20. ΚΑΛΟΚΑΙΡΙΝΑ PROJECTS III - PYTHON TURTLE GAMES

20.1.0 Εισαγωγή

Συνεχίζοντας τα καλοκαιρινά μας projects, ας δημιουργήσουμε ένα ακόμα παιχνιδάκι, λίγο μεγαλύτερο, αλλά και λίγο καλύτερο και πιο ενδιαφέρον. Όπως ήδη γνωρίζουμε, η python turtle, έχει ήδη όλες τις απαραίτητες βιβλιοθήκες ώστε να δημιουργήσουμε πολλά και ενδιαφέροντα παιχνίδια.

Το Space Invaders το οποίο και θα αναπτύξουμε, είναι ένα παιχνίδι τύπου shooter στο οποίο ο παίκτης πυροβολεί εξωγήινα σκάφη που βρίσκονται από πάνω του και κατεβαίνουν σιγά-σιγά, απειλητικά. Η «αναχαίτιση» γίνεται μετακινώντας ένα σκάφος οριζόντια στο κάτω μέρος της οθόνης, το οποίο με ένα όπλο ρίχνει βλήματα στους εχθρούς.

Σαν ομάδα, τα εξωγήινα σκάφη ταξιδεύουν αριστερά και δεξιά, μετακινούμενοι προς τα κάτω καθώς πλησιάζουν στην άκρη της οθόνης. Ο στόχος είναι να πυροβολήσουμε όλα τα σκάφη μέχρι να τα εξαλείψουμε όλα.

Το παιχνίδι τελειώνει γρήγορα εάν οι εισβολείς φτάσουν στο κάτω μέρος της οθόνης ενώ ο παίκτης έχει τρεις ζωές (ή ότι άλλο εμείς ορίσουμε, σαν δημιουργοί του παιχνιδιού).

Μπορούμε να κατεβάσουμε τα στοιχεία του παιχνιδιού από τον φάκελο που ήδη βρίσκεται στο GDrive μας εδώ.

20.1.1 Κανόνες παιχνιδιού

Όταν τα εξωγήινα σκάφη φτάνουν στην άκρη της οθόνης, θα πρέπει να κατεβαίνουν κατά ένα επίπεδο προς τα κάτω, προς το σκάφος του παίκτη.

Όταν οι εξωγήινοι φτάσουν στο κάτω μέρος της οθόνης, το παιχνίδι πρέπει να τελειώσει.

Το σκάφος του παίκτη μπορεί να κινηθεί μόνο προς τα αριστερά ή τα δεξιά.

Θα πρέπει να υπάρχει μετρητής βαθμολογίας. Προσθέτουμε 10 πόντους για κάθε επιτυχές χτύπημα εξωγήινου.

20.1.2 Εισαγωγή των βιβλιοθηκών

import turtle

import math

import random

Θα χρησιμοποιήσουμε όλες τις βασικές βιβλιοθήκες για αυτό το έργο. Όλοι γνωρίζουμε αυτές τις παραπάνω βιβλιοθήκες.

turtle για τα γραφικά του παιχνιδιού μας,

math για τη χρήση μαθηματικών συναρτήσεων και

random για τη δημιουργία τυχαίων αριθμών σε μια δεδομένη περιοχή.

20.1.3 Ρύθμιση της οθόνης του παιχνιδιού και του φόντου

window = turtle.Screen()

window.bgcolor("green")

window.title("Space Invaders - CopyAssignment")

window.bgpic("background.gif")

Στη γραμμή 1 : Σε αυτή τη γραμμή, χρησιμοποιούμε τη συνάρτηση screen() που δείχνει όλες τις χελώνες που βρίσκονται στην οθόνη.

Γραμμή 2 : Εδώ έχουμε ορίσει το χρώμα φόντου σε πράσινο χρησιμοποιώντας τη συνάρτηση bgcolor().

Γραμμή 3: Η συνάρτηση title() της βιβλιοθήκης της χελώνας χρησιμοποιείται για να ορίσετε τον τίτλο του παραθύρου. Εδώ έχουμε μεταβιβάσει τον τίτλο ως "Space Invaders - CopyAssignment"

Γραμμή 4: Για να ορίσουμε την εικόνα φόντου, χρησιμοποιούμε τη συνάρτηση bgpic().

