15. ΓΡΑΦΙΚΟ ΠΕΡΙΒΑΛΛΟΝ ΜΕ TKINTER

15.0 Λύσεις προηγούμενων ασκήσεων

1.

Σενάριο – Άσκηση (class_BankAccount.py):

Έχετε προσληφθεί για να δημιουργήσετε ένα απλό σύστημα τραπεζικού λογαριασμού. Το σύστημα πρέπει να υποστηρίζει δύο τύπους λογαριασμών: `***SavingsAccount***` και `***CheckingAccount***`. Και οι δύο τύποι λογαριασμών πρέπει να έχουν κοινές λειτουργίες όπως `***deposit()***`, `***withdraw()***` και `***get_balance()***`, αλλά κάθε τύπος λογαριασμού έχει κάποια μοναδικά χαρακτηριστικά.

1. Ορίστε μια βασική κλάση με το όνομα `***BankAccount***` με τις ακόλουθες μεθόδους:

- `***__init__(self, account_number, initial_balance)***`: Αρχικοποιεί τον λογαριασμό με έναν αριθμό λογαριασμού και έναν αρχικό υπόλοιπο.

- `***deposit(self, amount)***`: Προσθέτει το καθορισμένο ποσό στο υπόλοιπο του λογαριασμού.

- `***withdraw(self, amount)***`**:** Αφαιρεί το καθορισμένο ποσό από το υπόλοιπο του λογαριασμού.

- `***get_balance(self)***`: Επιστρέφει το τρέχον υπόλοιπο του λογαριασμού.

2. Δημιουργήστε μια υποκλάση με το όνομα `***SavingsAccount***` που κληρονομεί από την `***BankAccount***`. Προσθέστε τα ακόλουθα χαρακτηριστικά:

- Μια μεταβλητή κλάσης με το όνομα `***interest_rate***` που είναι ίση με 0.05 (5%).

- Αντικαταστήστε τη μέθοδο `***get_balance()***` για να υπολογίζει και να επιστρέφει το υπόλοιπο συν το επιτόκιο που έχει αποκτηθεί βάσει του επιτοκίου.

3. Δημιουργήστε μια άλλη υποκλάση με το όνομα `***CheckingAccount***` που κληρονομεί από την `***BankAccount***`. Προσθέστε τα ακόλουθα χαρακτηριστικά:

- Μια μεταβλητή κλάσης με το όνομα `***transaction_fee***` που είναι ίση με 1.0.

- Αντικαταστήστε τη μέθοδο `***withdraw()***` για να αφαιρεί το τέλος συναλλαγής από το υπόλοιπο του λογαριασμού.

4. Πραγματοποιήστε διάφορες λειτουργίες όπως καταθέσεις, αναλήψεις και ερωτήσεις υπολοίπου και παρατηρήστε τα αποτελέσματα.

Λύση:

```
class BankAccount:
    def __init__(self, account_number, initial_balance):
        self.account_number = account_number
        self.balance = initial_balance

    def deposit(self, amount):
        self.balance += amount
```

```
    def withdraw(self, amount):
        self.balance -= amount

    def get_balance(self):
        return self.balance


class SavingsAccount(BankAccount):
    interest_rate = 0.05

    def get_balance(self):
        interest = self.balance * self.interest_rate
        return self.balance + interest


class CheckingAccount(BankAccount):
    transaction_fee = 1.0

    def withdraw(self, amount):
        self.balance -= amount + self.transaction_fee


# Δοκιμή του κώδικα
savings = SavingsAccount("SAV001", 1000)
savings.deposit(500)
```

savings.withdraw(200)

print("Υπόλοιπο Λογαριασμού Αποταμίευσης:", savings.get_balance())

checking = CheckingAccount("CHK001", 2000)

checking.deposit(100)

checking.withdraw(50)

print("Υπόλοιπο Λογαριασμού Έλεγχου:", checking.get_balance())

Έξοδος:

Υπόλοιπο Λογαριασμού Αποταμίευσης: 1352.5

Υπόλοιπο Λογαριασμού Ελέγχου: 2049.0

Σε αυτήν τη λύση, καθορίζουμε την κλάση `**BankAccount**` ως τη βασική κλάση με κοινές μεθόδους όπως `**deposit**()`, `**withdraw**()` και `**get_balance()**`. Οι υποκλάσεις `**SavingsAccount**` και `**CheckingAccount**` κληρονομούν από την `**BankAccount**` και προσθέτουν τα δικά τους μοναδικά χαρακτηριστικά.

