε να είναι η nDok-1 (διότι στην python η αρίθμηση αρχίζει από το 0), αλλά επειδή υπάρχει ήδη ένα Label με το όνομα της σχολής στη γραμμή 0, τα νέα widgets θα δημιουργηθούν στη γραμμή nDok-1+1=Ndok. Τα νέα widgets αποθηκεύονται στις λίστες wids και labs. Επειδή πιθανότατα ο χρήστης θα θέλει να βάλει αντοχή στο νέο Entry, γίνεται εστίαση στο νέο Entry.

```
    def add1(self):
        "Προσθήκη ενός δοκιμίου."
        self.nDok += 1
        lab, ent = self.addLabEntry(text2="Δοκίμιο {} (Mpa)".format(self.nDok),
            width2=15, state2=tk.NORMAL, ir2=self.nDok, ic2=0)
        self.wids.append(ent)
        self.labs.append(lab)
        ent.focus_set()  #Focus to the new widget; probably we enter a new value there
```

Η μέθοδος που διαγράφει καλεί τη μέθοδος detroy() για το τελευταίο Label και τελευταίο Entry, και στη συνέχεια τα διαγράφει από τις αντίστοιχες λίστες.

```
    def delete1(self):
        "Αφαίρεση τελευταίου δοκιμίου."
        if self.nDok <= 1: return    #Τουλάχιστον ένα δοκίμιο αφήνουμε
        temp = self.wids.pop()
        temp.destroy()
```

```
        temp = self.labs.pop()
        temp.destroy()
        self.nDok -= 1
```

Για να τερματίσουμε την εφαρμογή, διαγράφουμε τις μεταβλητές στιγμιότυπου που αποθηκεύουν widgets, δηλαδή από προηγουμένως επιπλέον και τη λίστα labs, και καλούμε την μέθοδο destroy() του παραθύρου.

```
    def destroy(self):
          "Delete all references to widgets and the destroy the root window (close the
window)."
        del self.wids, self.labs, self.entAver
        self.root.destroy()
```

Η μέθοδος getVals() τώρα δεν χρειάζεται να αγνοεί τα τελευταία κενά widgets.

```
    def getVals(self):
        "Validate and return the values of the widgets."
        vals = []
        for i in range(self.nDok):
            wid = self.wids[i]
            t = wid.get().strip().replace(",", ".")
            try: temp = float(t)
            except ValueError: temp = None
            if temp is None or temp <= 0:
                tk.messagebox.showinfo("Error", "A positive number was expected",
                    parent=self.root, icon=tk.messagebox.ERROR)
                wid.focus_set()
                return None
            vals.append(temp)
        return vals
```

Το υπόλοιπο πρόγραμμα δεν αλλάζει.

3. Υπολογισμός αντοχής δοκιμίων – πολλά παράθυρα

Στο προηγούμενο παράδειγμα να δημιουργηθεί δεύτερο και τρίτο παράθυρο με τα ίδια widgets. Αλλάξτε τα παράθυρα έτσι ώστε να κληρονομούν το παράθυρο tk.Tk και το παράθυρο tk.Toplevel.

