17. ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΜΕ TTKBOOTSTRAP

17.0.0 Λύσεις προηγούμενων ασκήσεων

1.

'''

Δημιουργήστε ένα γραφικό περιβάλλον με το tkinter,

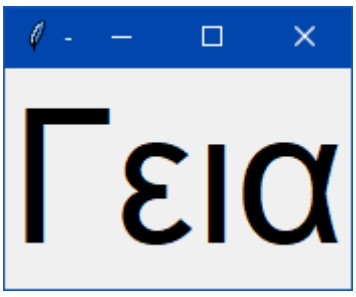
το οποίο να περιέχει ένα label και να αλλάξετε τη

γραμματοσειρά σε Arial Bold, καθώς και το μέγεθός

της σε 70 και να την κάνετε με έντονη γραφή.

'''

```
import tkinter as tk
parent = tk.Tk()
parent.title("-Αλλαγή label στο tkinter")
my_label = tk.Label(parent, text="Γεια", font=("Arial Bold", 70))
my_label.grid(column=0, row=0)
parent.mainloop()
```



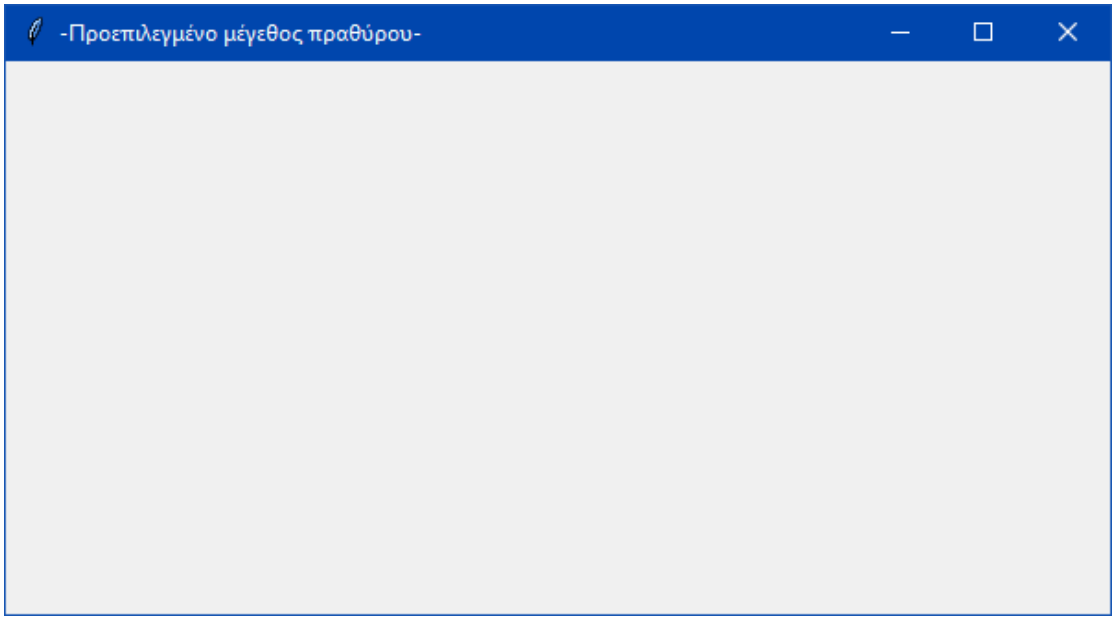
2.

'''

Δημιουργήστε ένα γραφικό περιβάλλον με ένα παράθυρο
στο οποίο να θέσετε το προεπιλεγμένο του μέγεθος
σε 600x300 κα τον τίτλο του σε «-Προεπιλεγμένο μέγεθος
πραθύρου-« με το tkinter.

'''

```
import tkinter as tk
parent = tk.Tk()
parent.title("-Προεπιλεγμένο μέγεθος πραθύρου-")
parent.geometry('600x300')
parent.mainloop()
```



3.

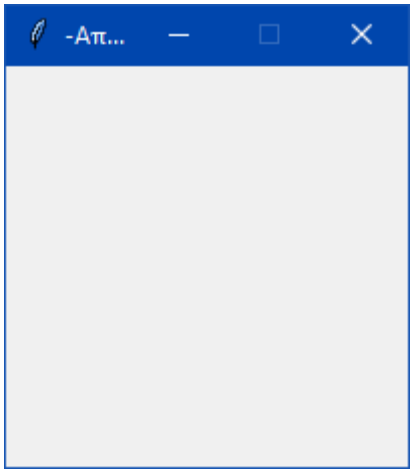
'''

Φτιάξτε ένα απλό παράθυρο στο tkinter στο οποίο

να απενεργοποιείτε το resizing του παραθύρου.

