Πανεπιστήμιο Πατρών
Τμήμα Ηλεκτρολόγων Μηχανικών & Τεχνολογίας Υπολογιστών

Υ106 Εισαγωγή Υπολογιστές

1ο εξάμηνο

Python tkinter

γραφικές διεπαφές χρήστη

[κεφάλαιο 10]

tkinter

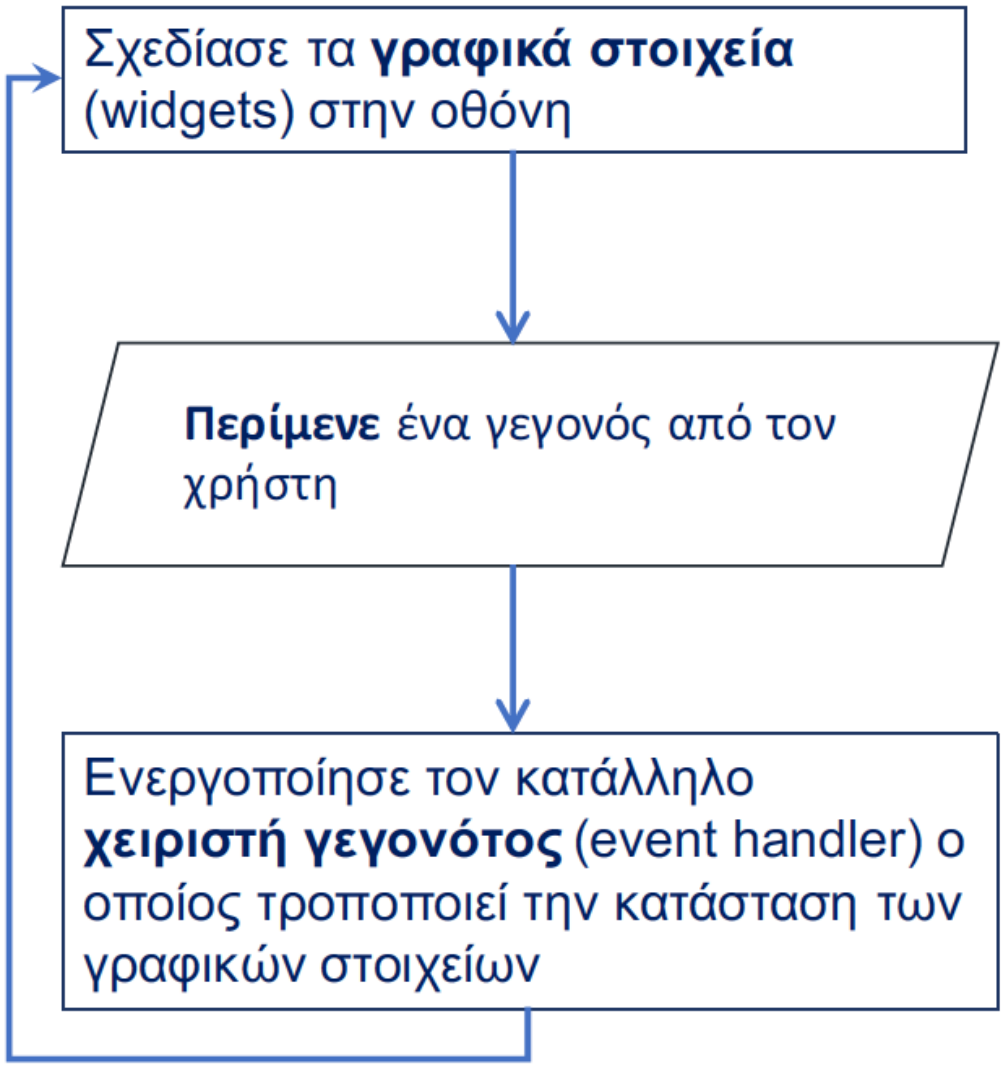
εισαγωγή στον προγραμματισμό με γεγονότα (event-based programming)

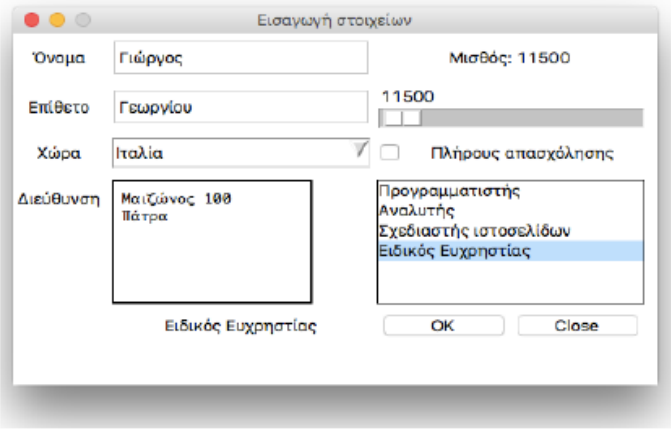
https://realpython.com/python-gui-tkinter/
https://dafarry.github.io/tkinterbook/

tkinter

- η πιο απλή γραφική βιβλιοθήκη της Python
- **tk interface** διεπαφή της Python 3 με τη γραφική βιβλιοθήκη **tk,** της tcl/tk
- περιλαμβάνεται στις βασικές βιβλιοθήκες της Python - δεν χρειάζεται να εγκατασταθεί.
- Εναλλακτικά υπάρχουν γραφικές βιβλιοθήκες όπως η **PyQt**, WxPython, PyGtk, kivy, **flet**

μοντέλο προγραμματισμού με γεγονότα





Event based programming / προγραμματισμός με γεγονότα

Το πρόγραμμα είναι ένας βρόχος επανάληψης:

```
σχεδίασε οθόνη
while True:
    περίμενε event από χρήστη
    ανάλογα με το event :
            κάλεσε event_handler
```

δομή ενός προγράμματος tkinter

a. δημιουργεί ένα αρχικό παράθυρο με τη μέθοδο **Tk()**
b. στη συνέχεια δημιουργεί **γραφικά στοιχεία** και τα τοποθετεί στο παράθυρο.
c. Τέλος καλεί τη μέθοδο **.mainloop()** του αρχικού παράθυρου που ξεκινάει το βρόχο επεξεργασίας γεγονότων

Σημείωση: Πολλά γεγονότα τα διαχειρίζεται η ίδια η tkinter (πχ ελαχιστοποίηση παράθυρου κλπ.)

Παραθυρικές εφαρμογές με Python
GUI = graphical user interface

- Με την tkinter δημιουργούμε παράθυρα που περιέχουν **γραφικά στοιχεία (widgets)** όπως labels, buttons, entry texts, menu, scroll bars
- Τα widgets δέχονται γεγονότα (**events**) από ενέργειες του χρήστη και ανταποκρίνονται με κλήση συναρτήσεων: event_handlers

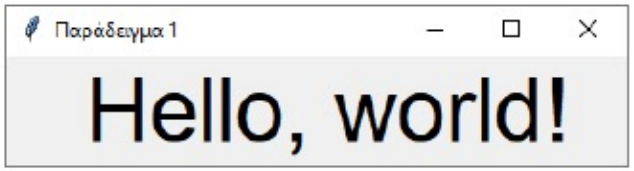
δημιουργία παράθυρου

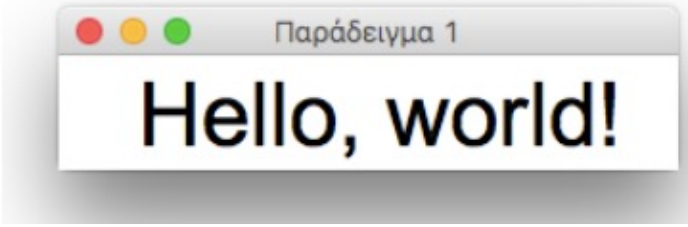
```
import tkinter as tk

w = tk.Tk()

w.title('mywindow')

w.geometry('200x200')

w.resizable(False, False)

w.destroy()
```

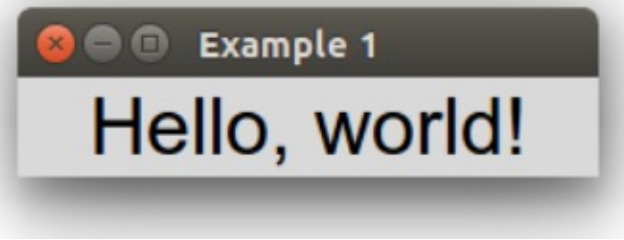
1ο πρόγραμμα: ένα **Label** στο παράθυρο

```
import tkinter as tk
root = tk.Tk() # Δημιουργία παράθυρου root
root.title('Παράδειγμα 1')
# δημιουργία label
w = tk.Label(root, text = "   Hello, world!  ", font = 'Arial 40')
w.pack() # τοποθέτηση w στο παράθυρο
root.mainloop() # έναρξη βρόχου γεγονότων χρήστη
```

