Γραφικά στο Raspberry pi

ΜΑΘΗΜΑΤΑ RASPBERRY

ΧΡΗΣΤΟΣ ΤΟΠΑΛΙΔΗΣ

ΔΟΥΜΑ ΑΝΑΣΤΑΣΙΑ

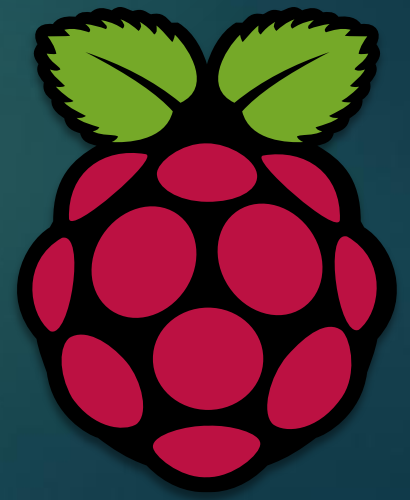
ΚΑΒΑΛΛΙΕΡΑΤΟΥ ΕΡΓΙΝΑ

Περιεχόμενα

- Εισαγωγή στο GUI
- Επιλογές για GUI σε python
- Εγκατάσταση guizero
- Προσθήκη στοιχείων widgets
- listeners
- GUI layout
- Άνοιγμα επιπλέον παραθύρων

Εισαγωγή στο GUI

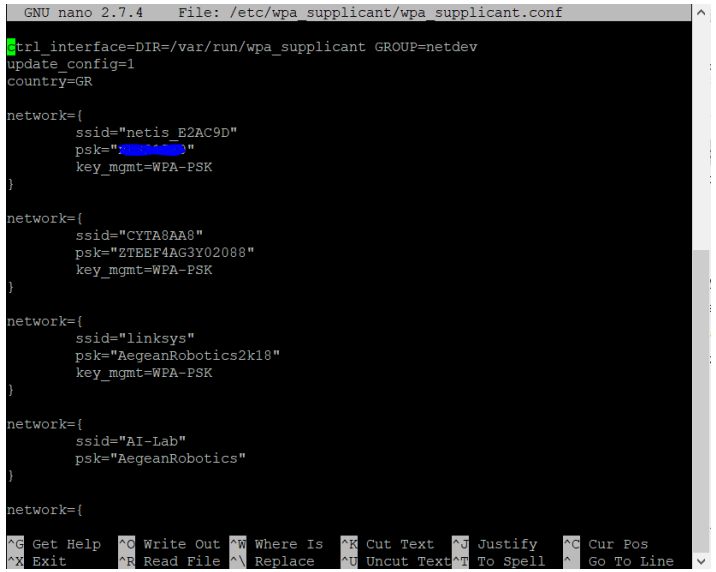
- Για να έχουμε μια ολοκληρωμένη εφαρμογή είναι απαραίτητο το γραφικό περιβάλλον (Graphical User Interface - GUI)
- Το γραφικό περιβάλλον βοηθάει του χρήστες να μπορούν να διαχειρίζονται την εφαρμογή πιο εύκολα και κατανοητά για αυτούς
- Πολλές εφαρμογές δεν θα μπορούσαν να υπάρξουν χωρίς GUI
- Στα ρομπότ τα οποία ενσωματώνουν οθόνες είναι σχεδόν απαραίτητη η προσθήκη γραφικού περιβάλλοντος



Εισαγωγή στο GUI

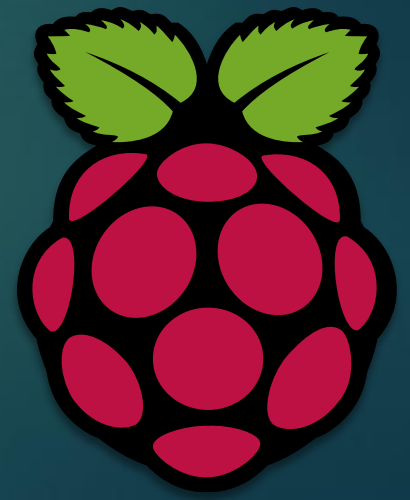
1. Μια εφαρμογή με χρήση CLI (command line interface)
2. Εφαρμογή με GUI

- Είναι πιο εύκολο για τον μέσο χρήση να διαχειριστεί μια εφαρμογή με την χρήση GUI



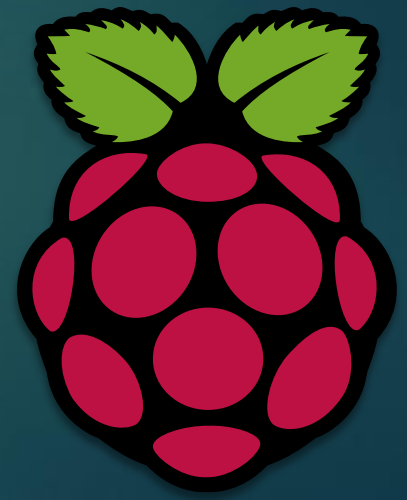
Επιλογές για GUI σε python

- Υπάρχουν πολλές και διαφορετικές βιβλιοθήκες για να κατασκευάσει κάποιος μια διεπαφή στην γλώσσα προγραμματισμού python και στο raspberry pi
- Μια από τις πιο γνωστές βιβλιοθήκες είναι η Tkinter όπου μπορεί να χρησιμοποιηθεί σε όλα τα γνωστά λειτουργικά συστήματα
- Ακόμα μια γνωστή επιλογή είναι η βιβλιοθήκη QT η οποία εκτελείται σε διαφορετικά λειτουργικά συστήματα αλλά και σε διαφορετικές γλώσσες προγραμματισμού



