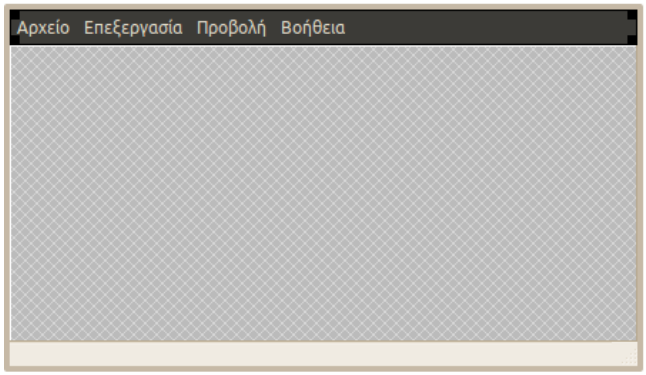
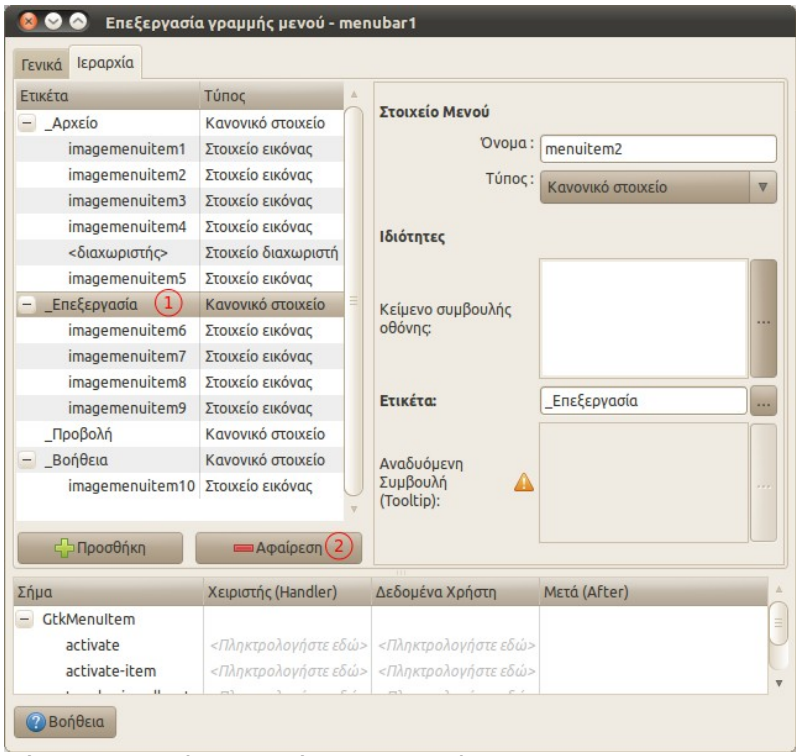
Σε αυτή την σειρά άρθρων θα δείξουμε ορισμένα νέα αντικείμενα του GTK και πως μπορούμε να τα δημιουργήσουμε και να τα παραμετροποιήσουμε μέσα από το Glade. Για όσους δεν έχουν εμπειρία με το Glade καλό είναι να ξεκινήσουν από την πρώτη σειρά άρθρων εδώ.

Θα φτιάξουμε το περιβάλλον χρήστη για ένα πρόγραμμα όπου θα μετατρέπει όλα τα αρχεία κειμένου που θα επιλέξει ο χρήστης σε κωδικοποίηση της αρεσκείας του. Θα μπορεί να μετατρέψει από και προς τα εξής συστήματα: utf-8, utf-16, windows-1253 και iso8859-7. Θα του δίνεται επίσης η δυνατότητα να αλλάξει το τέλος γραμμής των αρχείων μεταξύ των τριών που υπάρχουν σήμερα: \n,\r, \r\n.

Η δυνατότητα αυτή υπάρχει ενσωματωμένη τώρα πια σε όλους τους κειμενογράφους τόσο στα Linux και OSX όσο και στα Windows, αλλά το πρόγραμμα μας θα μπορεί να μετατρέψει πολλά αρχεία με διάφορες κωδικοποιήσεις με μια επιλογή μόνο (bulk ή αλλιώς "σωρηδόν"), και εκτός των άλλων χρησιμεύει για εκπαιδευτικούς σκοπούς τόσο στα Glade και GTK όσο και στην Python.

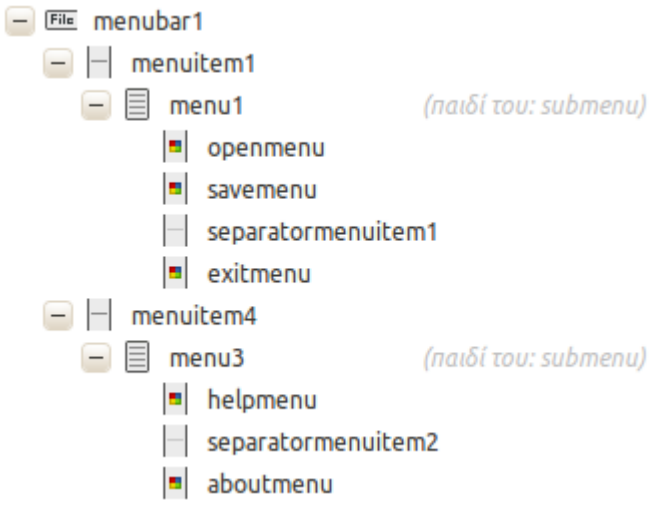


Ξεκινάμε λοιπόν με το Glade και δημιουργούμε το κύριο παράθυρο. *Όνομα*: convertwindow, *τίτλος*: Convert text files, αν θέλουμε *Εικονίδιο* δείνουμε το όνομα του (εμείς έχουμε το process.gif). Στο σήμα destroy του παραθύρου δίνουμε για handler close_window. Δώστε και 550 στην *Αίτηση πλάτους.* Στην συνέχεια προσθέτουμε ένα Vbox 3 γραμμών. Για την πρώτη γραμμή διαλέγουμε από τα Πλαίσια (containers) την "*Γραμμή μενού*" και για την τρίτη από τα Έλεγχος και Εμφάνιση την "*Γραμμή κατάστασης*". Θα ζητήσει πόσα αντικείμενα θέλουμε στην γραμμή κατάστασης, διαλέγουμε ένα. Την ονομάζουμε statusbar και δίνουμε διάκενο 5.



Στο μενού έχει βάλει από μόνο του ορισμένες έτοιμες επιλογές για όλες τις χρήσεις. Θα το αλλάξουμε λίγο, δεν τις χρειαζόμαστε όλες. Με το menubar1 επιλεγμένο πατάμε στο κουμπί Edit της μπάρας εργαλείων. Αυτό ανοίγει ένα νέο παράθυρο όπου μπορούμε να προσθαφαιρέσουμε επιλογές στο μενού. Επιλέγουμε το tab *Ιεραρχία* για να δούμε την δομή του μενού. Κάνοντας δεξί κλικ σε όποια επιλογή θέλουμε μπορούμε να προσθέσουμε μία νέα, θυγατρική ή όχι. Αντίστοιχα με το κουμπί Αφαίρεση αφαιρούμε όποια θέλουμε. Μπορούμε ακόμα να τις μετακινήσουμε με το ποντίκι και να τους αλλάξουμε σειρά και επίπεδο. Διαλέγουμε την *_Επεξεργασία* και πατάμε *Αφαίρεση*, το ίδιο και για την *Προβολή*. Δεν χρειαζόμαστε επίσης τα imagemenuitem1 το οποίο έχει εικόνα του (gtk-new), το imagemenuitem4 (gtk-save-as). Προσθέτουμε ένα διαχωριστικό πάνω από το imagemenuitem10 και ένα νέο θυγατρικό στοιχείο εικόνας πάνω από το διαχωριστικό. Το ονομάζουμε helpmenu, διαλέγουμε σαν έτοιμο στοιχείο stock το "Βοήθεια" (gtk-help) και στο σήμα του *activate* δίνουμε handler: helpmenu_clicked. Το imagemenuitem10 το ονομάζουμε aboutmenu και στο *activate* βάζουμε aboutmenu_clicked. Με το ίδιο τρόπο έχουμε: imagemenuitem5 σε exitmenu και