Η υποκλάση `**SavingsAccount**` υπολογίζει και επιστρέφει το υπόλοιπο συν τα επιτόκια που έχουν συσσωρευτεί βάσει του επιτοκίου. Η υποκλάση `CheckingAccount` αφαιρεί ένα τέλος συναλλαγής από το υπόλοιπο του λογαριασμού όταν γίνεται μια ανάληψη.

Στη συνέχεια, δημιουργούμε παραδείγματα τόσο για το `**SavingsAccount**` όσο και για το `**CheckingAccount**` και πραγματοποιούμε διάφορες λειτουργίες για να δοκιμάσουμε τον κώδικα. Η έξοδος εμφανίζει τα υπόλοιπα που προκύπτουν μετά την εκτέλεση των λειτουργιών.

Μπορείτε ελεύθερα να τροποποιήσετε τον κώδικα και να δοκιμάσετε διάφορα σενάρια για να εξερευνήσετε περαιτέρω την έννοια των μεταβλητών κλάσης, της ενθυλάκωσης και του πολυμορφισμού.

15.1 Εισαγωγή στο Tkinter

Το Tkinter προφέρεται ως tea-kay-inter. Το Tkinter είναι η διεπαφή της Python με το Tk, η οποία είναι η εργαλειοθήκη GUI (Graphical User Interface – Γραφική Διεπαφή Χρήστη) για το Tcl/Tk.

Η Tcl (προφέρεται ως tickle = γαργαλητό) είναι μια γλώσσα δέσμης ενεργειών που χρησιμοποιείται συχνά σε δοκιμές, πρωτότυπα και ανάπτυξη GUI.

Το Tk είναι μια εργαλειοθήκη γραφικών στοιχείων ανοιχτού κώδικα, πολλαπλών πλατφορμών που χρησιμοποιείται από πολλές διαφορετικές γλώσσες προγραμματισμού για τη δημιουργία προγραμμάτων GUI.

Η Python υλοποιεί το Tkinter σαν ένα module. Το Tkinter είναι ένας wrapper επεκτάσεων της C που χρησιμοποιούν βιβλιοθήκες Tcl/Tk.

Το Tkinter μας επιτρέπει να αναπτύσσουμε εφαρμογές desktop. Είναι ένα πολύ καλό εργαλείο για προγραμματισμό γραφικών διεπαφών στην Python.

Το Tkinter είναι μια καλή επιλογή για τους ακόλουθους λόγους:

- Είναι εύκολο στην εκμάθηση.
- Χρησιμοποιεί πολύ λίγο κώδικα για να δημιουργήσει μια λειτουργική εφαρμογή επιφάνειας εργασίας.

- Σχεδιάζεται σε πολλά επίπεδα (layers).
- Είναι φορητό σε όλα τα λειτουργικά συστήματα, συμπεριλαμβανομένων των Windows, macOS και Linux.
- Είναι προεγκατεστημένο στην Python.

15.2 Δημιουργία παραθύρου στο Tkinter

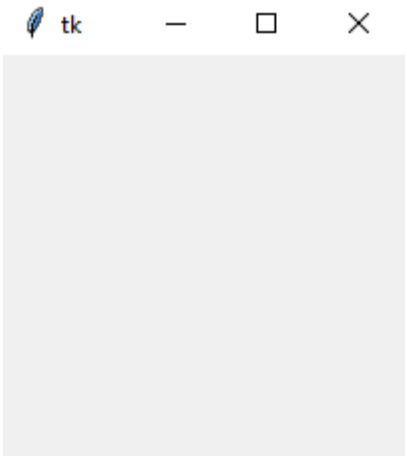
Με τον παρακάτω κώδικα εμφανίζουμε ένα παράθυρο στην οθόνη:

import tkinter as tk

root = tk.Tk()

root.mainloop()

Εάν εκτελέσουμε το πρόγραμμα, θα δούμε το ακόλουθο παράθυρο:



Πως δουλεύει.

Πρώτα, εισάγουμε το module tkinter ως tk στο πρόγραμμα:

import tkinter as tk

Δημιουργούμε ένα στιγμιότυπο της κλάσης ***tk.Tk*** που θα δημιουργήσει το παράθυρο ***root*** της εφαρμογής:

root = tk.Tk()

Κατά σύμβαση, το κύριο παράθυρο στο Tkinter ονομάζεται ***root***. Αλλά μπορούμε να χρησιμοποιήσουμε οποιοδήποτε άλλο όνομα όπως ***main***.