Επιλογές για GUI σε python

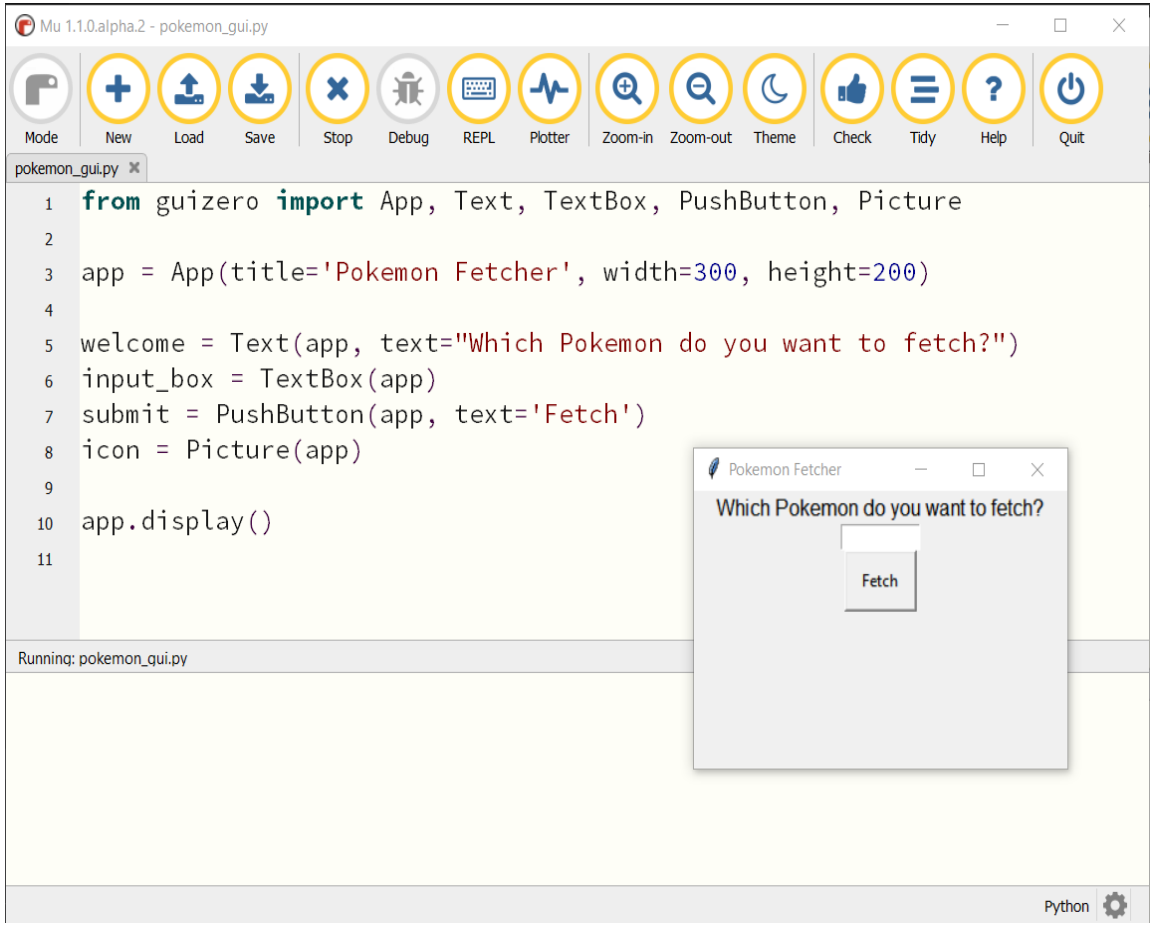
- Ακόμα υπάρχει και η επιλογή της swing. Αυτή η επιλογή είναι για γλώσσα προγραμματισμού java και όχι python. Μπορεί όμως να χρησιμοποιηθεί στο raspberry pi και είναι μια καλή επιλογή για αυτούς που έχουν γνώσεις προγραμματισμού σε java
- Η GTK+ είναι μια γνωστή επιλογή που μπορεί να χρησιμοποιηθεί και στην python καθώς επίσης σε C και java
- Τέλος υπάρχουν αρκετές ακόμα βιβλιοθήκες. Η καλύτερη επιλογή εξαρτάται από το τι θέλουμε να κάνουμε και το πόσο καλά γνωρίζουμε μια γλώσσα προγραμματισμού



Εγκατάσταση guizero

- Για αυτά τα μαθήματα έχει επιλεγεί η βιβλιοθήκη guizero για τους παρακάτω λόγους :

1. Είναι εύκολη στην εγκατάσταση
2. Έχει σχεδιαστεί για νέους χρήστες
3. Παρέχει όλα τα βασικά χαρακτηριστικά που θέλουμε
4. Είναι ελαφριά βιβλιοθήκη
5. Υπάρχει αρκετή πληροφορία στο διαδίκτυο για αυτήν
6. Και παρέχει χρήσιμη πληροφορία για να γίνονται κατανοητά τα λάθη

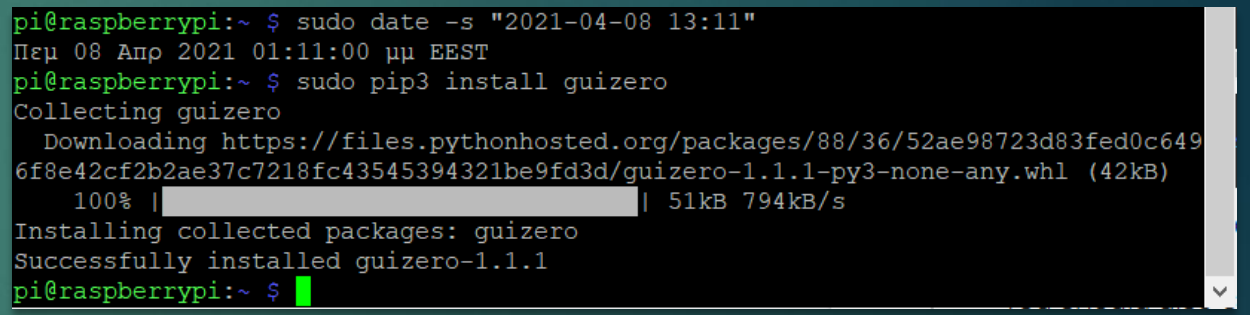


Εγκατάσταση guizero

- Για την εγκατάσταση του guizero πρέπει να ακολουθήσουμε τα παρακάτω βήματα :

1. Ορίζουμε την σωστή ημερομηνία και ώρα με την εντολή
   sudo date –s “2021-04-08 13:11”
2. Κάνουμε εγκατάσταση της βιβλιοθήκης
   sudo pip3 install guizero

Αν εμφανιστεί μήνυμα όπως αυτό της οθόνης η εγκατάσταση έχει γίνει με επιτυχία.