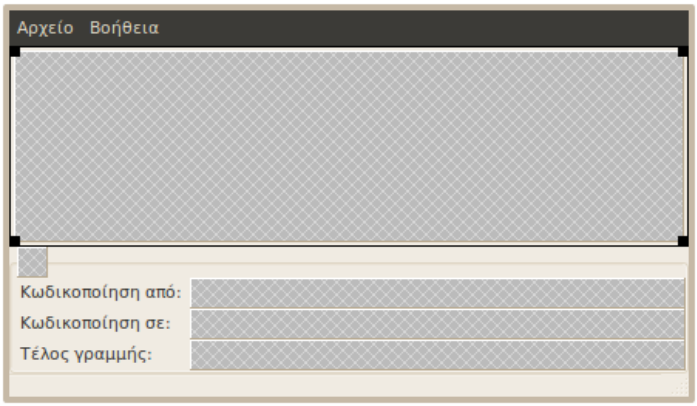
close_window (το ίδιο με του παραθύρου), imagemenuitem2 σε openmenu και openmenu_clicked, imagemenuitem3 σε savemenu και savemenu_clicked. Τώρα το μενού έχει αυτή την διάταξη:



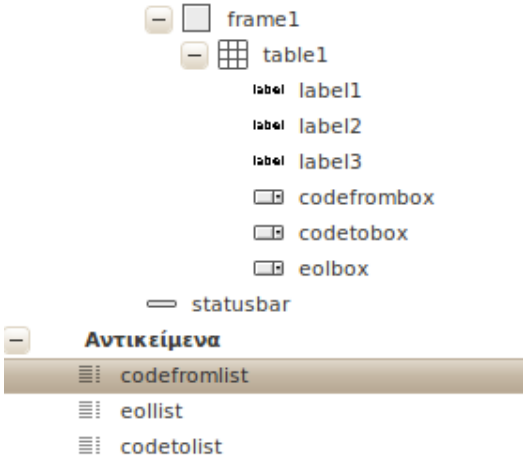
Την μεσαία θέση την χωρίζουμε σε δύο τμήματα με ένα Vbox. Στο πάνω τμήμα βάζουμε ένα *scrollwindow* και στην κάτω ένα *frame*. Αλλάζουμε και στα δύο το χαρακτηριστικό τους *Σκίαση* σε Μέσα. Σβήνουμε τα alignment1 και label1 από το frame. Αλλάζουμε τα *Ανάπτυξη* και *Γέμισμα* του frame στο P*acking* tab σε Όχι, δίνουμε 5 σαν τιμή στο αριθμητικό *Γέμισμα*.

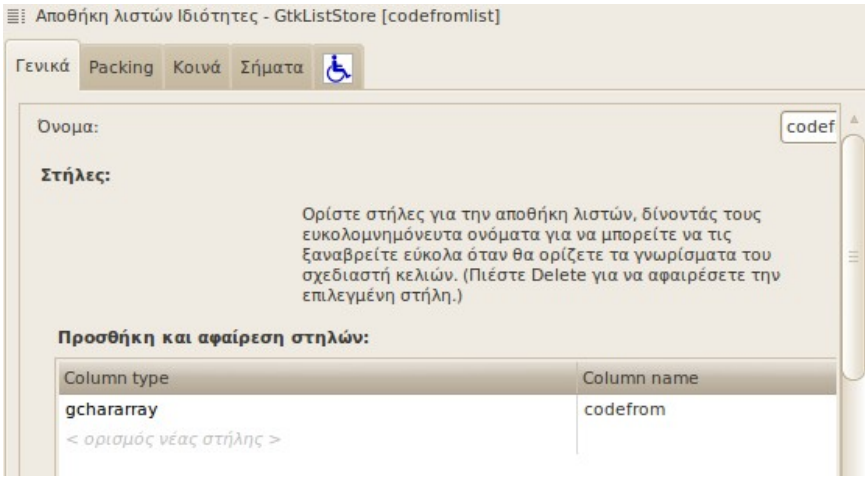
Στην συνέχεια προσθέτουμε έναν *Πίνακα* τριών γραμμών και 2 στηλών στο frame. Ο πίνακας έχει αντικαταστήσει το οριζόντιο box στις καινούργιες εκδόσεις του Glade και είναι αρκετά πιο χρηστικός από το να προσθέτουμε συνέχεια μία οριζόντια και μία κάθετα box.

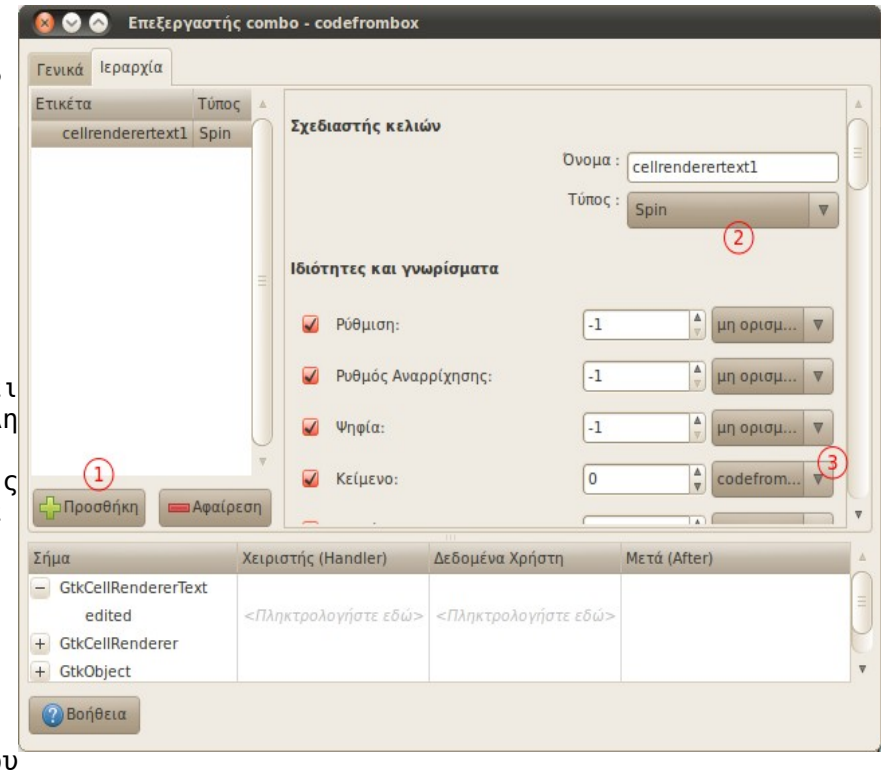
Στη πρώτη στήλη του πίνακα βάζουμε τρία label. Και στα τρία αλλάζουμε την *Στοίχιση Χ* σε 0 (0 σημαίνει αριστερή στοίχιση, 0,50 είναι κέντρο και 1,00 δεξιά στοίχιση) στο *Packing* αλλάζουμε στις *Οριζόντιες επιλογές* το *Επέκταση* σε όχι (unchecked) και δίνουμε στο *Οριζόντιο γέμισμα* τιμή 5. Αλλάζουμε τις ετικέτες των label σε, από πάνω προς τα κάτω: Κωδικοποίηση από, Κωδικοποίηση σε, Τέλος γραμμής.