20.1.4 Καταχώρηση των σχημάτων μας

Ένα από τα καλύτερα χαρακτηριστικά της βιβλιοθήκης turtle είναι η καταχώριση των δικών μας προσαρμοσμένων εικονιδίων και εικόνων στη βιβλιοθήκη όπως εικόνες, gif κ.λπ. Η συνάρτηση ***reigster_shape()*** χρησιμοποιείται για αυτόν τον σκοπό. Στην ουσία για να «εισάγει» τα δικά μας γραφικά στη βιβλιοθήκη, έτοιμα για χρήση.

turtle.register_shape("invader.gif")

turtle.register_shape("player.gif")

Στις γραμμές 1 & 2: Χρησιμοποιούμε τη ***register_shape*** για να καταχωρήσουμε τις δικές μας εικόνες. Εισάγουμε δύο αρχεία gif για αυτό το σκοπό.

20.1.5 Προσθήκη περιγράμματος

```
border_pen = turtle.Turtle()
border_pen.speed(0)
border_pen.color("white")
border_pen.penup()
border_pen.setposition(-300,-300)
border_pen.pendown()
border_pen.pensize(3)
for side in range(4):
    border_pen.fd(600)
    border_pen.lt(90)
border_pen.hideturtle()
```

Στη γραμμή 1 : ***turtle.Turtle()*** έχουμε τη δημιουργία του αντικειμένου.

Στη γραμμή 2 : Η ταχύτητα της χελώνας ρυθμίζεται χρησιμοποιώντας τη συνάρτηση ***speed()***

Γραμμή 3: Η αλλαγή του χρώματος της πένας είναι δυνατή χρησιμοποιώντας τη συνάρτηση ***color()***. Εδώ έχουμε ρυθμίσει το χρώμα της πένας σε λευκό.

Γραμμή 4: Η ***penup()*** σηκώνει την πένα και μπορεί να τη μετακινεί, χωρίς να «γράφει».

Γραμμή 5: η ***setposition()*** ορίζει τη θέση του δρομέα στην επιθυμητή θέση. Παίρνει τιμές με τη μορφή συντεταγμένων X και Y.

Γραμμή 6:Η ***pendown()*** είναι η προεπιλεγμένη κατάσταση της χελώνας. Με αυτήν την εντολή, μπορούμε να βεβαιώσουμε ότι η χελώνα είναι έτοιμη να σχεδιάσει.

Γραμμή 7: Σε αυτή τη γραμμή χρησιμοποιείται η ***pensize().*** Αυτή η μέθοδος μας βοηθά να ορίσουμε το πλάτος της πένας. Παίρνει ένα όρισμα για το πλάτος.

Γραμμή 8 έως 10: Εδώ, χρησιμοποιούμε τον βρόχο for που θα εκτελέσει τις εντολές στον βρόχο. Έχουμε χρησιμοποιήσει μεθόδους fd και lt για να μετακινήσουμε τη χελώνα προς τα εμπρός και να τη στρέψουμε προς τα αριστερά αντίστοιχα.

Γραμμή 11: ***tp.hideturtle()*** χρησιμοποιείται για την απόκρυψη της χελώνας και αυτό θα κάνει τελικά το σχέδιό μας πιο όμορφο κρύβοντας ανεπιθύμητα αντικείμενα.

20.1.6 Εμφάνιση βαθμολογίας

score = 0

score_pen = turtle.Turtle()

score_pen.speed(0)

score_pen.color("red")

score_pen.penup()

score_pen.setposition(-290, 280)

scorestring = "SCORE: %s" %score

score_pen.write(scorestring, move=False, align="left", font=("Arial", 14, "normal"))

score_pen.hideturtle()

Γραμμή 1: η μεταβλητή σκορ θα παρακολουθεί τη βαθμολογία του παίκτη. Την αρχικοποιούμε στο 0.

Γραμμή 2: Δημιουργούμε ένα αντικείμενο το οποίο κινείται και έχει χαρακτηριστικά (μια χελώνα). Η turtle() θα δημιουργήσει μια συνάρτηση κλήσης με αναφορά στο score_pen, οπότε κάθε φορά που θέλουμε να έχουμε πρόσβαση μπορούμε να τη χρησιμοποιούμε δίνοντας τη μεταβλητή score_pen.

Γραμμή 3: Ορίζουμε την ταχύτητα της πένας.

Γραμμή 4: Χρησιμοποιούμε τη συνάρτηση ***color()*** για να βάλουμε το χρώμα της πένας. Στην περίπτωσή μας, θα είναι το «SCORE» του οποίου το χρώμα θα είναι κόκκινο δίνοντας χρώμα («κόκκινο») στην πένα.

Γραμμή 5: Σηκώνεται η πένα και μετακινείται σε μια νέα τοποθεσία χωρίς να αφήνει ίχνη.