Διαφορετική εμφάνιση του παράθυρου ανάλογα με το **λειτουργικό σύστημα**







1ο πρόγραμμα με χρήση κλάσεων

```
import tkinter as tk

class MyApp():
    def __init__(self, root):
        self.w = root
        self.w.geometry('600x200’)
        self.w.title('Παράδειγμα 1’)
        self.l = tk.Label(self.w, text= "Καλή σας μέρα",
                  bg='darkgray', fg='white', font = 'Arial 80’)
        self.l.pack(expand=1, fill='both', padx=10, pady=10)

root = tk.Tk()
MyApp(root)
root.mainloop()
```

κλάση -Frame

```
import tkinter as tk
class MyApp(tk.Frame):
        def __init__(self, root, *args, **kwrds):
                tk.Frame.__init__(self, root, *args, **kwrds)
                self.root = root
                self.l = tk.Label(self, font="arial 40", \
                        text=" καλή σας μέρα ", bg = "yellow")
                self.l.pack(fill="both", expand=True)

if __name__ == "__main__":
        root = tk.Tk()
        root.title("ex0")
        MyApp(root).pack(fill="both", expand=True)
        root.mainloop()
```

```
#https://stackoverflow.com/questions/17466561/best-way-to-
structure-a-tkinter-application
```

μηχανές γεωμετρίας

Οι μηχανές γεωμετρίας είναι συστήματα που επιτρέπουν την τοποθέτηση των γραφικών στοιχείων στους υποδοχείς τους (παράθυρα, frames)

- **pack** Η πιο απλή μέθοδος για την τοποθέτηση widgets σε ένα υποδοχέα. Οργανώνει τα widgets είτε κάθετα είτε οριζόντια, ανάλογα με τις επιλογές που χρησιμοποιούνται.
- **grid**: Αυτή η μέθοδος επιτρέπει να τοποθετούμε widgets σε ένα δισδιάστατο πλέγμα. Ορίζουμε γραμμές και στήλες και καθορίζουμε σε ποιο κελί θα εμφανίζεται κάθε widget.
- **place**: Αυτή η μέθοδος δίνει τον μεγαλύτερο έλεγχο στη θέση των widgets, καθώς επιτρέπει ακριβείς συντεταγμένες.

```
widget.pack(fill=…, expand=…,
side=…, padx=…, pady=…)
```

Η **fill** παίρνει τιμές y, x, both και ορίζει την κατεύθυνση προς την οποία θα επεκταθεί το στοιχείο

Η **expand** παίρνει τιμές True, False και ορίζει αν το στοιχείο θα καταλάβει το διαθέσιμο χώρο

Η **side** παίρνει τιμές left, top, bottom και ορίζει αν το στοιχείο θα είναι στο πλάι ή κάτω από το προηγούμενο στοιχείο

Οι **padx**, **pady** ορίζουν τα περιθώρια σε pixel

pack() - παραδείγματα

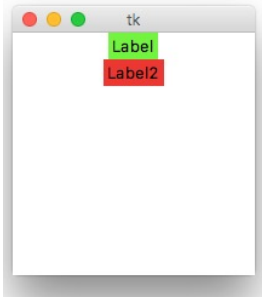
```
import tkinter as tk
class MyApp():
    def __init__(self, root):
        root.geometry('200x200+200+200')
        tk.Label(root, text='Label', bg='green').pack()
        tk.Label(root, text='Label2', bg='red').pack()
        # παράδειγμα 1
        tk.Label(root, text='Label', bg='green').pack(expand=1, fill ='y')
        tk.Label(root, text='Label2', bg='red').pack(fill = 'both')
        # παράδειγμα 2
        tk.Label(root, text='Label', bg='green').pack(expand=1)
        tk.Label(root, text='Label2', bg='red').pack(fill = 'both')
        # παράδειγμα 3
        tk.Label(root, text='Label', bg='green').pack(fill='both', expand=1, side='left')
        tk.Label(root, text='Label2', bg='red').pack(fill='both', expand=1, side='right')
        # παράδειγμα 4
        tk.Label(root, text='Label', bg='green').pack(fill = 'both', expand=1)
        tk.Label(root, text='Label2', bg='red').pack(fill = 'both', expand=1)
root = tk.Tk()
MyApp(root)
root.mainloop()
```

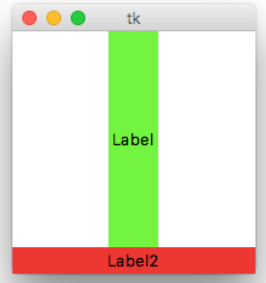
παράδειγμα 1

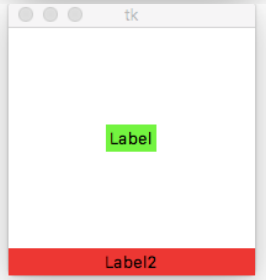
παράδειγμα 2

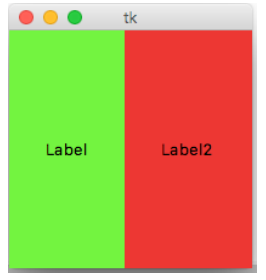
παράδειγμα 3

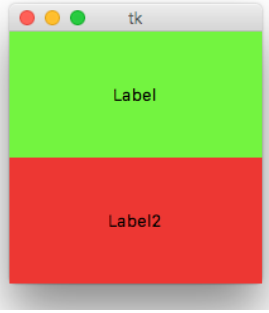
παράδειγμα 4











χρώματα

```
"#ffffff"=white,
"#000000"=black,
"#000fff000" =pure green,
κλπ
ονόματα χρωμάτων: "white", "black", "red",
"green", "blue", "cyan", "yellow", and
"magenta"
```

συνάρτηση που επιστρέφει τυχαίο χρώμα

```
import random

def randomColor():
        r = lambda : random.randint(0,255)
        return f"#{r():02X}{r():02X}{r():02X}"
```

widgets – γραφικά στοιχεία

Widget	Περιγραφή
Label	Ένα στοιχείο που χρησιμοποιείται για την εμφάνιση κειμένου στην οθόνη
Button	Ένα κουμπί που μπορεί να περιέχει κείμενο και εκτελεί μια ενέργεια όταν πατηθεί.
Entry	Ένα στοιχείο εισαγωγής κειμένου που επιτρέπει μόνο μία γραμμή κειμένου
Text	Ένα στοιχείο εισαγωγής κειμένου που επιτρέπει την εισαγωγή κειμένου πολλών γραμμών
Frame	Μια ορθογώνια περιοχή που χρησιμοποιείται για την ομαδοποίηση σχετικών widgets

widgets – γραφικά στοιχεία

Γραφικά στοιχεία: Button, Checkbutton, Entry (εισαγωγή κειμένου), Label (απλό κείμενο), Listbox, Menubutton, Menu, Message (Label πολλών γραμμών), Radiobutton, Scale, Scrollbar, Text (μορφοποιημένο κείμενο πολλών γραμμών), Combobox (επιλογή από τιμές)

Υποδοχείς: Canvas, Frame, LabelFrame (διαχωριστής Frame), Toplevel, PanedWindow

Σύνθετα στοιχεία διαλόγων: tkMessageBox

δημιουργία widget

```
# Δημιουργία widget
widget = <widgetname>(parent, attributes…)

# τοποθέτηση του widget στο παράθυρο
widget.pack( παράμετροι )
```

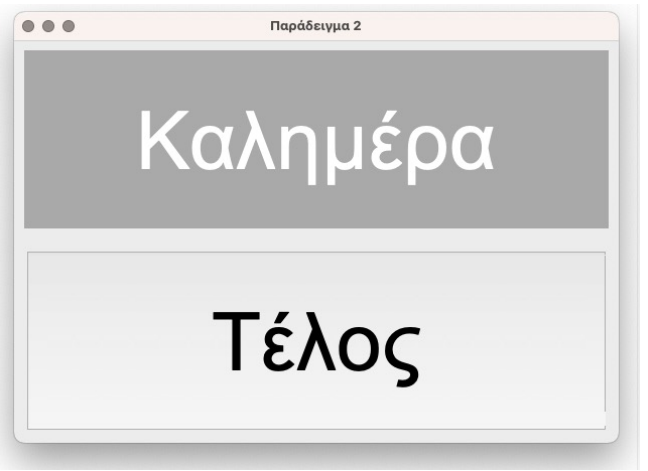
τροποποίηση χαρακτηριστικών γνωρισμάτων widget

