Εισαγωγή στην Python

Η βιβλιοθήκη turtle (turtle module)

Η βιβλιοθήκη turtle της Python

Η **Python** διαθέτει πολλές βιβλιοθήκες γραφικών. Ίσως η πιο απλή στη χρήση είναι η βιβλιοθήκη γραφικών χελώνας (**turtle graphics**). Παρουσιάστηκε μέσω της γλώσσας προγραμματισμού **Logo**, που το 1967 εισήγαγε ένα παρόμοιο σύστημα γραφικών το οποίο στη θέση του κέρσορα είχε μια χελώνα στην οποία ο προγραμματιστής έδινε εντολές να μετακινηθεί και να σχεδιάσει γραμμές πάνω στην οθόνη.

Για να φορτώσουμε τη βιβλιοθήκη **turtle** πληκτρολογούμε την εντολή:

```
>>> import turtle
```

Κυριότερες μέθοδοι της βιβλιοθήκης turtle

- **`forward()`**: Μετακινεί την χελώνα μπροστά ένα συγκεκριμένο αριθμό βημάτων, π.χ. η εντολή **`forward(100)`** μετακινεί τη χελώνα 100 pixel μπροστά.
- **`backward()`**: Μετακινεί την χελώνα πίσω ένα συγκεκριμένο αριθμό βημάτων, π.χ. η εντολή **`backward(50)`** μετακινεί τη χελώνα 50 pixel πίσω.
- **`right()`**: Στρίβει τη χελώνα δεξιά συγκεκριμένο αριθμό μοιρών, π.χ. η εντολή **`right(90)`** στρίβει τη χελώνα 90 μοίρες δεξιά.
- **`left()`**: Στρίβει τη χελώνα αριστερά συγκεκριμένο αριθμό μοιρών, π.χ. η εντολή **`left(90)`** στρίβει τη χελώνα 90 μοίρες αριστερά.
- **`circle()`**: Σχεδιάζει κύκλο συγκεκριμένης ακτίνας, π.χ. η εντολή **`circle(50)`** σχεδιάζει κύκλο ακτίνας 50 pixel.
- **`bgcolor()`**: Αλλάζει το χρώμα φόντου του παραθύρου γραφικών χελώνας, π.χ. η εντολή **`bgcolor("blue")`**, αλλάζει το χρώμα φόντου του παραθύρου γραφικών χελώνας σε μπλε.

Κυριότερες μέθοδοι της βιβλιοθήκης turtle

- **`penup()`**: Σηκώνει την πένα της χελώνας και η χελώνα μετακινείται χωρίς να αφήνει ίχνος.
- **`pendown()`**: Κατεβάζει την πένα της χελώνας και η χελώνα μετακινείται αφήνοντας ίχνος.
- **`color()`**: Αλλάζει το χρώμα της χελώνας και του ίχνους που αφήνει, π.χ. η εντολή **`color("green")`** αλλάζει το χρώμα της χελώνας και του ίχνους σε πράσινο.
- **`pensize()`**: Αλλάζει το πάχος του ίχνους της πένας της χελώνας, π.χ. η εντολή **`pensize(10)`** αλλάζει το πάχος του ίχνους της πένας σε 10 pixel.
- **`shape()`**: Αλλάζει τη μορφή της χελώνας. Οι διαθέσιμες μορφές είναι οι εξής: **`"arrow", "turtle", "circle", "square", "triangle", "classic"`**, π.χ. η εντολή **`shape("turtle")`** αλλάζει τη μορφή σε χελώνα.
- **`clear()`**: Καθαρίζει τα γραφικά χωρίς να φύγει από τη θέση της η χελώνα.