Καλούμε τη μέθοδο mainloop() του αντικειμένου του κύριου παραθύρου:

root.mainloop()

Η ***mainloop()*** διατηρεί το παράθυρο ορατό στην οθόνη. Εάν δεν καλέσουμε τη μέθοδο ***mainloop(),*** το παράθυρο θα εμφανιστεί και θα εξαφανιστεί αμέσως.

Επίσης, η μέθοδος ***mainloop()*** διατηρεί το παράθυρο να εμφανίζεται και να λειτουργεί μέχρι να το κλείσουμε.

Συνήθως, καλούμε τη μέθοδο mainloop() στο τέλος, μετά τη δημιουργία των γραφικών στοιχείων.

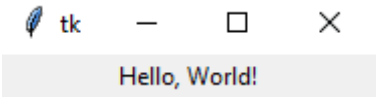
15.2.1 Εισαγωγή widget στο παράθυρο

Στο Tkinter, τα αντικείμενα ονομάζονται widgets.

Για να προσθέσουμε μια ετικέτα (label) στο παράθυρο, πληκτρολογούμε:

```
import tkinter as tk

root = tk.Tk()

# Τοποθέτηση μιας ετικέτας - label στο root window

message = tk.Label(root, text="Hello, World!")

message.pack()

# Συνεχής εμφάνιση του παραθύρου

root.mainloop()
```

Εκτελώντας τον κώδικα, βλέπουμε:



Πως δουλεύει:

Για να δημιουργήσουμε ένα γραφικό στοιχείο που ανήκει σε ένα container, χρησιμοποιούμε την ακόλουθη σύνταξη:

```
widget = Όνομα γραφικού στοιχείου (container, **options)
```

Το κοντέινερ είναι το γονικό παράθυρο ή πλαίσιο όπου θέλουμε να τοποθετήσουμε το γραφικό στοιχείο.

Οι επιλογές (***options***) είναι ένα ή περισσότερα ορίσματα λέξεων-κλειδιών που καθορίζουν τις διαμορφώσεις του γραφικού στοιχείου.

Στο πρόγραμμα, τα ακόλουθα δημιουργούν ένα γραφικό στοιχείο ***Label*** που τοποθετείται στο παράθυρο root:

message = tk.Label(root, text="Hello, World!")

Και η ακόλουθη δήλωση τοποθετεί την ετικέτα στο παράθυρο ***root***:

message.pack()

Εάν δεν καλέσουμε τη μέθοδο ***pack(),*** το Tkinter εξακολουθεί να δημιουργεί το γραφικό στοιχείο. Ωστόσο, το widget είναι αόρατο.

15.2.2 Χαρακτηριστικά του παραθύρου

Το παράθυρό μας έχει ένα τίτλο το οποίος έχει σαν προεπιλεγμένη τιμή "tk". Επίσης, έχει τρία κουμπιά συστήματος: Ελαχιστοποίηση, μεγέθυνση και κλείσιμο.

Ας δούμε πως αλλάζουμε τα χαρακτηριστικά του παραθύρου.

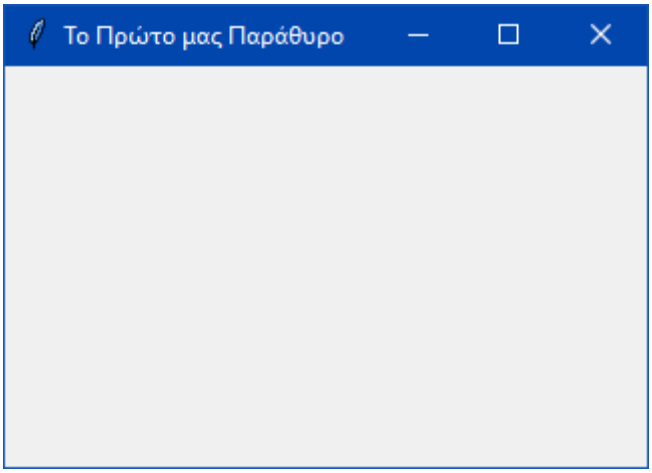
Μπορούμε να χρησιμοποιήσουμε τη μέθοδο ***title():***

window.title(newtitle)

Παράδειγμα:

Αλλάζουμε τον τίτλο του root window σε «Το Πρώτο μας Παράθυρο»

```
import tkinter as tk
root = tk.Tk()
root.title('Το Πρώτο μας Παράθυρο')
root.mainloop()
```



15.2.3 Αλλαγή μεγέθους και τοποθεσίας του παραθύρου

Στο Tkinter, το μέγεθος και η τοποθεσία του παραθύρου καθορίζεται από γεωμετρικά χαρακτηριστικά τα οποία δηλώνουμε ως εξής:

width x height ± x ± y

window.geometry('width x height + x + y')

- *Το width είναι το πλάτος του παραθύρου σε pixels*
- *Το height είναι το ύψος του παραθύρου σε pixels*
- Το x είναι η οριζόντια θέση του παραθύρου. Για παράδειγμα, +50 σημαίνει ότι η αριστερή άκρη του παραθύρου πρέπει να είναι 50px από την αριστερή άκρη της οθόνης και -50 σημαίνει, ότι η δεξιά άκρη του παραθύρου θα πρέπει να είναι 50 pixels από τη δεξιά άκρη της οθόνης.
- Το y είναι η κάθετη θέση του παραθύρου. Για παράδειγμα, +50 σημαίνει ότι το πάνω μέρος του παραθύρου πρέπει να είναι 50px

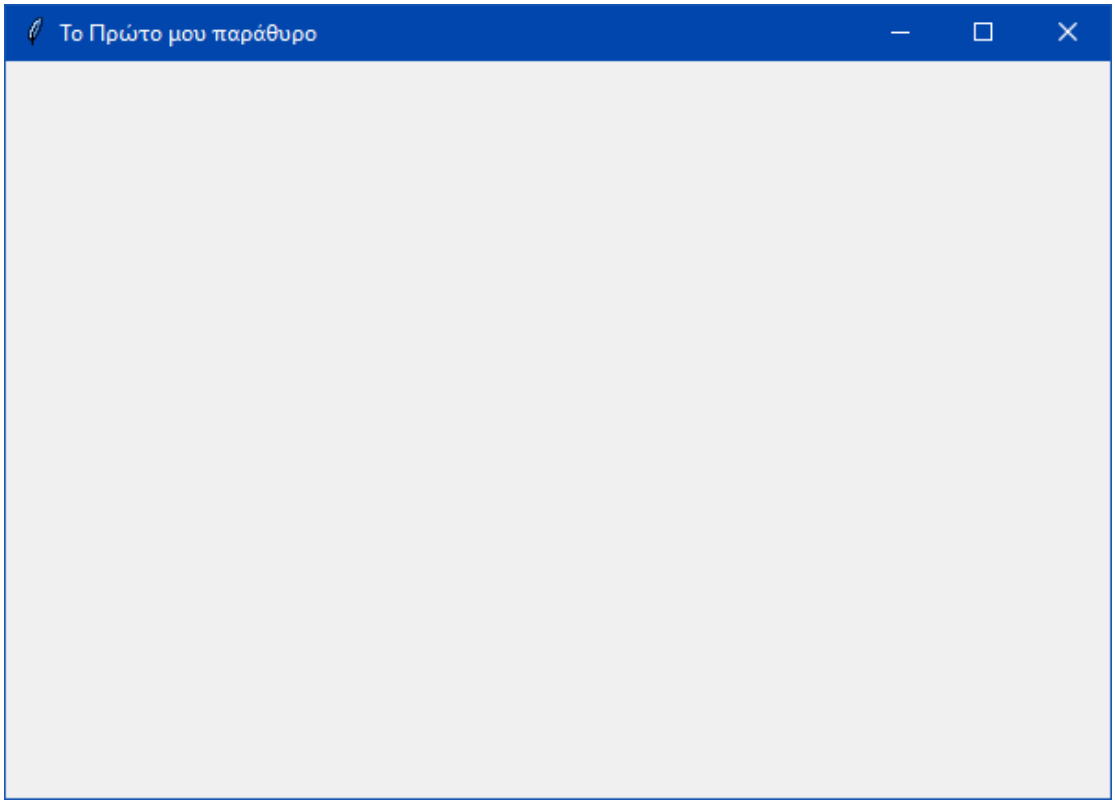
κάτω από το πάνω μέρος της οθόνης και -50 σημαίνει, ότι η κάτω άκρη του παραθύρου θα πρέπει να είναι 50 pixels πάνω από το κάτω μέρος της οθόνης.

