# 19. ΚΑΛΟΚΑΙΡΙΝΑ PROJECTS II - PYTHON TURTLE

## 19.1.0 Εισαγωγή

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 δούμε την Python Turtle.
Θα μάθουμε τα βασικά της βιβλιοθήκης Turtle, και πώς να προγραμματίζουμε και να δημιουργούμε μ΄αυτή.

### Τι θα μάθουμε;

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

## 19.1.0 Τι είναι η Python Turtle

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.

## 19.1.2 Επισκόπηση της Python turtle

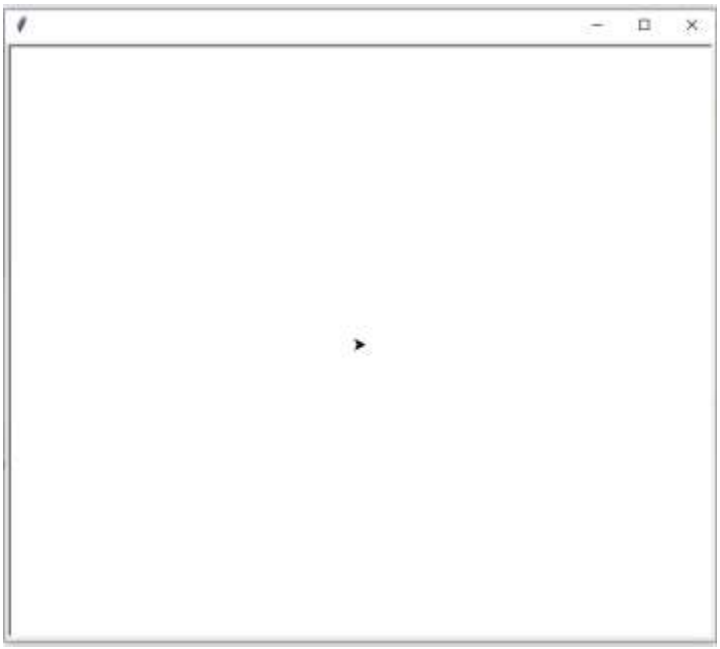
Όπως αναφέρθηκε προηγουμένως, η turtle είναι μια βιβλιοθήκη και επομένως δεν χρειάζεται να εγκαταστήσουμε κάτι επιπλέον. Μπορούμε απλά να χρησιμοποιήσουμε τη βιβλιοθήκη χρησιμοποιώντας τη δήλωση εισαγωγής και το όνομα της βιβλιοθήκης. Αυτή η βιβλιοθήκη περιέχει διάφορες προκαθορισμένες μεθόδους και λειτουργίες 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.

# 19.1.3 Συναρτήσεις και κινήσεις της χελώνας

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** .

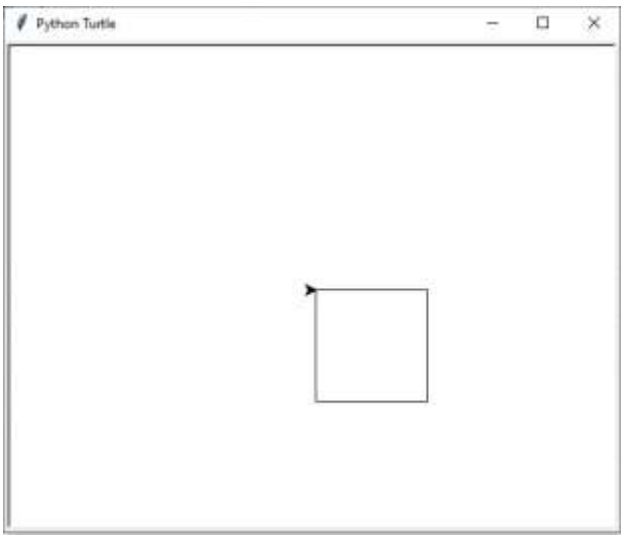
# 19.1.4 Σχήματα και σχέδια

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.