Κυριότερες μέθοδοι της βιβλιοθήκης turtle

- **`stamp()`**: Αφήνει το αποτύπωμά της χελώνας στο παράθυρο και το αποτύπωμα παραμένει αφού η χελώνα μετακινηθεί κάπου αλλού.
- **`speed()`**: Αλλάζει την ταχύτητα κίνησης της χελώνας. Οι δυνατές τιμές της ταχύτητας είναι οι φυσικοί αριθμοί **0-10** με την ακόλουθη σημασία:
  - **`“fastest”: 0`**
  - **`“fast”: 10`**
  - **`“normal”: 6`**
  - **`“slow”: 3`**
  - **`“slowest”: 1`**

Για παράδειγμα η εντολή **`speed(0)`** αλλάζει την ταχύτητα της χελώνας στη μέγιστη δυνατή, ενώ η εντολή **`speed(6)`** αλλάζει την ταχύτητα της χελώνας σε κανονική ταχύτητα.

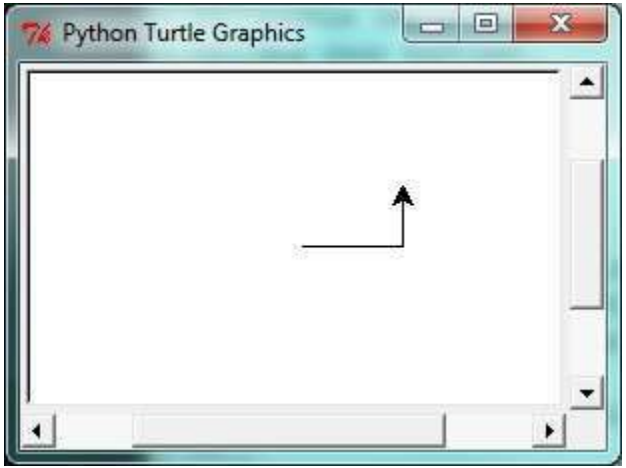
Το πρώτο μας turtle πρόγραμμα

Ας γράψουμε μερικές γραμμές προγράμματος Python για να δημιουργήσουμε μια νέα χελώνα και ας αρχίσουμε να σχεδιάζουμε ένα ορθογώνιο. (Θα ονομάσουμε τη μεταβλητή που αναφέρεται στην πρώτη μας χελώνα alex, αλλά μπορούμε να επιλέξουμε ένα οποιοδήποτε άλλο όνομα).

```
import turtle                   # Μας επιτρέπει να χρησιμοποιήσουμε το turtle module
window = turtle.Screen()        # Δημιουργεί ένα παράθυρο γραφικών χελώνας
alex = turtle.Turtle()          # Δημιουργεί μια χελώνα, την αντιστοιχίζει στο όνομα alex
alex.forward(50)                # Λέει στον alex να κινηθεί προς τα εμπρός κατά 50 pixel
alex.left(90)                   # Λέει στον alex να στρίψει κατά 90 μοίρες
alex.forward(30)                # Συμπλήρωση της δεύτερης πλευράς ενός ορθογωνίου
window.mainloop()               # Περιμένει να κλείσει το παράθυρο ο χρήστης
```

Το πρώτο μας turtle πρόγραμμα

Όταν εκτελέσουμε αυτό το πρόγραμμα, εμφανίζεται ένα νέο παράθυρο:



`window = turtle.Screen()`: Κάθε παράθυρο περιέχει μια περιοχή μέσα στην οποία μπορούμε να σχεδιάσουμε

`window.mainloop()`: Το παράθυρο περιμένει συμβάντα (όπως πατήματα πλήκτρων ή κλικ του ποντικιού). Το πρόγραμμα θα τερματιστεί όταν ο χρήστης κλείσει το παράθυρο