Λύση

Το κύριο (master) παράθυρο στο tkinter είναι στην κλάση tk.Tk και μπορεί να υπάρχει μόνο ένα σε κάθε εφαρμογή, ενώ αν χρειάζονται και άλλα δευτερεύοντα παράθυρα υπάρχει η κλάση tk.Toplevel. Αν κλείσει το κύριο παράθυρα, τερματίζεται όλη η εφαρμογή, ενώ αν κλείσει ένα δευτερεύον κλείνει μόνο αυτό. Συνεπώς στο παράδειγμα πρέπει να υπάρχει ένα παράθυρο που θα κληρονομεί από το tk.Tk (βασική κλάση) και 2 άλλα παράθυρα που θα κληρονομούν από το tk.Toplevel (βασική κλάση). Τα 3 παράθυρα έχουν τον ίδιο κώδικα εκτός από την βασική κλάση Tk ή Toplevel και τον τίτλο. Για αυτό θα συνταχθεί και άλλη μία βασική κλάση που θα περιέχει τον κοινό κώδικα. Τα παράθυρα θα κληρονομούν αυτή την κλάση και την κλάση tk.Tk ή tk.Toplevel (πολλαπλή κληρονομικότητα – multiple inheritance).
Επειδή οι νέες κλάσεις *είναι* παράθυρα, στον κώδικα του προηγούμενου παραδείγματος θα αντικαταστήσουμε το self.root με το self:

```
class BaseWin:
    "Base class with the common code."
    def init1(self):
        "Create the widgets."
        ir = 0
        temp = tk.Label(self, text="Σχολή Πολιτικών Μηχανικών, ΕΜΠ", background="yellow")
        temp.grid(row=ir, column=0, columnspan=2, sticky="w", pady=20)

        self.wids = []
        self.labs = []
        for i in range(self.nDok):
            ir += 1
            lab, ent = self.addLabEntry(text2="Δοκίμιο {} (MPa)".format(i+1), width2=15,
                state2=tk.NORMAL, ir2=ir, ic2=0)
            self.wids.append(ent)
            self.labs.append(lab)
```

```
        ir = 1
        temp = tk.Label(self, text="   ")   #Leave some horizontal space
        temp.grid(row=ir, column=2)

        temp = tk.Button(self, text="Προσθήκη δοκιμίου", command=self.add1,
            bg="lightcyan", activebackground="cyan")
        temp.grid(row=ir, column=3)
        temp = tk.Button(self, text="Αφαίρεση δοκιμίου", command=self.delete1,
            bg="lightcyan", activebackground="cyan")
        temp.grid(row=ir, column=4)

        ir += 1
        temp = tk.Label(self, text="")   #Κενή σειρά
        temp.grid(row=ir, column=3)

        ir += 1
        temp = tk.Button(self, text="Υπολογισμός", command=self.ypol, fg="white",
            bg="green", activebackground="lightgreen")
        temp.grid(row=ir, column=3)

        ir += 1
        temp, self.entAver = self.addLabEntry(text2="Μέση Αντοχή (MPa)", width2=15,
            state2="readonly", ir2=ir, ic2=3)

        self.columnconfigure(1, weight=1)

    def addLabEntry(self, text2, width2, state2, ir2, ic2):
        "Add a label and and an Entry at the same row."
        lab = tk.Label(self, text=text2)
        lab.grid(row=ir2, column=ic2, sticky="e")
        ent = Entry2(self, width=width2, state=state2)
        ent.grid(row=ir2, column=ic2+1, sticky="we")
        return lab, ent

    def add1(self):
        "Προσθήκη ενός δοκιμίου."
        self.nDok += 1
        lab, ent = self.addLabEntry(text2="Δοκίμιο {} (Mpa)".format(self.nDok),
            width2=15, state2=tk.NORMAL, ir2=self.nDok, ic2=0)
        self.wids.append(ent)
        self.labs.append(lab)
        ent.focus_set()  #Focus to the new widget; probably we enter a new value there

    def delete1(self):
        "Αφαίρεση τελευταίου δοκιμίου."
        if self.nDok <= 1: return    #Τουλάχιστον ένα δοκίμιο αφήνουμε
        temp = self.wids.pop()
        temp.destroy()
        temp = self.labs.pop()
        temp.destroy()
        self.nDok -= 1

    def setFonts(self):
        "Set the fonts a little bigger."
        for t in ("TkDefaultFont", "TkFixedFont", "TkMenuFont", "TkCaptionFont",
            "TkTextFont"):
            font1 = tk.font.Font(name=t, exists=True)
            font1.config(size=14)   # Negative size means size in pixels
        font1.config(family="Arial")

    def getVals(self):
        "Validate and return the values of the widgets."
        vals = []
        for i in range(self.nDok):
            wid = self.wids[i]
            t = wid.get().strip().replace(",", ".")
            try: temp = float(t)
            except ValueError: temp = None
            if temp is None or temp <= 0:
                tk.messagebox.showinfo("Error", "A positive number was expected",
```

```
                    parent=self, icon=tk.messagebox.ERROR)
                wid.focus_set()
                return None
            vals.append(temp)
        return vals

    def ypol(self):
        "Compute the average."
        vals = self.getVals()
        if vals is None: return   #An error has occured
        av = sum(vals)/len(vals)
        self.entAver.set("{:.1f}".format(av))

    def destroy(self):
        "Delete all references to widgets and the destroy the window (close the window)."
        del self.wids, self.labs, self.entAver
        super().destroy()   #Call the destroy method of the superclass
```

Στο κύριο παράθυρο αντικαθιστούμε τη μέθοδο __init__() αλλά καλούμε και την __init__ της βασικής κλάσης μέσω της συνάρτησης της Python super(), η οποία καλεί την __init__() σε όλες τις βασικές κλάσεις (αν την έχουν).

```
class DokWinMain(tk.Tk, BaseWin):
    "Κύριο Παράθυρο για την εισαγωγή και υπολογισμό δοκιμίων."
    def __init__(self):
        "Make the window."
        super().__init__()   #Call the __init__ of the superclass
        self.nDok = 1       #Πλήθος δοκιμίων
        self.title("Δοκίμια - Κύριο")
        self.setFonts()
        self.init1()
```

Όταν δημιουργείται το παράθυρο Toplevel, πρέπει να δοθεί το αντικείμενο που είναι γονέας του, δηλαδή το κύριο παράθυρο:

win = Toplevel(master)

Συνεπώς στη μέθοδο __init__ του δευτερεύοντος παραθύρου πρέπει να μπει το όρισμα master. Επίσης βάζουμε και το όρισμα inum που είναι ο αύξων αριθμός του δευτερεύοντος και ο οποίος θα μπει στον τίτλο.

```
class DokWin(tk.Toplevel, BaseWin):
    "Κύριο Παράθυρο για την εισαγωγή και υπολογισμό δοκιμίων."
    def __init__(self, master, inum):
        "Make the window."
        super().__init__(master)   #Call the __init__ of the superclass
        self.nDok = 1       #Πλήθος δοκιμίων
        self.title("Δοκίμια - Δευτερεύον "+str(inum))
        self.setFonts()
        self.init1()
```

Τέλος δημιουργούμε ένα κύριο και 2 δευτερεύοντα παράθυρα και ξεκινάμε το GUI.

```
winm = DokWinMain()
temp = DokWin(winm, 1)
temp = DokWin(winm, 2)
winm.mainloop()  #No new mainloop() function is needed
```