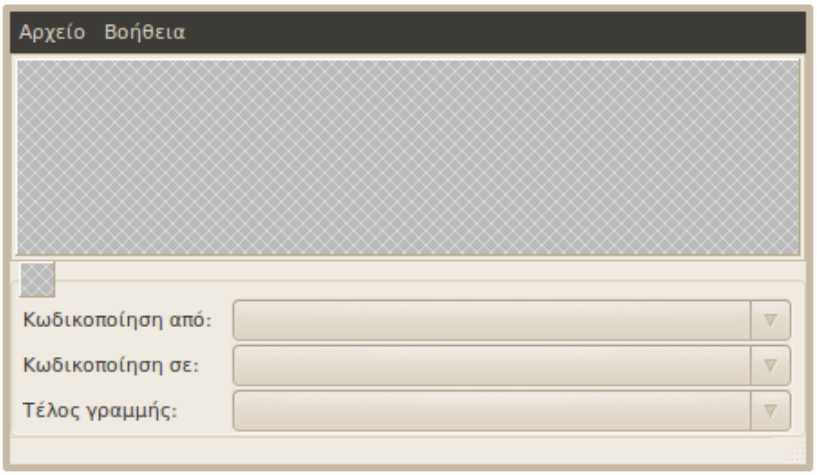
Στην δεύτερη στήλη προσθέτουμε τρία combobox. Δίνουμε και στα τρία την τιμή 5 για το *Οριζόντιο γέμισμα.* Ονομάζουμε το πρώτο codefrombox, το δεύτερο codetobox και το τρίτο eolbox. Για να μπορέσουμε να δώσουμε τιμές στα combobox πρέπει το καθένα να έχει ένα *Μοντέλο.* Διαλέγουμε από αριστερά από το τμήμα *Μοντέλο δέντρου* την *Αποθήκη λιστών(listsource).* Προσθέτουμε τρεις τέτοιες λίστες για τα τρία combo μας και τις ονομάζουμε αντίστοιχα codefromlist, codetolist, eollist. Στην κάθε λίστα πρέπει να ορίσουμε τι και πόσα δεδομένα θα εμφανίζει. Επειδή το ίδιο αντικείμενο χρησιμοποιείται και για την προβολή δέντρου (treeview) το οποίο εμφανίζει πολλές στήλες, μας δίνει την δυνατότητα να προσθέσουμε πολλές στήλες. Εμείς εδώ θέλουμε μόνο μία, να είναι gchararray (δηλαδή string) και θα δώσουμε όνομα σχετικό με την χρήση, όπως codefrom στο codefromlist, codeto στο codetolist και eol στο eolbox. Προχωρόντας πιο κάτω στο tab *Γενικά* συναντάμε την δυνατότητα να γεμίσουμε με τιμές τις λίστες. Εμείς εδώ θα τις γεμίσουμε μέσα από το πρόγραμμα. Τώρα, σε καθένα από τα combo επιλέγουμε στο tab *Γενικά* το αντίστοιχο *Μοντέλο* του. Στην συνέχεια, πατώντας το κουμπί Edit στην γραμμή εργαλείων εμφανίζει ένα καινούργιο παράθυρο επεξεργασίας του τρέχοντος combo. Στο tab *Ιεραρχία* θα δώσουμε τα χαρακτηριστικά κάθε στήλης της λίστας. Το βασικότερο από όλα είναι το τι θέλουμε να εμφανίζει για αυτή την στήλη. Μπορεί κάθε στήλη της λίστας να εμφανίζει διαφορετικό αντικείμενο, άλλη να εμφανίζει text, άλλη γραφικό, άλλη combobox. Εμείς εδώ έχουμε combo και θέλουμε να μπορεί ο χρήστης να επιλέγει από τις διάφορες επιλογές. Αυτή η διαδικασία ονομάζεται spin για το GTK. Έτσι πατάμε *Προσθήκη* και αμέσως βάζει μια γραμμή cellrendertext1, και επιλέγουμε Spin στον *Τύπο* του *Σχεδιαστή κελιών.* Στην γραμμή *Κείμενο* επιλέγουμε το πεδίο που θέλουμε να εμφανίζεται, εδώ έχουμε μόνο ένα πεδίο, επιλέγουμε αυτό. Αυτή η διαδικασία







επαναλαμβάνεται και για τα τρία combo που έχουμε.
Το παράθυρο μας έχει τώρα την εξής μορφή:

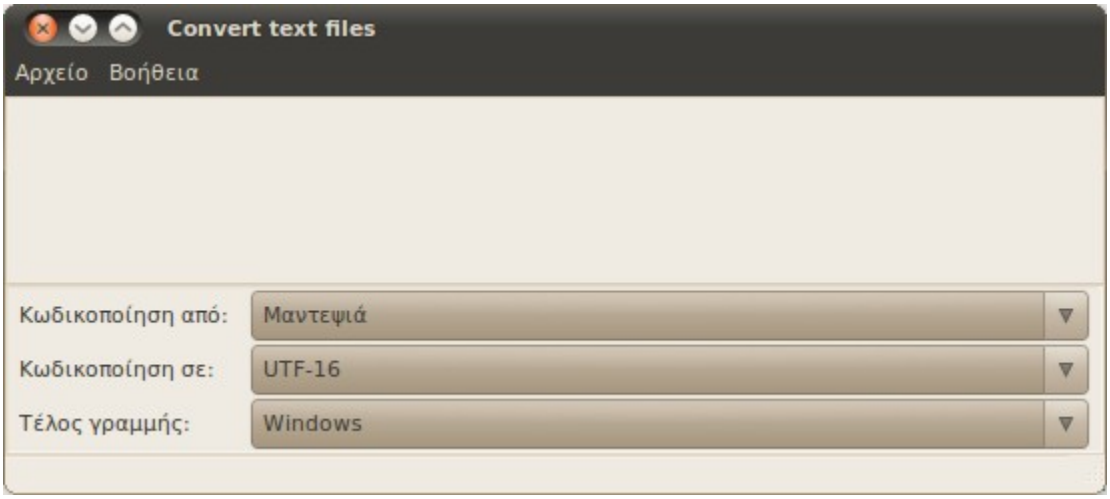


Το σώνουμε σαν convert.glade και πάμε να φτιάξουμε ένα προγραμματάκι σε python για να τεστάρουμε όσα έχουμε κάνει μέχρι τώρα.

```
import gtk

CODEPAGES = ['UTF-16','UTF-8','Windows-1253','ISO8859-7']
EOL = ['Windows','Unix','Mac']

class ConvertFiles(object):

    def close_window(self, widget, data=None):
        gtk.main_quit()

    def __init__(self):
        builder = gtk.Builder()
        builder.add_from_file("convert.glade")
        builder.connect_signals(self)
        self.window = builder.get_object("convertwindow")
        self.codefrombox = builder.get_object("codefrombox")
        self.codetobox = builder.get_object("codetobox")
        self.eolbox = builder.get_object("eolbox")
        self.codefromlist = builder.get_object("codefromlist")
        self.codetolist = builder.get_object("codetolist")
        self.eollist = builder.get_object("eollist")
        self.codefromlist.append([u"Μαντεψιά"])
        for item in CODEPAGES:
            self.codefromlist.append([item])
            self.codetolist.append([item])
        for item in EOL:
            self.eollist.append([item])
        self.codefrombox.set_active(0)
        self.codetobox.set_active(0)
        self.eolbox.set_active(0)

    def main(self):
        self.window.show()
        gtk.main()

if __name__== "__main__":
    app=ConvertFiles()
    app.main()
```

Φορτώνουμε το αρχείο Glade με τον builder και ορίζουμε τις δικές μας μεταβλητές για τα αντικείμενα μας. Μετά, με append σε κάθε λίστα την γεμίζουμε με τους πίνακες μας. Στην λίστα codefromlist βάζουμε μία παραπάνω επιλογή την “Μαντεψιά”.

Το πρόγραμμα, όταν επιλέξουμε αρχεία προς μετατροπή, θα τα ανοίγει και θα προσπαθεί να βρει μόνο του σε τι κωδικοποίηση είναι. Αυτή η “μαντεψιά” δεν είναι πάντοτε σωστή, για λόγους που θα αναφέρουμε αργότερα, οπότε δίνουμε την δυνατότητα στον χρήστη να επιλέξει αυτός την κωδικοποίηση “από”.
Μετά το γέμισμα των λιστών κάνουμε ενεργό το πρώτο στοιχείο κάθε combo (πρώτο είναι το στοιχείο 0) για να μην είναι άδεια τελείως όταν ξεκινάει το πρόγραμμα. Θα παρατηρήσετε ότι στην append το στοιχείο προς πρόσθεση είναι μέσα σε αγκύλες [], αυτό γιατί η append του liststore περιμένει σαν όρισμα πίνακα ή tuple, αλλιώς βαράει λάθος κατά το τρέξιμο.
Αν όλα έχουν πάει καλά θα δείτε ένα τέτοιο παράθυρο να εμφανίζεται (αγνοήστε τα λάθη που εμφανίζει για τις μεθόδους που δεν έχουμε ορίσει ακόμα):