Στη γραμμή 6: Για να ορίσουμε τη θέση της πένας χρησιμοποιούμε τη συνάρτηση setposition(), όχι μόνο θα ορίσει τη θέση αλλά θα την κάνει και απόλυτη.

Γραμμή 7: Η συμβολοσειρά που θέλουμε να γράψουμε με την πένα, στην περίπτωσή μας, θα γραφτεί "SCORE:".

Γραμμή 8: Η θέση της συμβολοσειράς που γράφτηκε στην προηγούμενη γραμμή ορίζεται με τη βοήθεια της συνάρτησης write(). Το Scorestring αποθηκεύει τη βαθμολογία, τότε η κίνηση είναι ψευδής σε αυτήν την περίπτωση, η συμβολοσειρά μπορεί να ευθυγραμμιστεί "αριστερά, δεξιά ή στο κέντρο" σύμφωνα με την επιλογή μας και στο τέλος, δίνεται η συμβολοσειρά στο στυλ γραμματοσειράς.

Γραμμή 9: Τέλος, για να κρύψουμε τη χελώνα χρησιμοποιούμε την εντολή hideturtle().

20.1.7 Δημιουργία του σκάφους του παίκτη

player = turtle.Turtle()

player.shape("player.gif")

player.penup()

player.speed(0)

player.setposition(0,-250)

player.setheading(90)

playerspeed = 15

Εδώ έχουμε χρησιμοποιήσει πολλαπλές λειτουργίες για να δημιουργήσουμε το σκάφος του παίκτη. Το σκάφος του παίκτη προστίθεται χρησιμοποιώντας το shape(player.gif) .

Χρησιμοποιούμε τη συνάρτηση speed() για να ορίσουμε την ταχύτητα του σκάφους. Για να ορίσουμε τη θέση του, χρησιμοποιούμε τη setposition() με συντεταγμένη X = 0 και συντεταγμένη Y = 250.

player = turtle.Turtle()

Δημιουργία εχθρού και καθορισμός της θέσης του:

number_of_enemies = 10

enemies = []

for i in range(number_of_enemies):

   enemies.append(turtle.Turtle())

for enemy in enemies:

   enemy.shape("invader.gif")

   enemy.penup()

   enemy.speed(0)

   x = random.randint(-200, 200)

   y = random.randint(100, 250)

   enemy.setposition(x, y)

enemyspeed = 5

Γραμμές 1 & 2: Αριθμός εχθρών και κενή λίστα εχθρών.

Στη γραμμή 3: ο βρόχος for θα εκτελεστεί 9 φορές και θα προσαρτήσει τη turtle.Turtle() στη λίστα εχθρών.

Γραμμή 4: Και πάλι ένας βρόχος for.

Γραμμή 5: Το σχήμα του εχθρού ορίζεται χρησιμοποιώντας το shape()

Στη γραμμή 6: penup() για να σηκώσει το στυλό χωρίς να αφήσει κανένα ίχνος.

Γραμμή 7: Η ταχύτητα των εχθρών έχει οριστεί στο 0.

Στη γραμμή 8,9 & 10: Εδώ χρησιμοποιούμε τη συνάρτηση randint της τυχαίας βιβλιοθήκης για να ορίσουμε τυχαία τη θέση του εχθρού.

Γραμμή 11: Η μεταβλητή εχθρική ταχύτητα έχει οριστεί στο 5.

20.1.8 Δημιουργία εχθρού και καθορισμός της θέσης του

```
number_of_enemies = 10
enemies = []

for i in range(number_of_enemies):
    enemies.append(turtle.Turtle())
for enemy in enemies:
    enemy.shape("invader.gif")
    enemy.penup()
    enemy.speed(0)
    x = random.randint(-200, 200)
    y =  random.randint(100, 250)
```

enemy.setposition(x, y)

enemyspeed = 5

Γραμμές 1 & 2: Αριθμός εχθρών και κενή λίστα εχθρών.

Στη γραμμή 3: ο βρόχος for θα εκτελεστεί 9 φορές και θα προσαρτήσει τη ***turtle.Turtle()*** στη λίστα εχθρών.

Γραμμή 4: Και πάλι ένας βρόχος for.

Γραμμή 5: Το σχήμα του εχθρού ορίζεται χρησιμοποιώντας το ***shape()***

Στη γραμμή 6: ***penup()*** για να μην αφεθεί κανένα ίχνος.

Γραμμή 7: Η ταχύτητα των εχθρών έχει οριστεί στο 0.

