35. GUI - TTKBOOTSTRAP – TURTLE GAMES & GRAPHICS

Επανάληψη κεφαλαίων 18 - 20

35.0.1 Λύσεις των προηγούμενων ασκήσεων

1. Άσκηση 1 (34.ask_1)

Γράψτε ένα πρόγραμμα το οποίο να έχει ένα παράθυρο με ένα πεδίο, το οποίο δέχεται κείμενο από τον χρήστη. Το πρόγραμμα πρέπει, με το πάτημα ενός κουμπιού, να αντιστρέφει και να παρουσιάζει το αντεστραμμένο κείμενο.

Λύση

```
from tkinter import ttk

root = Tk()
root.geometry('400x150')

def action():
    word = entryWord.get()
    # Αρχικοποίηση της λέξης
    reversed_word = ""
    for x in word:
        reversed_word = x + reversed_word
    lblResult['text'] = reversed_word


# Δημιουργία της ετικέτας και είσοδος της λέξης/φράσης
lblWord = Label(root , text ='Εισάγετε κείμενο : ')
lblWord.place( x = 20 , y = 20 )
entryWord = Entry(root )
entryWord.place( x = 150 , y = 20 , width = 225 , height =
25)
```

```
# Δημιουργία ετικέτας για προβολή του αποτελέσματος
lblResult = Label(root , text = "Αποτέλεσμα .......")
lblResult.place( x = 150 , y = 60)

# Δημιουργία του κουμπιού για την αντιστροφή
btnAction = Button(root , text = "Αντιστροφή" , command =
action)
btnAction.place( x = 150 , y = 100 , width = 225 , height =
30 )

root.mainloop()
```

2. Άσκηση 2 (34.ask_2)

Γράψτε ένα πρόγραμμα με γραφικό περιβάλλον, το οποίο να υπολογίζει τον Μέγιστο Κοινό Διαιρέτη (ΜΚΔ), καθώς και το Ελάχιστο Κοινό Πολλαπλάσιο (ΕΚΠ).

Θα πρέπει να δημιουργήσετε 2 πεδία που να δέχονται από έναν ακέραιο από τον χρήστη κι ένα combobox ώστε ο χρήστης να επιλέγει τη λειτουργία που χρειάζεται.

Λύση

```
from tkinter import *
from tkinter.ttk import *

root = Tk()
root.geometry('400x150')
```

```
def action (event):
    # Παίρνουμε την επιλογή από τη λίστα combobox
    select = comboList.get ()

    # Παίρνουμε την τιμή του αριθμού m από το πεδίο
εισαγωγής
    m = int (entry_m.get ())
    # Παίρνουμε την τιμή του αριθμού n από το πεδίο
εισαγωγής
    n = int (entry_n.get ())

    if (select == "ΜΚΔ"):
        lblResult ['text'] = " ΜΚΔ(m , n)  = " + str (gcd
(m, n))
    else:
```

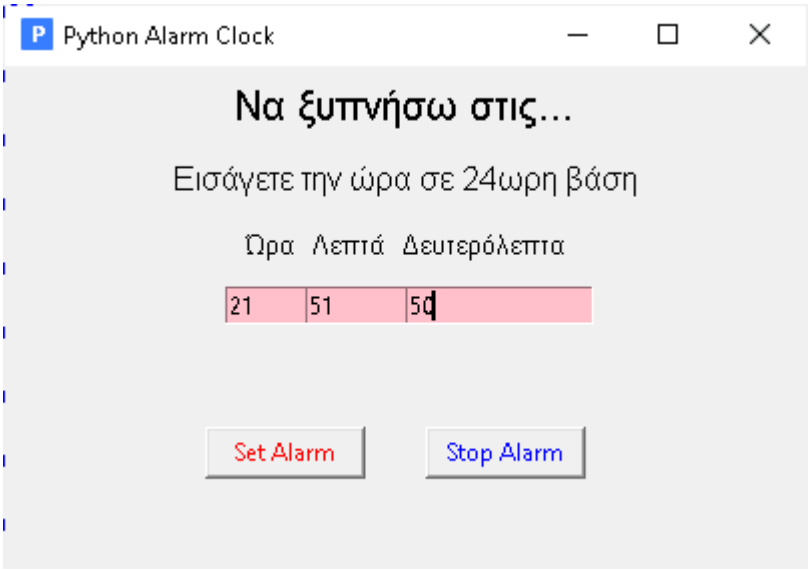
```
        lblResult ['text'] = " ΕΚΠ(m , n)  = " + str (lcm
(m, n))

# Δημιουργία ετικέτας και πεδίου εισαγωγής για τον ακέραιο m
lbl_m = Label (root, text = "Εισαγωγή 1ου ακεραίου: ")
entry_m = Entry (root)
lbl_m.place (x = 10, y = 20)
entry_m.place (x = 150, y = 20)

# Δημιουργία ετικέτας και πεδίου εισαγωγής για τον ακέραιο n
lbl_n = Label (root, text = "Εισαγωγή 2ου ακεραίου: ")
lbl_n.place (x = 10, y = 50)
entry_n = Entry (root)
entry_n.place (x = 150, y = 50)

lblChoose = Label (root, text = "Επιλογή λειτουργίας:")
lblChoose.place (x = 10, y = 80)

# Δημιουργία της λίστας του combobox για επιλογή ΜΚΔ ή ΕΚΠ
comboList = Combobox(root , values = ["ΜΚΔ" , "ΕΚΠ"])
comboList.place (x = 150, y = 80, width = 165)
comboList.bind ("<<ComboboxSelected>>", action)

# Δημιουργία ετικέτας για παρουσίαση του αποτελέσματος
lblResult = Label (root, text = "Αποτέλεσμα:")
lblResult.place (x = 150, y = 110)

root.mainloop ()
```

35.1.0 Python Alarm Clock

Σαν νέοι προγραμματιστές, είναι σημαντικό να εργαζόμαστε με έναν τρόπο πρακτικό κι ευχάριστο, εμπλουτίζοντας έτσι πιο γρήγορα, πιο πρακτικά και πιο ευχάριστα τις γνώσεις μας.