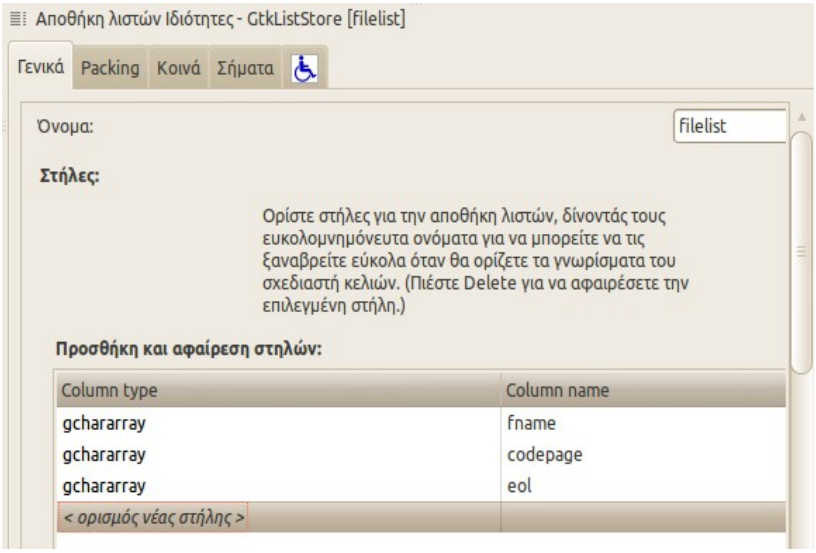


Το μενού μας είναι λειτουργικό, ανοίγουν δηλαδή τα υπομενού, βέβαια μόνο η επιλογή Έξοδος κάνει κάτι. Τα combo μας ανοίγουν και ο χρήστης επιλέγει. Κατά την αλλαγή του μεγέθους του παραθύρου πρέπει όλα να παραμένουν στο ίδιο ύψος, εκτός από τα combo που μεγαλώνουν μόνο κατά πλάτος και το μεσαία κενό μας το οποίο μεγαλώνει και στις δύο διαστάσεις.

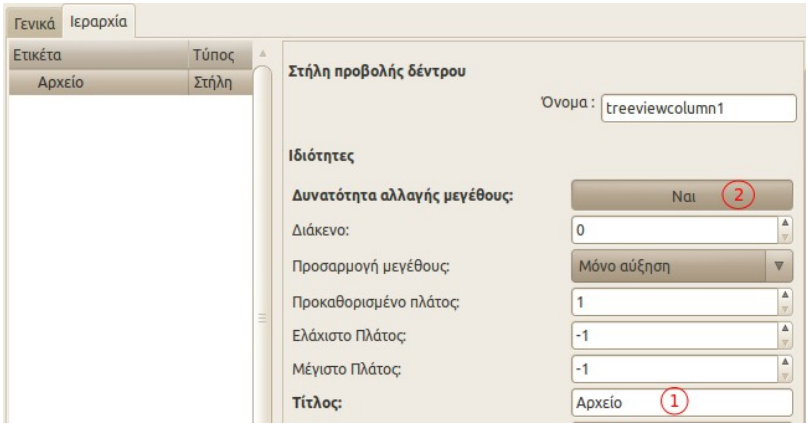
Στο επόμενο άρθρο θα βάλουμε στο κενό ένα treeview όπου θα παρουσιάζονται τα αρχεία που έχει επιλέξει ο χρήστης, το όνομά τους, η κωδικοποίηση τους και το τέλος γραμμής τους.

Σε αυτό το άρθρο θα συνεχίσουμε την κατασκευή του κεντρικού παραθύρου για το πρόγραμμα μας και θα προσθέσουμε ορισμένα βοηθητικά για να δούμε την χρήση τους. Στο τρίτο μέρος θα παρουσιάσουμε το πρόγραμμα και την σύνδεση τον αντικειμένων του Glade με το πρόγραμμα μας.

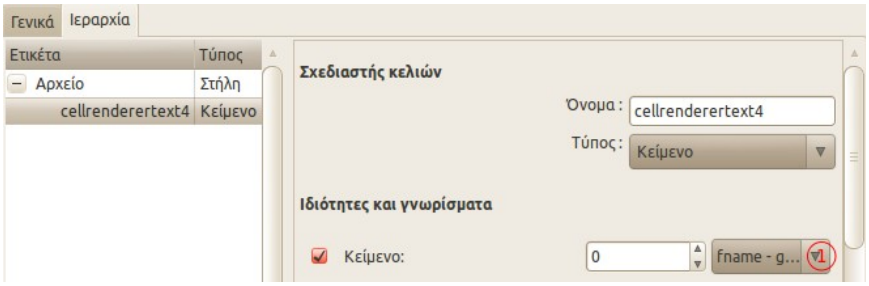
Στο κεντρικό κενό του παραθύρου μας θέλουμε να προσθέσουμε μια *Προβολή Δέντρου* (TreeView). Το TreeView χρειάζεται ένα μοντέλο για να δείχνει τα δεδομένα του. Με τον ίδιο τρόπο που προσθέσαμε τα τρία ListStore στο προηγούμενο άρθρο για τα τρία combobox μας, πάμε να προσθέσουμε μια νέα *Αποθήκη Λιστών* (ListStore). Την ονομάζουμε filelist και προσθέτουμε τρεις στήλες τύπου gchararray με ονόματα: fname, codepage, eol. Μπορούμε τώρα να προσθέσουμε το TreeView και όταν μας ζητήσει πιο μοντέλο θέλουμε θα επιλέξουμε το filelist. Αφού προστεθεί κάτω από το scrollwindow, αλλάζουμε το όνομα του σε filelistview.



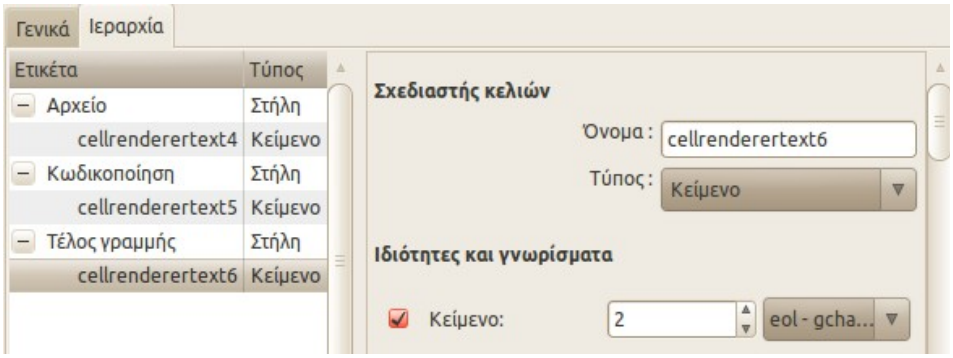
Μετά από αυτό πρέπει να ορίσουμε τι και πως θα εμφανίζει αυτό το TreeView. Πατάμε στο κουμπί *Edit* στην μπάρα εργαλείων, στο νέο παράθυρο που ανοίγει επιλέγουμε το tab *Ιεραρχία*. Εδώ θα ορίσουμε πως και τι θέλουμε να εμφανίσουμε από το μοντέλο μας. Πατάμε *Προσθήκη* και θα βάλει μια νέα στήλη. Αλλάζουμε τον *Τίτλο* σε Αρχείο και κάνουμε την *Δυνατότητα αλλαγής μεγέθους* Ναι.



Στην συνέχεια κάνουμε δεξί κλικ πάνω στην γραμμή και διαλέγουμε το *Προσθήκη θυγατρικού Κειμένου αντικειμένου.* Αυτό θα βάλει μια νέα γραμμή από κάτω με όνομα cellrendertext. Εδώ αλλάζουμε μόνο το *Κείμενο* και επιλέγουμε το fname από το μοντέλο μας. Ακριβώς τις ίδιες κινήσεις κάνουμε και για τις υπόλοιπες δύο στήλες που θέλουμε στο TreeView.