Εγκατάσταση guizero

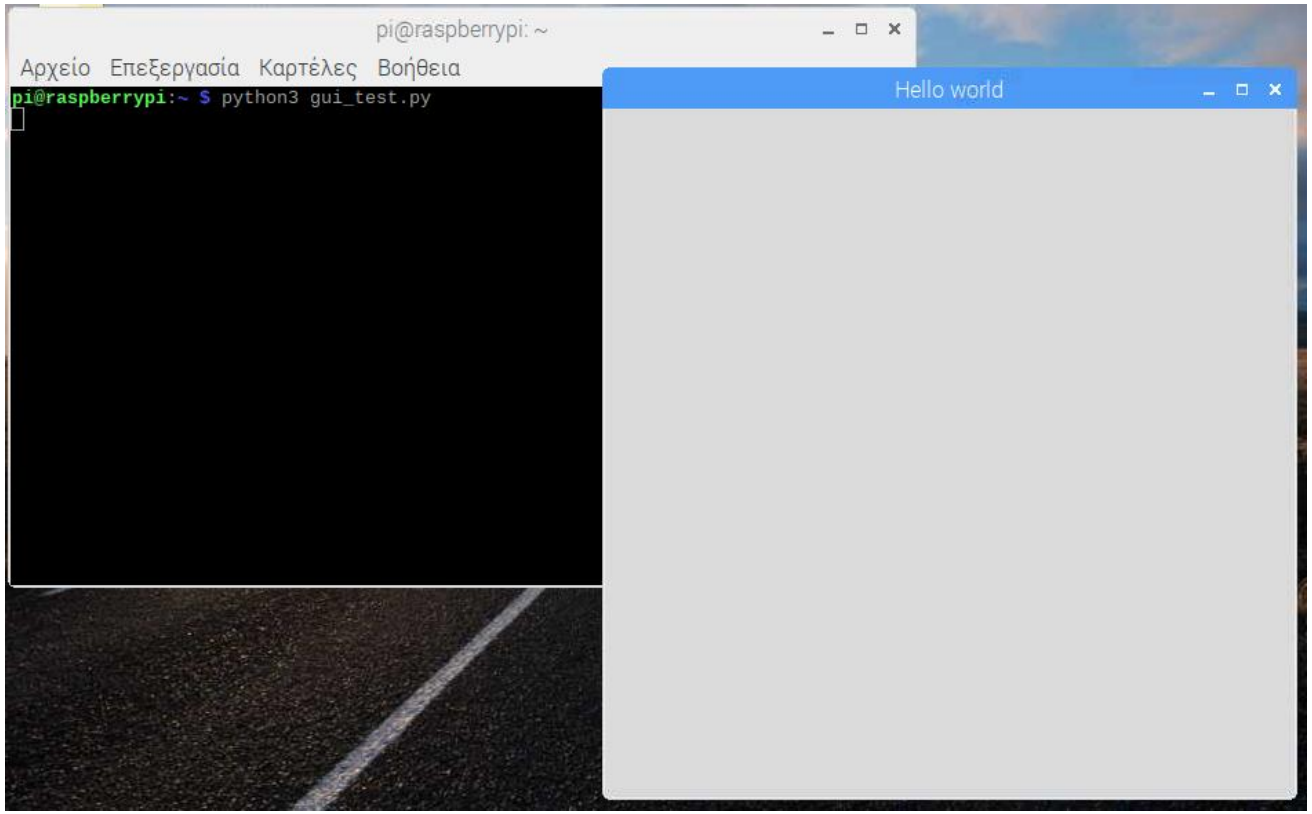
- Για να ελέγξουμε ότι η βιβλιοθήκη λειτουργεί και έχει εγκατασταθεί σωστά φτιάχνουμε το πρώτο μας πρόγραμμα
- Για τα προγράμματα με γραφικό περιβάλλον θα πρέπει να χρησιμοποιήσουμε remote πρόγραμμα που να υποστηρίζει γραφικά, όπως το vnc viewer
- Για να φτιάξουμε το πρόγραμμα μας γράφουμε
  sudo nano gui_test.py
- Και γράφουμε τον κώδικα που φαίνεται στην εικόνα

```
pi@raspberrypi:~ $ sudo nano gui_test.py
```

```
  GNU nano 2.7.4          File: gui_test.py

from guizero import App

app = App(title="Hello world")
app.display()
```

Εγκατάσταση guizero

- Εκτελούμε το αρχείο που φτιάξαμε γράφοντας την εντολή
  python3 gui_test.py
- Και μας εμφανίζει το πρώτο μας παράθυρο όπου γράφει επάνω hello world
- Προσοχή γράφουμε python3 και όχι python



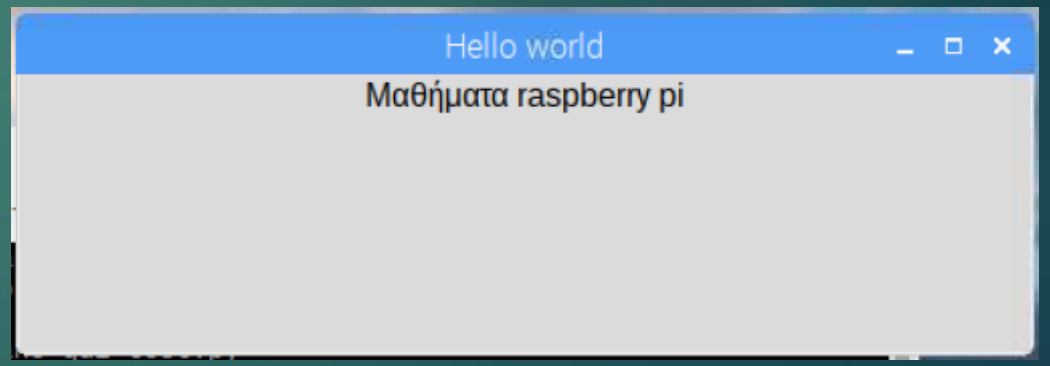
Προσθήκη στοιχείων widgets

- Εκτός από τον τίτλο σε ένα παράθυρο μπορούμε να προσθέσουμε και στοιχεία μέσα σε αυτό, όπως κείμενο (text), κουμπιά ενεργειών (buttons), πεδία κειμένου (textfields), slidebar, λίστες και πολλά άλλα widgets
- Για να προσθέσουμε ένα στοιχείο στο πρόγραμμα μας πρέπει πάντα να το κάνουμε import και έπειτα να το χρησιμοποιήσουμε: **from guizero import App, Text, TextBox**
- Στο παραπάνω παράδειγμα προσθέτουμε αρχικά το App που είναι η σελίδα μας, το Text που είναι απλό κείμενο και το TextBox που είναι μια περιοχή για να γράψουμε και να μεταφέρουμε πληροφορία στο πρόγραμμα μας

Προσθήκη στοιχείων widgets

- Για να χρησιμοποιήσουμε οποιοδήποτε widget πρέπει πρώτα να το εισάγουμε. Σε αυτή την περίπτωση εισάγουμε το Text για να γράψουμε κείμενο
- Για να τυπώσουμε το μήνυμα μας γράφουμε την εντολή Text(..)
- Το app.display() είναι ένα ατέρμονος βρόχος ώστε το παράθυρο να εμφανίζεται συνέχεια μέχρι να το κλείσουμε εμείς

```
from guizero import App,Text

app = App(title="Hello world")
message = Text(app, text="Μαθήματα raspberry pi")

app.display()
```