### 1. Σχεδίαση τετραγώνου:

```
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:



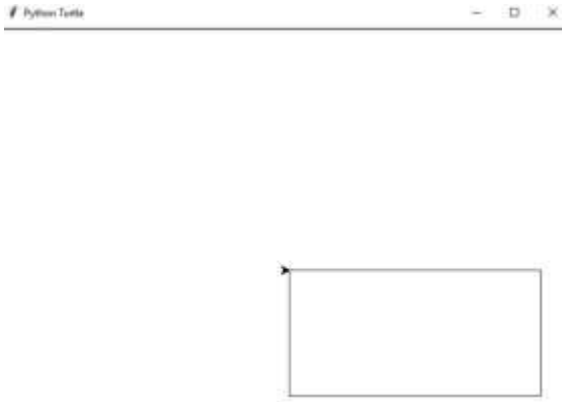

**Επεξήγηση:** Καθώς το τετράγωνο αποτελείται από 4 γραμμές, καθεμία συνδεδεμένη σε γωνία 90 μοιρών, σε αυτό έχουμε χρησιμοποιήσει όλες τις λειτουργίες εμπρός και δεξιά.
***tp.fd(100)*** : μετακινεί τη χελώνα 100 μονάδες προς τα εμπρός.

***tp.rt(90):*** στρέφει τη χελώνα 90 μοίρες προς τα δεξιά.
Με τον ίδιο τρόπο, η υπόλοιπη εντολή εκτελείται και παράγει το αποτέλεσμα.

## 2. 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:



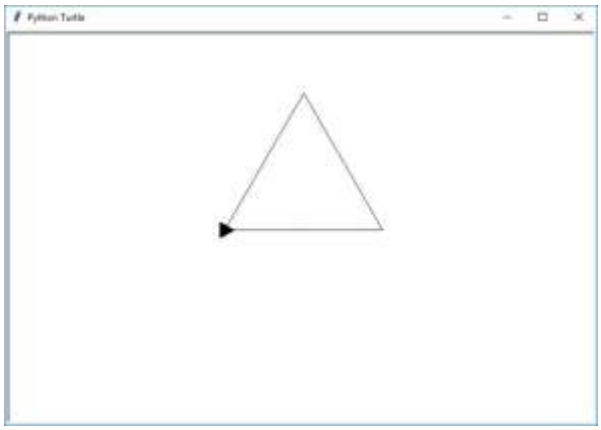
**Επεξήγηση:** Το ορθογώνιο αποτελείται από 4 γραμμές, καθεμία συνδεδεμένη σε γωνία 90 μοιρών, αλλά οι διπλανές πλευρές έχουν διαφορετικά μήκη. Έχουμε χρησιμοποιήσει τις συναρτήσεις για κίνηση εμπρός και δεξιά.
***tp.fd(300*)** : μετακινεί τη χελώνα 300 μονάδες προς τα εμπρός.
***tp.rt(90):*** στρέφει 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.

## 3. Σχεδίαση τριγώνου

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