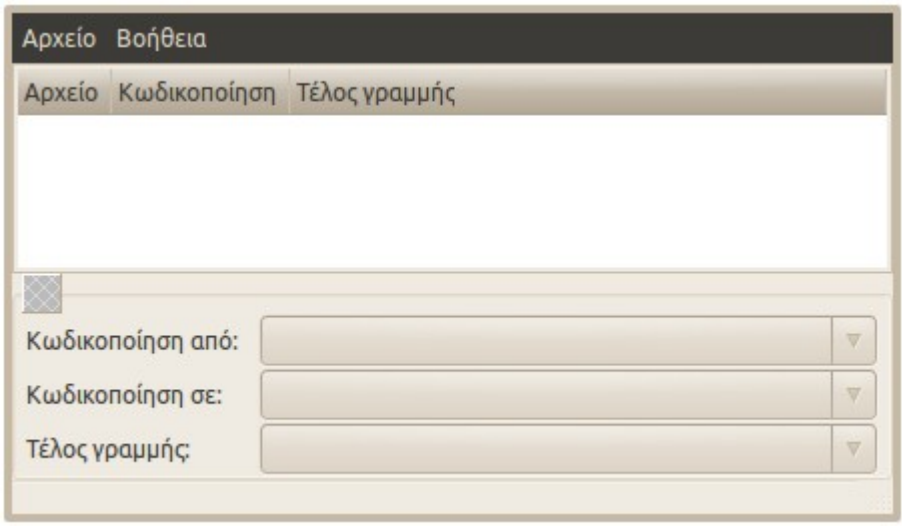


Τίτλος->Κωδικοποίηση, *Δυνατότητα αλλαγής μεγέθους*->Ναι, cellrendertext->Κείμενο->codepage.
Τίτλος->Τέλος γραμμής, Δυνατότητα αλλαγής μεγέθους->Ναι, cellrendertext->Κείμενο->eol.
Τώρα θα πρέπει να έχει την εξής μορφή η λίστα στην *Ιεραρχία:*

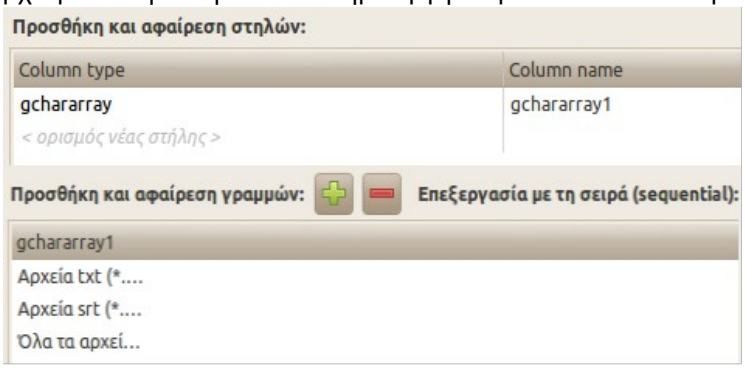


Το filelistview θα γεμίζει όταν ο χρήστης επιλέξει αρχεία προς μετατροπή. Θα πρέπει να προβλέψουμε τι δυνατότητα του χρήστη να αφαιρέσει ένα αρχείο από την λίστα. Θα μπορεί μαρκάροντας ένα αρχείο και πατώντας Delete να αφαιρείται το αρχείο από την λίστα. Χρειαζόμαστε οπότε μια μέθοδο που επιβλέπει αν πατήθηκε κουμπί. Στο tab *Σήματα* του filelistview στο GtkWidget και στο σήμα *key-press-event* δίνουμε σαν handler το *filelistview-key.* Σε αυτή θα αφαιρούμε την αντίστοιχη εγγραφή από το filelist. Θα έχει όμως το πρόγραμμα και έναν δικό του πίνακα με τα αρχεία, οπότε όταν αφαιρεθεί ένα από το filelist θα πρέπει να αφαιρέσουμε και από τον δικό μας. Για να γίνει αυτό χρειαζόμαστε να ελέγχουμε το σήμα *row_deleted*. Το επιλέγουμε έτσι από το tab *Σήματα* και δίνουμε σαν handler την μέθοδο filelist_row_deleted.

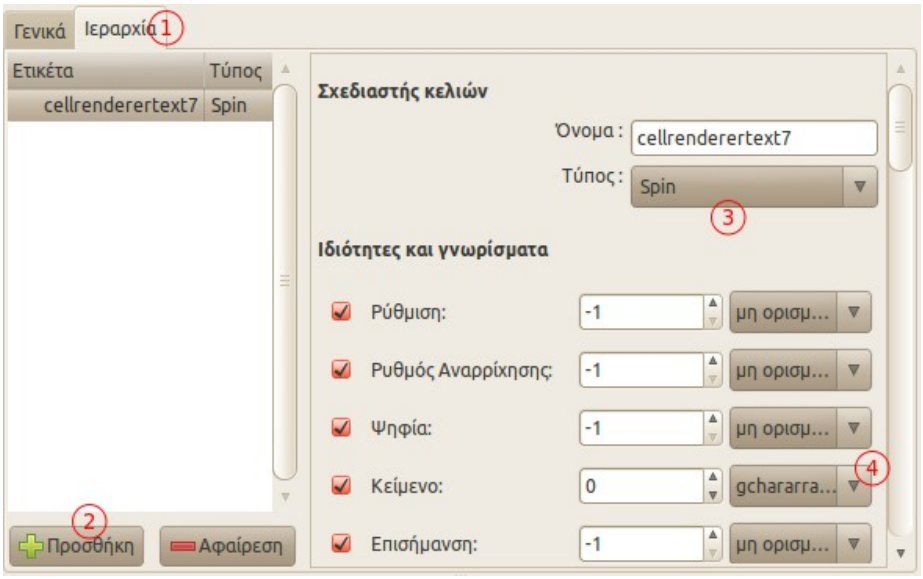
Το κεντρικό μας παράθυρο είναι τώρα έτοιμο. Έχει όλα τα σήματα και τα αντικείμενα που χρειαζόμαστε.



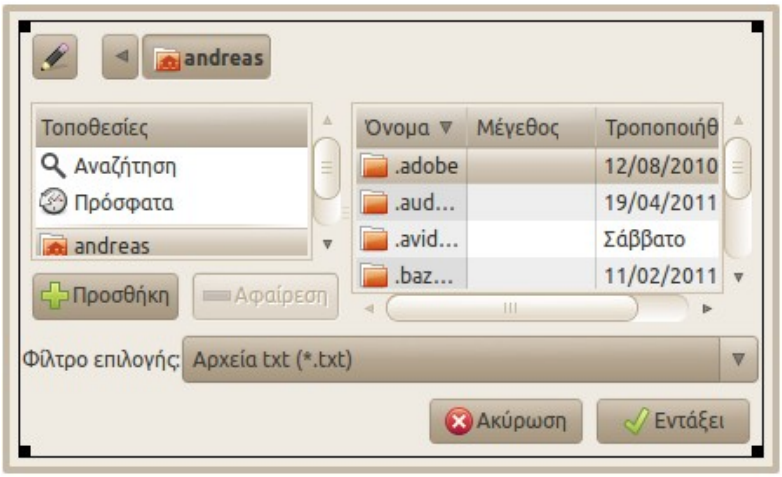
Για την επιλογή των αρχείων δημιουργούμε ένα filechooser, όπως στο προηγούμενο εκπαιδευτικό. Του δίνουμε όνομα filechooser, *Τιτλος παραθύρου*->Επιλέξτε αρχεία, *Σχηματικό*->Ναι, *Καταστροφή με το μητρικό*->Ναι, *Επιλογή πολλαπλών*->Ναι, *Να επιτρέπεται η δημιουργία φακέλων*->Οχι. Στο σήμα του delet-event δίνουμε σαν handler την delet_chooser (αναφερθήκαμε σ' αυτό στο προηγούμενο tutorial). Προσθέτουμε τα δύο κουμπιά, *Ακύρωση* και *Εντάξει* και τα αντίστοιχα σήματα τους στο *clicked,* filechooser_cancel και filechooser_ok. Πάνω ακριβώς από τις θέσεις των κουμπιών υπάρχει μια κενή θέση. Εκεί θα δημιουργήσουμε ένα combobox με φίλτρα αρχείων για να επιλέξει ο χρήστης πια θέλει να του δείχνει προς επιλογή. Θα βάλουμε ένα Hbox 2 στοιχείων, αλλάζουμε τα *Γεμισμα* και *Ανάπτυξη* σε Όχι. Στην αριστερή θέση βάζουμε ένα label και κάνουμε την *ετικέτα* του “Φίλτρο επιλογής”. Αλλάζουμε και τα *Γέμισμα* και *Ανάπτυξη σε Όχι.* Πριν βάλουμε το combobox να φτιάξουμε μια νέα *Αποθήκη λιστών* (liststore) και να της δώσουμε όνομα filterlist. Βάζουμε μια στήλη gchararray, το όνομα δεν είναι σημαντικό, και προσθέτουμε τρεις επιλογές: Αρχεία txt (*txt), Αρχεία srt (*.srt), Όλα τα αρχεία (*.*). Τρεις