Για να αλλάξουμε το μέγεθος και τη θέση του παραθύρου χρησιμοποιούμε τη μέθοδο geometry():

window.geometry(new_geometry)

Στο παρακάτω παράδειγμα, αλλάζουμε το μέγεθος του Παραθύρου σε 600 x 400 και τη θέση του παραθύρου σε 50px από επάνω και από αριστερά της οθόνης:

```
import tkinter as tk

root = tk.Tk()

root.title('Το Πρώτο μου παράθυρο')

root.geometry('600x400+50+50')


root.mainloop()
```



Κάποιες φορές, θέλουμε να κεντράρουμε το παράθυρο στην οθόνη:

```
import tkinter as tk

root = tk.Tk()
root.title('Tkinter Window - Center')

window_width = 300
window_height = 200

# Παίρνουμε το μέγεθος της οθόνης
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
# Βρίσκουμε το κέντρο της οθόνης
center_x = int(screen_width/2 - window_width / 2)
center_y = int(screen_height/2 - window_height / 2)
# Κεντράρισμα του παραθύρου
root.geometry(f'{window_width}x{window_height}+{center_x}+{center_y}')

root.mainloop()
```

Για να εμποδίσουμε το παράθυρο να αλλάζει μέγεθος, μπορούμε να χρησιμοποιήσουμε τη μέθοδο ***resizable():***

```
window.resizable(width,height)
```

Έχει δύο παραμέτρους οι οποίες καθορίζουν αν το πλάτος και το ύψος του παραθύρου μπορούν να αλλάξουν μέγεθος.

Επομένως, για παράθυρο με σταθερό μέγεθος έχουμε:

```
import tkinter as tk


root = tk.Tk()
root.title('Το Πρώτο μου Παράθυρο')
root.geometry('600x400+50+50')
root.resizable(False, False)


root.mainloop()
```

Όταν ένα παράθυρο αλλάζει μέγεθος, μπορούμε να καθορίσουμε το ελάχιστο και μέγιστο μέγεθος χρησιμοποιώντας τις μεθόδους minsize() και maxsize().

```
window.minsize(min_width, min_height)
window.maxsize(min_height, max_height)
```

Διαφάνεια

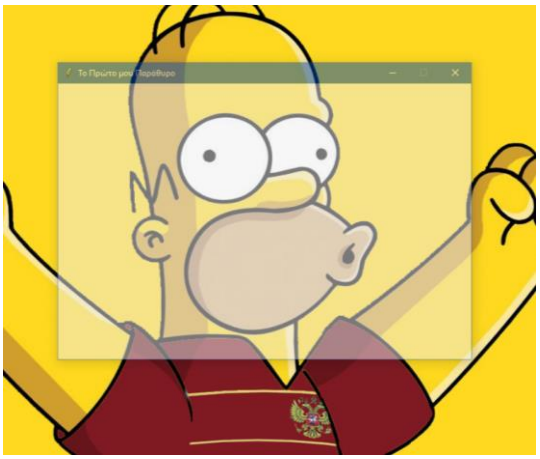
Το Tkinter μας επιτρέπει να καθορίζουμε τη διαφάνεια ενός παραθύρου ρυθμίζοντας το κανάλι άλφα του να κυμαίνεται από 0,0 (πλήρως διαφανές) έως 1,0 (πλήρως αδιαφανές):

```
window.attributes('-alpha',0,5)
```

Το ακόλουθο παράδειγμα απεικονίζει ένα παράθυρο με 50% διαφάνεια:

```
import tkinter as tk

root = tk.Tk()
root.title('Το πρώτο μου Παράθυρο')
root.geometry('600x400+50+50')
root.resizable (False, False)
root.attributes('-alpha', 0,5)

root.mainloop()
```

Η έξοδος είναι:



Σειρά στοίβαξης παραθύρων

Η σειρά στοίβαξης των παραθύρων αναφέρεται στη σειρά των παραθύρων που τοποθετούνται στην οθόνη από κάτω προς τα πάνω. Το πιο κοντινό παράθυρο βρίσκεται στο επάνω μέρος της στοίβας και επικαλύπτει το χαμηλότερο.