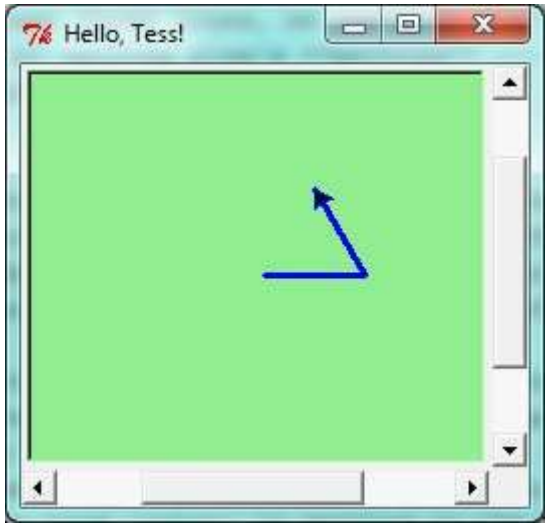
Μέθοδοι και ιδιότητες

Ας γράψουμε μερικές γραμμές προγράμματος Python για να τροποποιήσουμε μερικές από τις ιδιότητες της χελώνας και του παραθύρου και ας αρχίσουμε να σχεδιάζουμε ένα τρίγωνο.

```
import turtle                    # Εισάγουμε το turtle module
window = turtle.Screen()         # Δημιουργεί ένα παράθυρο για τις χελώνες
window.bgcolor("lightgreen")     # Ορισμός του χρώματος φόντου του παραθύρου
window.title("Hello, Tess!")     # Ορισμός του τίτλου του παραθύρου
tess = turtle.Turtle()           # Δημιουργεί μια χελώνα, την αντιστοιχίζει στο όνομα tess
tess.color("blue")               # Ορίζει το χρώμα της tess
tess.pensize(3)                  # Ορίζει το πλάτος του μαρκαδόρου της tess
tess.forward(50)                 # Λέει στην tess να κινηθεί προς τα εμπρός κατά 50 pixel
tess.left(120)                   # Λέει στην tess να στρίψει κατά 120 μοίρες
tess.forward(50)                 # Λέει στην tess να κινηθεί προς τα εμπρός κατά 50 pixel
window.mainloop()                # Περιμένει να κλείσει το παράθυρο ο χρήστης
```

Μέθοδοι και ιδιότητες

Όταν εκτελέσουμε αυτό το πρόγραμμα, εμφανίζεται αυτό το νέο παράθυρο και θα παραμείνει στην οθόνη μέχρι να το κλείσουμε.



Άσκηση 1

1. Τροποποιήστε το πρόγραμμα έτσι ώστε πριν δημιουργήσει το παράθυρο, να ζητά από τον χρήστη (με την εντολή **input**) να εισάγει το χρώμα φόντου (**bgcolor**). Η απάντηση του χρήστη θα πρέπει να αποθηκεύεται σε μια μεταβλητή και το χρώμα του παραθύρου να τροποποιείται σύμφωνα με την απάντηση του χρήστη.
2. Τροποποιήστε το πρόγραμμα έτσι ώστε να ζητά από τον χρήστη, κατά τον χρόνο εκτέλεσης, να ορίσει το χρώμα (**color**) της χελώνας.
3. Τροποποιήστε το πρόγραμμα έτσι ώστε να ζητά από τον χρήστη να ορίσει το πλάτος της πένας (**pensize**) της χελώνας. Υπόδειξη: ο διάλογος με τον χρήστη θα επιστρέψει μια συμβολοσειρά, θα πρέπει να μετατρέψετε τη συμβολοσειρά σε ακέραιο αριθμό πριν την εισάγετε στην `pensize`.

Συνδυάζουμε πολλά αντικείμενα μαζί

Μπορούμε να έχουμε πολλές χελώνες σε ένα πρόγραμμα. Κάθε χελώνα έχει τις δικές της ιδιότητες και μεθόδους.