```
tp.fd(100)
tp.lt(120)
tp.fd(100)
```

Αποτέλεσμα:



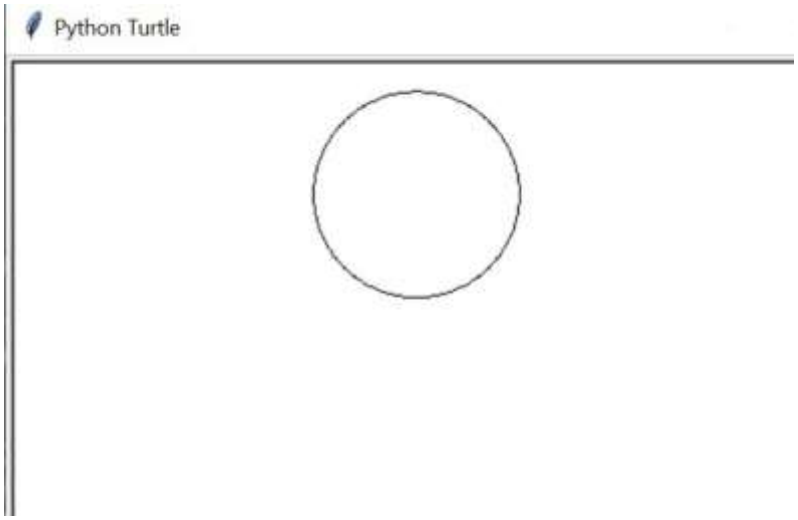
**Εξήγηση:** Ο παραπάνω κώδικας σχηματίζει ένα ισόπλευρο τρίγωνο δηλ. με όλες τις πλευρές του ίσες. Σε αυτό, χρησιμοποιήσαμε όλες τις λειτουργίες εμπρός και αριστερά.
***tp.fd(100)*** : μετακινεί τη χελώνα 100 μονάδες προς τα εμπρός.
***tp.lt(120):*** στρέφει τη χελώνα 120 μοίρες προς τα αριστερά.
Με τον ίδιο τρόπο, ο υπόλοιπος κώδικας εκτελείται και παράγει το αποτέλεσμα.

## 4. 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.

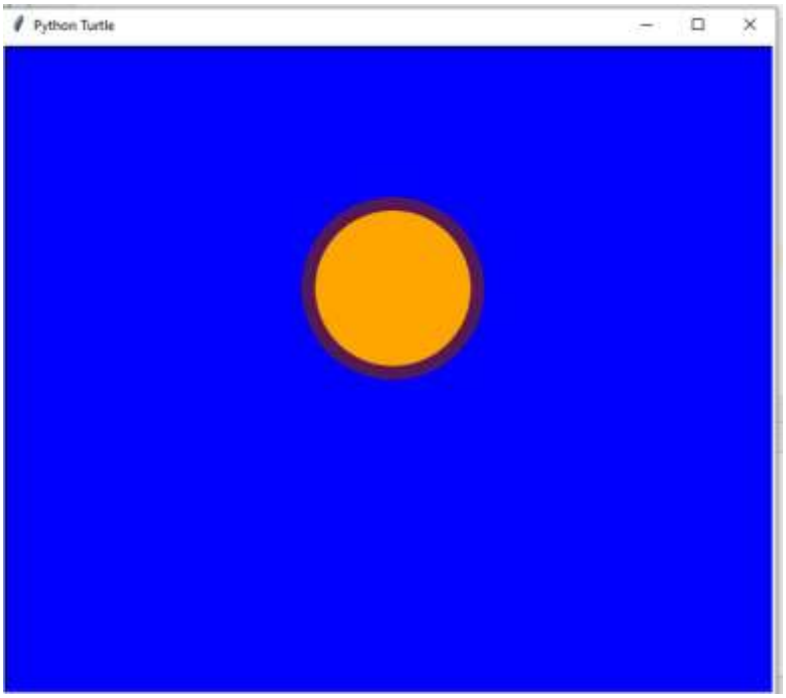
## 19.1.5 Σχήματα ενδιάμεσου Επιπέδου

Χρησιμοποιώντας αυτές τις συναρτήσεις, θα δημιουργήσουμε έναν προσαρμοσμένο κύκλο όπου ο χρήστης μπορεί να αλλάξει το χρώμα του , να του δώσει ένα περίγραμμα και να προσθέσει χρώμα σε αυτόν.