Για να διασφαλίσουμε ότι ένα παράθυρο βρίσκεται πάντα στην κορυφή της σειράς στοίβαξης, μπορούμε να χρησιμοποιήσουμε το χαρακτηριστικό -topmost ως εξής:

window.attributes('-topmost', 1)

Για να μετακινήσουμε ένα παράθυρο πάνω ή κάτω της στοίβας, μπορούμε να χρησιμοποιήσουμε τις μεθόδους ***lift()*** και ***low():***

window.lift()

window.lift(another_window)

window.lower()

window.lower(another_window)

Το παρακάτω παράδειγμα τοποθετεί το παράθυρο root πάνω από όλα τα άλλα παράθυρα. Με άλλα λόγια, το παράθυρο root είναι πάντα στην κορυφή:

```
import tkinter as tk
root = tk.Tk()
root.title('Tkinter Window Demo')
root.geometry('300x200+50+50')
root.resizable(0, 0)
root.attributes('-topmost', 1)

root.mainloop()
```

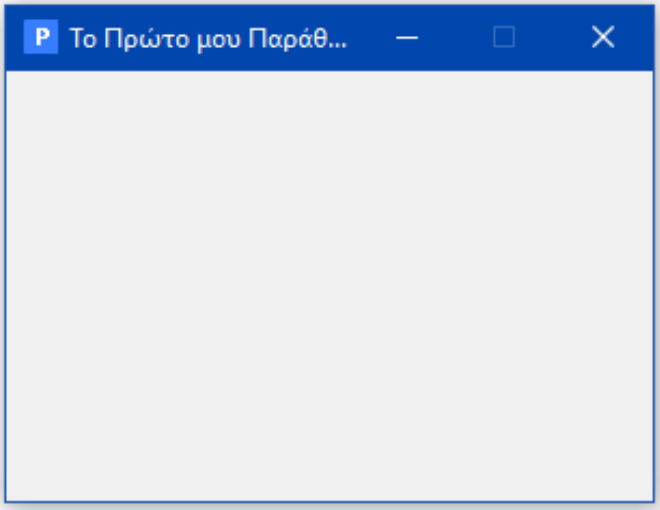
Αλλαγή του προεπιλεγμένου εικονιδίου

Το παράθυρο του Tkinter εμφανίζει ένα προεπιλεγμένο εικονίδιο. Για να αλλάξουμε αυτό το εικονίδιο:

1. Προετοιμάζουμε μια εικόνα σε μορφή .ico. Εάν έχουμε την εικόνα σε άλλες μορφές όπως png ή jpg, θα χρειαστεί να τη μετατρέψουμε σε μορφή .ico (μπορούμε να το κάνουμε στο διαδίκτυο).
2. Τοποθετούμε το εικονίδιο σε ένα φάκελο που είναι προσβάσιμος από το πρόγραμμα και καλούμε τη μέθοδο ***iconbitmap()*** του παραθύρου.

Ας δούμε παρακάτω:

```
import tkinter as tk
root = tk.Tk()
root.title('Το Πρώτο μου Παράθυρο')
root.geometry('300x200+50+50')
root.resizable(False, False)
root.iconbitmap('D:\Cookoo_Home\Documents\Ευδόκιμος\Παρουσιάσεις Μαθήματος Python\python_new.ico')

root.mainloop()
```



Ανακεφαλαίωση – Νέες Μέθοδοι

- Χρήση της μεθόδου ***title()*** για να αλλάξετε τον τίτλο του παραθύρου.
- Χρήση της μεθόδου ***geometry()*** για να αλλάξετε το μέγεθος και τη θέση του παραθύρου.
- Χρήση της μεθόδου ***resizable()*** για να καθορίσετε εάν ένα παράθυρο μπορεί να αλλάξει μέγεθος οριζόντια ή κάθετα.
- Χρήση του ***window.attributes('-alpha',0.5)*** για να ορίσουμε τη διαφάνεια του παραθύρου.
- Χρήση του ***window.attributes('-topmost', 1)*** για να κάνουμε το παράθυρο να είναι πάντα στην κορυφή.
- Χρήση των μεθόδων ***lift()*** και ***low()*** για να μετακινήσουμε το παράθυρο πάνω και κάτω στη σειρά στοίβαξης παραθύρων.
- Χρήση της μεθόδου ***iconbitmap()*** για να αλλάξουμε το προεπιλεγμένο εικονίδιο του παραθύρου.

15.3 Εισαγωγή στα Tk themed widgets

Το Tkinter έχει δύο γενιές widget:

Τα παλιά κλασικά γραφικά στοιχεία tk. Παρουσιάστηκαν το 1991.