Για να κατακτήσουμε μια γλώσσα προγραμματισμού, θα πρέπει να εργαστούμε σε έργα. Η δημιουργία έργων μας δίνει πρώτα απ’ όλα τη χαρά της δημιουργίας, αλλά και μας βάζει σε μια νοοτροπία προγραμματιστική, στην οποία

καταλαβαίνουμε ότι ο προγραμματισμός είναι μια δεξιότητα που απαιτεί συγκεκριμένο κι έξυπνο τρόπο σκέψης, γνώση των βιβλιοθηκών και ολοκλήρωση σε συγκεκριμένο χρόνο. Πάμε λοιπόν να φτιάξουμε ένα ξυπνητήρι. Το τελικό μας αποτέλεσμα θα είναι το παρακάτω:



35.1.1 Στόχος & Προαπαιτούμενα

Στόχος μας είναι να φτιάξουμε ένα ξυπνητήρι σαν το παραπάνω. Βέβαια, όταν ξεκινάμε κάτι, δεν έχουμε την απεικόνιση του τελικού αποτελέσματος, παρά μόνο στο νου μας. Ήδη υπάρχει δηλαδή ένα νοητικό κατασκεύασμα, το οποίο πρέπει να υλοποιήσουμε γράφοντας τον αντίστοιχο κώδικα και δίνοντάς του την αντίστοιχη μορφή.

Αυτό το έργο απαιτεί καλή γνώση της Python καθώς και της δημιουργίας GUI (Graphic User Interface). Η Python όταν συνδυάζεται με το Tkinter και ακόμα καλύτερα με το ttkbootstrap, παρέχει έναν γρήγορο και εύκολο τρόπο δημιουργίας εφαρμογών GUI.

Το Tkinter με το ttkbootstrap παρέχουν όμορφες και εύκολα παραμετροποιήσιμες γραφικές διεπαφές χρήστη.

Δεν χρειάζεται να κατεβάσουμε ή να χρησιμοποιήσουμε κάτι που δεν έχει ενσωματωμένο η Python, όμως, πρέπει να αφιερώσουμε ελάχιστο χρόνο για να δούμε βιβλιοθήκες συνεργασίας της γλώσσας με το λειτουργικό μας

σύστημα, όπως τις “date, datetime και winsound” τις οποίες και εισάγουμε στην αρχή του project.

35.1.2 Δομή του έργου

Κάθε έργο μας, από αυτό το μέγεθος και πάνω, θα πρέπει να έχει μια δομή, ώστε το «νοητικό μας κατασκεύασμα» που αναφέραμε παραπάνω, να αρχίσει να παίρνει μορφή.

Μια υλοποίηση του ξυπνητηριού, θα μπορούσε να χρειάζεται τα παρακάτω βήματα για να υλοποιηθεί:

Αρχικά, ας ελέγξουμε τα βήματα για τη δημιουργία ενός προγράμματος Ξυπνητήρι στην Python:

- Εισαγωγή όλων των βιβλιοθηκών και ενοτήτων που απαιτούνται
- Ένας βρόχος while που παίρνει το όρισμα της ώρας, στην οποία ο χρήστης θέλει να βάλει το ξυπνητήρι να χτυπήσει και διακόπτεται αυτόματα όταν αυτή η ώρα φτάσει, με ήχο
- Γραφικό περιβάλλον με το tk.

35.1.3 Κατασκευή – Περιγραφή – Σχόλια

Ξεκινάμε την κατασκευή του προγράμματός μας με την εισαγωγή των απαραίτητων βιβλιοθηκών:

```
from tkinter import *
import datetime
import time
import winsound
```

Εισάγουμε τα πάντα από το Tkinter ώστε να έχουμε διαθέσιμα όλα τα στοιχεία για τα γραφικά μας.

Τα modules ***datetime*** και ***time*** μάς βοηθούν να εργαστούμε με τις ημερομηνίες και την ώρα της τρέχουσας ημέρας στην οποία ο χρήστης θα χρειαστεί το ξυπνητήρι.
Το module winsound παρέχει πρόσβαση στη χρήση βιβλιοθηκών ήχου των Windows. Αυτό θα μας είναι χρήσιμο για τη «χτύπημα του ξυπνητηριού» μόλις ικανοποιηθεί η συνθήκη και κληθεί η συνάρτησή μας.
Πληροφορίες για την datetime, μπορείτε να βρείτε εδώ:
https://docs.python.org/3/library/datetime.html

35.1.4 Βρόχος while – Η «μηχανή» του ξυπνητηριού

Πάμε να δούμε πως θα υλοποιήσουμε το ξυπνητήρι μας. Μπορούμε να σκεφτούμε: Όσο ο χρόνος μετράει και πλησιάζει τον χρόνο της αφύπνισης, μην κάνεις τίποτα. Όταν ο χρόνος της αφύπνισης, γίνει ίσος με τον χρόνο του τώρα, (της στιγμής), χτύπα το ξυπνητήρι.

Δηλαδή, τύπωσε ένα μήνυμα και κάλεσε τον ήχο του ξυπνητηριού.