'''

```
import tkinter as tk
parent = tk.Tk()
parent.title("-Απενεργοποίηση resizing-")
# Απενεργοποίηση resizing
parent.resizable(0,0)
parent.mainloop()
```



17.2.0 Ολοκλήρωση Tkinter widgets

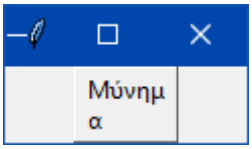
17.2.1 Message - Μήνυμα

Το widget Message χρησιμοποιείται για την εμφάνιση πεδίων κειμένου πολλών γραμμών για αποδοχή τιμών από έναν χρήστη.

Ας δούμε το παράδειγμα:

```
from tkinter import *

root = Tk()

var = StringVar()
label = Message( root, textvariable = var, relief = RAISED )

var.set("Μύνημα")
label.pack()
root.mainloop()
```

Το παράθυρο είναι:



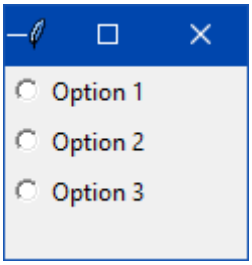
17.2.2 Radiobutton - Κουμπί ραδιοφώνου

Το γραφικό στοιχείο Radiobutton χρησιμοποιείται για την εμφάνιση ενός αριθμού επιλογών ως κουμπιά επιλογής. Ο χρήστης μπορεί να επιλέξει μόνο μία επιλογή κάθε φορά.

Παράδειγμα:

```
from tkinter import *

def sel():
   selection = "You selected the option " + str(var.get())
   label.config(text = selection)

root = Tk()
var = IntVar()
R1 = Radiobutton(root, text = "Option 1", variable = var, value = 1,
                 command = sel)
R1.pack( anchor = W )

R2 = Radiobutton(root, text = "Option 2", variable = var, value = 2,
                 command = sel)
R2.pack( anchor = W )

R3 = Radiobutton(root, text = "Option 3", variable = var, value = 3,
                 command = sel)
R3.pack( anchor = W)
```

```
label = Label(root)
label.pack()
root.mainloop()
```



17.2.3 Scale - Κλίμακα

Το γραφικό στοιχείο Scale χρησιμοποιείται για την παροχή ενός γραφικού στοιχείου ρυθμιστικού.

Παράδειγμα:

```
from tkinter import *

def sel():
   selection = "Τιμή = " + str(var.get())
   label.config(text = selection)

root = Tk()
var = DoubleVar()
scale = Scale( root, variable = var )
scale.pack(anchor = CENTER)
```

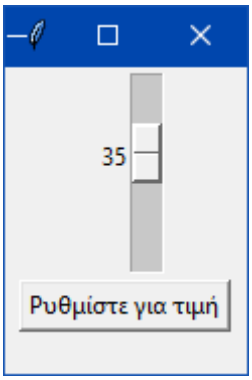
button = Button(root, text = "Ρυθμίστε για τιμή", command = sel)

button.pack(anchor = CENTER)

label = Label(root)

label.pack()

root.mainloop()



17.2.4 Scrollbar - Γραμμή κύλισης

Το γραφικό στοιχείο Scrollbar χρησιμοποιείται για την προσθήκη δυνατότητας κύλισης σε διάφορα γραφικά στοιχεία, όπως πλαίσια λίστας.

Παράδειγμα:

from tkinter import *

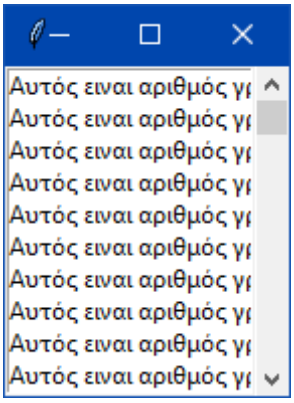
root = Tk()

scrollbar = Scrollbar(root)

scrollbar.pack(side = RIGHT, fill = Y)

```
mylist = Listbox(root, yscrollcommand = scrollbar.set )
for line in range(100):
   mylist.insert(END, "Αυτός ειναι αριθμός γραμμής " + str(line))

mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )

mainloop()
```

Έξοδος:



17.2.5 Text - Κείμενο

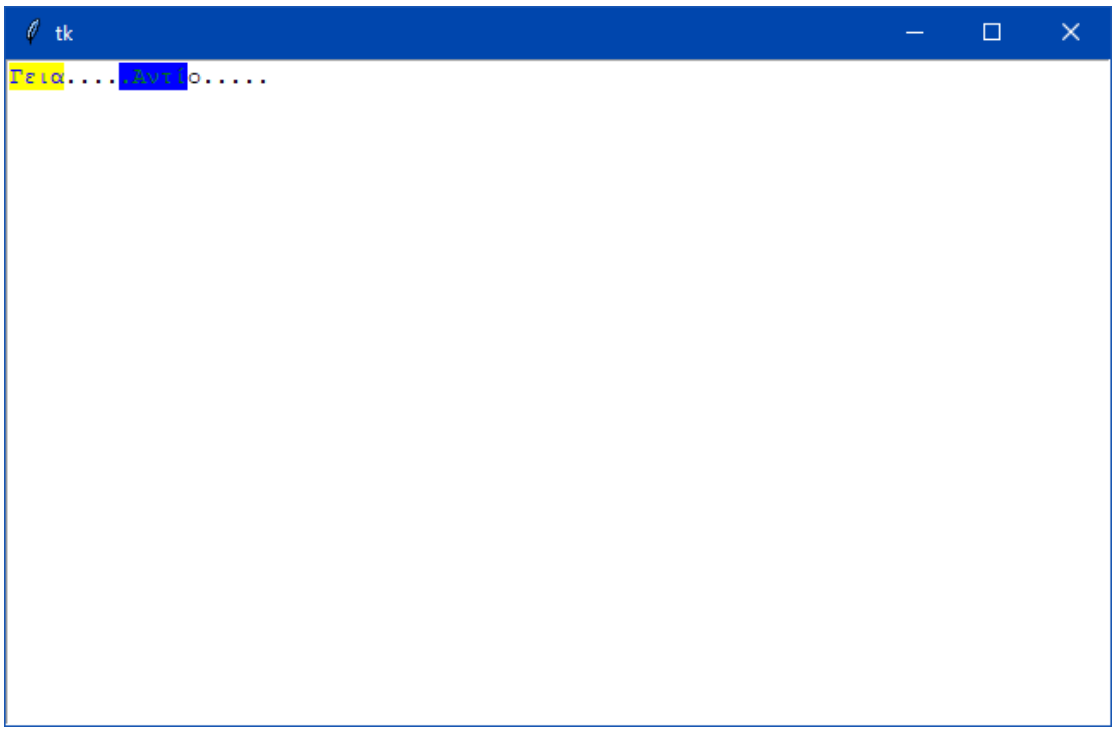
Το γραφικό στοιχείο κειμένου χρησιμοποιείται για την εμφάνιση κειμένου σε πολλές γραμμές.

Παράδειγμα:

```
from tkinter import *
```

```
root = Tk()
text = Text(root)
text.insert(INSERT, "Γεια.....")
text.insert(END, "Αντίο.....")
text.pack()

text.tag_add("εδώ", "1.0", "1.4")
text.tag_add("αρχή", "1.8", "1.13")
text.tag_config("εδώ", background = "yellow", foreground = "blue")
text.tag_config("αρχή", background = "blue", foreground = "green")
root.mainloop()
```

Έξοδος:

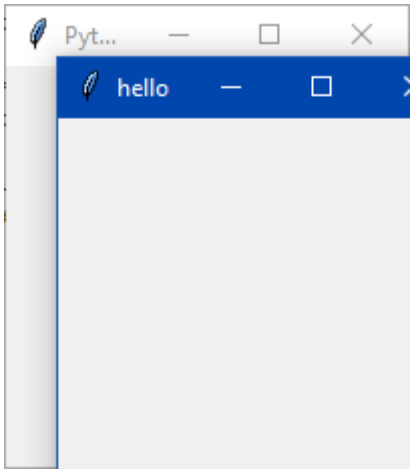


17.2.6 Toplevel - Κορυφαίο επίπεδο

Το γραφικό στοιχείο Toplevel χρησιμοποιείται για την παροχή ενός ξεχωριστού κοντέινερ παραθύρου.

Παράδειγμα:

```
from tkinter import *

root = Tk()
root.title("hello")
top = Toplevel()
top.title("Python")
top.mainloop()
```



17.2.7 Spinbox

Το γραφικό στοιχείο Spinbox είναι μια παραλλαγή του τυπικού γραφικού στοιχείου Tkinter Entry, το οποίο μπορεί να χρησιμοποιηθεί για επιλογή από έναν σταθερό αριθμό τιμών.