μπορούμε να τροποποιήσουμε τα γνωρίσματα ενός widget με τη μέθοδο configure()

παράδειγμα

```
self.label.configure(text =
     self.entry.get())
```

2ο πρόγραμμα: εισαγωγή Button

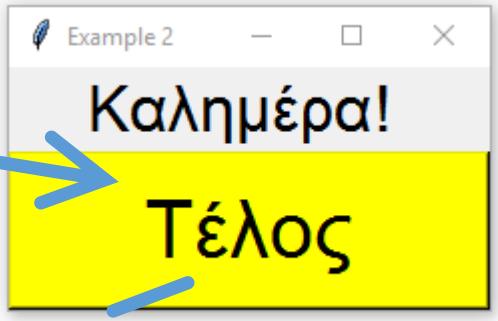



```
import tkinter as tk

class MyApp():
    def __init__(self, root):
        self.w = root
        self.w.geometry('600x400’)
        self.w.title('Παράδειγμα 2’)
        self.font = 'Arial 80’
        self.l = tk.Label(self.w, text= "Καλημέρα",
                          bg='darkgray', fg='white', font = self.font)
        self.l.pack(expand=1, fill='both', padx=10, pady=10)
        self.b = tk.Button(self.w, text='Τέλος', font = self.font,
                           command=self.button_pushed)
        self.b.pack(expand=1, fill='both', padx=10, pady=10)
    def button_pushed(self):
        self.w.destroy()

root = tk.Tk()
MyApp(root)
root.mainloop()
```

2ο πρόγραμμα : κλήση handler

Αν επιθυμούμε το **γεγονός** επιλογής του πλήκτρου myButton να καλεί τη συνάρτηση ButtonPushed θα πρέπει να τροποποιήσουμε τη δήλωση του:

```
myButton = Button(root, text= "Τέλος",
command=buttonPushed)
```



Και να δηλώσουμε τη συνάρτηση buttonPushed()

Event handler

Πώς θα ορίσουμε την ButtonPushed ώστε να καταργεί το παράθυρο;

```
def buttonPushed(self):
    self.p.destroy() # Kill the root window!
```

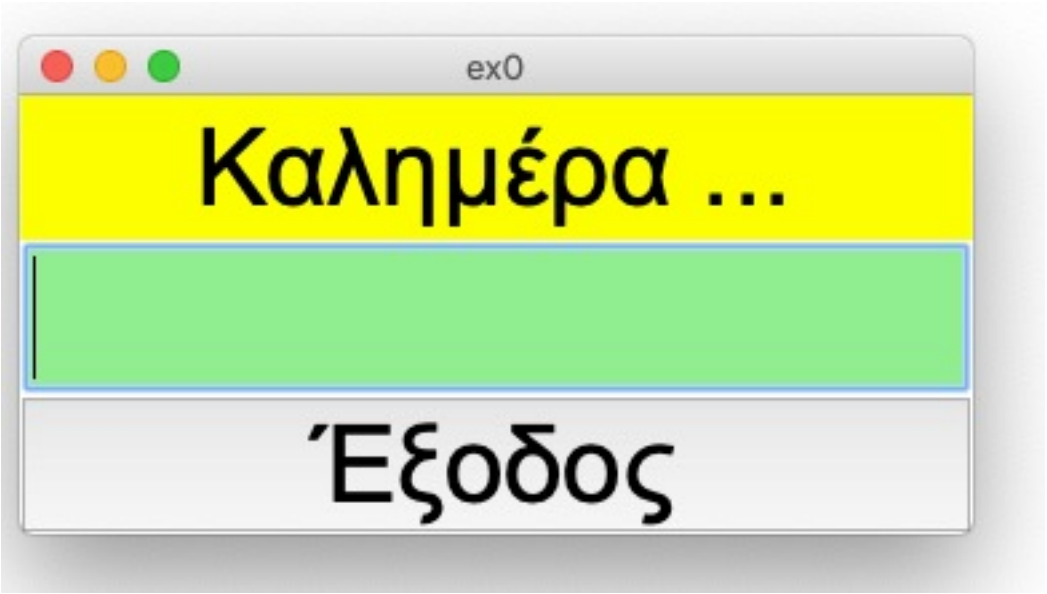
Καλείται η μέθοδος destroy() στο αντικείμενο παράθυρο.
Προσοχή: το παράθυρο είναι γνωστό στη μέθοδο μέσω της class variable self.p

Πώς θα προσθέσουμε ένα γραφικό στοιχείο εισαγωγής κειμένου **Entry**

Label

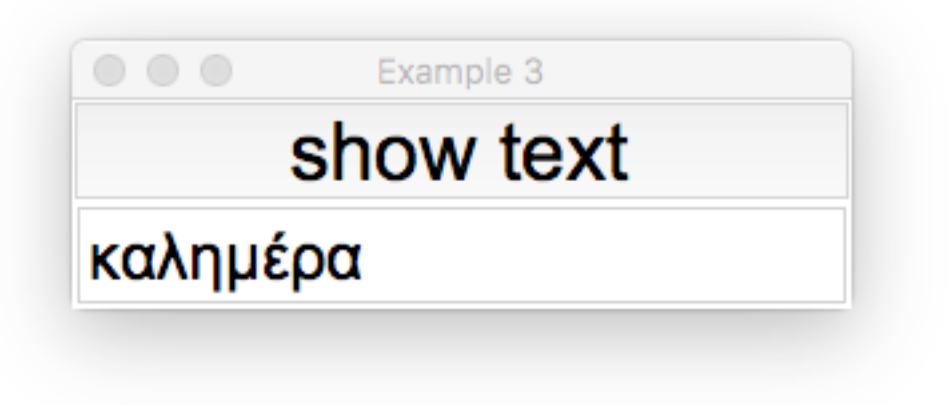
Entry

Button




```
tbox=Entry(root, παράμετροι…)
tbox.pack(expand='both', fill=True)
```

get() συνάρτηση για να πάρουμε το περιεχόμενο ενός Entry widget

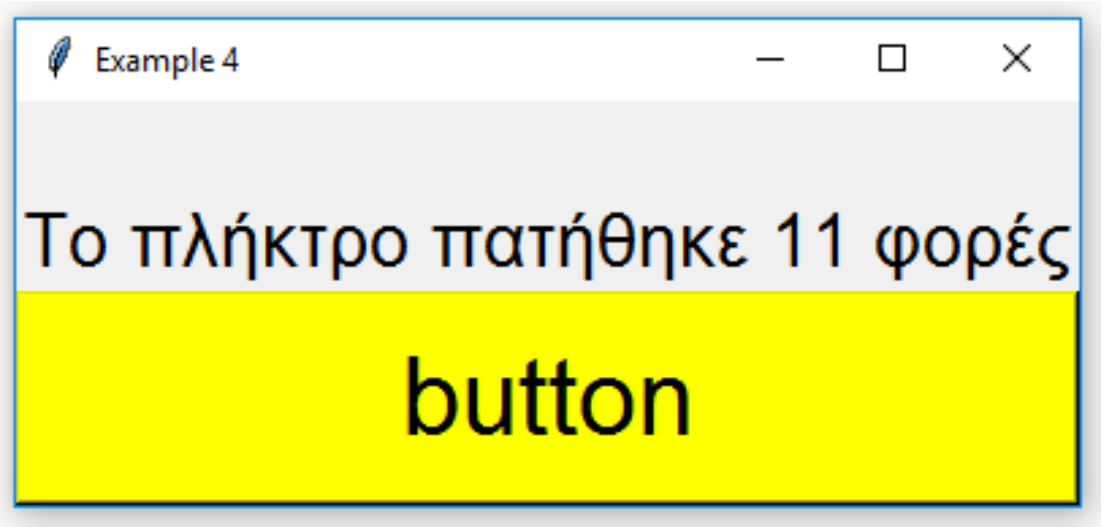


Στη συνάρτηση handler του Button θα πρέπει να ορίσουμε μεταβλητή η οποία θα παίρνει την τιμή του Entry με τη μέθοδο get και θα την τυπώνει π.χ.

```
mytext = tbox.get()
print (mytext)
```

Μεταβολή κειμένου Label

Να δημιουργήσετε ένα παράθυρο με πλήκτρο (Button) και κείμενο (Label) το οποίο όταν πατηθεί το πλήκτρο να εμφανίζει το μήνυμα : «Το πλήκτρο πατήθηκε x φορές»