```
import turtle
tp = turtle.Turtle()
tp.pencolor("purple")
tp.fillcolor("orange")
tp.pensize(8)
tp.speed(5)
tp.begin_fill()
tp.circle(110)
turtle.bgcolor("blue")
```

***tp.end_fill()***

**Αποτέλεσμα:**




**Επεξήγηση:** Οι μέθοδοι που χρησιμοποιούνται είναι ***pencolor(), fillcolor(), pensize(), speed(), begin_fill()*** και ***end_fill()***.

***pencolor():*** Αυτή η μέθοδος βοηθά τον χρήστη να αλλάξει το χρώμα του πένα. Λαμβάνει τα ορίσματα με τη μορφή συμβολοσειρών χρώματος ή σε κωδικούς χρώματος RGB. Το προεπιλεγμένο χρώμα του μελανιού είναι μαύρο και επομένως αν δεν περάσουμε
ορίσματα, το χρώμα του μελανιού θ
α είναι μαύρο.
Εδώ, λοιπόν, χρησιμοποιήσαμε pencolor ("μωβ").

**fillcolor():** Όπως υποδηλώνει το ίδιο το όνομα, αυτή η μέθοδος χρησιμοποιείται για να γεμίσει το χρώμα στο σχέδιο. Εδώ, καθώς έχουμε περάσει το όρισμα «πορτοκαλί» στη μέθοδο ***fillcolor***, θα γεμίσει το εσωτερικό του σχήματος με πορτοκαλί χρώμα.

**pensize():**Αυτή η μέθοδος μας βοηθά να ορίσουμε το πλάτος του πένα. Παίρνει τα ορίσματα με τη μορφή πλάτους.
Έχουμε περάσει το "5" σε αυτήν τη συνάρτηση και ως εκ τούτου το πλάτος του πένα ή του περιγράμματος θα είναι 5 μονάδες.

**speed():** Αυτό απλώς αποφασίζει πόσο γρήγορα ή αργά θα κινηθεί η χελώνα με βάση τα ορίσματα που έχουμε περάσει. Η ταχύτητα κυμαίνεται μεταξύ 1-10.
Το 1 θεωρείται η πιο αργή, το 6 ως η κανονική ταχύτητα και το 10 ως η ταχύτερη.

**Begin_fill():** Όποτε υπάρχει ανάγκη να γεμίσουμε οποιοδήποτε σχήμα, αυτή η μέθοδος πρέπει να καλείται πριν από το σχήμα. Δεν γίνονται δεκτά ορίσματα.

**bgcolor():** Αυτό βοηθά στην αλλαγή του χρώματος του φόντου. Εδώ έχουμε βάλει μπλε χρώμα.

**end_fill():** Χρησιμοποιείται για να γεμίσει πραγματικά το σχήμα.

## 19.1.6 Κύκλοι μέσα σε κύκλο

Βασικά, θέλουμε να κάνουμε έναν κύκλο μέσα σε έναν μεγαλύτερο κύκλο και μετά έναν κύκλο μέσα στον δεύτερο μεγαλύτερο κύκλο και ούτω καθεξής. Πώς θα το κάνουμε αυτό; Υπάρχουν πολλοί τρόποι, αλλά ο καλύτερος είναι η χρήση των βρόχων.

Ας δούμε την περίπτωση χρήσης των βρόχων στον προγραμματισμό της χελώνας μας.