Στη γραμμή 8,9 & 10: Εδώ χρησιμοποιούμε τη συνάρτηση ***randint*** της βιβλιοθήκης random για να ορίσουμε τυχαία τη θέση του εχθρού.

Γραμμή 11: Η μεταβλητή ***enemyspeed*** έχει οριστεί στο 5.

20.1.9 Δημιουργία σφαίρας σκάφους

bullet = turtle.Turtle()

bullet.color("white")

bullet.shape("triangle")

bullet.penup()

bullet.speed(0)

bullet.setheading(90)

bullet.shapesize(0.5,0.5)

bullet.hideturtle()

bulletspeed = 30

bulletstate = "ready"

Το χρώμα της σφαίρας ορίζεται σε λευκό χρησιμοποιώντας τη συνάρτηση ***color()***. Για να ορίσουμε το σχήμα της κουκκίδας σε τρίγωνο, χρησιμοποιούμε τη συνάρτηση ***shape()***.

Και για να ορίσουμε το μέγεθος του τριγώνου, χρησιμοποιούμε τη συνάρτηση ***shapesize()***. Δηλώσαμε μια μεταβλητή ***bulletspeed*** και εκχωρήσαμε μια τιμή 30. Το ***bulletstate*** αποθηκεύει την κατάσταση της κουκκίδας.

20.1.10 Συνάρτηση μετακίνησης του σκάφους μας

```
def move_left():
    x = player.xcor()
    x -= playerspeed
    if x < -280:
        x = -280
    player.setx(x)

def move_right():
    x = player.xcor()
    x += playerspeed
    if x > 280:
        x = 280
    player.setx(x)
```

move_left(): Αυτή η συνάρτηση μπορεί να μετακινήσει το σκάφος προς τα αριστερά. Χρησιμοποιούμε τη συνάρτηση xcor() για να πάρουμε την τιμή συντεταγμένων X της χελώνας για την τρέχουσα θέση της χελώνας. Για να περιορίσουμε την κίνηση της χελώνας, χρησιμοποιούμε έναν υπό συνθήκη βρόχο.

move_right: Αυτή η λειτουργία βοηθά στη μετακίνηση του σκάφοςυ του παίκτη προς τα δεξιά. Οι υπόλοιπες εντολές σε αυτή τη συνάρτηση είναι ίδιες με τη συνάρτηση move_left.

20.1.11 Συνάρτηση για την εκτόξευση σφαιρών

```
def fire_bullet():
    global bulletstate
    if bulletstate == "ready":
        bulletstate = "fire"
        # Τοποθετούμε τη σφαίρα ακριβώς πάνω από τη θέση του σκάφους του παίκτη
        x = player.xcor()
        y = player.ycor() + 10
        bullet.setposition(x,y)
        bullet.showturtle()
```

fire_bullet(): Αυτή η συνάρτηση χρησιμοποιείται για τον έλεγχο της εκτόξευσης της σφαίρας.

Γραμμή 1: Δηλώνεται μια καθολική μεταβλητή ***bulletstate***.

Γραμμή 2: Χρησιμοποιούμε μια συνθήκη που ελέγχει την κατάσταση των σφαιρών. Εάν η σφαίρα είναι έτοιμη, τότε η ροή θα εκτελέσει τον κώδικα μέσα στο μπλοκ if.

Στη γραμμή 3: Η πρώτη γραμμή του μπλοκ if αλλάζει την κατάσταση της σφαίρας σε πυροδότηση.

Στη γραμμή 4 έως 5: Για να μετακινήσουμε την κουκκίδα και να δώσουμε στον παίκτη μια λειτουργικότητα χρησιμοποιούμε τα xcor και ycor για να πάρουμε τιμές συντεταγμένων x και y.

Γραμμή 6 έως 7: Χρησιμοποιούμε τη θέση ρύθμισης για να ορίσουμε τη θέση της σφαίρας. Οι παραπάνω δύο γραμμές και αυτές οι δύο γραμμές χρησιμοποιούνται βασικά για τον καθορισμό της θέσης της σφαίρας ακριβώς πάνω από το σκάφος.