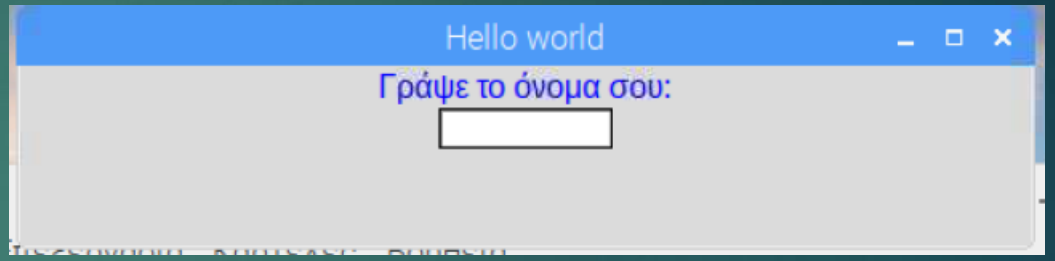
Προσθήκη στοιχείων widgets

- Επίσης μπορούμε να προσαρμόσουμε όπως θέλουμε το κείμενο μας
- Μέσα την συνάρτηση Text μπορούμε να κάνουμε τις αλλαγές που θέλουμε προσθέτοντας το αντίστοιχο στοιχείο όπως για παράδειγμα size=35 και color="blue"
- Το αποτέλεσμα θα είναι να έχουμε μεγάλα μπλε γράμματα

```
app = App(title="Hello world")
message = Text(app, text="Μαθήματα raspberry pi", size=35, color="blue")
```

Προσθήκη στοιχείων widgets

- Για να προσθέσουμε TextBox το κάνουμε αρχικά import
- Και το εισάγουμε με την εντολή name = TextBox(app)
- Επομένως η μεταβλητή name θα είναι μια μεταβλητή τύπου TextBox, η οποία θα περιέχει τα στοιχεία που θα καταχωρεί ο χρήστης

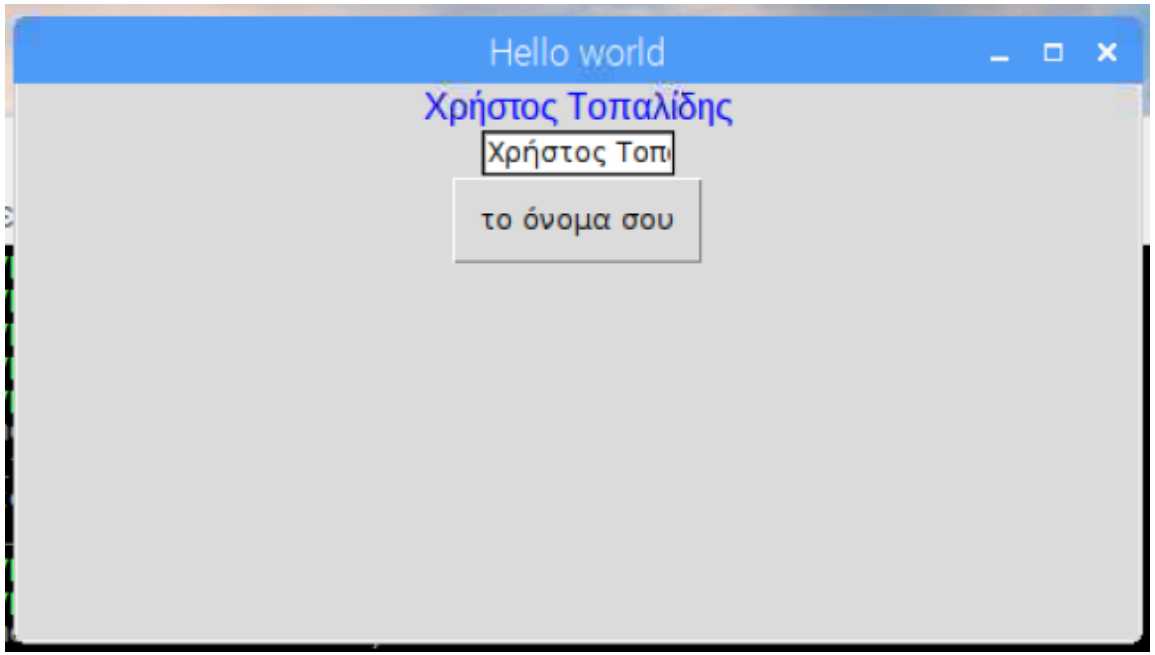



```
from guizero import App,Text,TextBox

app = App(title="Hello world")
message = Text(app, text="Γράψε το όνομα σου:", size=12, color="blue")
name = TextBox(app)

app.display()
```

Προσθήκη στοιχείων widgets

- Καλό είναι να χρησιμοποιούμε συναρτήσεις στον κώδικα μας και να τις γράφουμε κάτω από τα import έτσι ώστε να τις καλούμε όποτε χρειάζεται
- Η συνάρτηση your_name τυπώνει ένα μήνυμα σε μορφή Text

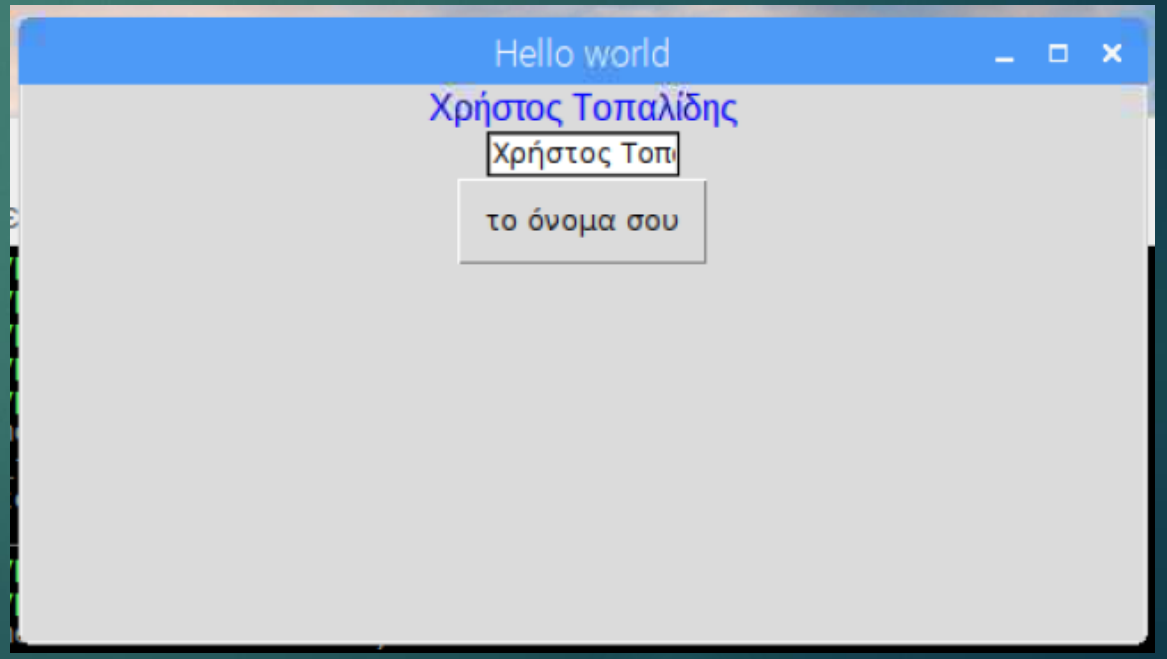
```
from guizero import App,Text,TextBox,PushButton

def your_name():
        message.value = name.value

app = App(title="Hello world")
message = Text(app, text="Γράψε το όνομα σου:", size=12, color="blue")
name = TextBox(app)
update_text = PushButton(app, command=your_name, text="το όνομα σου")

app.display()
```