Παράδειγμα:

```
from tkinter import *

master = Tk()

w = Spinbox(master, from_ = 0, to = 10)
w.pack()

mainloop()
```

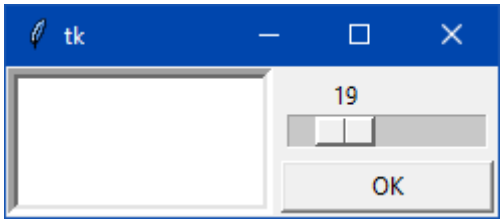
17.2.8 PanedWindow

Ένα PanedWindow είναι ένα γραφικό στοιχείο κοντέινερ που μπορεί να περιέχει οποιονδήποτε αριθμό παραθύρων, διατεταγμένα οριζόντια ή κάθετα.

Παράδειγμα:

```
from tkinter import *
```

```
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)

left = Entry(m1, bd = 5)
m1.add(left)

m2 = PanedWindow(m1, orient = VERTICAL)
m1.add(m2)

top = Scale( m2, orient = HORIZONTAL)
m2.add(top)

bottom = Button(m2, text = "OK")
m2.add(bottom)

mainloop()
```

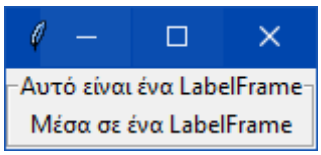


17.2.9 Label frame - Πλαίσιο ετικέτας

Ένα πλαίσιο ετικέτας είναι ένα απλό widget κοντέινερ. Ο πρωταρχικός του σκοπός είναι να λειτουργεί ως διαχωριστικό ή δοχείο για πολύπλοκες διατάξεις παραθύρων.

Παράδειγμα:

```
from tkinter import *

root = Tk()

labelframe = LabelFrame(root, text = "Αυτό είναι ένα LabelFrame")
labelframe.pack(fill = "both", expand = "yes")

left = Label(labelframe, text = "Μέσα σε ένα LabelFrame")

left.pack()

root.mainloop()
```



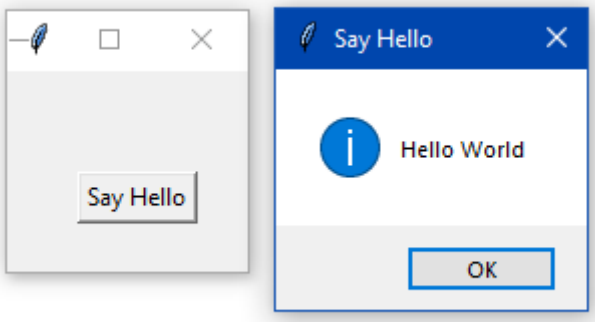
17.2.10 tkMessageBox

Αυτή η ενότητα χρησιμοποιείται για την εμφάνιση πλαισίων μηνυμάτων στις εφαρμογές μας.

Παράδειγμα:

```
from tkinter import

from tkinter import messagebox

top = Tk()

top.geometry("100x100")

def hello():

  messagebox.showinfo("Say Hello", "Hello World")

B1 = Button(top, text = "Say Hello", command = hello)

B1.place(x = 35,y = 50)

top.mainloop()
```

Έξοδος:



17.3.0 Εισαγωγή στο ttkbootstrap

Όσοι γνωρίζετε το bootstrap, θα γνωρίζετε ότι πρόκειται για ένα css framework, το οποίο είναι αρκετά δημοφιλές στο διαδίκτυο και κυρίως στους front-end developers.

Το ttkbootstrap λοιπόν είναι μια «συνεργασία» του ttk με το bootstrap, το οποίο μας εξασφαλίζει όλα τα καλά εμφανισιακά χαρακτηριστικά του bootstrap.

Για να ξεκινήσουμε λοιπόν, πρέπει να κατεβάσουμε την αντίστοιχη βιβλιοθήκη, την οποία κατόπιν θα μπορούμε να κάνουμε import.