20.1.12 Λειτουργία ελέγχου σύγκρουσης

```
def isCollision_enemy_bullet(t1, t2):
    distance = math.sqrt(math.pow(t1.xcor()-
t2.xcor(),2)+math.pow(t1.ycor()-t2.ycor(),2))
    if distance < 25:
        return True
    else:
        return False


def isCollision_enemy_player(t1, t2):
    distance = math.sqrt(math.pow(t1.xcor()-
t2.xcor(),2)+math.pow(t1.ycor()-t2.ycor(),2))
    if distance < 30:
        return True
    else:
        return False
```

isCollision_enemy_bullet() **:** Αυτή η συνάρτηση χρησιμοποιείται για τον έλεγχο της σύγκρουσης της σφαίρας με τον εχθρό. Χρησιμοποιούμε τη

math για να υπολογίσουμε την απόσταση και βάσει αυτής της απόστασης χρησιμοποιούμε τις υπό όρους συνθήκες.

isCollision_enemy_player(): Αυτή η συνάρτηση φροντίζει για τη σύγκρουση μεταξύ του σκάφους και της σφαίρας.

20.1.13 Χειρισμός εισόδων πληκτρολογίου

turtle.listen()

turtle.onkey(move_left, "Left")

turtle.onkey(move_right, "Right")

turtle.onkey(fire_bullet, "space")

Γραμμή 1: Το ***turtle.listen()*** μας βοηθά να ακούμε και να ανιχνεύουμε πότε ο χρήστης πατάει πλήκτρα στο πληκτρολόγιο ή το ποντίκι.

Γραμμή 2: Χρησιμοποιούμε την ***onkey()*** για να «ακούμε» τα συμβάντα. Σε αυτή τη συνάρτηση, έχουμε περάσει τη συνάρτηση move_left για να μετακινούμε το σκάφος προς τα αριστερά.

Γραμμή 3: Χρησιμοποιούμε ξανά την ***onkey()*** για να ακούσουμε τα γεγονότα. Σε αυτή τη συνάρτηση, έχουμε περάσει τη συνάρτηση move_right για να μετακινούμε το σκάφος προς τα δεξιά.

Γραμμή 4: Αυτή τη φορά περάσαμε τη συνάρτηση ***fire_bullet*** για να χειριστούμε την πυροδότηση της σφαίρας.

20.1.14 Λογική του παιχνιδιού

Game_Over = False

missed_enemies = 0

while True:

```
for enemy in enemies:
    # Move the enemy
    x = enemy.xcor()
    x += enemyspeed
    enemy.setx(x)

    # Move the enemy back and down
    if enemy.xcor() > 270:
        # Move all enemies down
        for e in enemies:
            y = e.ycor()
            y -= 40
            e.sety(y)
            if e.ycor() < -285 and Game_Over == False:
                e.hideturtle()
                missed_enemies += 1
                if missed_enemies == 5:
                    Game_Over = True
                x = random.randint(-200, 200)
                y = random.randint(100, 250)
                e.setposition(x, y)
                e.showturtle()
        # Change enemy direction
        enemyspeed *= -1

    if enemy.xcor() < -270:
```

```
    # Move all enemies down
    for e in enemies:
        y = e.ycor()
        y -= 40
        e.sety(y)
        if e.ycor() < -285 and Game_Over == False:
            e.hideturtle()
            missed_enemies += 1
            if missed_enemies == 5:
                Game_Over = True
            x = random.randint(-200, 200)
            y = random.randint(100, 250)
            e.setposition(x, y)
            e.showturtle()
    # Change enemy direction
    enemyspeed *= -1

# check for a collision between the bullet and the enemy
if isCollision_enemy_bullet(bullet, enemy):
    # Reset the bullet
    bullet.hideturtle()
    bulletstate = "ready"
    bullet.setposition(0, -400)
    # Reset the enemy
    x = random.randint(-200, 200)
    y = random.randint(100, 250)
```

```
            enemy.setposition(x, y)
            enemyspeed += 0.5
            # update the score
            score += 10
            scorestring = "Score: %s" % score
            score_pen.clear()
            score_pen.write(
                scorestring, False, align="left", font=("Arial", 14, "normal")
            )
        # check for a collision between the player and enemy
        if isCollision_enemy_player(player, enemy):
            Game_Over = True
        if Game_Over == True:
            player.hideturtle()
            bullet.hideturtle()
            for e in enemies:
                e.hideturtle()
            window.bgpic("end.gif")
            break

    # Move the bullet
    if bulletstate == "fire":
        y = bullet.ycor()
        y += bulletspeed
        bullet.sety(y)
```

```
# Check to see if the bullet has gone to the top
if bullet.ycor() > 275:
    bullet.hideturtle()
    bulletstate = "ready"
```

Χρησιμοποιούμε έναν βρόχο while για τη συνεχή ροή του κώδικα.

Μέσα στον βρόχο χρησιμοποιούμε ένα for loop που βοηθάει στη συνεχή κίνηση των εχθρών.