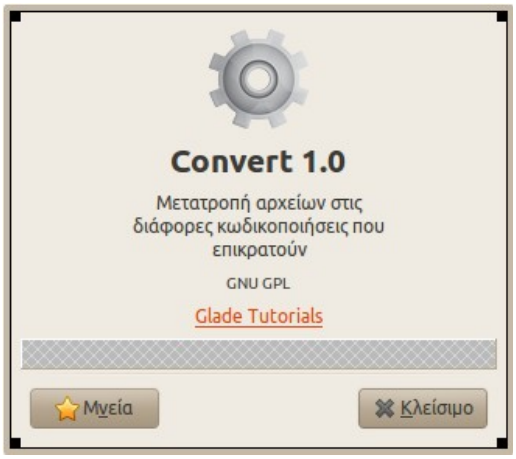
επιλογές όπου μπορεί να είναι χρήσιμη η μετατροπή αρχείων. Μπορείτε εσείς να προσθαφαιρέσετε ανάλογα. Προσθέτουμε τώρα το combobox στην δεξιά θέση, διαλέγουμε σαν μοντέλο του το filterlist, το ονομάζουμε filterbox και κάνουμε 0 (μηδέν) το *Ενεργό αντικείμενο* του. Στα *Σήματα* βάζουμε σαν handler στο changed την filterbox_changed. Στην συνέχεια πρέπει να ορίσουμε το spin του combobox, όπως και στα προηγούμενα. Πατάμε *Edit* στην γραμμή εργαλείων, στο tab *Ιεραρχία* πατάμε Προσθήκη και αλλάζουμε σε spin των *Τύπο*, και τέλος διαλέγουμε για *Κείμενο* το πεδίο gchararray1 της στήλης μας. Θα χρειαστούμε βέβαια και τρία *filefilter* για να μπορέσουμε να δώσουμε τις αντίστοιχες τιμές. Βάλτε λοιπόν τρία με ονόματα txtfilter, srtfilter, allfilter.



Το filechooser έχει τώρα της εξής μορφή:



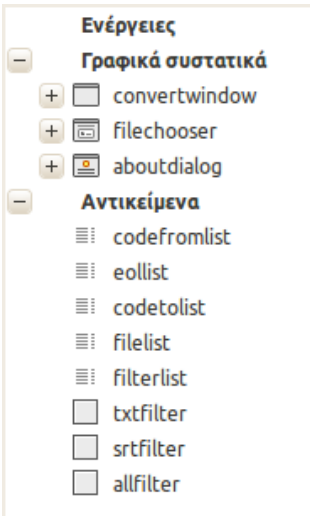
Ένα δεύτερο βοηθητικό παράθυρο που θα δημιουργήσουμε είναι το *Περί...* Βρίσκεται σε σχεδόν σε όλα τα προγράμματα σαν τελευταία επιλογή στο μενού *Βοήθεια* και εμφανίζει γενικές πληροφορίες για το πρόγραμμα και τους δημιουργούς του. Το GTK παρέχει ένα έτοιμο αντικείμενο για αυτό το σκοπό, το aboutdialog. Προσθέτουμε ένα λοιπόν ένα από τα *Ανώτατα επίπεδα.* Η ελληνική μετάφραση είναι *Διάλογος πληροφοριών.* Θα το ονομάσουμε aboutdialog. Στα πεδία που υπάρχουν στο tab *Γενικά* μπορούμε να συμπληρώσουμε ότι θέλουμε. Στο *Όνομα προγράμματος* δίνουμε Convert για να είμαστε συνεπής και από εκεί και πέρα μπορείτε να συμπληρώσετε όσα θέλετε για να δείτε πως επεκτείνετε μόνο του. Για παράδειγμα αν συμπληρώσετε *Συγγραφείς* θα προστεθεί αυτόματα ένα νέο κουμπί το *Μνεία.* Στο συγκεκριμένο έχουμε προσθέσει ένα γραφικό, σχόλια και συγγραφείς.



Για την *Βοήθεια* μπορούμε να δημιουργήσουμε ένα νέο παράθυρο με ένα textview αντικείμενο και να προβάλουμε πληροφορίες χρήσης του προγράμματος. Υπάρχει όμως ένα μειονέκτημα στο textview. Δεν μπορείς να διαμορφώσεις το κείμενο σου με συνδέσεις ή με έντονους ή πλάγιους χαρακτήρες. Θα έχει μια πολύ παλη μορφή χωρίς περιεχόμενα και αν η βοήθεια που χρειάζεται να γραφτεί είναι μεγάλη, θα είναι πολύ δύσχρηστο. Για αυτό προτείνουμε την δημιουργία ενός html αρχείου, όπου μπορούμε να το διαμορφώσουμε όπως θέλουμε, με ευρετήριο, ωραίο κείμενο και εικόνες, και προβολή του μέσω του εξ ορισμού browser του συστήματος. Στην python αυτό γίνεται πολύ εύκολα με τις παρακάτω γραμμές κώδικά:

```
import webbrowser
webbrowser.open("help.html")
```

Αυτές οι δυο γραμμές κώδικά θα ανοίξουν το αρχείο help.html στον εξ ορισμού browser του συστήματος.



Όλα τα αντικείμενα μας είναι τώρα τα εξής:

Μπορούμε με το παρακάτω πρόγραμμα να κάνουμε ένα έλεγχο για το πως εμφανίζονται όλα τα αντικείμενα μας πριν προχωρήσουμε στην δημιουργία του τελικού προγράμματος.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import gtk

CODEPAGES = ['UTF-16','UTF-8','Windows-1253','ISO8859-7']
EOL = ['Windows','Unix','Mac']

class ConvertFiles(object):

    def close_window(self, widget, data=None):
        gtk.main_quit()

    def delete_chooser(self, widget, data=None):
        return True

    def aboutmenu_clicked(self, widget, data=None):
        self.aboutdialog.run()
        self.aboutdialog.hide()

    def helpmenu_clicked(self, widget, data=None):
        import webbrowser
        webbrowser.open("help.html")

    def openmenu_clicked(self, widget, data=None):
        self.filechooser.show()

    def filterbox_changed(self, widget, data=None):
        if widget.get_active() == 0:
            self.filechooser.set_filter(self.txtfilter)
        if widget.get_active() == 1:
            self.filechooser.set_filter(self.srtfilter)
        if widget.get_active() == 2:
            self.filechooser.set_filter(self.allfilter)

    def filechooser_cancel(self, widget, data=None):
        self.filechooser.hide()

    def filechooser_ok(self, widget, data=None):
        self.filechooser.hide()

    def savemenu_clicked(self, widget, data=None):
        pass

    def __init__(self):
        builder = gtk.Builder()
```

```
        builder.add_from_file("convert.glade")
        builder.connect_signals(self)
        self.window = builder.get_object("convertwindow")
        self.codefrombox = builder.get_object("codefrombox")
        self.codetobox = builder.get_object("codetobox")
        self.eolbox = builder.get_object("eolbox")
        self.codefromlist = builder.get_object("codefromlist")
        self.codetolist = builder.get_object("codetolist")
        self.eollist = builder.get_object("eollist")
        self.codefromlist.append([u"Μαντεψιά"])
        for item in CODEPAGES:
            self.codefromlist.append([item])
            self.codetolist.append([item])
        for item in EOL:
            self.eollist.append([item])
        self.codefrombox.set_active(0)
        self.codetobox.set_active(0)
        self.eolbox.set_active(0)
        self.filechooser = builder.get_object("filechooser")
        self.aboutdialog = builder.get_object("aboutdialog")
        self.txtfilter = builder.get_object("txtfilter")
        self.srtfilter = builder.get_object("srtfilter")
        self.allfilter = builder.get_object("allfilter")
        self.filterbox = builder.get_object("filterbox")
        self.txtfilter.add_pattern('*.txt')
        self.txtfilter.add_pattern('*.TXT')
        self.srtfilter.add_pattern('*.srt')
        self.srtfilter.add_pattern('*.SRT')
        self.allfilter.add_pattern('*.*')
        self.filterbox_changed(self.filterbox)
        self.statusbar=builder.get_object("statusbar")

    def main(self):
        self.window.show()
        gtk.main()