Για να τα κάνουμε όλα αυτά, χρειάζεται να παίρνουμε τον χρόνο του τώρα και μπορούμε επίσης, για να έχουμε έναν έλεγχο του τι συμβαίνει, να τυπώνουμε τον χρόνο, παρατηρώντας ότι πλησιάζει τον χρόνο αφύπνισης. Έτσι, μπορούμε και εύκολα να αποσφαλματώσουμε το προγραμματάκι μας:

```
def alarm(set_alarm_timer): # Η παράμετρος εισάγεται από την
actual_time()
    while True: # Όσο δεν συμβαίνει τίποτα, τύπωνε στην
κονσόλα την ημ/νία και την ώρα
        time.sleep(1) # Ο χρόνος θα μετράει με καθυστέρηση
ενός δευτ/του
        current_time = datetime.datetime.now()
        now = current_time.strftime("%H:%M:%S")
        date = current_time.strftime("%d/%m/%Y")
        print("Η ημερομηνία είναι:",date) # Εκτύπωση στην
κονσόλα (IDLE)
        print(now) # Εκτύπωση στην κονσόλα (IDLE)
        if now == set_alarm_timer: # Όταν η ώρα γίνει ίση με
την ώρα που έχει ορίσει ο χρήστης
```

```
            print("Ώρα να ξυπνήσεις!") # τύπωσε μήνυμα στην
κονσόλα
            #
winsound.PlaySound("sound.wav",winsound.SND_ASYNC)
#Beep(1200, 500)
# Εδώ έχουμε έναν κλασικό ήχο των Windows

winsound.PlaySound("D://Cookoo_Home/Λήψεις/mixkit-retro-
game-emergency-alarm-1000.wav",winsound.SND_FILENAME) # και
κάλεσε τον ήχο του ξυπνητηριού (ήχος που έχουμε κατεβάσει)
            break # βγες από τον βρόχο (η λειτουργία του
ξυπνητηριού ολοκληρώθηκε)
```

Βέβαια, παρατηρούμε ότι η παράμετρος ***set_alarm_timer*** εισήχθη από την ***activate_alarm*** την οποία και πρέπει να δημιουργήσουμε:

```
def activate_alarm(): # Παίρνει τον χρόνο αφύπνισης και
"καλεί" τη "μηχανή" που θα χτυπήσει το ξυπνητήρι
    set_alarm_timer =
f"{hour.get()}:{min.get()}:{sec.get()}"
    alarm(set_alarm_timer) # Κλήση της μηχανής του
ξυπνητηριού
```

35.1.5 Δημιουργία γραφικού περιβάλλοντος

Έχουμε αναφερθεί αρκετά σ' αυτό το κομμάτι στα προηγούμενα μαθήματα στα οποία και μπορείτε να ανατρέξετε. Μια εύκολη παραλλαγή του GUI, μπορούμε να δημιουργήσουμε απλά αλλάζοντας το themename, εφόσον χρησιμοποιούμε το ttkbootstrap (όλες οι ονομασίες που μπορούμε να χρησιμοποιήσουμε, παρέχονται για ευκολία σας στα σχόλια).

```
# Δημιουργία γραφικού περιβάλλοντος
clock = ttk.Window(themename="cerculean")
clock.title("Python Alarm Clock")
#clock.iconbitmap(r"python.ico") # Μπορούμε να βάλουμε το
δικό μας εικονίδιο στο παράθυρο
clock.geometry("400x200")
time_format=ttk.Label(clock, text= "Εισάγετε την ώρα σε
24ωρη βάση", font=('Helvetica', 15), bootstyle =
PRIMARY).pack(side = TOP, padx=5, pady=5)
addTime = ttk.Label(clock, text = "Ώρα  Λεπτά
Δευτερόλεπτα", font=('Helvetica', 12), bootstyle =
SUCCESS).pack(side = TOP, padx=5, pady=5)
setYourAlarm = ttk.Label(clock, text = "Να ξυπνήσω στις...",
font=('Helvetica', 10), bootstyle = SECONDARY).pack(side =
BOTTOM, padx=5, pady=5)
```

35.1.6 Μεταβλητές και ολοκλήρωση του προγράμματος

Χρειαζόμαστε κάποιες μεταβλητές οι οποίες θα αποθηκεύουν τον χρόνο που θα μας δίνει ο χρήστης. Αυτές πρέπει να είναι τύπου StringVar(), καθότι μ' αυτό τον τύπο λειτουργούν και τα modules datetime και date, παραπάνω.

```
# Οι μεταβλητές που χρειαζόμαστε για το ξυνητήρι
(αρχικοποίηση):
hour = StringVar()
min = StringVar()
sec = StringVar()
```

Κατόπιν, φτιάχνουμε 3 πεδία (Entry) στο παράθυρο, ώστε ο χρήστης να μπορει να ορίζει την ώρα αφύπνισης. Μπορούμε εδώ, να πειραματιστούμε και πάλι με τα ttk widgets, ή να αφήσουμε την κλασική έκδοση του Tk().

```
# Ώρα στην οποία βάζουμε το ξυπνητήρι να χτυπήσει
hourTime= Entry(clock,textvariable = hour, width =
10).place(x=105,y=70)
```

```
minTime= Entry(clock,textvariable = min, width =
10).place(x=145,y=70)
secTime = Entry(clock,textvariable = sec, width =
10).place(x=185,y=70)
```

Ολοκληρώνουμε με τη δημιουργία του αντίστοιχου κουμπιού και το απαραίτητο συνεχές ***mainloop():***

```
# Το ξυπνητήρι μπαίνει σε λειτουργία με το πάτημα του
κουμπιού
submit = ttk.Button(clock,text = "Βάζω ξυπνητήρι", width =
15, command = activate_alarm).pack(side = BOTTOM, padx=5,
pady=5)

clock.mainloop()
```

ΤΟ ΞΥΠΝΗΤΗΡΙ ΜΑΣ ΕΙΝΑΙ ΕΤΟΙΜΟ!