Στα windows, ανοίγουμε το command line και δίνουμε την εντολή:

pip install ttkbootstrap

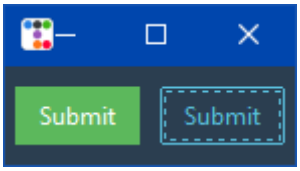
Ας το γνωρίσουμε:

Αντί να χρησιμοποιούμε μεγάλες δηλώσεις του ttk, μπορούμε με μικρότερες και απλούστερες δηλώσεις να έχουμε ένα καλύτερο εμφανισιακά αποτέλεσμα:

```
import ttkbootstrap as ttk
from ttkbootstrap.constants import *

root = ttk.Window(themename="superhero")

b1 = ttk.Button(root, text="Submit", bootstyle="success")
b1.pack(side=LEFT, padx=5, pady=10)

b2 = ttk.Button(root, text="Submit", bootstyle="info-outline")
b2.pack(side=LEFT, padx=5, pady=10)
```

root.mainloop()

Το αποτέλεσμα στην οθόνη μας είναι:



Το νέο API είναι ελαστικό και οι παρακάτω δηλώσεις έχουν όλες το ίδιο αποτέλεσμα, δίνοντάς μας έτσι μεγάλη ευελιξία και πολύ λιγότερα σφάλματα:

bootstyle="info-outline"

bootstyle="info outline"

bootstyle=("info", "outline")

bootstyle=(INFO, OUTLINE)

Για πλήρη αναφορά στα χαρακτηριστικά και τις δυνατότητες του ttkbootstrap, μπορεί κάποιος να ψάξει σε βάθος στην τεκμηρίωσή του και στο Github:

Τεκμηρίωση: https://ttkbootstrap.readthedocs.io/en/latest/

GitHub: https://github.com/israel-dryer/ttkbootstrap

Για την ακρίβεια, τα χαρακτηριστικά του είναι:

Ενσωματωμένα themes
Πάνω από δώδεκα dark και light themes

Προκαθορισμένα στυλ

Έτοιμα widgets με στυλ όπως κουμπιά με στρογγυλές γωνίες ή με διακεκομμένο περίγραμμα.

Απλή χρήση με λέξεις κλειδιά

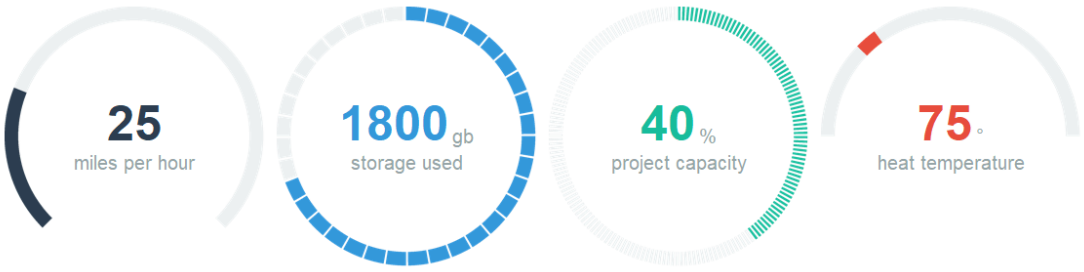
Εφαρμογή χρωμάτων και τύπων χρησιμοποιώντας λέξεις κλειδιά όπως **primary** και **striped** αντί για την κλασική προσέγγιση του ***primary.Striped.Horizontal.TProgressbar***. Αν κάποιος έχει χρησιμοποιήσει το Bootstrap για web development, θα του είναι ήδη προσφιλής η χρήση των κλάσεων της css.

Πολλά νέα Widgets:
Το ***ttkbootstrap*** έρχεται με πολλά σύγχρονα και με εξαιρετική εμφάνιση widgets όπως τα: Meter, DateEntry, και Floodgauge. Επιπροσθέτως, οι διάλογοι είναι τώρα εμπλουτισμένοι με themes και πλήρως παραμετροποιήσιμοι. dialogs are now themed and fully customizable.

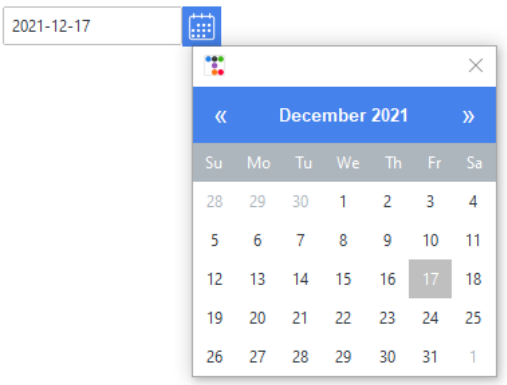
Ενσωματωμένος δημιουργός θεμάτων.