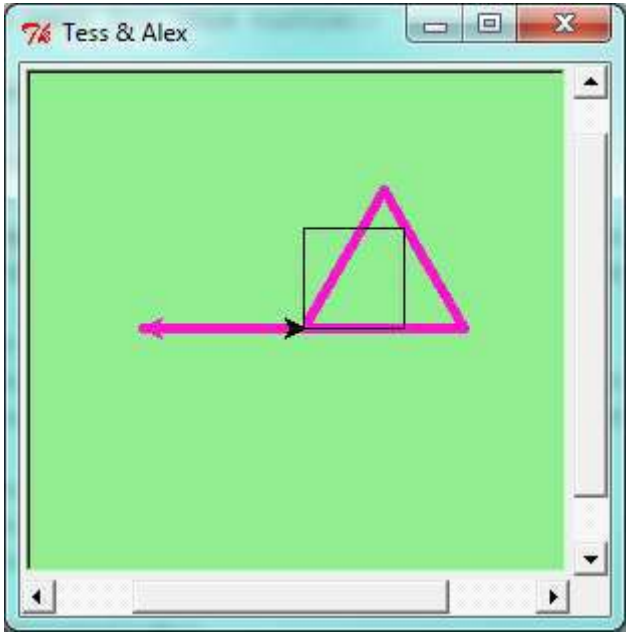
```
import turtle
window = turtle.Screen()        # Ρύθμιση του παραθύρου και των χαρακτηριστικών του
window.bgcolor("lightgreen")    # Ορισμός του χρώματος φόντου του παραθύρου
window.title("Tess & Alex")     # Ορισμός του τίτλου του παραθύρου
alex = turtle.Turtle()          # Δημιουργεί μια χελώνα, την αντιστοιχίζει στο όνομα alex
alex.forward(50)                # Λέει στον alex να σχεδιάσει ένα τετράγωνο
alex.left(90)
alex.forward(50)
alex.left(90)
alex.forward(50)
alex.left(90)
alex.forward(50)
alex.left(90)
```

Συνδυάζουμε πολλά αντικείμενα μαζί

```
tess = turtle.Turtle()          # Δημιουργεί μια χελώνα, την αντιστοιχίζει στο όνομα tess
tess.color("purple")            # Ορίζει το χρώμα της tess
tess.pensize(5)                 # Ορίζει το πλάτος του μαρκαδόρου της tess
tess.forward(80)                # Λέει στην tess να σχεδιάσει ένα ισόπλευρο τρίγωνο
tess.left(120)
tess.forward(80)
tess.left(120)
tess.forward(80)
tess.left(120)                  # Ολοκληρώνει το τρίγωνο
tess.right(180)                 # Γυρίζει την tess επιτόπου
tess.forward(80)                # Απομακρύνει την tess από την αρχική της θέση
window.mainloop()
```

Συνδυάζουμε πολλά αντικείμενα μαζί

Να τι συμβαίνει όταν εκτελέσουμε αυτό το πρόγραμμα:



Απλοποιούμε το πρόγραμμα μας με την επανάληψη

Όταν σχεδιάσαμε το τετράγωνο έπρεπε να επαναλάβουμε τα βήματα της κίνησης και της στροφής τέσσερις φορές. Αν σχεδιάζαμε ένα εξάγωνο, ή ένα οκτάγωνο, ή ένα πολύγωνο με 42 πλευρές, θα έπρεπε να επαναλάβουμε πολύ περισσότερες εντολές. Η δυνατότητα επανάληψης κάποιου κώδικα μας λύνει αυτό το πρόβλημα.

Για να σχεδιάσουμε ένα τετράγωνο, θέλουμε να κάνουμε το ίδιο πράγμα τέσσερις φορές – να μετακινήσουμε τη χελώνα και να στρίψουμε. Έχοντας βρει ένα «επαναλαμβανόμενο μοτίβο» εντολών μπορούμε να αναδιοργανώσουμε το πρόγραμμά μας για να επαναλάβουμε το μοτίβο με τον **βρόχο επανάληψης `for`**:

```
for i in range(4): # Εκτελεί το σώμα εντολών με i = 0, μετά 1, μετά 2, μετά 3
    alex.forward(50)
    alex.left(90)
```

Η συνάρτηση **`range()`** παράγει μια ακολουθία τιμών για τον βρόχο **`for`**. Ξεκινά από το 0 (μπορούμε να το παραλείψουμε) και δεν περιλαμβάνει το 4 (την τελική τιμή).