Τα νεότερα θεματικά γραφικά στοιχεία ttk προστέθηκαν το 2007 με το Tk 8.5. Τα νεότερα γραφικά στοιχεία με θέμα Tk αντικαθιστούν πολλά (αλλά όχι όλα) κλασικά γραφικά στοιχεία.

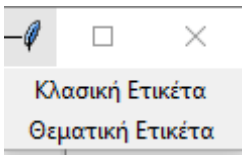
Σημειώστε ότι το ttk σημαίνει Tk theme. Επομένως, τα γραφικά στοιχεία με θέμα Tk (themed) είναι τα ίδια με τα γραφικά στοιχεία ttk

Η ενότητα tkinter.ttk περιέχει όλα τα νέα γραφικά στοιχεία ttk. Είναι καλή πρακτική να χρησιμοποιούμε πάντα θεματικά γραφικά στοιχεία όποτε είναι διαθέσιμα.

Οι ακόλουθες δηλώσεις εισάγουν τα κλασικά και τα νέα γραφικά στοιχεία με θέμα Tk και δημιουργούν κλασικές και θεματικές ετικέτες:

```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()

tk.Label(root, text='Κλασική ετικέτα').pack()
ttk.Label(root, text='Θεματική ετικέτα').pack()

root.mainloop()
```

Έξοδος:



Και οι δύο ετικέτες μοιάζουν παρόμοιες. Όμως, η εμφάνισή τους εξαρτάται από την πλατφόρμα στην οποία εκτελείται το πρόγραμμα.

15.3.1 Πλεονεκτήματα της χρήσης Tk themed widgets

Χρησιμοποιώντας τα themed widgets επιτυγχάνεται:

1. Διαχωρισμός της συμπεριφοράς και της εμφάνισης του γραφικού στοιχείου – τα γραφικά στοιχεία ttk προσπαθούν να διαχωρίσουν τον κώδικα που υλοποιεί τις συμπεριφορές των γραφικών στοιχείων από την εμφάνισή τους μέσω του συστήματος στυλ.
2. Εγγενής εμφάνιση και αίσθηση – τα γραφικά στοιχεία ttk έχουν την εγγενή εμφάνιση και αίσθηση της πλατφόρμας στην οποία εκτελείται το πρόγραμμα.

3. Απλοποίηση της συμπεριφοράς των γραφικών στοιχείων σε ειδικές καταστάσεις

Τα παρακάτω γραφικά στοιχεία ttk αντικαθιστούν τα γραφικά στοιχεία Tkinkter με τα ίδια ονόματα:

- Button
- Checkbutton
- Entry
- Frame
- Label
- LabelFrame
- Menubutton
- PanedWindow
- Radiobutton
- Scale
- Scrollbar
- Spinbox

Νέα στοιχεία ttk:

- Combobox
- Notebook
- Progressbar
- Separator
- Sizegrip
- Treeview

Πλήρη αναφορά για τα themed widgets μπορείτε να βρείτε στην τεκμηρίωση της Python.

Συμπερασματικά:

- το Tkinter έχει κλασικά αλλά και themed widgets (ttk)
- Το module, tkinter.ttk περιέχει όλα τα ttk widgets

- Όταν είναι διαθέσιμα, είναι καλό να χρησιμοποιούμε αυτά τα widgets.

15.4 Δημιουργία μετατροπέα θερμοκρασίας

Βασικά, η εφαρμογή έχει ένα label (ετικέτα), ένα πεδίο εισαγωγής αριθμού (entry field) και ένα κουμπί (button). Όταν εισάγουμε μια θερμοκρασία σε Φαρενάιτ και κάνουμε κλικ στο κουμπί «Μετατροπή», θα μετατραπεί η τιμή στο πεδίο εισαγωγής από Φαρενάιτ σε Κελσίου.

Εάν εισαγάγουμε μια τιμή που δεν μπορεί να μετατραπεί σε αριθμό, το πρόγραμμα θα εμφανίσει ένα σφάλμα.