Με το ***ttkbootstrap*** μπορούμε πλέον σχετικά εύκολα να δημιουργήσουμε και να φορτώσουμε τα δικά μας θέματα.

Ας δούμε τα meters:

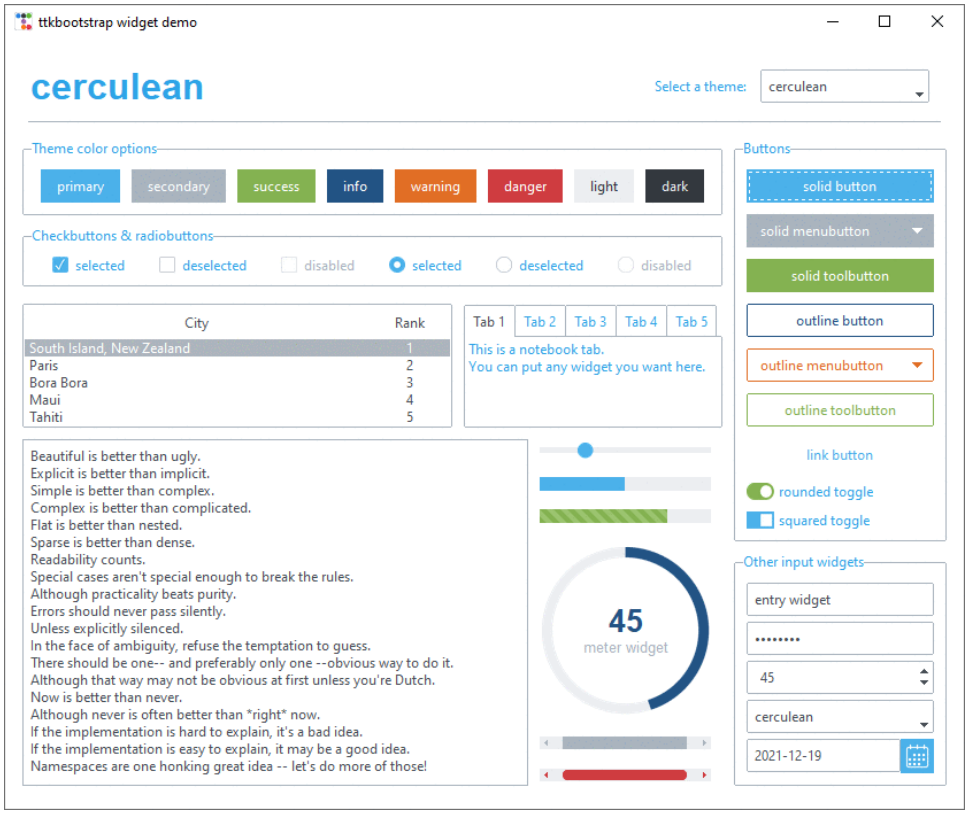


το date-entry:



και το floodgauge:

καθώς και ένα gif με εναλλασσόμενα θέματα:



17.3.1 Δημιουργία εφαρμογής

Γνωρίζοντας ήδη το tkk, η κλασική προσέγγιση θα ήταν να τα κάνουμε όλα όπως ήδη γνωρίζουμε, με τη διαφορά, να εισάγουμε το ttkbootstrap αντί για το ttk και να χρησιμοποιήσουμε το bootstyle αντί για το boot, καθώς και να εισάγουμε τις «σταθερές» (constants) του bootstrap.

Δηλαδή:

import tkinter as tk

import ttkbootstrap as ttk

from ttkbootstrap.constants import *

root = tk.Tk()

b1 = ttk.Button(root, text="Button 1", bootstyle=SUCCESS)

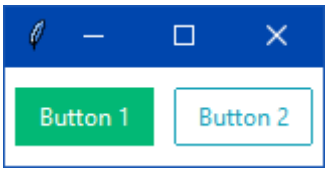
b1.pack(side=LEFT, padx=5, pady=10)

b2 = ttk.Button(root, text="Button 2", bootstyle=(INFO, OUTLINE))

b2.pack(side=LEFT, padx=5, pady=10)

root.mainloop()

για να πάρουμε το παραθυράκι:



17.3.2 Νέα προσέγγιση

Το ίδιο αποτέλεσμα μπορούμε να έχουμε χρησιμοποιώντας τη νέα κλάση Window. Στην αρχή, η διαφορά μοιάζει μικρή, όμως αυτή η κλάση χρησιμοποιεί ειδικές παραμέτρους για να ορίζει πολλά από τα χαρακτηριστικά, ενώ με την κλασική προσέγγιση με την κλάση Tk, θα χρειαζόμασταν μεθόδους. Επίσης, το στυλιζαρισμένο αντικείμενο, ενσωματώνεται στο παράθυρο αυτόματα, όπως θα δούμε παρακάτω:

import ttkbootstrap as ttk

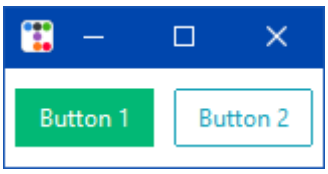
from ttkbootstrap.constants import *

root = ttk.Window()

b1 = ttk.Button(root, text="Button 1", bootstyle=SUCCESS)

b1.pack(side=LEFT, padx=5, pady=10)

```
b2 = ttk.Button(root, text="Button 2", bootstyle=(INFO, OUTLINE))
b2.pack(side=LEFT, padx=5, pady=10)

root.mainloop()
```



17.3.3 Επιλογή theme

Το default theme είναι το litera, αλλά μπορούμε να θέσουμε όποιο άλλο θέλουμε για να ξεκινήσουμε, επιλέγοντας έναν από τους παρακάτω τρόπους:

```
import ttkbootstrap as ttk

# κλασική προσέγγιση
root = ttk.Tk()
style = ttk.Style("darkly")

# νέα προσέγγιση
root = ttk.Window(themename="darkly")
```

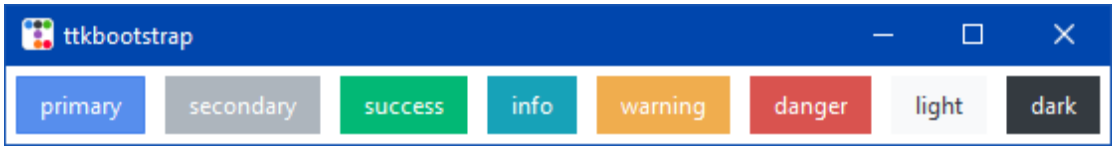
Στο ttkbootstrap υπάρχουνε δεκάδες προκαθορισμένα στυλ, τα οποία εφαρμόζονται χρησιμοποιώντας λέξεις κλειδιά, που παραμετροποιούν και τον τύπο και το χρώμα του widger. Για την ακρίβεια, το χρώμα καθορίζεται κάθε φορά από την επιλογή του αντίστοιχου theme.

Σαν παράδειγμα, χρησιμοποιώντας τη λέξη κλειδή outline, δημιουργούμε ένα κουμπί με τύπο outline (περίγραμμα), αλλά χρησιμοποιώντας επίσης τη λέξη κλειδή info θα αλλάξει το χρώμα του περιγράμματος και του κειμένου.

Πάμε να δούμε τα χρώματα για κάθε κουμπί:

```
import ttkbootstrap as ttk
from ttkbootstrap.constants import *

root = ttk.Window()

b1 = ttk.Button(root, text='primary', bootstyle=PRIMARY)
b1.pack(side=LEFT, padx=5, pady=5)

b2 = ttk.Button(root, text='secondary', bootstyle=SECONDARY)
b2.pack(side=LEFT, padx=5, pady=5)

b3 = ttk.Button(root, text='success', bootstyle=SUCCESS)
b3.pack(side=LEFT, padx=5, pady=5)

b4 = ttk.Button(root, text='info', bootstyle=INFO)
b4.pack(side=LEFT, padx=5, pady=5)

b5 = ttk.Button(root, text='warning', bootstyle=WARNING)
```

```
b5.pack(side=LEFT, padx=5, pady=5)

b6 = ttk.Button(root, text='danger', bootstyle=DANGER)
b6.pack(side=LEFT, padx=5, pady=5)

b7 = ttk.Button(root, text='light', bootstyle=LIGHT)
b7.pack(side=LEFT, padx=5, pady=5)

b8 = ttk.Button(root, text='dark', bootstyle=DARK)
b8.pack(side=LEFT, padx=5, pady=5)

root.mainloop()
```



Ας δούμε τώρα πόσο πιο απλά θα μπορούσαμε να έχουμε δημιουργήσει όλα αυτά τα κουμπιά, χρησιμοποιώντας το αντικείμενο Style.colors το οποίο εμπεριέχει μια αναφορά σε όλα τα χρώματα του theme και το οποίο επίσης είναι ένας iterator.