Μεταβολή κειμένου Label (3 τρόποι)

(α)

```
myText = tkinter.StringVar()
myLabel=tkinter.Label(root, textvariable=myText)
myText.set('xyz')
```

(β)

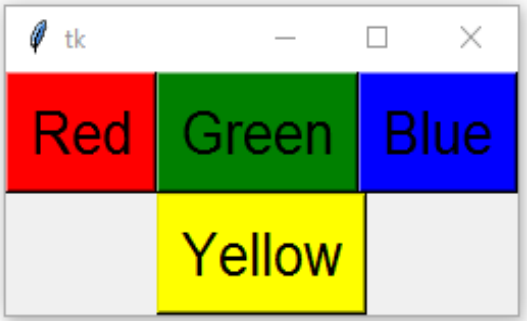
```
myLabel=tkinter.Label(root, text='xyz')
myLabel.configure(text= 'abc')
```

(γ)

```
myLabel=tkinter.Label(root, text='xyz')
myLabel['text']= 'abc'
```

Ιεραρχική διάταξη widgets: χρήση **Frames**

```
import tkinter
class App():
    def __init__(self, root):
        self.r = root
        self.f1 = tkinter.Frame(root)
        self.f1.pack()
        self.f2 = tkinter.Frame(root)
        self.f2.pack( side = "bottom" )
        redb = tkinter.Button(self.f1, text="Red", font="Arial 20", bg="red")
        redb.pack( side = "left")
        greenb = tkinter.Button(self.f1, text = "Green", font="Arial 20", bg="green")
        greenb.pack( side = "left" )
        blueb = tkinter.Button(self.f1, text="Blue", font="Arial 20", bg="blue")
        blueb.pack( side = "left" )
        yellowb = tkinter.Button(self.f2, text="Yellow", font="Arial 20", bg="yellow")
        yellowb.pack( side = "bottom")

def main():
    root = tkinter.Tk()
    app = App(root)
    root.mainloop()
main()
```



νέο παράθυρο

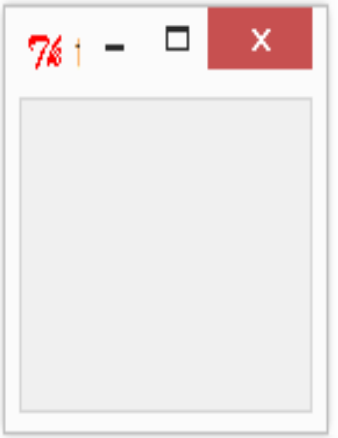
Μπορούμε να δημιουργήσουμε νέο παράθυρο με κλήση της μεθόδου

```
newWindow = tk.Toplevel()
```

μπορούμε όπως και το αρχικό παράθυρο να ορίσουμε γεωμετρία, τίτλο και να ορίσουμε αν επιτρέπεται να αλλάζει το μέγεθος

```
newWindow.geometry('{}χ{}+{}+{}'.format(width, height, x, y))
newWindow.title('newTitle')
newWindow.resizable(False, False)
```

bind (event, συνάρτηση)
event.x, event.y --> x,y



```
import tkinter

def callback(event):
    print ("clicked at", event.x, event.y)

root = tkinter.Tk()
frame = tkinter.Frame(root, width=300, height=300,
                      borderwidth=2, bg='lightyellow')
frame.bind("<Button-1>", callback)
frame.pack()
root.mainloop()
```

clicked at 32 47

Άσκηση: περιγράψτε τον κώδικα

```
import tkinter

def mouseEntered(event):
    button = event.widget
    button.config(text = "click!")

def mouseExited(event):
    button = event.widget
    button.config(text = "Logon")


def main():
    global root
    root = tkinter.Tk() # Create the root window
    b = tkinter.Button(root, text="Logon",
                       font = "Arial 30")
    b.bind("<Enter>",mouseEntered)
    b.bind("<Leave>",mouseExited)
    b.pack()
    root.mainloop() # Start the event loop
main()
```

Γεγονότα από το ποντίκι ή από το πληκτρολόγιο

Χρειάζεται η αντιστοίχιση γεγονότων χρήστη με widget

widget.bind(event, handler)

Κατηγορίες γεγονότων:

<Button-1> (αριστερό πλήκτρο ποντικιού)

<Double-Button-1> (διπλό κλικ)

<B1-Motion> η κίνηση με πατημένο αριστερό πλήκτρο

<Return> - πάτημα πλήκτρου “**enter”**

<Enter> - το ποντίκι εισέρχεται στο widget

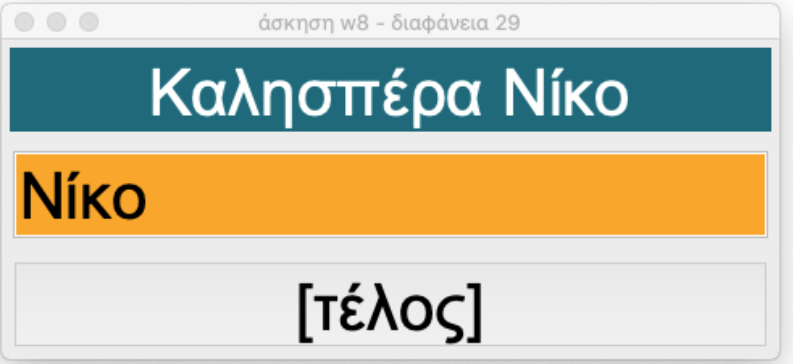
<Leave> - το ποντίκι αφήνει το widget

<Key> - event.char == “a” για πάτημα πλήκτρου “a”

άσκηση

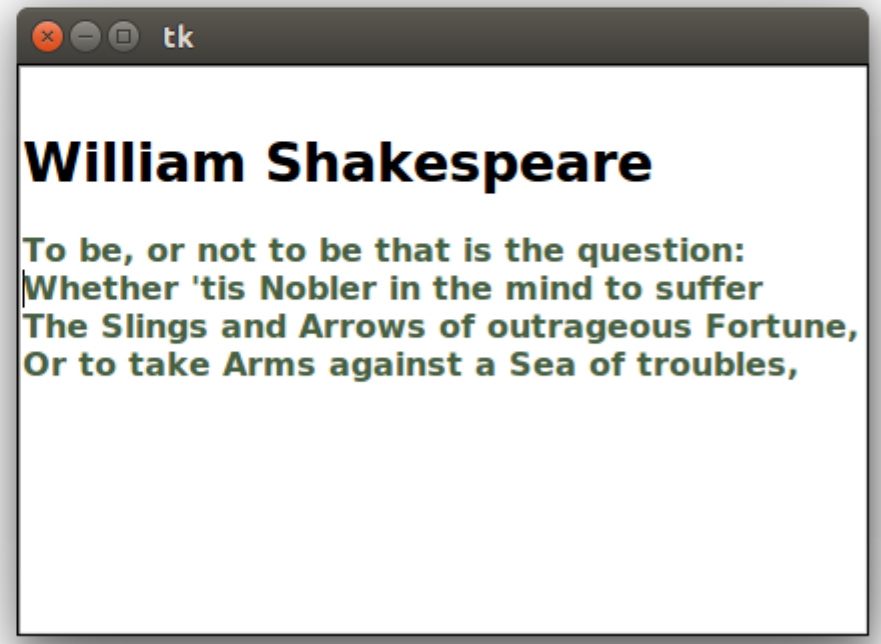
Να γράψετε πρόγραμμα που περιέχει ένα Label, ένα Entry και ένα Button
το Label αρχικά περιέχει το κείμενο Καλημέρα… ή Καλησπέρα… ανάλογα με την ώρα της μέρας
Στο Entry γράφουμε ένα όνομα πχ Μαρία
Όταν πατήσουμε <Return> τότε το μήνυμα στο Label γίνεται Καλημέρα Μαρία
Το Button τερματίζει το πρόγραμμα