Στη συνέχεια χρησιμοποιούμε ένα μπλοκ if και μέσα σε αυτό το μπλοκ χρησιμοποιούμε έναν βρόχο for που βοηθά στον καθορισμό της θέσης των εχθρών και στην επανατοποθέτηση νέων εχθρών.

Αυτό γίνεται ελέγχοντας την τιμή των x συντεταγμένων τους (καθώς κινούνται από αριστερά προς τα δεξιά). Χρησιμοποιώντας αυτή την υπό όρους συνθήκη, υπολογίζουμε επίσης πότε πρέπει να τελειώσει το παιχνίδι.

Αυτό γίνεται όταν οποιοδήποτε από τα εχθρικά πλοία αγγίξει το σκάφος του παίκτη. Αυτή η λειτουργία αντιμετωπίζεται επίσης μέσα σε αυτό το μπλοκ.

Για να ελέγξουμε τη σύγκρουση μεταξύ των εχθρών και της σφαίρας χρησιμοποιούμε τη συνάρτηση isCollision_enemy_bullet() και isCollision_enemy_player().

Εάν υπάρξει σύγκρουση μεταξύ της σφαίρας και του εχθρού, τότε θα πρέπει να αυξηθεί η βαθμολογία και αυτό αντιμετωπίζεται από μια συνάρτηση που ονομάζεται isCollision_enemy_bullet()

Τώρα έξω από το βρόχο for, ελέγχουμε την κατάσταση της σφαίρας, δηλαδή εάν έχει εκτοξευθεί ή όχι.

Και τέλος, εάν η σφαίρα εκτοξευθεί και πάει στην κορυφή, τότε η κατάσταση της επόμενης σφαίρας θα πρέπει να είναι έτοιμη. Στο τέλος, ελέγχουμε αυτήν την κατάσταση.

20.1.15 Έναρξη του παιχνιδιού:

turtle.done()

Για την έναρξη των βρόχων συμβάντος χρησιμοποιείται η turtle.done(). Δεν χρειάζεται να περάσετε κανένα όρισμα σε αυτή τη συνάρτηση.