Προσθήκη στοιχείων widgets

- Προσθέτουμε το στοιχείο PushButton το οποίο όταν πατηθεί θα εκτελεί την συνάρτηση your_name
- Στην μεταβλητή text γράφουμε αυτό που θέλουμε να γράφει το κουμπί
- Αφού πατηθεί το κουμπί ότι γράφει μέσα στο TextBox θα τυπώνεται στο επάνω text

```
from guizero import App,Text,TextBox,PushButton

def your_name():
        message.value = name.value

app = App(title="Hello world")
message = Text(app, text="Γράψε το όνομα σου:", size=12, color="blue")
name = TextBox(app)
update_text = PushButton(app, command=your_name, text="το όνομα σου")

app.display()
```



listeners

- Listeners (ακροατές) είναι κάποιες κλάσεις οι οποίες μας παρέχουν συναρτήσεις για ορισμένα γεγονότα ώστε μετά από αυτά να εκτελεστεί μια άλλη ενέργεια. Όπως για παράδειγμα μετά το πάτημα ενός πλήκτρου να γίνει ο υπολογισμός μιας πράξης
- Υπάρχουν πολλά και διαφορετικά events (γεγονότα) που μπορούν να εγείρουν ένα listener όπως το πάτημα οποιοδήποτε πλήκτρου ή το πάτημα ενός κουμπιού (button) η ακόμα και η κίνηση του ποντικιού
- Κάθε φορά που επιθυμούμε ένα event να επηρεάσει την εκτέλεση του προγράμματος μας, αυτό θα πρέπει να το έχουμε ορίσει από πριν

GUI layout

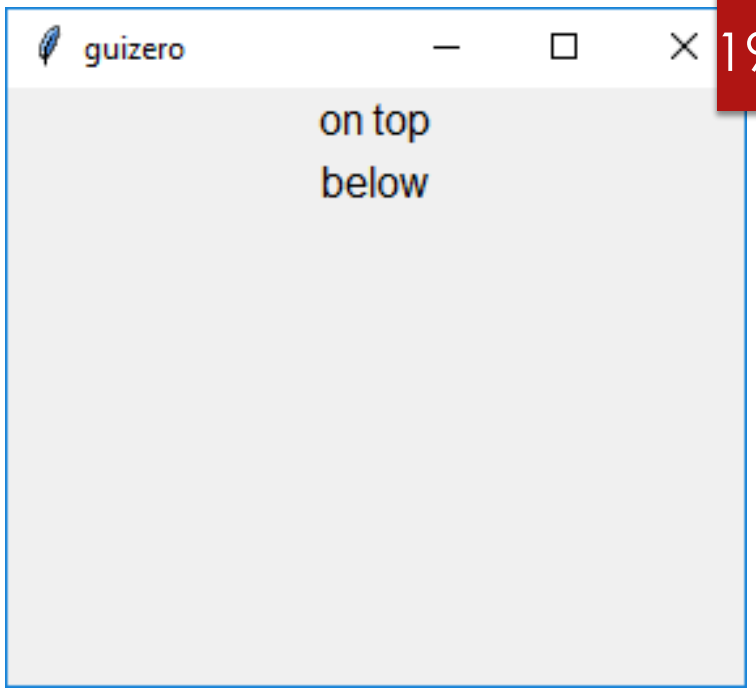
- Layout είναι ο τρόπος με τον οποίο επιλέγει ο προγραμματιστής να κατανεμηθούν τα στοιχεία σε ένα app, window ή και box
- Στην γλώσσα προγραμματισμού python έχουμε 2 layout
  1. Auto
  2. Grid
- Για να δηλώσουμε το layout που θέλουμε να χρησιμοποιηθεί, γράφουμε:

```
app = App(layout="auto") ή app = App(layout="grid")
```

- Η διαφορά ανάμεσα στις δυο επιλογές είναι ότι στην auto επιλέγει το πρόγραμμα να κατανέμει τα στοιχεία όπως αυτό θέλει με μια βασική σειρά το ένα κάτω από το άλλο, ενώ με την grid επιλογή ορίζεται ότι ο προγραμματιστής ρυθμίζει την θέση των στοιχείων, σε μορφή πίνακα

GUI layout

- Στο παράδειγμα της εικόνας βλέπουμε πως τοποθετεί το auto layout τα στοιχεία το ένα κάτω από το άλλο
- Όπως παρατηρείται δεν έχει δηλωθεί πουθενά το layout και έτσι το πρόγραμμα το όρισε από μόνο του
- Στα προηγούμενα παραδείγματα που δείξαμε χρησιμοποιήθηκε το ίδιο layout

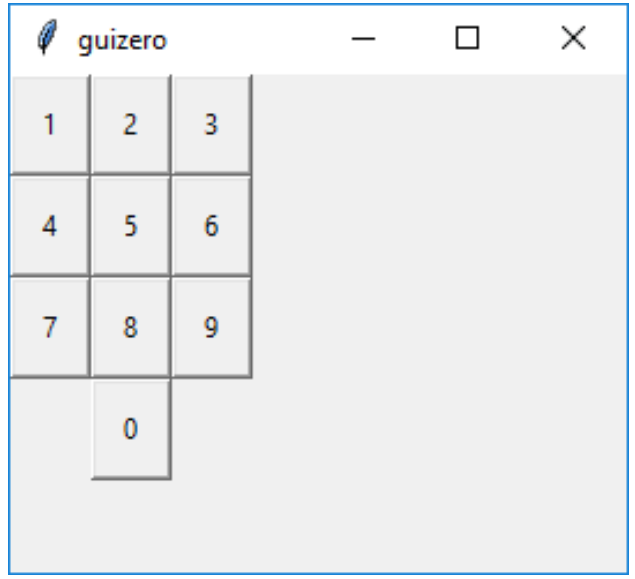



```
from guizero import App, Text
app = App()
text_1 = Text(app, text="on top")
text_2 = Text(app, text="below")
app.display()
```