Απλοποιούμε το πρόγραμμα μας με την επανάληψη

Τι θα συμβεί αν κάνουμε αυτή την αλλαγή στο πρόγραμμα μας;

```
colors = ["yellow", "red", "green", "blue"]   # Λίστα με χρώματα
for i in range(4):
    alex.color(colors[i])   # Άλλαξε στο επόμενο χρώμα
    alex.forward(50)        # Σχεδίασε την επόμενη πλευρά
    alex.left(90)
```

Η χελώνα αλλάζει το χρώμα της στο επόμενο χρώμα της λίστας πριν σχεδιάσει την επόμενη πλευρά του τετραγώνου.

Άσκηση 2

1. Τροποποιήστε το προηγούμενο πρόγραμμα έτσι ώστε η χελώνα να χρησιμοποιεί έναν βρόχο επανάληψης `for` για να σχεδιάσει το ισόπλευρο τρίγωνό της.
2. Τροποποιήστε το προηγούμενο πρόγραμμα έτσι ώστε η χελώνα να αλλάζει το χρώμα της πριν σχεδιάσει την επόμενη πλευρά του τριγώνου.

Μερικές ακόμη turtle μέθοδοι

Μπορούμε να σηκώσουμε ή να κατεβάσουμε την πένα μιας χελώνας, έτσι μπορούμε να μετακινήσουμε τη χελώνα χωρίς να σχεδιάσουμε γραμμή.

```
alex.penup()       # Σήκωμα της πένας
alex.forward(100)  # Δεν σχεδιάζεται γραμμή
alex.pendown()
```

Κάθε χελώνα μπορεί να έχει το δικό της σχήμα. Οι διαθέσιμες επιλογές είναι: `"arrow", "circle", "classic", "square", "triangle", "turtle"`.

```
alex.shape("turtle")
```

Μερικές ακόμη turtle μέθοδοι

Μπορούμε να ορίσουμε την ταχύτητα κίνησης της χελώνας από το 1 (πιο αργή) έως το 10 (πιο γρήγορη). Αν ορίσουμε την ταχύτητα στο 0, απενεργοποιείται το animation και η χελώνα προχωρά όσο πιο γρήγορα γίνεται.

```
alex.speed(10)
```

Μια χελώνα μπορεί να αφήσει το αποτύπωμά της στο παράθυρο και το αποτύπωμα αυτό να παραμείνει αφού η χελώνα μετακινηθεί κάπου αλλού.

```
alex.stamp()
```

Μερικές ακόμη turtle μέθοδοι

Ας δούμε ένα παράδειγμα που παρουσιάζει μερικά από αυτά τα χαρακτηριστικά:

```
import turtle
window = turtle.Screen()
window.bgcolor("lightgreen")
tess = turtle.Turtle()
tess.shape("turtle")
tess.color("blue")
tess.speed(5)
tess.penup()
step = 10                       # ορίζουμε το βήμα κίνησης
for i in range(5):
    tess.stamp()                # άφησε ένα αποτύπωμα
    tess.forward(step)          # κίνησε τη χελώνα
    step = step + 10            # αυξάνουμε το βήμα σε κάθε επανάληψη
window.mainloop()
```

Μπορείτε να βρείτε τι θα συμβεί πριν τρέξετε το πρόγραμμα;

Μερικές ακόμη turtle μέθοδοι

Μπορούμε να σχεδιάσουμε έναν κύκλο με δεδομένη ακτίνα με την μέθοδο `circle()`.

```
aktina = 50           # ορίζουμε την ακτίνα
alex.circle(aktina)   # πλήρης κύκλος
```

Μπορούμε να σχεδιάσουμε ένα τόξο κύκλου δίνοντας στην `circle()` την παράμετρο `extent` σε μοίρες.

```
aktina = 50                                 # ορίζουμε την ακτίνα
moires = 180                                # ορίζουμε τις μοίρες του τόξου
alex.circle(aktina, extent=moires)          # ημικύκλιο
```