35.1.7 Τι μάθαμε

- *Μάθαμε πως να χρησιμοποιούμε την ώρα του συστήματος στα προγράμματά μας.*
- *Μάθαμε πώς να αναζητούμε και να βάζουμε δικούς μας ήχου και εικονίδια στο πρόγραμμα μας.*
- *Τέλος, μάθαμε πώς να δομούμε ένα πρόγραμμα. Αυτό φυσικά αποκτάται όλο και καλύτερα με την εμπειρία.*

35.2.0 Μετατροπή αρχείων py-to-exe

Είναι επόμενο πως σαν χρήστες Windows, θα μας εξυπηρετούσε κάποιες φορές να έχουμε διαθέσιμα αρχεία executables, δηλαδή με κατάληξη .exe, που όπως γνωρίζετε είναι η κατάληξη των εκτελέσιμων αρχείων των Windows.

35.2.1 Εγκατάσταση του πακέτου PyInstaller

Ανοίγουμε το command prompt και πληκτρολογούμε:

pip install pyinstaller

Πάμε τώρα να δημιουργήσουμε το εκτελέσιμο αρχείο:

Πλοηγούμαστε στον φάκελο όπου βρίσκεται το αρχείο .py, το οποίο θέλουμε να μετατρέψουμε επίσης σε .exe. Αυτό το κάνουμε στο command prompt των Windows, πληκτρολογώντας cd και το path, δηλαδή:

cd C:\Users\User\AppData\Local\Programs\Python\Python311

Τώρα, δίνουμε την εντολή:

pyinstaller –onefile onomaarxeiou.py

και πατάμε το Enter.

Αφού η διαδικασία ολοκληρωθεί, θα βρούμε το εκτελέσιμό μας στον φάκελο dist με το όνομα ***onomaarxeiou.exe***.

Αν όμως, όταν είχαμε εγκαταστήσει τα Windows, είχαμε χρησιμοποιήσει ελληνικούς χαρακτήρες στο όνομα χρήστη, θα λαμβάνουμε σφάλμα στην κονσόλα των Windows και η διαδικασία δεν θα ολοκληρώνεται.

35.2.2 Εγκατάσταση του auto-py-to-exe

Το παραπάνω πρόβλημα μπορούμε να ξεπεράσουμε αν εγκαταστήσουμε το auto-py-to-exe, το οποίο διαθέτει γραφικό περιβάλλον και μπορεί επίσης να ξεπεράσει αυτό το θέμα, ολοκληρώνοντας με παρόμοιο τρόπο τη δημιουργία του exe μας.

pip install auto-py-to-exe

35.3.1 Η βιβλιοθήκη Turtle

Μπορούμε να χρησιμοποιήσουμε την Python για να φτιάξουμε εντυπωσιακά γραφικά, σχέδια, κινούμενα σχέδια και παιχνίδια. Η δημιουργία γραφικών χρησιμοποιώντας μια γλώσσα προγραμματισμού είναι πολύ ενδιαφέρουσα, αλλά το πιο ωραίο είναι η αυτόματη δημιουργία αυτών των γραφικών μπροστά στα μάτια μας, απλά γράφοντας μερικές γραμμές κώδικα.

Σήμερα θα θυμηθούμε την Python Turtle.

- Τι είναι η βιβλιοθήκη Turtle;
- Επισκόπηση
- Βασικές λειτουργίες και κινήσεις της χελώνας.
- Σχεδιάζοντας διάφορα σχήματα και σχέδια.

Η Turtle είναι μια βιβλιοθήκη της Python για τη δημιουργία γραφικών,

εικόνων και παιχνιδιών. Η χρήση της είναι παρόμοια με έναν πίνακα σχεδίασης που επιτρέπει στους χρήστες να σχεδιάζουν εικόνες και σχήματα με προγραμματισμό. Ο προγραμματισμός με το χελωνάκι είναι επίσης γνωστός ως προγραμματισμός LOGO. Αυτή η βιβλιοθήκη συνιστάται ευρέως για κάποιον που ξεκινάει με προγραμματισμό σχεδίασης, ή για παιδιά τα οποία ξεκινούν τον προγραμματισμό. Είναι όμως απαραίτητη γνώση και για ενήλικες που ξεκινούν τον προγραμματισμό, μιας και η Python Turtle είναι μια ενσωματωμένη και πολύ γνωστή βιβλιοθήκη την οποία πρέπει οπωσδήποτε να εντάξουμε στις βασικές μας γνώσεις.

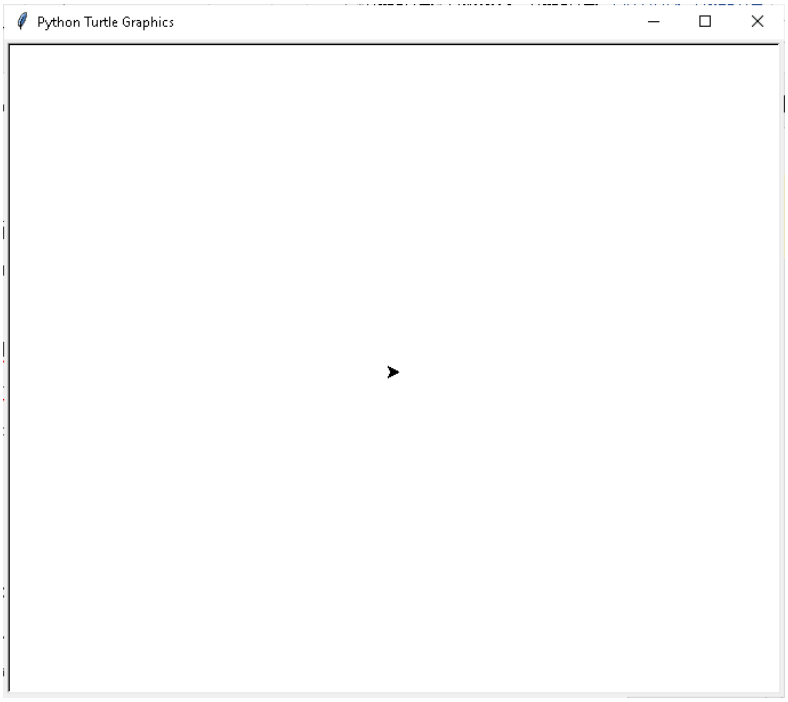
Όπως αναφέρθηκε προηγουμένως, η turtle είναι μια βιβλιοθήκη και επομένως δεν χρειάζεται να εγκαταστήσουμε κάτι επιπλέον. Μπορούμε απλά να χρησιμοποιήσουμε τη βιβλιοθήκη χρησιμοποιώντας τη δήλωση εισαγωγής και το όνομα της βιβλιοθήκης. Αυτή η βιβλιοθήκη περιέχει διάφορες προκαθορισμένες μεθόδους και λειτουργίες που είναι πολύ χρήσιμες για τη δημιουργία γραφικών χρησιμοποιώντας τη χελώνα.
Χρησιμοποιήστε την παρακάτω εντολή για να εισάγετε τη βιβλιοθήκη της χελώνας:

```
import turtle
```