GUI layout

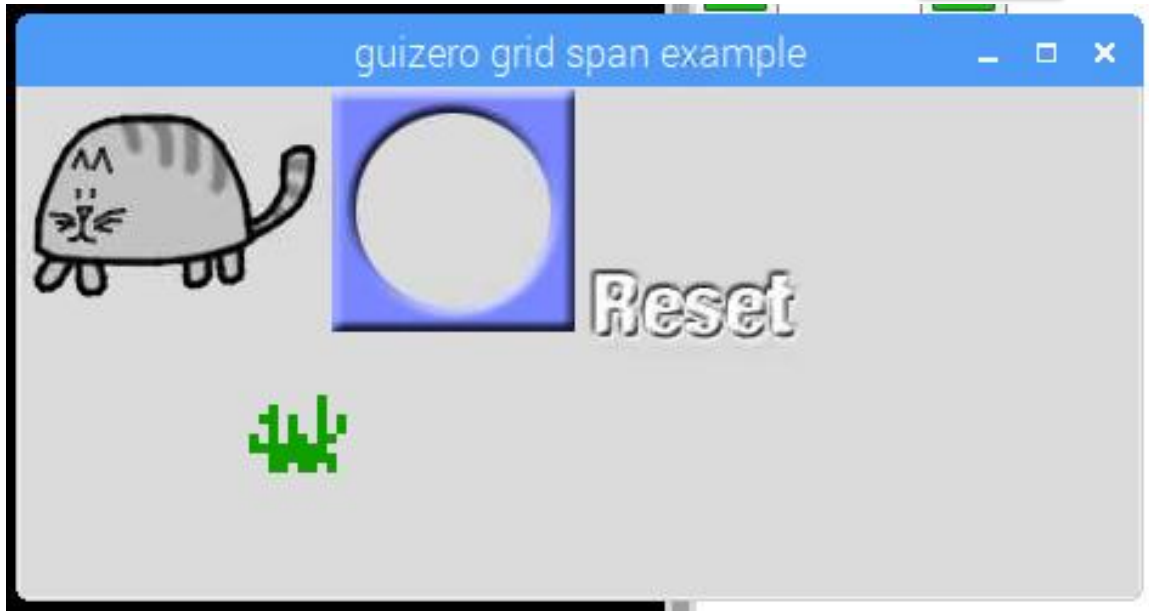
- Στο παράδειγμα της εικόνας βλέπουμε πως τοποθετεί το grid layout τα στοιχεία σε μορφή πίνακα
- Σε αυτή την περίπτωση βλέπουμε ότι έχει δηλωθεί το layout = “grid”

```
from guizero import App, PushButton

app = App(layout="grid")

button1 = PushButton(app, text="1", grid=[0,0])
button2 = PushButton(app, text="2", grid=[1,0])
button3  = PushButton(app, text="3", grid=[2,0])
button4  = PushButton(app, text="4", grid=[0,1])
button5  = PushButton(app, text="5", grid=[1,1])
button6  = PushButton(app, text="6", grid=[2,1])
button7  = PushButton(app, text="7", grid=[0,2])
button8  = PushButton(app, text="8", grid=[1,2])
button9  = PushButton(app, text="9", grid=[2,2])
button0  = PushButton(app, text="0", grid=[1,3])

app.display()
```



GUI layout

- Σε αυτό το παράδειγμα προσθέτουμε τυχαίες εικόνες στο παράθυρο και τις εμφανίζουμε με την σειρά που θέλουμε εμείς




```
from guizero import App, Picture

app = App(title="guizero grid span example", width=460, height=210, layout="grid")
picture1 = Picture(app, image="cat.png", grid=[0,0])
picture2 = Picture(app, image="4row_board.png", grid=[1,0])
picture3 = Picture(app, image="inkspillresetbutton.png", grid=[2,0,1,2])
picture4 = Picture(app, image="grass2.png", grid=[0,1,2,1])

app.display()
```

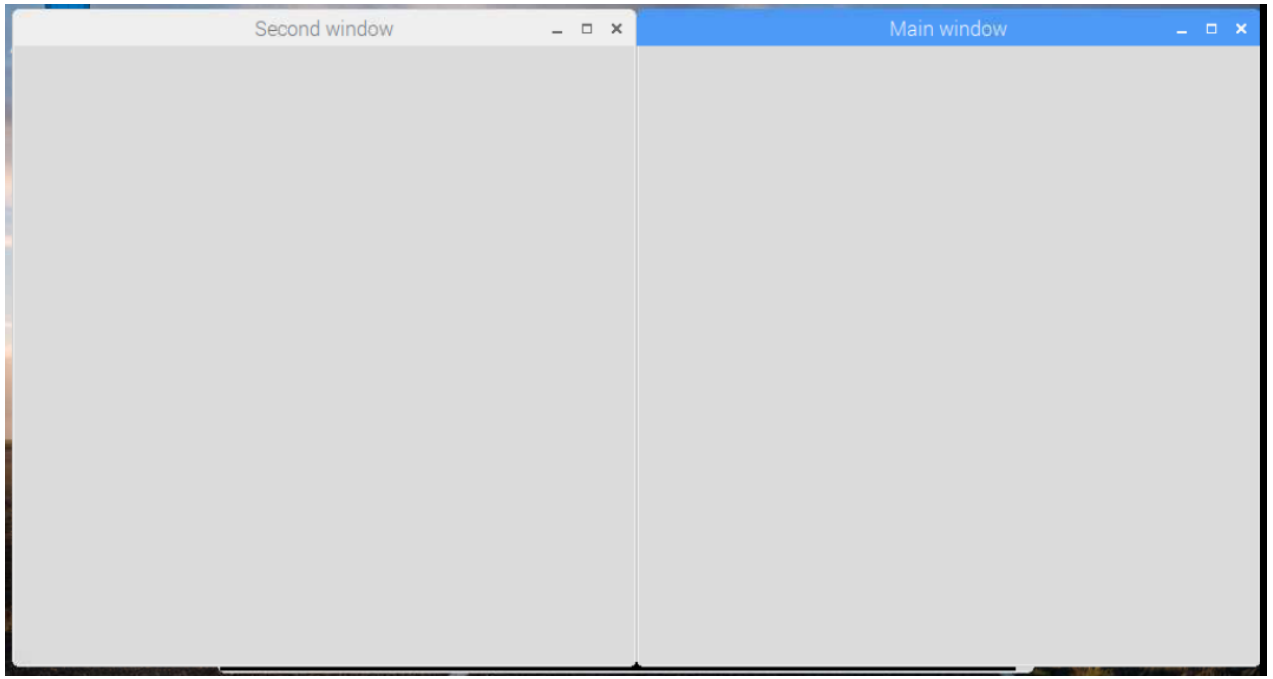
Άνοιγμα επιπλέον παραθύρων

- Σημαντικό σε μια εφαρμογή είναι και το άνοιγμα επιπλέον παραθύρων για την ομαλότερη λειτουργία του προγράμματος
- Όταν δημιουργούμε περισσότερα παράθυρα τότε είναι πιθανό να αυξηθεί πολύ το μέγεθος του κώδικα και να επηρεαστεί η ευκολία στην ανάγνωση του και στην παραμετροποίηση του