Δηλαδή:

```
import ttkbootstrap as ttk
from ttkbootstrap.constants import *

root = ttk.Window()
```

```
for color in root.style.colors:
    b = ttk.Button(root, text=color, bootstyle=color)
    b.pack(side=LEFT, padx=5, pady=5)
```

Η λέξη κλειδί μπορεί να ελέγξει τον τύπο του widget που παρουσιάζεται. Ας δούμε στο επόμενο παράδειγμα ένα solid κι ένα outline κουμπί. Είναι και τα δύο κουμπιά, διαφορετικών όμως τύπων:

```
import ttkbootstrap as ttk
from ttkbootstrap.constants import *

root = ttk.Window()

b1 = ttk.Button(root, text="Solid Button", bootstyle=SUCCESS)
b1.pack(side=LEFT, padx=5, pady=10)

b2 = ttk.Button(root, text="Outline Button", bootstyle=(SUCCESS, OUTLINE))
b2.pack(side=LEFT, padx=5, pady=10)

root.mainloop()
```

Όπως βλέπουμε, προσθέτοντας τη λέξη κλειδί outline, το κουμπί μετατρέπεται από solid σε outline.

17.4.0 Μελέτη και βελτίωση παιχνιδιού Tic Tac Toe

Κατανοήστε πώς λειτουργεί, μελετήστε και προτείνετε πιθανές βελτιώσεις στον παρακάτω κώδικα του παιχνιδιού Tic Tac Toe. Να έχετε έτοιμες καταγεγραμμένες απαντήσεις.

```
import tkinter as tk
#from tkinter import ttk
import ttkbootstrap as ttk

def array_match(arrayA, arrayB):
    result = []
    for i in arrayA:
        if i in arrayB:
            result.append(i)

    return result

def wincon():
```

```
    global player_move
    global playermovelogs
    winningConditions = [
        [0,3,6],
        [1,4,7],
        [2,5,8],
        [0,1,2],
        [3,4,5],
        [6,7,8],
        [0,4,8],
        [2,4,6]
    ]

    for i in winningConditions:
        matches = array_match(playermovelogs.get("X"),i)
        if len(matches) == 3:
            return 1

    for i in winningConditions:
        matches = array_match(playermovelogs.get("O"),i)
        if len(matches) == 3:
            return 2

    return 0


def capture_spot(player, button):
    global player_move
    global outputtext
```

```
global playermovelogs

print(player,button)
if player == "X":
        player_color = "red"
        player_move = "O"
        outputlabel.config(text = "[O] Player 2's Move")

elif player == "O":
        player_color = "blue"
        player_move = "X"
        outputlabel.config(text = "[X] Player 1's Move")

buttons[button].config(text = player)
buttons[button].config(state = "disabled")
buttons[button].config(bg = player_color)
buttons[button].config(fg = "#ffffff")
buttons[button].config(font = "Calibri 10 bold")

playermovelogs.get(player).append(button)

if(wincon() == 1):
        for i in buttons:
                i.config(state = "disabled")
        outputlabel.config(text = "Player 1 [X] Has Won!")
elif(wincon() == 2):
        for i in buttons:
                i.config(state = "disabled")
        outputlabel.config(text = "Player 2 [O] Has Won!")
```

```
# window
window = ttk.Window(themename = "darkly")
window.title("Tic Tac Toe")
window.geometry("400x500")
window.resizable(False, False)

# variables
player_move = "X"
outputtext = "[X] Player 1's Move"
playermovelogs = {"X" : [], "O" : []}


# title
title = ttk.Label(
        master=window,
        text="Tic Tac Toe",
        font="Calibri 24 bold"
)
title.grid(row=0, column=0, columnspan=3, pady=10,padx=10)

# frame
frame = tk.Frame(
        master=window
)
frame.grid(row=1, column=0, padx=15)

# buttons
buttons = []
```

```
for i in range(9):
        button = tk.Button(
                master=frame,
                text=str(i+1),
                width = 15,
                #padding = (0,30),
                height = 5,
                command = lambda index = i:
capture_spot(player_move, index)
        )
        buttons.append(button)

# grid layout for buttons
for i, button in enumerate(buttons):
        button.grid(
                row=i // 3,
                column=i % 3,
                padx=5,
                pady=5,
        )

# output label
outputlabel = ttk.Label(
        master = frame,
        text = outputtext,
        font = "Calibri 12"
)
outputlabel.grid(column = 1, row = 5)
```

```
# run
window.mainloop()
```