if __name__== "__main__":
    app=ConvertFiles()
    app.main()
```

Αυτό είναι σχεδόν ολοκληρωμένο το πρόγραμμα μας, δεν έχουν αναπτυχθεί μόνο οι openmenu_clicked και savemenu_clicked μέθοδοι. Λείπουν ακόμα οι filelistview_key και filelist_row_deleted. Αυτά θα τα δούμε στο επόμενο άρθρο. Μέχρι εδώ δεν έχει τίποτα καινούργιο εκτός από την filterbox_changed όπου ελέγχει πιο είναι το ενεργό αντικείμενο του combobox και αναλόγως επιλέγει το αντίστοιχο φίλτρο για το filechooser. Την καλούμε και από την __init__ αφού έχουμε γεμίσει τα φίλτρα μας για να ενεργοποιηθεί το φίλτρο στον filechooser.
Η aboutmenu_clicked δείχνει το αντίστοιχο παράθυρο και η helpmenu_clicked ανοίγει τον browser με το αρχείο help.html.

Σε αυτό το άρθρο θα παρουσιάσουμε το πρόγραμμα, γραμμένο σε python, που μετατρέπει τις κωδικοποιήσεις των αρχείων. Το πρόγραμμα καθώς και τα συνοδευτικά του αρχεία βρίσκονται εδώ.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import gtk
import os
import codecs

CODEPAGES = ['utf_16','utf_8','cp1253','iso8859_7']
EOL = ['Windows','Unix','Mac']
DICT_EOL = {'Windows':'\r\n','Unix':'\n','Mac':'\r'}

class FileToConvert(object):

    def __init__(self,fn = None, cp = None, eol=None):
        self.fn = fn
        self.cp = cp
        self.eol = eol
        self.to_cp = ''
        self.to_eol = ''

    def checkfile(self,codefrom=None):
        encodings=CODEPAGES
        if codefrom:
            encodings=[codefrom]
        for enc in encodings:
            t1=codecs.open(self.fn,'r',enc)
            try:
                line=t1.readline()
                if '\r\n' in line:
                    self.eol = EOL[0]
                elif '\n' in line:
                    self.eol = EOL[1]
                elif '\r' in line:
                    self.eol = EOL[2]
                self.cp=enc
                t1.close()
                ret=True
                break
            except:
                ret=False
        return ret

class ConvertFiles(object):

    def close_window(self, widget, data=None):
        gtk.main_quit()

    def delete_chooser(self, widget, data=None):
        return True

    def aboutmenu_clicked(self, widget, data=None):
        self.aboutdialog.run()
        self.aboutdialog.hide()

    def helpmenu_clicked(self, widget, data=None):
```

```
        import webbrowser
        webbrowser.open("help.html")

    def openmenu_clicked(self, widget, data=None):
        self.filechooser.show()

    def filterbox_changed(self, widget, data=None):
        if widget.get_active() == 0:
            self.filechooser.set_filter(self.txtfilter)
        if widget.get_active() == 1:
            self.filechooser.set_filter(self.srtfilter)
        if widget.get_active() == 2:
            self.filechooser.set_filter(self.allfilter)

    def filelistview_key(self, widget, data=None):
        treeviewmodel, treeviewindex = widget.get_selection().get_selected()
        if (gtk.gdk.keyval_name(data.keyval) == 'Delete') and treeviewindex:
            treeviewmodel.remove(treeviewindex)

    def filelist_row_deleted(self,widget,data=None):
        del self.files[data[0]]
        self.statusbar.push(0,"Αφαίρεση αρχείου")

    def filechooser_cancel(self, widget, data=None):
        self.filechooser.hide()

    def filechooser_ok(self, widget, data=None):
        filenames=[]
        if self.files:
            for fi in self.files:
                filenames.append(fi.fn)
        errortxt=''
        i=0
        for fi in self.filechooser.get_filenames():
            if fi not in filenames:
                newfile = FileToConvert(fi)
                if newfile.checkfile():
                    self.files.append(newfile)
                    self.filelist.append([newfile.fn,newfile.cp,newfile.eol])
                    i+=1
                else:
                    errortxt+=u"Πρόβλημα στην ανάγνωση του αρχείου: %s.\n" % cf.fn
                    self.errordialog(errortxt)
        self.filechooser.hide()
        if i>0:
            self.statusbar.push(0,u"Επιτυχής αναγνωση %d αρχείων" % i)
        else:
            self.statusbar.push(0,"")

    def errordialog(self, errortxt):
        dialog=gtk.MessageDialog(parent=None,flags=gtk.DIALOG_MODAL &
gtk.DIALOG_DESTROY_WITH_PARENT, type=gtk.MESSAGE_ERROR, buttons=gtk.BUTTONS_OK)
        dialog.set_title(u"Σφάλμα")
        dialog.set_markup(u"<b>"+errortxt+"</b>")
        response=dialog.run()
        if response == gtk.RESPONSE_DELETE_EVENT or response == gtk.RESPONSE_OK:
          dialog.destroy()

    def savemenu_clicked(self, widget, data=None):
```

```
        errortxt=''
        if not self.files:
            self.errordialog(u"Δεν έχετε επιλέξει αρχεία.\n")
            return
        for fi in self.files:
            if self.codefrombox.get_active()>0:
                if not fi.checkfile(CODEPAGES[self.codefrombox.get_active()-1]):
                    errortxt+=u"Πρόβλημα στην ανάγνωση του αρχείου: %s.\n" % fi.fn
        if errortxt:
            self.errordialog(errortxt)
            self.statusbar.push(0,u"Δεν ήταν σωστή η κωδικοποίηση που ορίσατε.")
            return
        for i in range(len(self.files)):
            self.files[i].to_cp=CODEPAGES[self.codetobox.get_active()]
            self.files[i].to_eol=EOL[self.eolbox.get_active()]
        treeiter=self.filelist.get_iter_first()
        for fi in self.files:
            try:
                t1=codecs.open(fi.fn,'r',fi.cp)
                t2=codecs.open('tmp','w',fi.to_cp)
                for line in t1:
                    line=line.replace(DICT_EOL[fi.eol],DICT_EOL[fi.to_eol])
                    t2.write(line)
                t2.close()
                t1.close()
                os.rename('tmp',fi.fn)
            except:
                self.errordialog(u"Πρόβλημα στην μετατροπή του %s.\n" % fi.fn)
                success=False
            else:
                if fi.checkfile():
                    self.filelist.set_value(treeiter,1,fi.cp)
                    self.filelist.set_value(treeiter,2,fi.eol)
                    fi.to_cp = ''
                    fi.to_eol = ''
                    success=True
                else:
                    self.errordialog(u"Πρόβλημα στην τελική ανάγνωση του %s.\n" % fi.fn)
                    success=False
            treeiter=self.filelist.iter_next(treeiter)
        if success:
            self.statusbar.push(0,u"Επιτυχής μετατροπή")
        else:
            self.statusbar.push(0,u"Προβλήματα στην μετατροπή")

    def __init__(self):
        builder = gtk.Builder()
        builder.add_from_file("convert.glade")
        builder.connect_signals(self)
        self.window = builder.get_object("convertwindow")
        self.codefrombox = builder.get_object("codefrombox")
        self.codetobox = builder.get_object("codetobox")
        self.eolbox = builder.get_object("eolbox")
        self.codefromlist = builder.get_object("codefromlist")
        self.codetolist = builder.get_object("codetolist")
        self.eollist = builder.get_object("eollist")
        self.filelist = builder.get_object("filelist")
        self.codefromlist.append([u"Μαντεψιά"])
        for item in CODEPAGES:
```

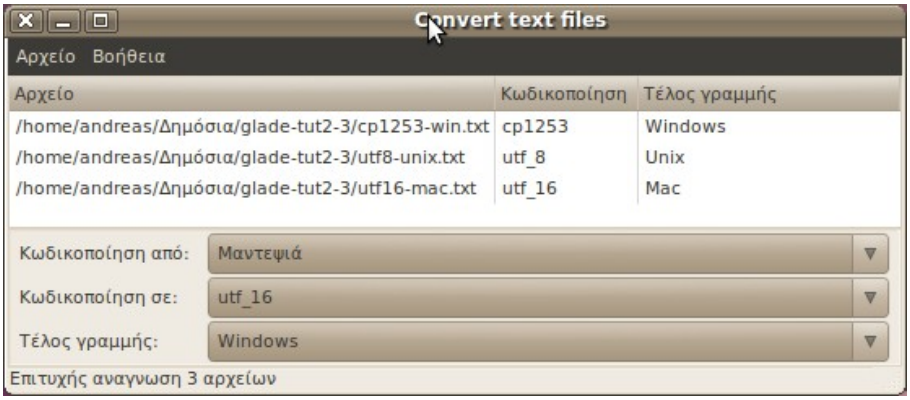
```
            self.codefromlist.append([item])
            self.codetolist.append([item])
        for item in EOL:
            self.eollist.append([item])
        self.codefrombox.set_active(0)
        self.codetobox.set_active(0)
        self.eolbox.set_active(0)
        self.filechooser = builder.get_object("filechooser")
        self.aboutdialog = builder.get_object("aboutdialog")
        self.txtfilter = builder.get_object("txtfilter")
        self.srtfilter = builder.get_object("srtfilter")
        self.allfilter = builder.get_object("allfilter")
        self.filterbox = builder.get_object("filterbox")
        self.txtfilter.add_pattern('*.txt')
        self.txtfilter.add_pattern('*.TXT')
        self.srtfilter.add_pattern('*.srt')
        self.srtfilter.add_pattern('*.SRT')
        self.allfilter.add_pattern('*.*')
        self.filterbox_changed(self.filterbox)
        self.statusbar=builder.get_object("statusbar")
        self.files=[]

    def main(self):
        self.window.show()
        gtk.main()

if __name__== "__main__":
    app=ConvertFiles()
    app.main()
```