20.1.16 Όλος ο κώδικας

```
# Εισαγωγή των απαραίτητων modules
import turtle
import math
import random

# Δημιουργία της οθόνης του παιχνιδιού
window = turtle.Screen()
window.bgcolor("green")
window.title("Space Invaders - CopyAssignment")
window.bgpic("background.gif")

# Εγγραφή του σχήματος
turtle.register_shape("invader.gif")
turtle.register_shape("player.gif")

# «Ζωγράφισμα» του περιγράμματος
border_pen = turtle.Turtle()
border_pen.speed(0)
border_pen.color("white")
```

```
border_pen.penup()
border_pen.setposition(-300, -300)
border_pen.pendown()
border_pen.pensize(3)
for side in range(4):
    border_pen.fd(600)
    border_pen.lt(90)
border_pen.hideturtle()

# Αρχικοποίηση του σκορ
score = 0

# Δημιουργία της πένας για εγγραφή του σκορ
score_pen = turtle.Turtle()
score_pen.speed(0)
score_pen.color("red")
score_pen.penup()
score_pen.setposition(-290, 280)
scorestring = "SCORE: %s" % score
score_pen.write(scorestring, False, align="left", font=("Arial", 14,
"normal"))
score_pen.hideturtle()

# Δημιουργία του σκάφους του παίκτη
player = turtle.Turtle()
player.shape("player.gif")
player.penup()
```

```
player.speed(0)
player.setposition(0, -250)
player.setheading(90)

playerspeed = 15

# Αριθμός εχθρικών σκαφών
number_of_enemies = 10
# Δημιουργία κενής λίστας εχθρικών σκαφών
enemies = []

# Προσθήκη εχθρικών σκαφών στη λίστα
for i in range(number_of_enemies):
    # δημιουργία εχθρικών σκαφών
    enemies.append(turtle.Turtle())

for enemy in enemies:
    # enemy.color("Red")
    enemy.shape("invader.gif")
    enemy.penup()
    enemy.speed(0)
    x = random.randint(-200, 200)
    y = random.randint(100, 250)
    enemy.setposition(x, y)

enemyspeed = 5
```

```
# Δημιουργία της σφαίρας του παίκτη
bullet = turtle.Turtle()
bullet.color("white")
bullet.shape("triangle")
bullet.penup()
bullet.speed(0)
bullet.setheading(90)
bullet.shapesize(0.5, 0.5)
bullet.hideturtle()

bulletspeed = 30

# Καθορισμός κατάστασης σφαίρας
# ready – έτοιμη προς πυροδότηση
# fire – η σφαίρα φεύγει
bulletstate = "ready"


# Κίνηση του παίκτη δεξιά κι αριστερά
def move_left():
    x = player.xcor()
    x -= playerspeed
    if x < -280:
        x = -280
    player.setx(x)
```

```
def move_right():
    x = player.xcor()
    x += playerspeed
    if x > 280:
        x = 280
    player.setx(x)



def fire_bullet():
    # Δήλωση κατάστασης της σφαίρας, ορατή σε όλο το παιχνίδι
    global bulletstate
    if bulletstate == "ready":
        bulletstate = "fire"
        # Τοποθέτηση της σφαίρας ακριβώς πάνω από το σκάφος του παίκτη
        x = player.xcor()
        y = player.ycor() + 10
        bullet.setposition(x, y)
        bullet.showturtle()



# Για τη σύγκρουση μεταξύ του εχθρού και της σφαίρας
def isCollision_enemy_bullet(t1, t2):
    distance = math.sqrt(
```

```
        math.pow(t1.xcor() - t2.xcor(), 2) + math.pow(t1.ycor() - t2.ycor(),
2)
    )
    if distance < 25:
        return True
    else:
        return False


# Συνάρτηση σύγκρουσης εχθρών και παίκτη
def isCollision_enemy_player(t1, t2):
    distance = math.sqrt(
        math.pow(t1.xcor() - t2.xcor(), 2) + math.pow(t1.ycor() - t2.ycor(),
2)
    )
    if distance < 30:
        return True
    else:
        return False


# Δημιουργία δεσμών με το πληκτρολόγιο
turtle.listen()
turtle.onkey(move_left, "Left")
turtle.onkey(move_right, "Right")
turtle.onkey(fire_bullet, "space")
```

```
# Αρχικοποίηση του παιχνιδιού
Game_Over = False
missed_enemies = 0
while True:

    for enemy in enemies:
        # Κίνηση των εχθρικών σκαφών
        x = enemy.xcor()
        x += enemyspeed
        enemy.setx(x)

        # Κίνηση των εθχρών πίσω και κάτω
        if enemy.xcor() > 270:
            # Κίνηση όλων των εχθρικών σκαφών προς τα κάτω
            for e in enemies:
                y = e.ycor()
                y -= 40
                e.sety(y)
                if e.ycor() < -285 and Game_Over == False:
                    e.hideturtle()
                    missed_enemies += 1
                    if missed_enemies == 5:
                        Game_Over = True
                    x = random.randint(-200, 200)
                    y = random.randint(100, 250)
                    e.setposition(x, y)
```

```
            e.showturtle()
    # Αλλαγή κατεύθυνσης των εχθρικών σκαφών
    enemyspeed *= -1

if enemy.xcor() < -270:
    # Move all enemies down
    for e in enemies:
        y = e.ycor()
        y -= 40
        e.sety(y)
        if e.ycor() < -285 and Game_Over == False:
            e.hideturtle()
            missed_enemies += 1
            if missed_enemies == 5:
                Game_Over = True
            x = random.randint(-200, 200)
            y = random.randint(100, 250)
            e.setposition(x, y)
            e.showturtle()
    # Αλλαγή κατεύθυνσης των εχθρικών σκαφών
    enemyspeed *= -1

# Έλεγχος αν η σφαίρα «πέτυχε» το εχθρικό σκάφος
if isCollision_enemy_bullet(bullet, enemy):
    # Reset the bullet
    bullet.hideturtle()
```

```
            bulletstate = "ready"
            bullet.setposition(0, -400)
            # Reset the enemy
            x = random.randint(-200, 200)
            y = random.randint(100, 250)
            enemy.setposition(x, y)
            enemyspeed += 0.5
            # ανανέωση του σκορ
            score += 10
            scorestring = "Score: %s" % score
            score_pen.clear()
            score_pen.write(
                scorestring, False, align="left", font=("Arial", 14, "normal")
            )
        # Έλεγχος σύγκρουσης του σκάφους του παίκτη με εχθρικό σκάφος
        if isCollision_enemy_player(player, enemy):
            Game_Over = True
        if Game_Over == True:
            player.hideturtle()
            bullet.hideturtle()
            for e in enemies:
                e.hideturtle()
            window.bgpic("end.gif")
            break


    # Κίνηση της σφαίρας
```

```
    if bulletstate == "fire":
        y = bullet.ycor()
        y += bulletspeed
        bullet.sety(y)

    # Έλεγχος για το αν η σφαίρα έχει φτάσει στο πάνω μέρος της οθόνης
    if bullet.ycor() > 275:
        bullet.hideturtle()
        bulletstate = "ready"

turtle.donε()
```