```
i=5
while i <= 55:
    tp.circle(i)
    i = i+10
```

Αποτέλεσμα:


**Εξήγηση**

***Γραμμή 1* :** Δηλώσαμε μια μεταβλητή “i” και ορίσαμε την τιμή της σε 5. Εδώ “i” είναι η ακτίνα του κύκλου που σχεδιάστηκε.
***Γραμμή* 2:** Χρησιμοποιήσαμε έναν βρόχο while με την προϋπόθεση ότι το "i" πρέπει να είναι μικρότερο ή ίσο με το 55.
***Γραμμή* 3** : Εδώ χρησιμοποιήσαμε τη συνάρτηση ***circle()*** για να σχεδιάσουμε τον κύκλο. Περάσαμε το «i» ως όρισμα.
***Γραμμή* 4** : Και στην τελευταία γραμμή, μετά από κάθε βρόχο, αυξάνουμε την τιμή του "i" (ακτίνα) κατά 10.
Μόλις η τιμή του "i" φτάσει στο 55, ο βρόχος τερματίζεται και έχουμε το απαιτούμενο σχέδιο.

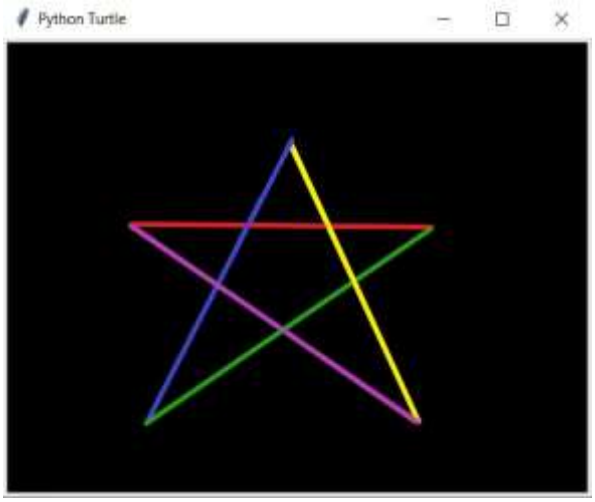
# 19.1.7 Πολύχρωμο αστέρι

Τώρα ας φτιάξουμε ένα πολύχρωμο αστέρι.

```
import turtle
import time
screen=turtle.Screen()
tp=turtle.Turtle()
screen.setup(420,320)
screen.bgcolor('black')
colour_set=['red','green','blue','yellow','purple']
```