Μετά την εισαγωγή της βιβλιοθήκης, πρέπει να έχουμε μια οθόνη στην οποία να μπορούμε να σχεδιάσουμε και να υλοποιήσουμε τα σχέδια μας. Αυτό μπορεί να γίνει χρησιμοποιώντας μια μεταβλητή που αποθηκεύει μια συνάρτηση της turtle που ονομάζεται getscreen()

```
sc = turtle.getscreen()
```

Έτσι μπορούμε να έχουμε το παρακάτω αποτέλεσμα στην οθόνη:



Και τώρα, θα δηλώσουμε μια μεταβλητή που θα αποθηκεύει τη χελώνα και θα χρησιμοποιήσουμε αυτήν τη μεταβλητή σε όλο το πρόγραμμα για να αποκτήσουμε πρόσβαση σε συναρτήσεις, μεθόδους κ.λ.π. που βρίσκονται στη βιβλιοθήκη turtle.

tp = turtle.Turtle()

Ωραία, τώρα το πρώτο πράγμα που θα μάθουμε είναι πώς να κάνουμε τη χελώνα να κινηθεί προς την κατεύθυνση που θέλουμε να πάει και τελικά να κάνει διαφορετικά σχήματα και σχέδια.

Μετά από αυτό, θα δούμε πώς να προσαρμόσουμε τη χελώνα μας και το περιβάλλον της.

Τέλος, θα μάθουμε μερικές επιπλέον εντολές που θα μας επιτρέψουν να εξοικειωθούμε με τον προγραμματισμό της χελώνας και θα μας βοηθήσουν σε μελλοντικά projects.

Ο σχεδιασμός γίνεται με τη μετακίνηση της χελώνας και για να γίνουν αυτά

χρειαζόμαστε συγκεκριμένες συναρτήσεις οι οποίες εισάγονται μαζί με τη βιβλιοθήκη και μπορούμε να τις χρησιμοποιούμε.

• forward(): Συνάρτηση για μετακίνηση της χελώνας προς τα εμπρός.

• backward(): Συνάρτηση για μετακίνηση της χελώνας προς τα πίσω.

• left(): Μετακίνηση της χελώνας προς τα αριστερά.

• right(): Αυτή η συνάρτηση μετακινεί τη χελώνα προς τα δεξιά.

Υπάρχουν επίσης κάποιες συντομογραφίες που μπορούμε να χρησιμοποιούμε για τις παραπάνω λειτουργίες.

Είναι fd για εμπρός , bk για πίσω , rt για δεξιά και lt για αριστερά .

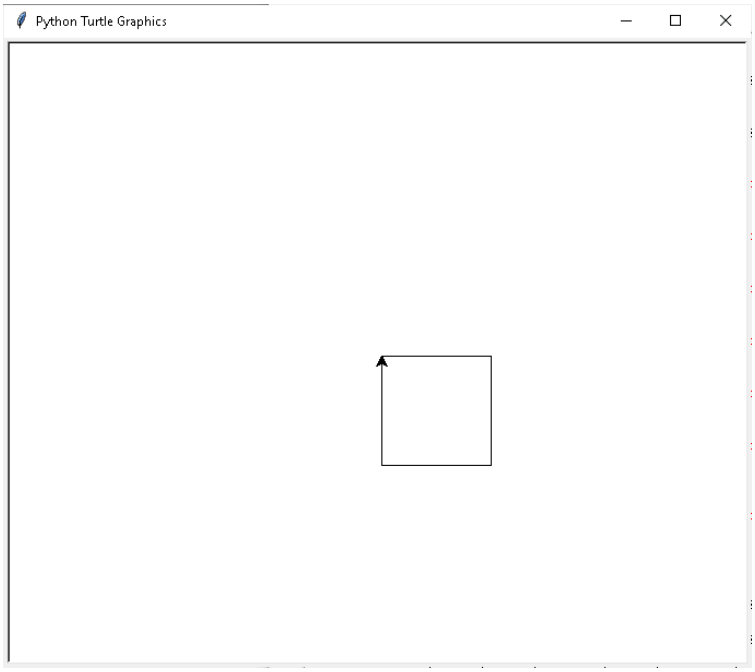
Πάμε να δούμε τη δημιουργία βασικών σχημάτων όπως τετράγωνα, κύκλοι, ορθογώνια και τρίγωνα και στη συνέχεια θα προχωρήσουμε για να φτιάξουμε μερικά όμορφα και μοναδικά πολύχρωμα σχέδια.