Μπορούμε να σχεδιάσουμε κανονικό πολύγωνο δίνοντας στην `circle()` την παράμετρο `steps` που καθορίζει τον αριθμό πλευρών του πολυγώνου.

```
aktina = 50                                 # ορίζουμε την ακτίνα
plevres = 8                                 # ορίζουμε τον αριθμό πλευρών του πολυγώνου
alex.circle(aktina, steps=plevres)          # πολύγωνο 8 πλευρών
```

Άσκηση 3

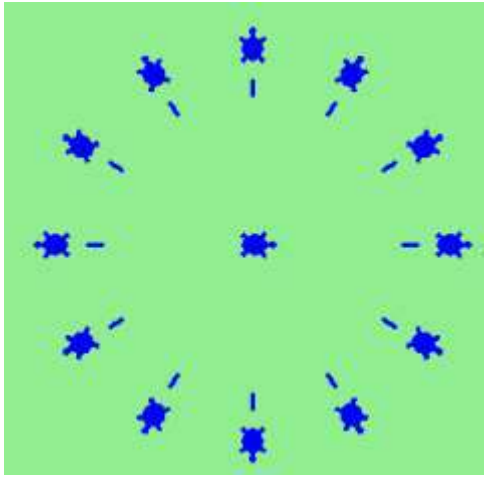
Σχεδιάστε με μια χελώνα τα παρακάτω κανονικά πολύγωνα, χρησιμοποιώντας τον βρόχο επανάληψης `for` (κανονικά είναι τα πολύγωνα που έχουν όλες τις πλευρές ίσες μεταξύ τους και όλες τις γωνίες ίσες μεταξύ τους):

1. Ένα εξάγωνο (έξι πλευρές)
2. Ένα οκτάγωνο (οκτώ πλευρές)
3. Ένα πολύγωνο με 18 πλευρές

Υπόδειξη: Μια πλήρης περιστροφή είναι 360 μοίρες (πλήρης κύκλος). Αν διαιρέσετε αυτό με τον αριθμό των πλευρών, μπορείτε να υπολογίσετε πόσες μοίρες πρέπει να περιστρέψετε τη χελώνα σε κάθε στροφή.

Άσκηση 4

Δημιουργήστε ένα πρόγραμμα που να σχεδιάζει την πρόσοψη ενός ρολογιού όπως το παρακάτω:



Γενίκευση του κώδικα με συναρτήσεις

Μια συνάρτηση είναι ένα κομμάτι κώδικα που γράφεται μία φορά και χρησιμοποιείται πολλές φορές. Μέσα σε μια συνάρτηση ομαδοποιούμε μια ακολουθία εντολών που πραγματοποιούν μια συγκεκριμένη εργασία.
Ας δημιουργήσουμε μια συνάρτηση με όνομα `tetragono` που θα χρησιμοποιεί μια χελώνα για να σχεδιάσει ένα τετράγωνο. Θα δέχεται μια παράμετρο `t`, που θα είναι μια χελώνα και μια παράμετρο με όνομα `mikos`, που θα είναι το μήκος των πλευρών.

```
def tetragono(t, mikos):
    for i in range(4):
        t.forward(mikos)
        t.left(90)
```

Έχουμε αντικαταστήσει κάτι συγκεκριμένο (τον αριθμό των pixel της πλευράς) με κάτι γενικότερο (την παράμετρο `mikos`). Η κλήση της συνάρτησης για να σχεδιάσουμε ένα τετράγωνο με πλευρές με `mikos` 100 γίνεται ως εξής:

```
tetragono(bob, 100)
```

Γενίκευση του κώδικα με συναρτήσεις

Ας τροποποιήσουμε την συνάρτηση `tetragono` έτσι ώστε να σχεδιάζει ένα κανονικό πολύγωνο με οποιονδήποτε αριθμό πλευρών. Θα δώσουμε στην συνάρτηση το όνομα `polygono` και θα προσθέσουμε την παράμετρο `n`, τον αριθμό των πλευρών. Ας θυμηθούμε επίσης ότι οι εξωτερικές γωνίες ενός κανονικού πολυγώνου με n πλευρές είναι `360 / n` μοίρες.