Ξεκινάμε στην αρχή με τα απαραίτητα import. Το νέο εδώ είναι η βιβλιοθήκη codecs, η οποία θα μας χρησιμεύσει στις μετατροπές. Στην συνέχεια υπάρχουν τρεις σταθερές. Η πρώτη περιέχει τα λεκτικά των κωδικοποιήσεων έτσι όπως τα χρειάζεται η codecs και με αυτά γεμίζουμε και τα δύο μας combobox. Η EOL σταθερά είναι αυτή που περιέχει τα λεκτικά για τα διάφορα τελειώματα γραμμής που υπάρχουν. Στην επόμενη σταθερά γίνεται συσχέτιση μεταξύ αυτών των λεκτικών και των πραγματικών τιμών που έχει το κάθε τέλος γραμμής.

Θα χρειαστούμε να ορίσουμε ένα αντικείμενο για κάθε αρχείο που θα διαλέξει ο χρήστης κι αυτό θα φτιαχτεί από την κλάση FileToConvert. Το αντικείμενο θα περιέχει τις εξής πληροφορίες: το όνομα του αρχείου, την κωδικοποίηση και το τέλος γραμμής που έχει πριν την μετατροπή καθώς και σε τι θέλουμε να το μετατρέψουμε. Η κλάση ακόμα περιλαμβάνει την μέθοδο checkfile η οποία προσπαθεί να μαντέψει ποια από τις τέσσερις κωδικοποιήσεις και ποιο από τα τρία EOL έχει το αρχείο. Αν είναι επιτυχής επιστρέφει True, αλλιώς False.

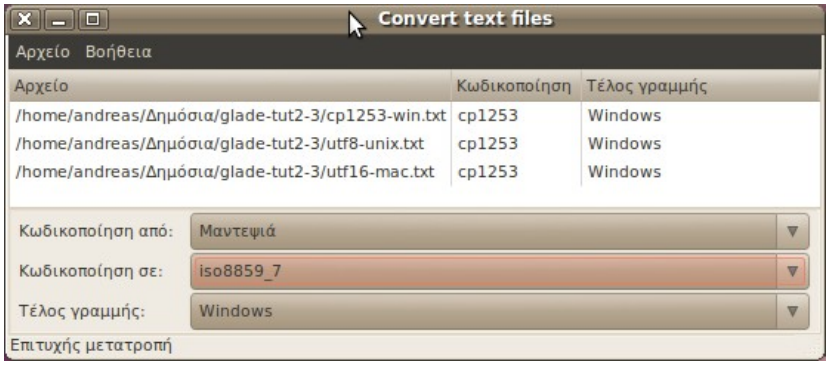
Στην κύρια κλάση έχουμε προσθέσει στην __init__ μέθοδο δύο μεταβλητές, την self.filelist η οποία είναι το μοντέλο του TreeView και την self.files στην οποία θα κρατάμε τα αντικείμενα FilesToConvert.

Η μέθοδος filelistview_key εκτελείται όταν ο χρήστης πατήσει ένα πλήκτο. Η get_selected() επιστρέφει δύο τιμές το μοντέλο στο οποίο ανήκει το TreeView και τον αύξων αριθμό της επιλογής. Έτσι λοιπόν ελέγχουμε αν ο χρήστης έχει πατήσει Delete και αν έχει επιλέξει γραμμή. Τότε μόνο διαγράφουμε την αντίστοιχη επιλογή από το μοντέλο. Όταν διαγραφεί ένα στοιχείο από το μοντέλο εκτελείται η filelist_row_deleted στην οποία διαγράφουμε το αντίστοιχο αντικείμενο από την self.files.



Όταν ο χρήστης επιλέξει αρχεία φτιάχνουμε έναν προσωρινό πίνακα με τα ονόματα των αρχείων που υπάρχουν ήδη έτσι ώστε να αποφύγουμε τις διπλές εισαγωγές. Στην συνέχεια για κάθε αρχείο φτιάχνουμε το αντίστοιχο αντικείμενό του και καλούμε την chekfile για να προσδιορίσει τον τύπο του αρχείου και του τέλους της γραμμής του. Για κάθε επιτυχή αναγνώριση προσθέτουμε τα αντίστοιχα πεδία στην self.filelist για να εμφανιστούν στο TreeView.

Εδώ θα θέλαμε να αναφέρουμε ορισμένα στοιχεία για την codecs.open. Αυτή όταν ανοίγει ένα αρχείο για διάβασμα το ανοίγει πάντα σε binary mode. Αυτό έχει σαν αποτέλεσμα να μην χτυπάει λάθος όταν αρχείο δεν είναι text. Ακόμα επειδή οι λατινικοί χαρακτήρες έχουν την ίδια κωδικοποίηση τόσο σε utf-8 όσο και σε windows-1253 ή ISO8859-7 μπορεί να μην δώσει σωστό αποτέλεσμα σε αρχεία με μόνο τέτοιους χαρακτήρες. Ένα άλλο σημείο προσοχής είναι ότι η μόνη διαφορά, στους ελληνικούς εμφανίσιμους χαρακτήρες, μεταξύ των windows-1253 και ISO8859-7 είναι το άλφα κεφαλαίο τονούμενο (Ά), οπότε πάλι μπορεί να έχουμε λάθος προσδιορισμό. Γι΄ αυτό και βάλαμε την επιλογή “Κωδικοποίηση από” για να μπορεί να επιλέξει ο χρήστης αν γνωρίζει την κωδικοποίηση του αρχείου προς μετατροπή.



Η μετατροπή γίνεται στην savemenu_clicked, ανοίγοντας με την σειρά τα αρχεία που υπάρχουν στην self.files με την κωδικοποίηση που έχει ορίσει ο χρήστης και σώζοντας τα σε ένα προσωρινό με την νέα. Αν όλα πάνε καλά αντικαθιστά το αρχικό με το προσωρινό. Στην συνέχεια τα ξαναδιαβάζει για να ενημερώσει το TreeView και για να δει ο χρήστης το αποτέλεσμα.

Καθ' όλη την διαδικασία της μετατροπής το πρόγραμμα ενημερώνει τον χρήστη με δύο τρόπους. Ο ένας είναι ο διάλογος μηνυμάτων που έχουμε δει και σε προηγούμενο άρθρο και ο άλλος είναι μέσω της statusbar. Η statusbar λειτουργεί σαν στοίβα LIFO. Βάζουμε στην στοίβα αυτό που θέλουμε να εμφανίσει με push και το αφαιρούμε με pop. Κάθε μήνυμα μπορεί να έχει και ένα αναγνωριστικό κωδικό, προσδιορίζοντας έτσι από πιο μέρος του προγράμματος προήλθε. Εδώ δεν χρησιμοποιούμε αυτήν την δυνατότητα γι' αυτό όλα τα μηνύματα έχουν σαν προσδιοριστικό το μηδέν.

Είδαμε σ' αυτό το άρθρο ορισμένα από τα πιο βασικά σύνθετα αντικείμενα του Gnome και του GTK και πως τα αναπτύσσουμε αυτά μέσα από το Glade. Βλέπουμε ότι ενώ αυτά είναι αρκετά πιο πολύπλοκα από ένα απλό label ή ένα text input, ωστόσο όλη η διαδικασία δημιουργίας και ελέγχου τους ολοκληρώνεται μέσα από το Glade προσθέτοντας ελάχιστο προγραμματιστικό βάρος.