Για να δημιουργήσουμε αυτήν την εφαρμογή, ακολουθούμε τα παρακάτω βήματα:

1. Κάνουμε εισαγωγή όλων των απαραίτητων modules:

import tkinter as tk

from tkinter import ttk

from tkinter.messagebox import showerror

2. Δημιουργούμε το παράθυρο root και τις ρυθμίσεις του:

 # Παράθυρο root

root = tk.Tk()

root.title('Μετατροπέας Θερμοκρασίας')

root.geometry('300x70')

root.resizable(False, False)

Γράφουμε τη συνάρτηση που μετατρέπει τη θερμοκρασία:

def fahrenheit_to_celsius(f):

 # Μετατροπή Φαρενάιτ σε Κελσίου

return (f - 32) * 5/9

Δημιουργούμε ένα frame το οποίο κρατάει το πεδίο εισαγωγής αριθμού:

frame = ttk.Frame(root)

Καθορίζουμε μια επιλογή που θα χρησιμοποιείται από όλα τα πεδία:

options = {'padx': 5, 'pady': 5}

Καθορίζουμε την ετικέτα (label) το πεδίο εισαγωγής (entry) και το κουμπί. Η ετικέτα θα δείξει το αποτέλεσμα μόλις πατηθεί το κουμπί «Μετατροπή»

```
# Ετικέτα θερμοκρασίας

temperature_label = ttk.Label(frame, text='Fahrenheit')

temperature_label.grid(column=0, row=0, sticky='W', **options)


# Πεδίο εισαγωγής θερμοκρασίας

temperature_entry = ttk.Entry(frame, textvariable=temperature)

temperature_entry.grid(column=1, row=0, **options)

temperature_entry.focus()


# Κουμπί μετατροπής

convert_button = ttk.Button(frame, text=Μετατροπή)

convert_button.grid(column=2, row=0, sticky='W', **options)

convert_button.configure(command=convert_button_clicked)


# Ετικέτα αποτελέσματος

result_label = ttk.Label(frame)
```

result_label.grid(row=1, columnspan=3, **options)

Τέλος, τοποθετούμε το frame στο παράθυρο root και τρέχουμε τη μέθοδο ***mainloop():***

frame.grid(padx=10, pady=10)

root.mainloop()

Όλος ο κώδικας του προγράμματος:

```
import tkinter as tk

from tkinter import ttk

from tkinter.messagebox import showerror


# Παράθυρο root

root = tk.Tk()

root.title('Μετατροπέας Θερμοκρασίας')

root.geometry('300x70')

root.resizable(False, False)


def fahrenheit_to_celsius(f):

    # Μετατροπή Φαρενάιτ σε Κελσίου

    return (f - 32) * 5/9


# Δημιουργία frame

frame = ttk.Frame(root)
```

```
# Ρυθμίσεις του πεδίου

options = {'padx': 5, 'pady': 5}


# Ετικέτα θερμοκρασίας

temperature_label = ttk.Label(frame, text='Φαρενάιτ')

temperature_label.grid(column=0, row=0, sticky='W', **options)


# Πεδίο εισαγωγής αριθμού (θερμοκρασίας)

temperature = tk.StringVar()

temperature_entry = ttk.Entry(frame, textvariable=temperature)

temperature_entry.grid(column=1, row=0, **options)

temperature_entry.focus()


# Κουμπί μετατροπής

def convert_button_clicked():

    # Διαχείριση γεγονότος εισαγωγής αριθμού και πατήματος του κουμπιού

        try:

        f = float(temperature.get())

        c = fahrenheit_to_celsius(f)

        result = f'{f} βαθμοί Φαρενάιτ είναι {c:.2f} βαθμοί Κελσίου'
```

```
        result_label.config(text=result)
    except ValueError as error:
        showerror(title='Error', message=error)


convert_button = ttk.Button(frame, text='Μετατροπή')
convert_button.grid(column=2, row=0, sticky='W', **options)
convert_button.configure(command=convert_button_clicked)

# Ετικέτα αποτελέσματος
result_label = ttk.Label(frame)
result_label.grid(row=1, columnspan=3, **options)

# Προσθήκη padding στο frame
frame.grid(padx=10, pady=10)

# Εκκίνηση εφαρμογής
root.mainloop()
```

15.5 Ασκήσεις

1. Γράψτε ένα προγραμματάκι το οποίο να έχει τρία πεδία εισαγωγής:

Όνομα, User ID, και Password, καθώς κι ένα κουμπί εισαγωγής, "ΥΠΟΒΟΛΗ". (gui_3fields.py)

2. Γράψτε ένα προγραμματάκι χρησιμοποιώντας το tkinter module, το οποίο πρέπει να δημιουργεί ένα checkbutton με κείμενο: "Τσεκάρισμα όταν είναι True". Χρησιμοποιείστε μια μεταβλητή boolean για να διευκολυνθείτε.(gui_checkbutton.py)

ΚΑΛΗ ΜΕΛΕΤΗ