```
tp.pensize(4)
tp.penup()
tp.setpos(-90,30)
tp.pendown()
for i in range(5):
    tp.pencolor(colour_set[i])
    tp.forward(200)
    tp.right(144)
tp.penup()
tp.setpos(80,-140)
tp.pendown()
tp.ht()
```

Αποτέλεσμα:




**Εξήγηση:** Ας δούμε πώς το σχεδιάσαμε

***Γραμμή 1 έως 4:*** Εισαγωγή βασικής βιβλιοθήκης turtle, αποθήκευση turtle σε μεταβλητή κ.λπ.

***Γραμμή 5:*** Χρησιμοποιείται για τον καθορισμό του μεγέθους του σχεδίου. Εδώ έχουμε περάσει δύο ορίσματα, πλάτος και ύψος, 420 και 320 αντίστοιχα.

***Γραμμή 6:*** Για να ορίσουμε το χρώμα φόντου της οθόνης σε μαύρο, χρησιμοποιήσαμε την bgcolor().

***Γραμμή 7:*** Αυτή είναι μια σειρά χρωμάτων που ονομάζεται colour_set. Είναι η λίστα με τα χρώματα που θέλουμε στο αστέρι μας.

***Γραμμή 8:*** το pensize(4) θα ορίσει το πλάτος του πένα σε 4.

***Γραμμή 9:***Η penup() βοηθά τον χρήστη να πάρει το πένα. Πολλές φορές θέλουμε να σηκώσουμε το πένα και δεν θέλουμε να αφήσουμε επιπλέον γραμμές στην οθόνη και έτσι χρησιμοποιείται η penup().

***Γραμμή 10:*** Η setpos(-90,30) θα μετακινήσει τη χελώνα στην επιθυμητή

θέση, Χρειάζονται 2 ορίσματα που βασικά έχουν τη μορφή συντεταγμένων x και y.
***Γραμμή 11:*** Η pendown() είναι η προεπιλεγμένη κατάσταση της χελώνας. Με αυτήν την εντολή, μπορούμε να διαβεβαιώσουμε ότι η χελώνα είναι έτοιμη να σχεδιάσει.
***Γραμμή 12:*** Σε αυτή τη γραμμή, έχουμε αρχικοποιήσει έναν βρόχο for. Αυτό γίνεται για να φτιάξουμε τις πλευρές του αστεριού.
***Γραμμή 13:*** tp.pencolor(colour_set[i]) ξεκινώντας την πρώτη φορά, έχει πρόσβαση στο πρώτο στοιχείο του πίνακα colour_set και θα λειτουργεί ως όρισμα στη μέθοδο pencolor().
***Γραμμή 14:***Το tp.forward(200) θα μετακινήσει τη χελώνα μπροστά κατά 200 μονάδες.
***Γραμμή 15:*** tp.right(144) στρέφει τη χελώνα κατά 144 μοίρες προς τα δεξιά.
***Γραμμές 16 έως 18:*** Εδώ χρησιμοποιήσαμε ξανά την penup(), ορίσαμε τη θέση της χελώνας σε x = 80 και y = -140 και τέλος χρησιμοποιήσαμε την pendown().
***Γραμμή 19:*** tp.ht() χρησιμοποιείται για να κρύψει τη χελώνα και αυτό θα κάνει τελικά το σχέδιό μας πιο ελκυστικό κρύβοντας ανεπιθύμητα πράγματα.

# 19.2.0 Προχωρημένο Επίπεδο

Ο προγραμματισμός χελωνών προηγμένου επιπέδου περιλαμβάνει περισσότερη χρήση συνθηκών, βρόχων κ.λπ. Όποτε υπάρχει ανάγκη δημιουργίας επαναλαμβανόμενων μοτίβων σχεδίασης ή συνθηκών παιχνιδιού, οι βρόχοι χρησιμοποιούνται ευρέως.
Μέχρι στιγμής μάθαμε πώς να φτιάχνουμε βασικά σχήματα, μετά μερικά σχέδια μεσαίου επιπέδου και τώρα είναι η ώρα να χρησιμοποιήσουμε επιτέλους τη χελώνα για την ανάπτυξη παιχνιδιών, για την οποία είναι και διάσημη.

Ας φτιάξουμε ένα παιχνιδάκι χρησιμοποιώντας τη βιβλιοθήκη turtle.

# 19.2.1 Η πιο γρήγορη χελώνα

Ο στόχος αυτού του παιχνιδιού είναι να έχουμε δύο χελώνες και τους συγκεκριμένους στόχους που πρέπει να φτάσουν.

Θα είναι ένα παιχνίδι δύο παικτών, όπου κάθε παίκτης θα ρίξει ένα ζάρι και η χελώνα θα μετακινηθεί σ' αυτό τον αριθμό βημάτων και το ίδιο ισχύει και για τον δεύτερο παίκτη.

Το παιχνίδι θα τελειώσει μόλις κάποιος από τους παίκτες φτάσει στον προορισμό-στόχο.

**Ξεκινώντας:**
**Βήμα 1:** Εισαγωγή απαιτούμενων βιβλιοθηκών

**Βήμα 2:** Εργασία στο σχεδιαστικό μέρος που είναι οι χελώνες και ο προορισμός στόχος.

**Βήμα 3:** Κωδικοποίηση λογικού μέρους. Πώς και πότε πρέπει να κινηθεί η χελώνα, τα κριτήρια νίκης κ.λ.π..

**<u>Βήμα 1</u> : Ρύθμιση του περιβάλλοντος και εισαγωγή της βιβλιοθήκης**