Σημειωση: διαλέγουμε χρώματα από χρωματικές παλέττες, πχ από την colorhunt.com

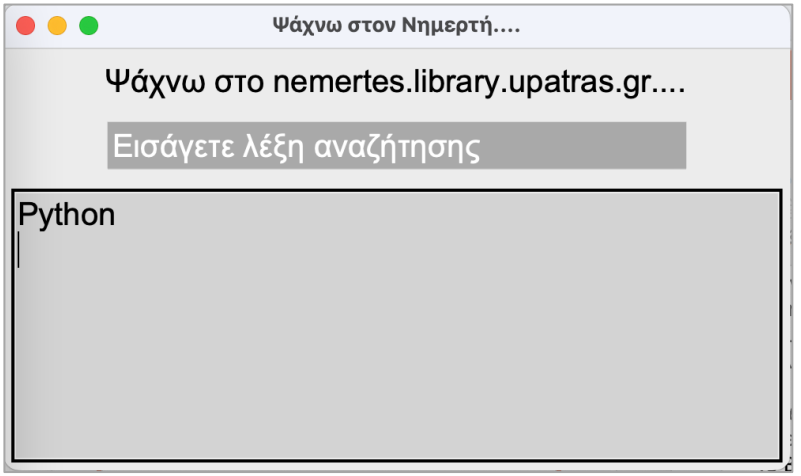


Text : υποδοχέας μορφοποιημένου κειμένου πολλών γραμμών

```
import tkinter

root = tkinter.Tk()
text1 = tkinter.Text(root, height=20, width=60)
text1.tag_configure('big_text', font=('Verdana', 20, 'bold'))
text1.tag_configure('color_text', foreground='#476042',
                    font=('Tempus Sans ITC', 12, 'bold'))
text1.insert('end','\nWilliam Shakespeare\n', 'big_text')
quote = """
To be, or not to be that is the question:
Whether 'tis Nobler in the mind to suffer
The Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
"""
text1.insert('end', quote, 'color_text')
text1.pack(side='left')

root.mainloop()
```



άσκηση



Να κατασκευάσετε μια μηχανή αναζήτησης στο αποθετήριο του Πανεπιστημίου Πατρών `https://nemertes.library.upatras.gr`
Να χρησιμοποιήσετε ένα γραφικό αντικείμενο τύπου Text για να αποθηκεύετε τις αναζητήσεις σας
Να δείχνετε τα αποτελέσματα στον φυλλομετρητή (χρησιμοποιήστε τη βιβλιοθήκη webbrowser)
Να υλοποιήσετε την υπόδειξη μέσα στο γραφικό αντικείμενο Entry

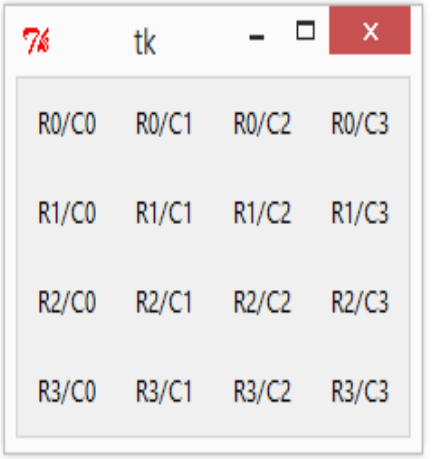
Εναλλακτικός τρόπος διάταξης widget : ο διαχειριστής γεωμετρίας **grid**

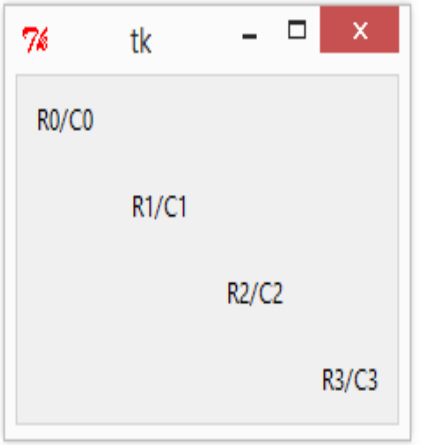
Αντί για την μέθοδο pack μπορεί να χρησιμοποιηθεί η grid
w.grid (row = 0, column = 0, rowspan=3)
Αυτή ορίζει το χώρο του παράθυρου ή του Frame με λογική στηλών και γραμμών όπως ένας πίνακας
Δεν επιτρέπεται να χρησιμοποιήσουμε τις μεθόδους pack και grid μαζί

Άσκηση: διάταξη widgets με τον διαχειριστή γεωμετρίας grid

Να διαταχθούν 16 label σε πίνακα όπως στο σχήμα, εναλλακτικά μόνο τα διαγώνια στοιχεία

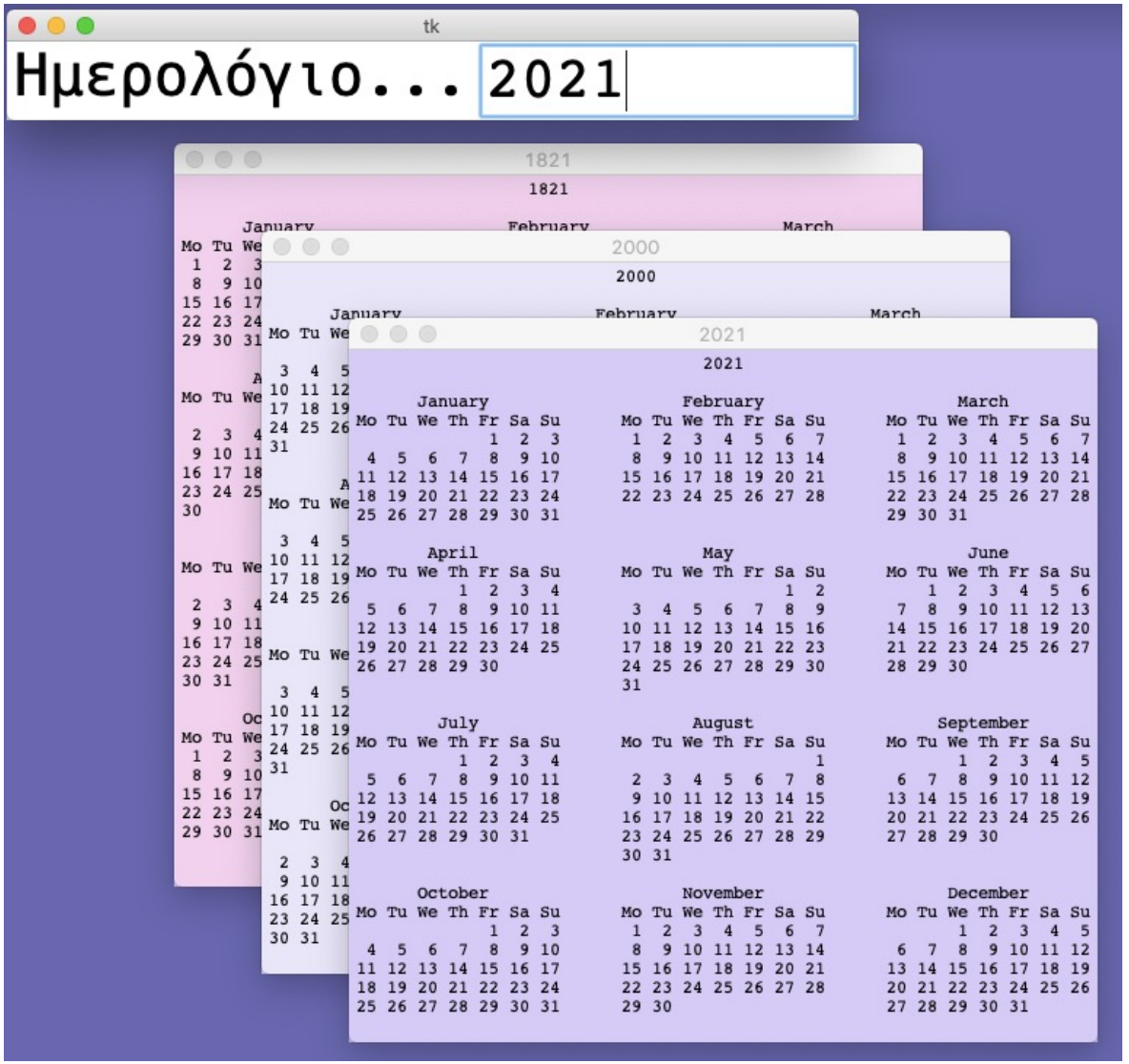
```
from tkinter import *
root = Tk(  )
for r in range(4):
    for c in range(4):
        lab=Label(root, text='R%s/C%s'%(r,c),
            borderwidth=10 )
        lab.grid(row=r, column=c)
root.mainloop()
```





Άσκηση

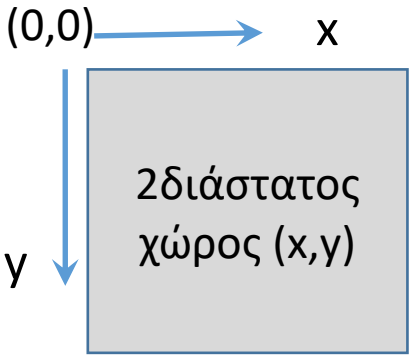
Να κατασκευάσετε μια εφαρμογή εκτύπωσης ημερολογίων:
Το αρχικό παράθυρο θα περιέχει ένα Label και ένα Entry, όπως στο σχήμα. Όταν ο χρήστης δώσει έναν θετικό ακέραιο, τυπώνεται το ημερολόγιο της αντίστοιχης χρονιάς σε ξεχωριστό παράθυρο.