Άνοιγμα επιπλέον παραθύρων

- Η μεταβλητή app αποτελεί το κύριο παράθυρο ενώ η window το δευτερεύον
- Όταν τρέξουμε τον κώδικα μας ανοίγουν 2 παράθυρα μαζί
- Αν κλείσουμε το παράθυρο Main window τότε κλείνει τελείως το πρόγραμμα, ενώ αν κλίσουμε το Second windows δεν σταματάει η λειτουργία του προγράμματος, απλά κλείνει το παράθυρο

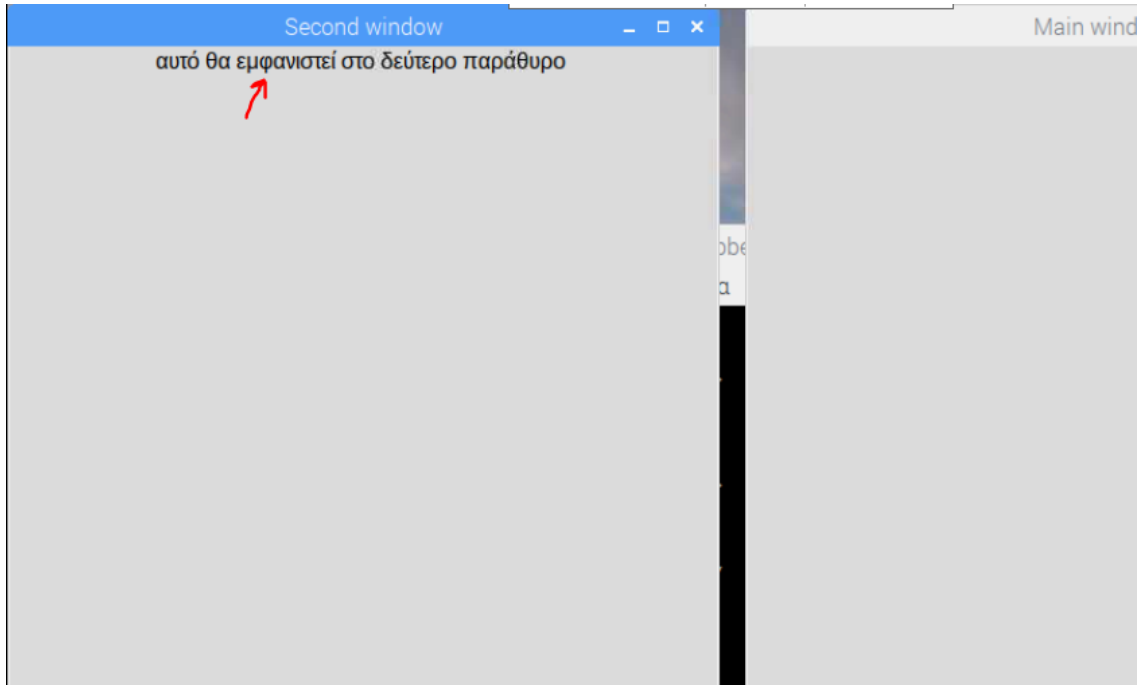
```
 GNU nano 2.7.4                                File: window

from guizero import App, Window

app = App(title="Main window")
window = Window(app, title="Second window")

app.display()
```



Άνοιγμα επιπλέον παραθύρων

- Για να προσθέσουμε οτιδήποτε θέλουμε μέσα στο δεύτερο παράθυρο δουλεύουμε όπως και πριν μόνο που ορίζουμε μέσα στην συνάρτηση το παράθυρο που θέλουμε να προστεθεί το στοιχείο
- Δηλαδή στην συνάρτηση Text(..., ...) αντί για να γράψουμε Text(app, ...) γράφουμε Text(window, ...)
- Και το μήνυμα εμφανίζεται στο δεύτερο παράθυρο

```
from guizero import App, Window, Text

app = App(title="Main window")
window = Window(app, title="Second window")
text = Text(window, text="αυτό θα εμφανιστεί στο δεύτερο παράθυρο ")

app.display()
```