35.3.2 Σχεδίαση τετραγώνου

Ας γράψουμε τον κώδικα:

```
import turtle
tp = turtle.Turtle()
tp.fd(100)
tp.rt(90)
tp.fd(100)
tp.rt(90)
tp.fd(100)
tp.rt(90)
tp.fd(100)
```

Αποτέλεσμα:



Επεξήγηση: Καθώς το τετράγωνο αποτελείται από 4 γραμμές, καθεμία συνδεδεμένη σε γωνία 90 μοιρών, σε αυτό έχουμε χρησιμοποιήσει όλες τις λειτουργίες εμπρός και δεξιά.

tp.fd(100) : μετακινεί τη χελώνα 100 μονάδες προς τα εμπρός.

tp.rt(90): στρέφει τη χελώνα 90 μοίρες προς τα δεξιά.

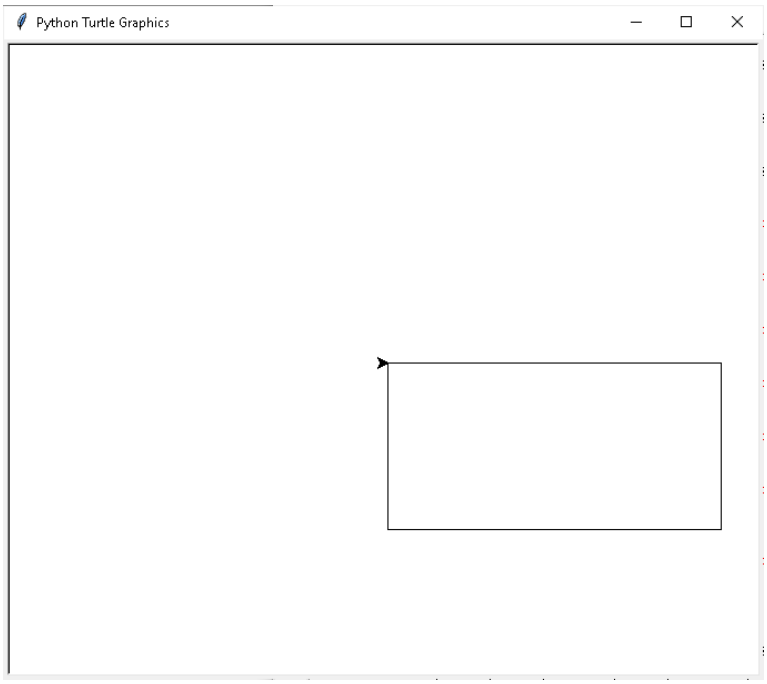
Με τον ίδιο τρόπο, η υπόλοιπη εντολή εκτελείται και παράγει το αποτέλεσμα.

35.3.3 Σχεδίαση ορθογωνίου

```
import turtle
tp = turtle.Turtle()
tp.fd(300)
tp.rt(90)
```

```
tp.fd(150)
tp.rt(90)
tp.fd(300)
tp.rt(90)
tp.fd(150)
tp.rt(90)
```

Αποτέλεσμα:



Επεξήγηση: Το ορθογώνιο αποτελείται από 4 γραμμές, καθεμία συνδεδεμένη σε γωνία 90 μοιρών, αλλά οι διπλανές πλευρές έχουν διαφορετικά μήκη. Έχουμε χρησιμοποιήσει τις συναρτήσεις για κίνηση εμπρός και δεξιά.

tp.fd(300) : μετακινεί τη χελώνα 300 μονάδες προς τα εμπρός.

tp.rt(90): στρέφει τη χελώνα 90 μοίρες προς τα δεξιά.

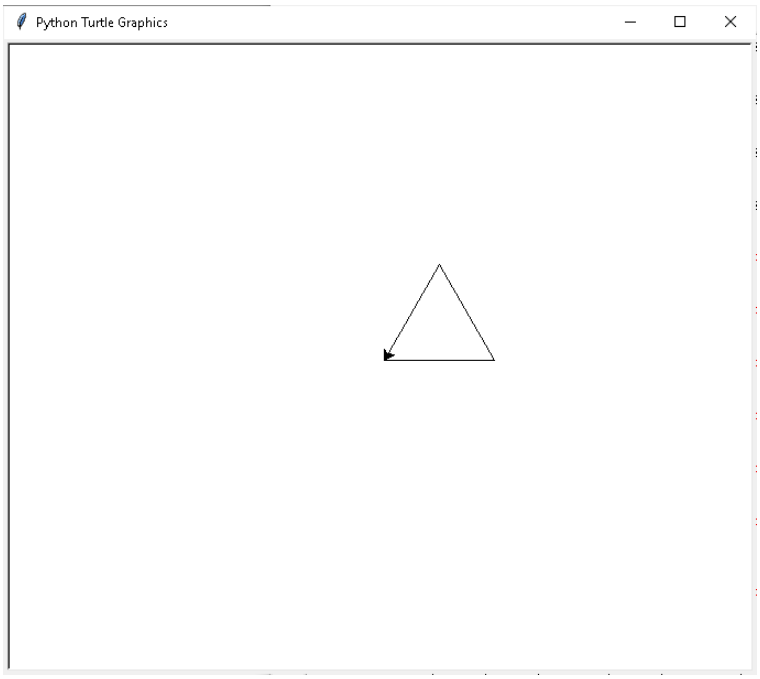
tp.fd(150): μετακινεί τη χελώνα 150 μονάδες προς τα εμπρός. Αυτό είναι 150 γιατί όλες οι πλευρές στο ορθογώνιο δεν είναι ίσες, μόνο οι απέναντι είναι ίσες.

Με τον ίδιο τρόπο, η υπόλοιπη εντολή εκτελείται και παράγει το αποτέλεσμα.