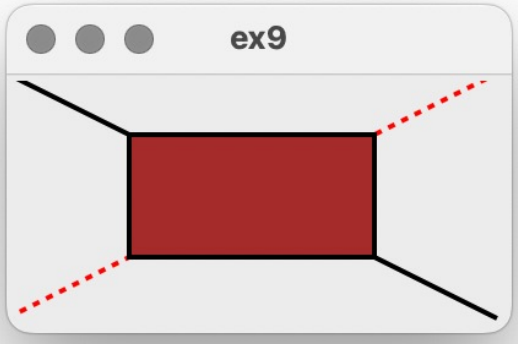
Άσκηση

Να γράψετε εφαρμογή που χαιρετάει (καλημέρα, καλησπέρα, καληνύχτα ανάλογα με την ώρα της μέρας (χρήση της datetime.datetime.now())
Παραλλαγή: να γεμίζει την οθόνη με χρωματιστή ευχή σε τυχαία χρώματα.

Canvas : υποδοχέας γραφικών αντικειμένων, εικόνων κλπ.

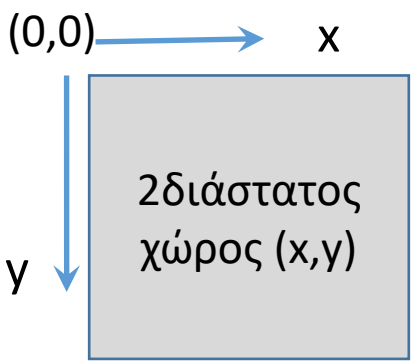



```
import tkinter as tk

class MyApp():
    def __init__(self, root):
        self.root = root
        self.root.title('ex9’)
        self.c = tk.Canvas(self.root, width=200, height=100)
        self.c.pack()
        self.c.create_line(0,0, 200,100, width=2)
        self.c.create_line(0,100, 200, 0, fill='red', width=2,
                           dash=(3,3))
        self.c.create_rectangle(50,25, 150, 75, width=2, fill='brown')

root = tk.Tk()
MyApp(root)
root.mainloop()
```



Canvas : είδη γραφικών αντικειμένων

- arc (arc, chord, or pieslice)
- bitmap (από αρχείο XBM)
- image (εικόνα BitmapImage ή PhotoImage)
- line
- oval (κύκλος ή έλλειψη)
- polygon
- rectangle
- text
- window



https://dafarry.github.io/tkinterbook/

mydraw


```
import tkinter as tk
class MyApp():
    def __init__(self, root):
        self.root = root
        self.root.geometry('800x600')
        self.create_canvas()
    def create_canvas(self):
        self.canvas = tk.Canvas(self.root)
        self.canvas.pack(fill='both', expand=1)
        self.canvas.bind('<Button-1>', self.start_draw)
        self.canvas.bind('<B1-Motion>', self.draw_line)
    def start_draw(self, event):
        self.lastx,self.lasty = event.x, event.y
    def draw_line(self,event):
        self.canvas.create_line((self.lastx, self.lasty, event.x, event.y), width=2)
        self.lastx, self.lasty = event.x, event.y
w = tk.Tk()
MyApp(w)
w.mainloop()
```

τροποποίηση αντικειμένων καμβά

- Μπορούμε να αλλάξουμε τις ιδιότητες ενός αντικειμένου με κλήση της μεθόδου c.itemconfig(αντικείμενο, ιδιότητα=τιμη)
- Μπορούμε να μετακινήσουμε ένα αντικείμενο με κλήση της μεθόδου c.**move**(αντικείμενο, συνεταγμένες) ή της c.**coords**()

```
# παράδειγμα: αλλαγή χρώματος και μετακίνηση
ορθογωνίου στον καμβά
w = tk.Tk()
c = tk.Canvas(w)
c.pack(expand=1, fill='both')
r = [50,50,150,150]
l = c.create_rectangle(*r, fill='red')
c.itemconfig(l, fill='blue') #αλλαγή χρώματος
c.move(l, 50,50) # μετακίνηση
r = [100,100,200,200]
c.coords(l,*r) # μετακίνηση
w.mainloop()
```

κίνηση αντικειμένων

Ο συνδυασμός της μεθόδου **move(item, dx, dy)** που κινεί το item κατά dx, dy με την μέθοδο **after(t, συνάρτηση)** που καλεί τη συνάρτηση που περιέχει τη move αναδρομικά μετά από t msec επιτρέπουν την κίνηση του αντικειμένου item

Παράδειγμα: κίνηση αυτοκινήτου σε ορεινή διαδρομή

προχωρημένα κεφάλαια tkinter

Χρήσιμα modules της tkinter

tkinter.messagebox κλάση για πλαίσια μηνυμάτων
tkinter.simpledialog ορίζει την askinteger, askfloat and askstring για παράθυρα επιλογής τιμών.
tkinter.scrolledtext επιτρέπει την κύλιση παράθυρου
tkinter.colorchooser ορίζει την askcolor(initialcolor), για το παράθυρο επιλογής χρώματος
tkinter.filedialog παράθυρο επιλογής αρχείου
tkinter.font κλάση για τους μορφότυπους χαρακτήρων
tkinter.ttk themed tk (περισσότερα widget, χρήση στυλ)
PIL για διαχείριση εικόνων τύπου jpg, png

ttk (themed tk)

```
import tkinter
from tkinter import ttk
s = tkinter.ttk.Style()
s.configure(“my.Tlabel”, font = (“Arial”, 20),
background = “lightyellow”)

L1 = tkinter.ttk.Label(f1, text = “Hello world”,
style=“my.TLabel”)
```

Από την έκδοση 8.5 περιλαμβάνει themes, styles, εμφάνιση στα widgets όπως αυτή του λειτουργικού, επίσης πρόσθετα στοιχεία όπως το combobox

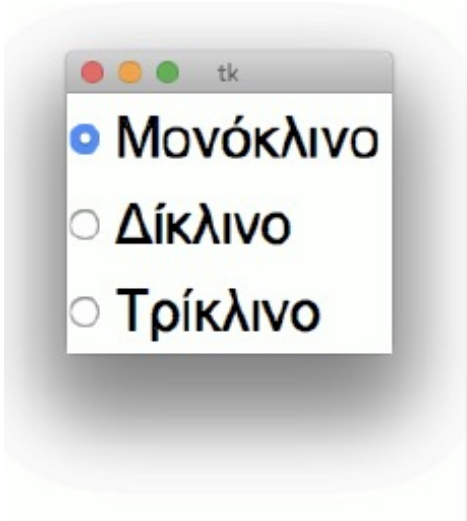
ttk widgets

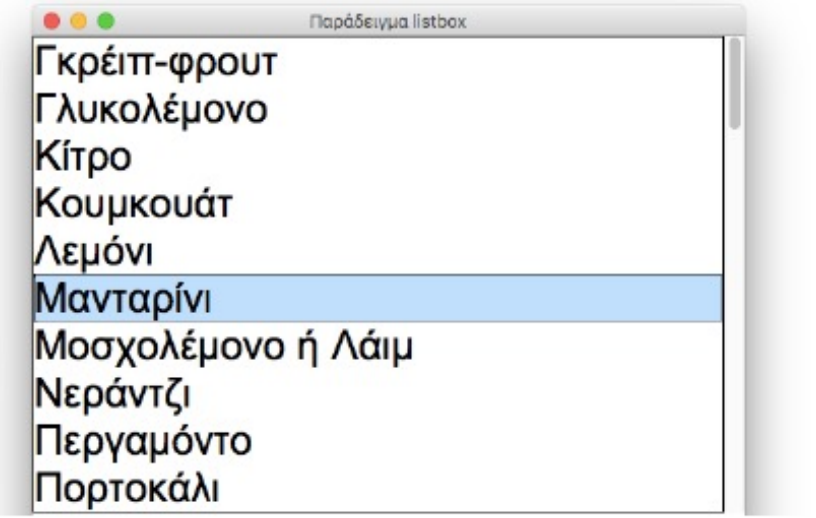
η **ttk** περιέχει 17 widgets, τα 11 υπήρχαν ήδη στην tkinter: Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale and Scrollbar.