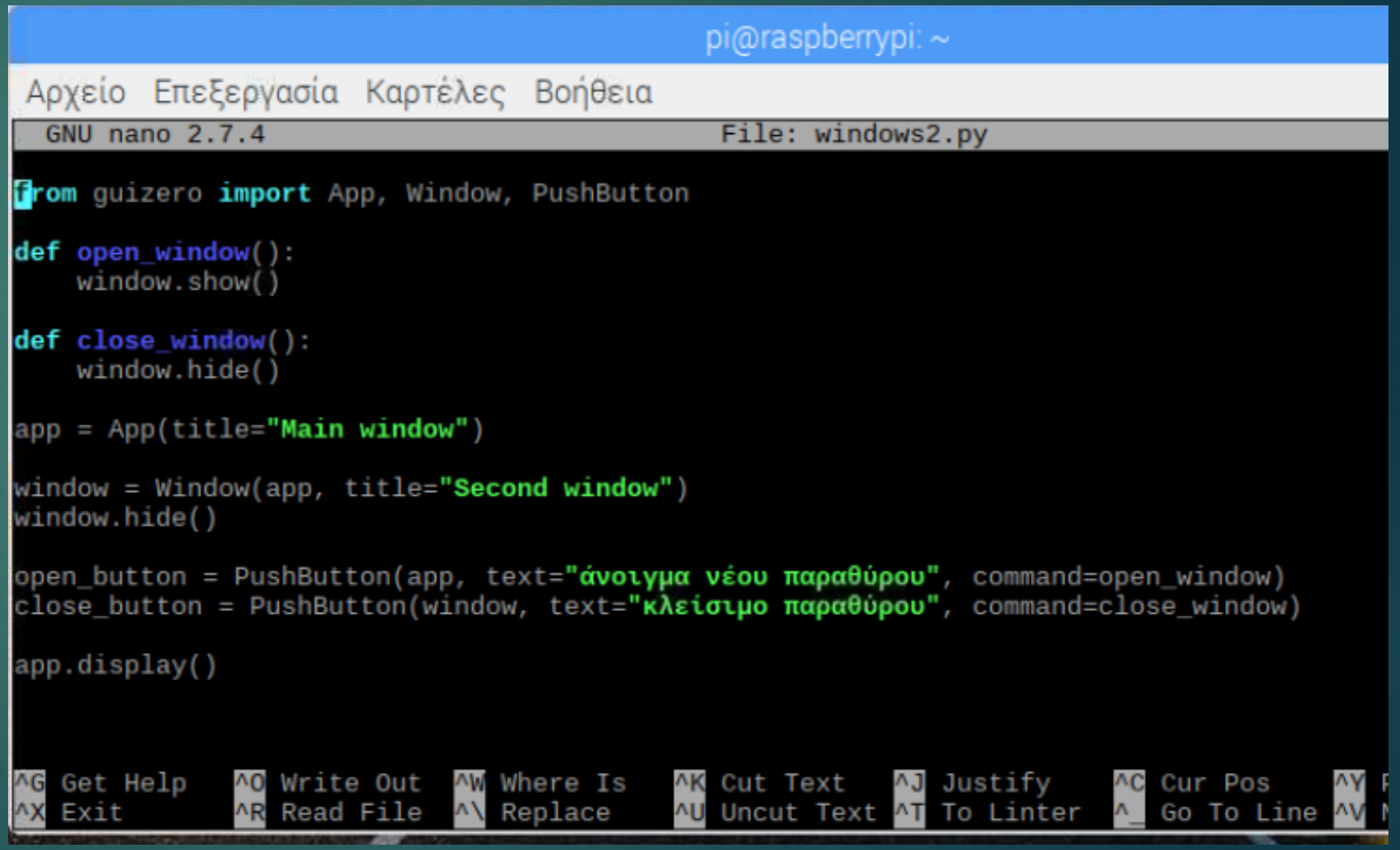


Άνοιγμα επιπλέον παραθύρων

- Είναι σημαντικό να μπορούμε να ελέγξουμε το πότε ανοίγει και κλείνει ένα παράθυρο γιατί συνήθως δεν θέλουμε μια εφαρμογή να ανοίγει όλα τα παράθυρα με το που ξεκινάει
- Όταν δημιουργούμε ένα αντικείμενο window αυτό εμφανίζεται απευθείας στην οθόνη. Μπορούμε όμως να ελέγξουμε το πότε θα εμφανίζεται και πότε θα κρύβεται με τις εντολές show() και hide()

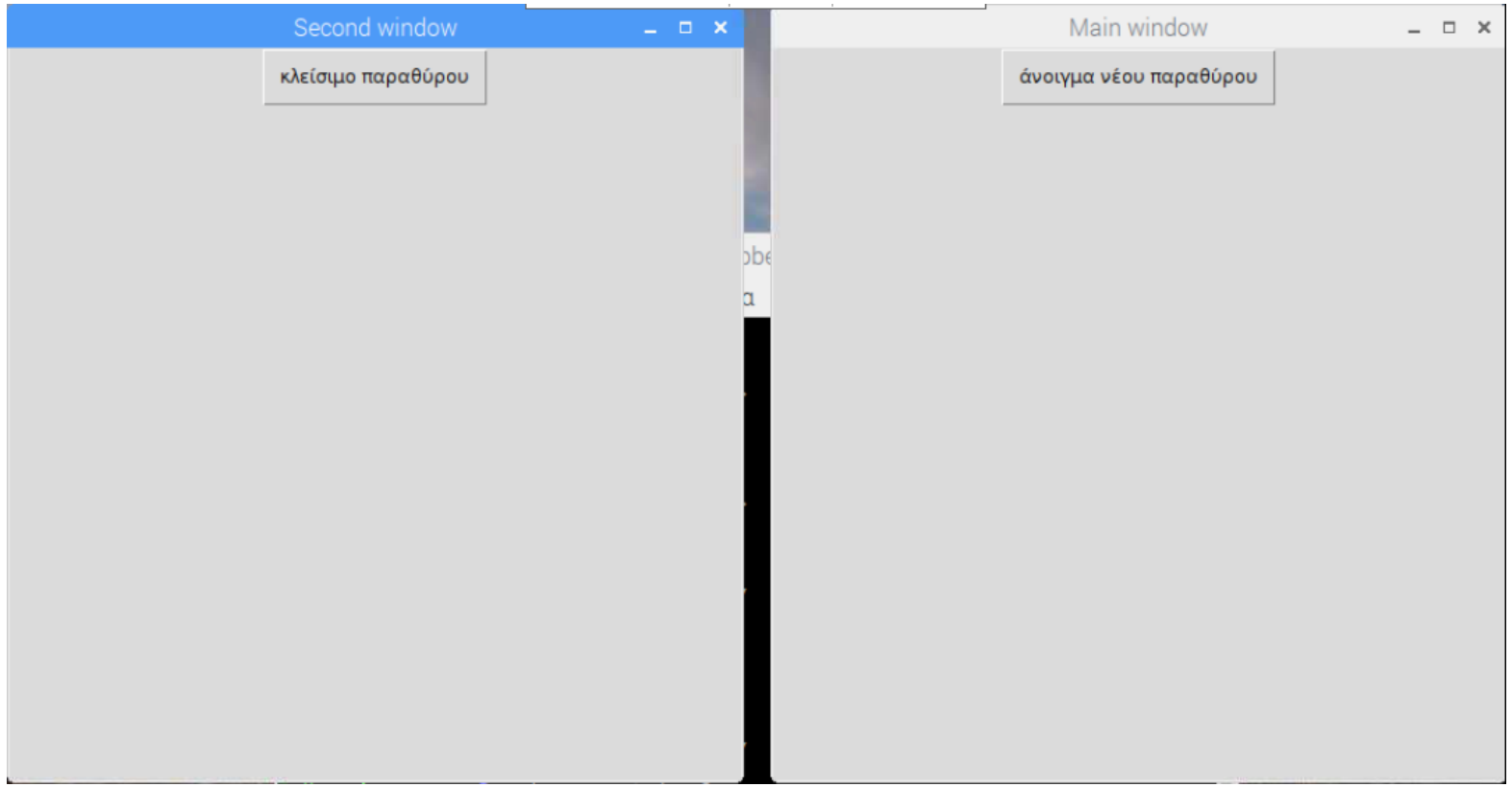
Άνοιγμα επιπλέον παραθύρων

- Δημιουργούμε 2 συναρτήσεις open_window και close_window και χρησιμοποιούμε τις συναρτήσεις show και hide
- Γράφουμε την εντολή για το άνοιγμα του παραθύρου window και κάνουμε hide απευθείας
- Δημιουργούμε το κουμπί open_button και close_button και ορίζουμε στον constructor σε ποιο παράθυρο ανήκει, τι text γράφει και ποια εντολή εκτελεί



Άνοιγμα επιπλέον παραθύρων

- Με το που ξεκινήσει η εφαρμογή εμφανίζεται μόνο το "main window"
- Με το πάτημα του κουμπιού "άνοιγμα νέου παραθύρου" μας ανοίγει το "Second window"
- Με το πάτημα του κουμπιού "κλείσιμο παραθύρου" κλείνει το παράθυρο "Second window"



Βιβλιογραφία

[1] https://www.digikey.com/en/maker/blogs/2018/the-best-gui-widgets-for-raspberry-pi

[2] https://projects.raspberrypi.org/

[3] https://pythonpyqt.com/pyqt-button/

[4] https://python-can.readthedocs.io/en/master/listeners.html