20.2.1 Συνάρτηση turtle.getshapes()

Αυτή η συνάρτηση χρησιμοποιείται για να επιστρέφει μια λίστα με τα ονόματα των σχημάτων της χελώνας, αυτή τη στιγμή. Μπορούμε να προσθέτουμε στοιχεία σ' αυτή τη λίστα, με τη συνάρτηση ***turtle.register_shape()***

με σύνταξη: ***turtle.register_shape(name, shape=None)***

turtle.getshapes()

ενώ η ***turtle.getshapes()***

έχει σύνταξη: ***turtle.getshapes()*** και δεν χρειάζεται παραμέτρους

Ας δούμε ένα παράδειγμα χρήσης της ***getshapes():***

```
# import πακέτου
import turtle
print(turtle.getshapes())
```

Έξοδος :

['arrow', 'blank', 'circle', 'classic', 'square', 'triangle', 'turtle']

Παράδειγμα 2 :

```
# import πακέτου
import turtle
shapes = turtle.getshapes()


# Βάζουμε την ταχύτητα στο πιο αργό
turtle.speed(1)

# Χρησιμοποιούμε όλα τα σχήματα
for i in range(len(shapes)):

  # Σχήμα σε χρήση
  turtle.shape(shapes[i])

  # Κίνηση χελώνας
  turtle.forward(100)
  turtle.right(51.42)
  turtle.stamp()
```

Έξοδος :

20.2.2 Συνάρτηση turtle.seth()

Αυτή η μέθοδος χρησιμοποιείται για να μας δώσει την κατεύθυνση της χελώνας σε γωνία (to_angle). Η μέθοδος παίρνει μια παράμετρο, τη γωνία.

Μπορεί να χρησιμοποιηθεί και η setheading() και πάλι με όρισμα μόνο τη γωνία.

Σύνταξη : ***turtle.seth(to_angle) or turtle.setheading(to_angle)***

Παράμετρος: ***to_angle***: αριθμός (ακέραιος ή δεκαδικός)

Εδώ έχουμε κάποιες συνηθισμένες κατευθύνσεις σε μοίρες:

0 – east

90 – north

180 – west

270 – south

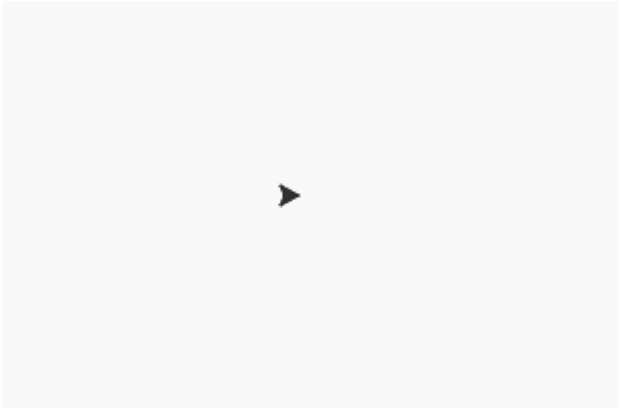
Παράδειγμα με εφαρμογή της συνάρτησης:

```
# import package
import turtle

# Θέτουμε την κατεύθυνση στο 0 (= Ανατολικά)
# Γωνία με χρήση της seth
turtle.seth(0)

# Κίνηση χελώνας
turtle.forward(80)
turtle.write("East")

# Πίσω στο αρχικό σημείο (0,0)
turtle.home()

# Θέτουμε την κατεύθυνση στις 90 μοίρες
# Χρησιμοποιώντας την sethading
turtle.setheading(90)

# Κίνηση της χελώνας
```

```
turtle.forward(80)
turtle.write("North")

# Πίσω στο home (αρχικό σημείο)
turtle.home()

# Θέτουμε την κατεύθυνση στις 180 μοίρες
# χρησιμοποιώντας την seth
turtle.seth(180)

# Κίνηση της χελώνας
turtle.forward(80)
turtle.write("West",align="right")

# Πίσω στο home
turtle.home()

# Θέτουμε την κατεύθυνση στις 270 μοίρες
# χρησιμοποιώντας την setheading
turtle.setheading(270)

# Κίνηση
turtle.forward(80)
turtle.write("South")

# Κρύβουμε τη χελώνα
```

turtle.ht()

Έξοδος :



ΚΑΛΟ ΚΑΛΟΚΑΙΡΙ ΜΕ PYTHON!