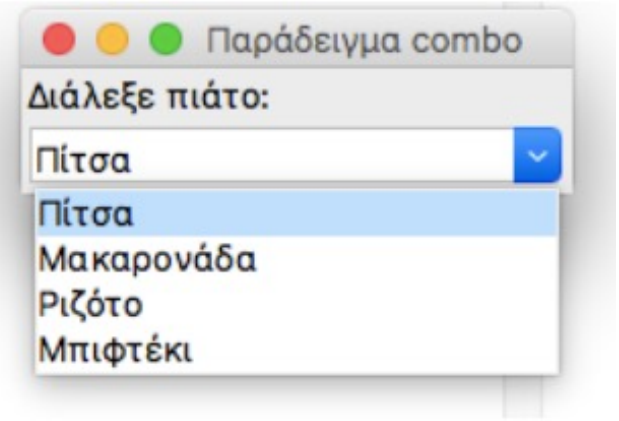
Υπάρχουν 6 νέα widgets
Combobox, Notebook, Progressbar, Separator, Sizegrip and Treeview. Όλα είναι υπόκλάσεις του Widget.

Εικονικά στοιχεία επιλογής

(α) Radiobutton (β) Listbox (γ) Combobox

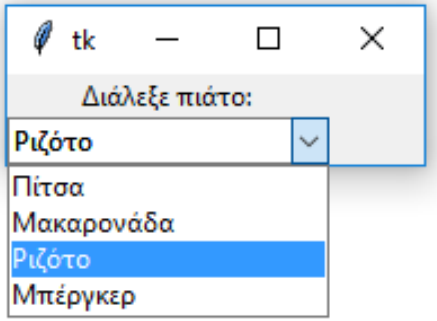






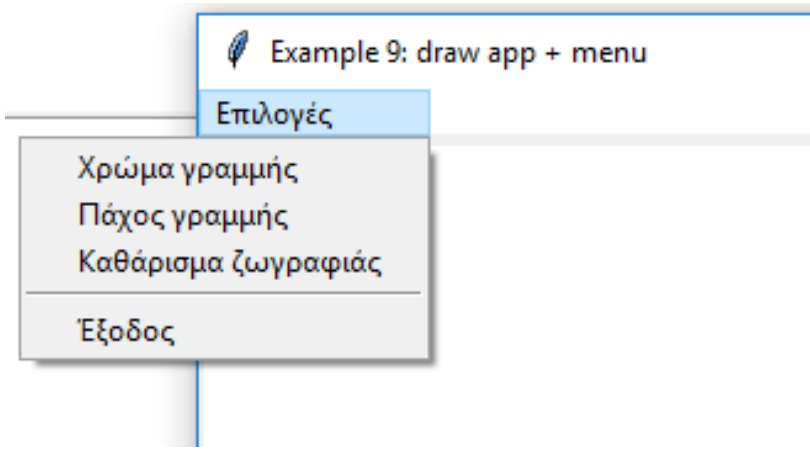
Combobox

```
import tkinter
from tkinter import ttk
## example of simple combobox
class App:
    def __init__(self, parent):
        self.parent = parent
        self.label = tkinter.ttk.Label(self.parent, text = "Διάλεξε πιάτο:")
        self.label.grid(column=0, row=0)
        self.combo()

    def combo(self):
        self.box_value = tkinter.StringVar()
        self.box = tkinter.ttk.Combobox(self.parent, textvariable=self.box_value)
        self.box.bind("<<ComboboxSelected>>", self.newselection)
        self.box['values'] = ('Πίτσα', 'Μακαρονάδα', 'Ριζότο', 'Μπέργκερ')
        self.box.current(0)
        self.box.grid(column=0, row=1)

    def newselection(self, event):
        self.value_of_combo = self.box.get()
        print(self.value_of_combo)

if __name__ == '__main__':
    root = tkinter.Tk()
    app = App(root)
    root.mainloop()
```



Δημιουργία μενού

Ένα μενού επιλογών είναι widget με ιεραρχική δομή (υπο-μενού κλπ)

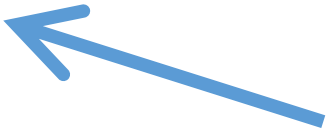



```
class MyApp ():
    def __init__(self, parent):
        self.f1 = tkinter.Frame(parent).pack()
        self.parent = parent
        self.menubar = tkinter.Menu(self.parent)
        self.selection = tkinter.Menu(self.menubar, tearoff=0)
        self.menubar.add_cascade(  label="Επιλογές           ", menu=self.selection)
        self.selection.add_command(label="Χρώμα γραμμής", command=self.color_select)
        self.selection.add_command(label="Πάχος γραμμής", command=self.line_select)
        self.selection.add_command(label="Καθάρισμα ζωγραφιάς", command=self.clear_all)
        self.selection.add_separator()
        self.selection.add_command(label="Έξοδος", command=self.parent.destroy)
        self.parent.config(menu = self.menubar)
        self.create_canvas()
        self.line_color = 'black' # default line drawing color
        self.line_width = 2
```

Εικόνες

Εικόνες μόνο τύπου **gif** στο tkinter
Για εικόνες τύπου jpg, png απαιτείται η χρήση της βιβλιοθήκης PIL

```
image1 = tkinter.PhotoImage(file= "myimage.gif")
l= tkinter.Label(root, image=image1)
l.pack(side='top', fill='both', expand='yes')
```



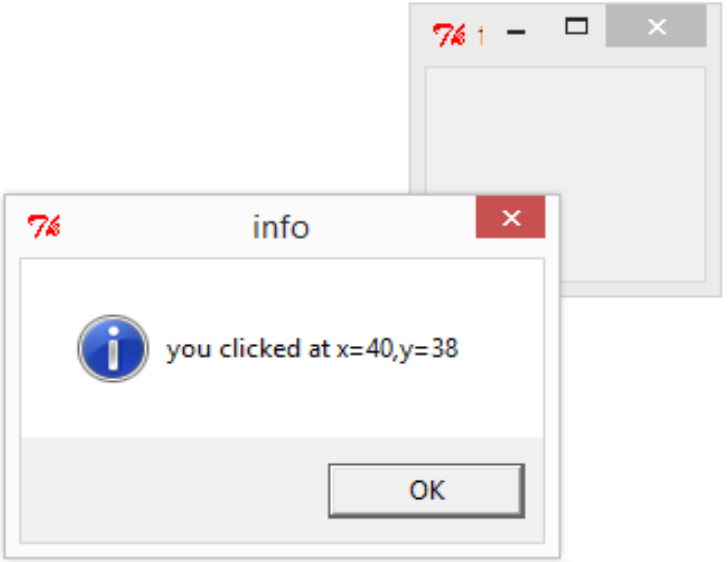
Η εικόνα εισάγεται σε widget τύπου label. Μπορεί επίσης να εισαχθεί σε canvas

Dialog box

Ένα παράθυρο διαλόγου dialog box είναι ένα **μονοτροπικό (modal) παράθυρο** που θέτει ένα ερώτημα στο χρήστη, δείχνει ένα μήνυμα σφάλματος ζητάει μια πληροφορία κλπ.

Πολλά τέτοια παράθυρα διαλόγου, όπως το παράθυρο επιλογής αρχείου ή επιλογής χρώματος, κλπ προσφέρονται από το γραφικό περιβάλλον και για αυτό είναι ίδια σε πολλές εφαρμογές.