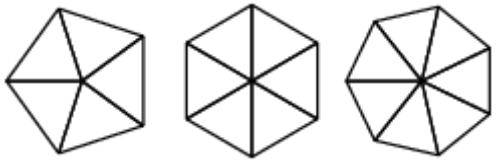
```
def polygono(t, n, mikos):
    gonia = 360 / n  # Αποθηκεύουμε στην μεταβλητή gonia την γωνία του πολυγώνου
    for i in range(n):
        t.forward(mikos)
        t.left(gonia)
```

Αντικαταστήσαμε τον αριθμό των πλευρών του τετραγώνου (4) με κάτι γενικότερο (την παράμετρο `n`). Η κλήση της συνάρτησης για να σχεδιάσουμε ένα πολύγωνο με 8 πλευρές με `mikos` 200 γίνεται ως εξής:

```
polygono(bob, 8, 200)
```

Άσκηση 5

Δημιουργήστε συναρτήσεις που μπορούν να σχεδιάζουν σχήματα όπως τα παρακάτω:



Υπόδειξη: Μπορείτε να ξεκινήστε από το κέντρο του σχήματος προς τα έξω. Στη συνέχεια, μπορείτε να χρησιμοποιήσετε τη μέθοδο `circle()` με παραμέτρους την ακτίνα και των αριθμό των πλευρών του εξωτερικού σχήματος.

Άσκηση 6

Μπορούμε να σχεδιάσουμε τα γράμματα του αλφαβήτου χρησιμοποιώντας μερικά βασικά στοιχεία, όπως κάθετες και οριζόντιες γραμμές και πολύγωνα. Σχεδιάστε στο χαρτί ένα αλφάβητο χρησιμοποιώντας έναν ελάχιστο αριθμό βασικών στοιχείων και στη συνέχεια δημιουργήστε συναρτήσεις που σχεδιάζουν τα γράμματα.

Δημιουργήστε μία συνάρτηση για κάθε γράμμα, δίνοντας ονόματα όπως `sxediase_a`, `sxediase_b`.

Άλλες μέθοδοι της βιβλιοθήκης turtle

- **`pos()`**: Επιστρέφει την τρέχουσα θέση της χελώνας π.χ. (100.0, 50.0).
- **`heading()`**: Επιστρέφει την τρέχουσα κατεύθυνση της χελώνας, π.χ. 90.0 (σε μοίρες).
- **`setheading()`**: Αλλάζει την τρέχουσα κατεύθυνση της χελώνας, με ενδεικτικές τιμές 0-ανατολικά, 90-βόρεια, 180-δυτικά, 270-νότια, π.χ. η εντολή **`setheading(180)`** κάνει τη χελώνα να δείχνει δυτικά.
- **`setx()`**: Αλλάζει την 1η συντεταγμένη (τετμημένη-άξονας x) της τρέχουσας θέσης της χελώνας.
- **`sety()`**: Αλλάζει την 2η συντεταγμένη (τεταγμένη-άξονας y) της τρέχουσας θέσης της χελώνας.

Άλλες μέθοδοι της βιβλιοθήκης turtle

- **`goto()`**: Αλλάζει την τρέχουσα θέση της χελώνας, π.χ. η εντολή goto(100, 120) μετακινεί τη χελώνα στη θέση (100,120).
- **`reset()`**: Δημιουργεί ένα παράθυρο γραφικών χελώνας (**Turtle Graphics**) ή καθαρίζει τα γραφικά και η χελώνα επιστρέφει στην αρχική της θέση (0, 0).
- **`bye()`**: Κλείνει το παράθυρο γραφικών χελώνας.
- **`colormode(255)`**: Αλλάζει το χρωματικό μοντέλο που χρησιμοποιεί η συνάρτηση **`color()`** στο RGB, π.χ. για να εμφανίσει το κόκκινο χρώμα η μέθοδος **`color()`** μπορεί να χρησιμοποιηθεί ως εξής: **`color(255, 0, 0)`** αντί για **`color(1, 0, 0)`**.