35.3.4 Σχεδίαση τριγώνου

```
import turtle
tp = turtle.Turtle()
tp.fd(100)
tp.lt(120)
tp.fd(100)
tp.lt(120)
tp.fd(100)
```

Αποτέλεσμα:



Εξήγηση: Ο παραπάνω κώδικας σχηματίζει ένα ισόπλευρο τρίγωνο δηλ. με όλες τις πλευρές του ίσες. Σε αυτό, χρησιμοποιήσαμε όλες τις λειτουργίες εμπρός και αριστερά.

tp.fd(100) : μετακινεί τη χελώνα 100 μονάδες προς τα εμπρός.

tp.lt(120): στρέφει τη χελώνα 120 μοίρες προς τα αριστερά.

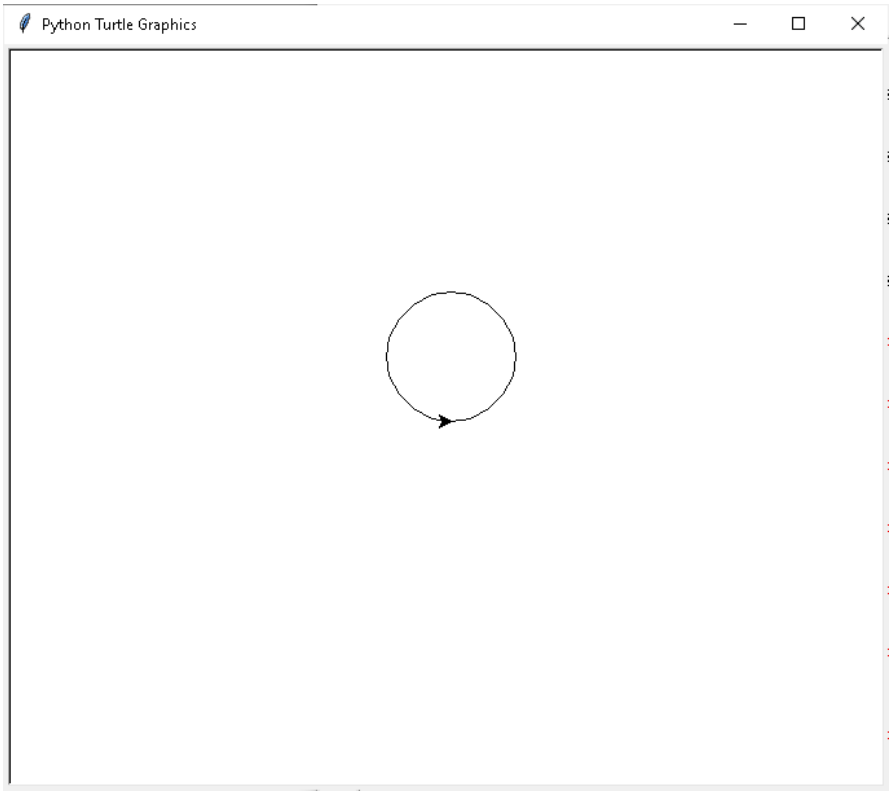
Με τον ίδιο τρόπο, ο υπόλοιπος κώδικας εκτελείται και παράγει το

αποτέλεσμα.

35.3.5 Σχεδιασμός κύκλου

```
import turtle
tp = turtle.Turtle()
tp.circle(50)
```

Αποτέλεσμα:



Επεξήγηση: Ο σχεδιασμός κύκλου είναι πιο εύκολος καθότι η βιβλιοθήκη Turtle παρέχει μια συνάρτηση με το όνομα circle().

Το όρισμα που χρειάζεται είναι απλώς η ακτίνα του κύκλου.

tp.circle(50) :

Εδώ έχουμε περάσει την ακτίνα ως 50 μονάδες στη συνάρτηση κύκλου.

Ας δούμε και τον παρακάτω σχολιασμένο κώδικα για τον σχηματισμό ενός συμμετρικού σχεδίου με την turtle.
Τέτοιου είδους σχέδια είναι συνηθισμένο να επιτυγχάνονται με τη χρήση for loops.

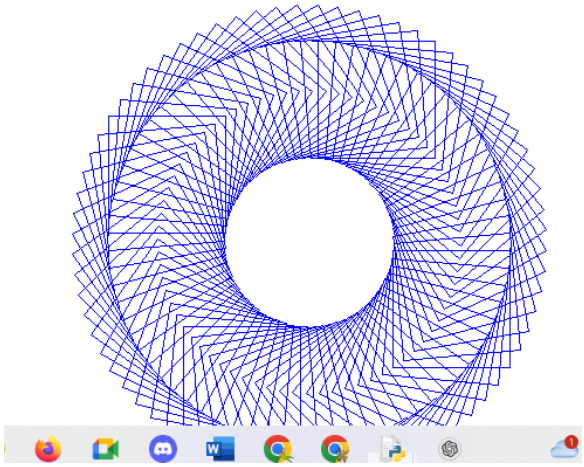
35.3.6 Κατασκευή συμμετρικού σχεδίου

Ας γράψουμε τον κώδικα:

```
import turtle as tl

# Η χελώνα είναι η μεταβλητή pen
pen = tl.Turtle()
pen.color("blue")
pen.speed(6)

# Ζωγραφική ενός μεγάλου τετραγώνου με 4 τετράγωνα στο εσωτερικό του
def draw_square():
    for side in range(4):
        pen.forward(100)
        pen.right(90)

    # Σήκωσε την πένα
    pen.penup()
    # Πήγαινε πίσω 20 μονάδες
    pen.back(20)
    # Τοποθέτησε την πένα έτοιμη να γράψει
    pen.pendown()

for square in range(80):
    # κλήση της συνάρτησης για σχηματισμό των τετραγώνων
    draw_square()
    # Αφού έχει ολοκληρωθει ένα μεγάλο τετράγωνο με 4 μικρά,
    # πήγαινε μπροστά κατά 5 μονάδες και στρίψε 5 μοίρες
```

```
    pen.forward(5)
    pen.left(5)

pen.hideturtle()
```

Το αποτέλεσμα είναι:



Ας δούμε και σχολιάσουμε ακόμα μερικά παραδείγματα σχηματισμού επαναλαμβανόμενων σχημάτων:

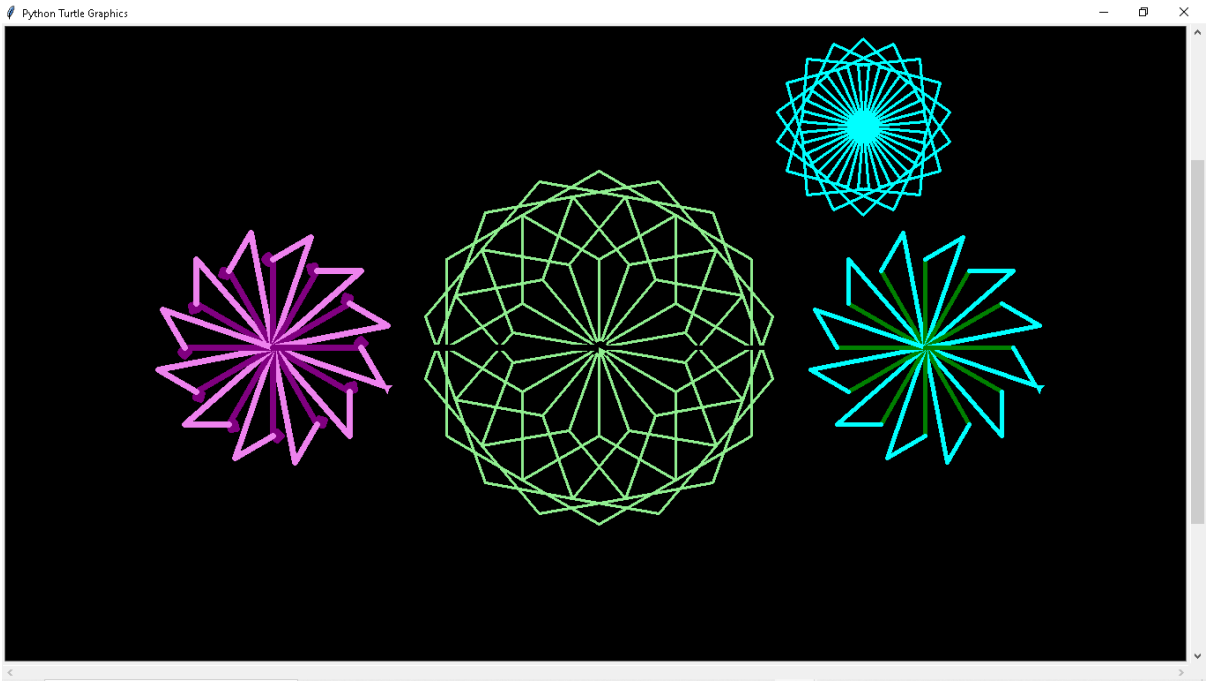
35.3.7 Κατασκευή επαναλαμβανόμενων σχημάτων

Ας γράψουμε τον παρακάτω κώδικα:

```
import time
Python_rules=turtle.Screen()
turtle.screensize(1200,1200)
turtle.bgcolor('black')
turtle.shape('turtle')
t2=turtle.Turtle()
t3=turtle.Turtle()
```

```
t4=turtle.Turtle()
t5=turtle.Turtle()
move=1

t2.speed(0)
for i in range(10):
    for i in range(4):
          t2.pu() #penup
          t2.goto(300,250)
          t2.pd() #pendown
          t2.color('cyan')
          t2.pensize(3)
          t2.circle(50,steps=4)# Φτιάξει ένα κύκλο με ακτίνα
50px σε 4 βήματα, δηλ. με 4 ακμές - ένα τετράγωνο
          t2.right(100)#στρίψε δεξιά 100 μοίρες

t3.speed(0)
for i in range(6):
    for i in range(4):
          t3.pu() #penup
          t3.goto(0,0)
          t3.pd() #pendown
          t3.color('light green')
          t3.pensize(3)
          t3.circle(100,steps=6)
          t3.right(100)

t4.speed(0)
for i in range (7):
    for i in range (2):
        t4.pensize(5)
        t4.goto(370,0)
        t4.color("green")
        t4.forward(100)
        t4.right(60)
        t4.color("cyan")
        t4.forward(50)
        t4.right(120)
    t4.right(30)

t5.speed(0)
for i in range (7):
    for i in range (2):
        t5.pensize(7)
```

```
        t5.goto(-370,0)
        t5.color("purple")
        t5.forward(100)
        t5.circle(5,steps=4)
        t5.right(60)
        t5.color("violet")
        t5.forward(50)
        t5.right(120)
    t5.right(30)
```



35.4.0 Ασκήσεις

Άσκηση 1

Γράψτε ένα πρόγραμμα twosquares.py που χρησιμοποιεί τη μονάδα Turtle για να σχεδιάσει δύο τετράγωνα μεγέθους 100. Σχεδιάστε τα με τέτοιο τρόπο ώστε να απέχουν 100 βήματα το ένα από το άλλο. Βεβαιωθείτε ότι το σχέδιο δεν περιέχει άλλες γραμμές. Ως τελευταίο βήμα ανάπτυξης, βεβαιωθείτε ότι το σχέδιο βρίσκεται στο κέντρο του παραθύρου.

Άσκηση 2

Γράψτε ένα πρόγραμμα triangle.py που χρησιμοποιεί τη μονάδα Turtle για να σχεδιάσει ένα τρίγωνο που έχει μια πλευρά μήκους 200, την άλλη πλευρά μήκους 100 και η γωνία μεταξύ αυτών των δύο πλευρών είναι 75 μοίρες. Βεβαιωθείτε ότι το σχέδιο βρίσκεται (περίπου) στο κέντρο του παραθύρου Python Turtle Graphics.