```
import turtle
import random
```

**Επεξήγηση:** Εδώ έχουμε εισαγάγει δύο βιβλιοθήκες, την ***turtle*** και την ***random***. Η βιβλιοθήκη ***turtle*** θα μας βοηθήσει στο σχεδιαστικό κομμάτι, ενώ η ***random*** θα βοηθήσει στη δημιουργία του λογικού μέρους του παιχνιδιού.
Η ***random*** θα μας χρειαστεί για την επιλογή τυχαίων στοιχείων για το ρίξιμο των ζαριών.

**<u>Βήμα 2</u> : Εργασία στο σχεδιαστικό μέρος**

Σε αυτό το βήμα, πρέπει να δημιουργήσουμε δύο χελώνες και τον προορισμό τους, δηλαδή έναν κύκλο. Το χρώμα της κάθε χελώνας πρέπει να είναι διαφορετικό και ο προορισμός τους πρέπει να ταιριάζει με το αντίστοιχο χρώμα.

**Κώδικας για τις δύο χελώνες:**

```
p_one = turtle.Turtle()
p_one.color("purple")
```

```
p_one.shape("turtle")
p_one.penup()
p_one.goto(-200,100)
p_two = p_one.clone()
p_two.color("orange")
p_two.penup()
p_two.goto(-200,-100)
```

**Επεξήγηση:** Χρησιμοποιήσαμε ένα μωβ χρώμα για τον παίκτη 1 και ένα πορτοκαλί χρώμα για τον παίκτη 2.
***p_one.shape("turtle"):*** Αυτή η συνάρτηση shape() ορίζει το σχήμα της χελώνας σε αυτό που μεταβιβάζεται ως όρισμα.
***p_one.goto(-200,100):*** Αυτό θα ορίσει τη θέση της χελώνας σε x = -200 και y = 100 (Αρχή).
***p_one.clone():*** Αυτή η μέθοδος απλώς αντιγράφει τα χαρακτηριστικά της χελώνας ***p_one***.

Η χελώνα του παίκτη δύο φτιάχτηκε κλωνοποιώντας τη χελώνα του παίκτη ένα, αλλάζοντας το χρώμα της και μεταφέροντάς την σε μια νέα αρχική θέση.

Πρέπει να έχουμε υπόψη μας ότι μόλις σχηματιστούν οι χελώνες, πρέπει να τις τοποθετήσουμε στις αρχικές τους θέσεις και να διασφαλίσουμε ότι είναι ευθυγραμμισμένες.

Πρέπει τώρα να δημιουργήσουμε σπίτια για τις χελώνες. Αυτά τα σπίτια θα χρησιμεύσουν ως προορισμοί των χελωνών.

Ένας κύκλος θα αντιπροσωπεύει το καθένα από τα σπίτια των χελωνών. Πρέπει να διασφαλίσουμε ότι και οι δύο προορισμοί βρίσκονται σε ίση απόσταση από το σημείο εκκίνησης σε αυτήν την περίπτωση:

Κώδικας για τους προορισμούς:

```
p_one.goto(300,60)
p_one.pendown()
p_one.circle(40)
p_one.penup()
p_one.goto(-200,100)
p_two.goto(300,-140)
```

```
p_two.pendown()
p_two.circle(40)
p_two.penup()
p_two.goto(-200,-100)
```

**Επεξήγηση:** Για το σχεδιασμό δύο προορισμών, χρησιμοποιήσαμε έναν κύκλο (). Για να ορίσουμε τη θέση των προορισμών χρησιμοποιήσαμε τη μέθοδο goto().

**<u>Βήμα 3:</u> Κωδικοποίηση για το λογικό μέρος.**

Ας δούμε πώς θα υλοποιήσουμε το υπόλοιπο παιχνίδι.

Επειδή θα χρησιμοποιήσουμε βρόχους και υπό όρους συνθήκες, θα πρέπει να προσέξουμε το indentation.

Αυτή η διαδικασία επαναλαμβάνεται έως ότου μία από τις χελώνες πετύχει τον στόχο, οπότε και τελειώνει το παιχνίδι.

Ας δούμε μια έκδοση του κώδικα:

```
die = [1,2,3,4,5,6]
for i in range(20):
    if p_one.pos() >= (300,100):
        print("Player One Wins!")
        break
    elif p_two.pos() >= (300,-100):
        print("Player Two Wins!")
        break
    else:
        p_one_turn = input("Πατήστε το 'Enter' για να ρίξετε το ζάρι ")
        die_outcome = random.choice(die)
        print("Φέρατε: ")
        print(die_outcome)
        print("Ο αριθμός των βημάτων είναι: ")
        print(20*die_outcome)
        p_one.fd(20*die_outcome)
        p_two_turn = input("Πατήστε το 'Enter' για να ρίξετε το ζάρι ")
```

```
die_outcome = random.choice(die)
print("Φέρατε: ")
print(die_outcome)
print("Ο αριθμός των βημάτων είναι: ")
print(20*die_outcome)
p_two.fd(20*die_outcome)
```

**Επεξήγηση:**
***Γραμμή 1:*** Αυτή η γραμμή είναι μια μεταβλητή που παρατίθεται και αποθηκεύει τους αριθμούς που κυμαίνονται από το 1 έως το 6, δηλαδή το ζάρι.
**Γραμμή 2:** Αυτή η γραμμή κώδικα αντιπροσωπεύει τον βρόχο for που θα τρέχει στην περιοχή από 1 έως 20.
**Γραμμές 3 έως 8:** Αυτές οι γραμμές ελέγχουν την κατάσταση του παίκτη που έχει κερδίσει ή χάσει.
Με βάση τη θέση του βρόχου αποφασίζεται εάν θα τερματιστεί ο βρόχος ή όχι. Εάν επιτευχθεί η καθορισμένη συνθήκη, τότε ο βρόχος διακόπτεται και εκτυπώνει τις δηλώσεις.
***Γραμμή 9:*** Εάν κανένας παίκτης δεν έχει κερδίσει, τότε η ροή του κώδικα μετακινείται στην πρόταση else.
***Γραμμή 10:*** Ζητάει από τον χρήστη να ρίξει το ζάρι πατώντας το πλήκτρο Enter.
***Γραμμή 11:***Επιλέγει ένα τυχαίο στοιχείο από τη λίστα die και εκχωρεί την τιμή στη μεταβλητή die_outcome.
***Γραμμή 12:*** Αυτή η γραμμή εκτυπώνει τη γραμμή "Φέρατε: "
***Γραμμή 13:*** Αυτή η γραμμή εκτυπώνει την τιμή που είναι αποθηκευμένη στο die_outcome
***Γραμμή 14:*** Ίδια με τη γραμμή 11.
***Γραμμή 15:*** Αυτό γίνεται για να μειωθεί ο αριθμός των βημάτων που απαιτείται.
***Γραμμή 16:*** Πάρτε τη χελώνα προς την κατεύθυνση προώθησης.
***Γραμμές 17 έως 23:*** Αυτές οι γραμμές είναι ίδιες για τον παίκτη 2. Οι ίδιες λειτουργίες ισχύουν και για τον παίκτη 2.

Ο βρόχος for συνεχίζει τη ροή του έως ότου οποιοσδήποτε από τους παίκτες φτάσει στον προορισμό του.

Εδώ είναι το τέλος του παιχνιδιού μας. Προσπαθήστε να προσθέσετε νέες λειτουργίες σε αυτό το παιχνίδι. Κάντε τη διεπαφή χρήστη να φαίνεται καλή, προσθέστε χρώμα φόντου, προσθέστε εικόνες, αλλάξτε

την ταχύτητα της χελώνας κ.λπ. Αυτή η βιβλιοθήκη είναι το μόνο που χρειάζεται κάποιος για να αναπτύξει καταπληκτικά έργα.

**Η επίσημη τεκμηρίωση της βιβλιοθήκης turtle βρίσκεται εδώ:** Python Turtle Library

## 19.3.0 Συμμετρικό σχέδιο

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.

```
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)
        for side in range(4):
            pen.forward(50)
            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:*