Example: showinfo




```
import tkinter
from tkinter import messagebox

def callback(event):
    messagebox.showinfo("info",
            "πάτησες στη θέση: x=%d,y=%d" % (event.x, event.y))

root = tkinter.Tk()
frame = tkinter.Frame(root, width=500, height=500)
frame.bind("<Button-1>", callback)
frame.pack()
root.mainloop()
```

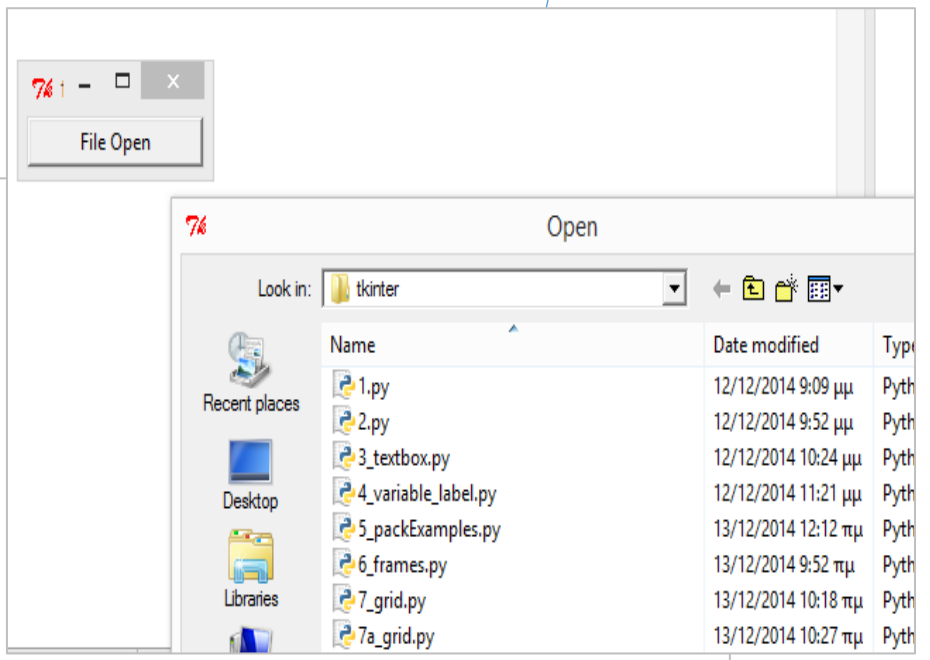
Άλλα παράθυρα διαλόγου: askyesno, askokcancel, askretrycancel, askquestion

from tkinter import messagebox
ans = askyesno("Continue", "Should I continue?”)

Η μεταβλητή ans θα είναι είτε True (Yes) ή False (No).
Άλλα παράθυρα διαλόγου: askokcancel,
askretrycancel,
askquestion
Προσοχή: askquestion επιστρέφει “yes” “no” και όχι True / False!

Άνοιγμα αρχείου: askopenfilename

C:/Users/user/PycharmProjects/eclass/tkinter/1.py



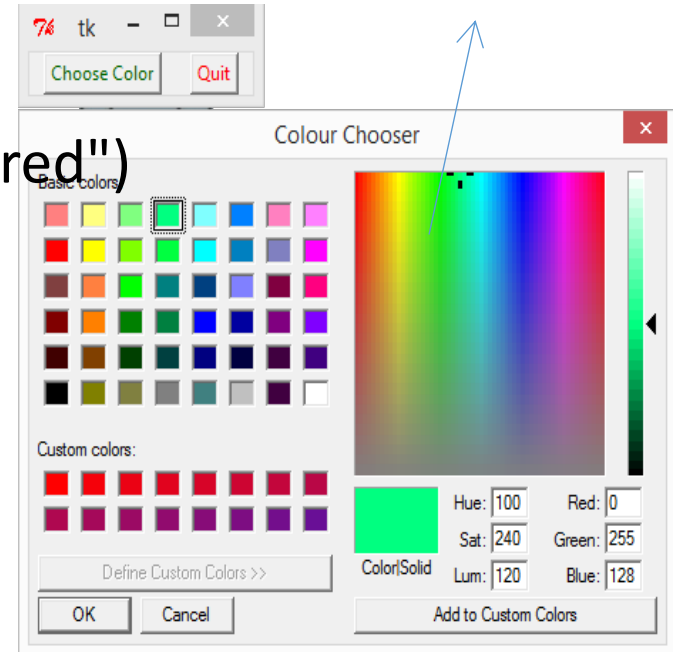

```
import tkinter as tk
from tkinter import filedialog

def callback():
    name= filedialog.askopenfilename()
    print (name)

root = tk.Tk()
b = tk.Button(root, text='File Open', command=callback)
b.pack(fill="x")
mainloop()
```

http://www.python-course.eu/tkinter_dialogs.php

Επιλογή χρώματος : askcolor

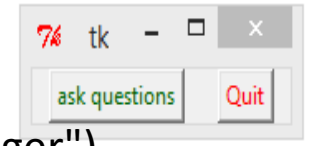
```
from tkinter import *
from tkinter import colorchooser
def callback():
    result = colorchooser.askcolor(color="#6A9662",  title = "Colour
Chooser")
    print (result)

root = Tk()
b1=Button(root, text='Choose Color', fg="darkgreen",
      command=callback)
b1.pack(side=LEFT, padx=10)
b2=Button(text='Quit',  command=root.quit, fg="red")
b2.pack(side=LEFT, padx=10)
mainloop()
```

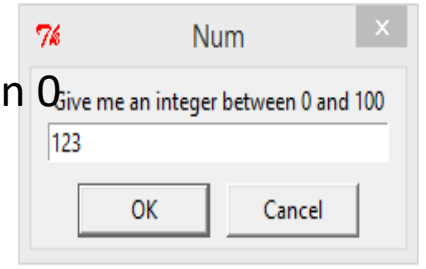
((0, 255, 128), '#00ff80'

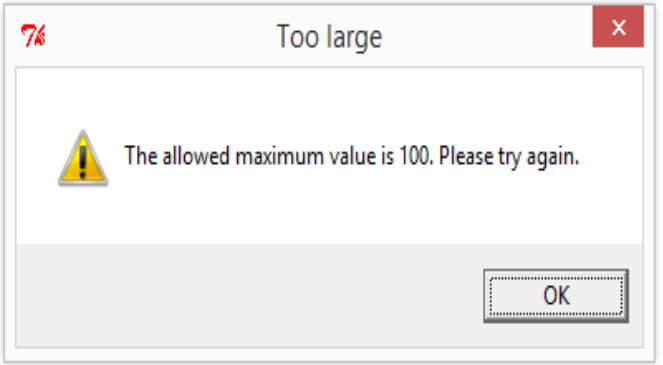


Διάλογος εισαγωγής τιμής από το χρήστη: askstring, askinteger

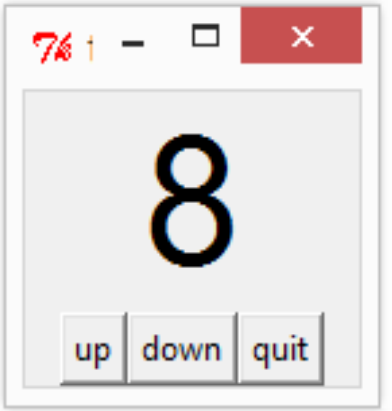
```
from tkinter import *
from tkinter import simpledialog
def callback():
    ans = simpledialog.askstring("Title", "Give me your name")
    print (ans)
    ans = simpledialog.askinteger("Dialog Title", "Give me an integer")
    print (ans)
    ans = simpledialog.askinteger("Num", "Give me an integer between 0
and 100",
    minvalue=0, maxvalue=100)
    print (ans)

root = Tk()
b1=Button(root, text='ask questions',fg="darkgreen",
command=callback)
b1.pack(side=LEFT, padx=10)
b2=Button(text='Quit', command=root.quit,fg="red")
b2.pack(side=LEFT, padx=10)
mainloop()
```







Άσκηση



Να κατασκευάσετε πρόγραμμα Python που δημιουργεί παράθυρο, όπως στο σχήμα, το οποίο εμφανίζει έναν αριθμό (αρχική τιμή 1) και 3 πλήκτρα: up, down, quit
Όταν πατηθεί το **up** ο αριθμός αυξάνει κατά 1, **down** μειώνεται κατά 1 (ως την κατώτερη τιμή 1) και η επιλογή **quit** κλείνει η εφαρμογή.

Άσκηση

Να κατασκευάσετε πρόγραμμα Python που υλοποιεί την εφαρμογή μετατροπής θερμοκρασιών από Κελσίου σε Φαρενάιτ, Κέλβιν και αντίθετα με γραφικό τρόπο (να χρησιμοποιήσετε **Combobox** για επιλογή κλίμακας, **Entry** για εισαγωγή τιμής και **Label** για εμφάνιση αποτελέσματος

- F = C*(9/5) + 32
- C = (F-32)*(5/9)
- K = (F+459.67)*(5/9),
- F = 300*K*(9/5) – 459.67

Άσκηση

Να κατασκευάσετε μια αριθμομηχανή 4 πράξεων, με μια θέση μνήμης.
Η αριθμομηχανή θα πρέπει να έχει λειτουργίες όπως στο σχήμα: **+m** (αποθήκευση στη μνήμη), **mr** (ανάκτηση από τη μνήμη), **mc** (καθαρισμός μνήμης), **c** (καθαρισμός οθόνης).

πηγές

https://docs.python.org/3/library/tk.html

https://realpython.com/python-gui-tkinter/
https://dafarry.github.io/tkinterbook/

http://www.tkdocs.com/index.html

διαλέξεις στο mathesis.cup.gr "Προχωρημένος προγραμματισμός με Python (Ε)" εβδομάδες 4,5,6