



Lecția 9 – Python Games

Trainer: Hrișcă Miruna

Aplicații folosite + alternative ▾

01

PyCharm

[Link - windows](#)

[Link - Apple](#)

02

One Compiler

[Link](#)

03

VS Code

[Link](#)

04

Online GDB

[Link](#)

Biblioteca Turtle

1. `hideturtle()`

- Ascunde țestoasa, ca să nu o mai vedem cum merge.
- Astfel, rămân doar desenele pe ecran (linii, forme, etc.), fără „personajul” care le face.
👉 Gândește-te că e ca un **magician invizibil** care desenează fără să-l vezi. 🎩✨



2. `tracer(False)`

- Oprește animația pas cu pas.
- Dacă nu folosim asta, vedem țestoasa desenând încet (ca atunci când desenezi tu cu creionul).
- Dacă folosim `tracer(False)`, tot desenul apare **instant** pe ecran.
👉 E ca diferența dintre a colora încet cu creionul sau a lipi direct un autocolant colorat. 🎨



3. `listen()`

- Pune programul pe modul „ascult”.
- Adică, acum calculatorul așteaptă să vadă ce taste apeși.
👉 Gândește-te că e ca un **paznic al tastelor**, mereu atent să vadă ce faci.



4. `onkey(acțiune, tastă)`

- Leagă o tastă de o acțiune.



Biblioteca FreeGames



Modulul **vector**

Unul dintre cele mai importante instrumente din FreeGames este **vectorul**.

- Un **vector** este o pereche de numere (x , y) care ne spune **unde este un obiect** pe ecran și **cum se mișcă**.
- În jocuri, vectorii sunt folosiți pentru **poziții și mișcări în planul bidimensional** (adică pe o tablă cu X și Y).



👉 Imaginează-ți că ai o săgeată 🏹:

- Capătul săgeții arată **direcția mișcării**.
- Lungimea săgeții arată **cât de repede se mișcă**.



Biblioteca Random



Funcția `choice(seq)`



- Imaginează că ai o **cutie cu bomboane de diferite culori**: roșu, galben, verde și albastru.
- Dacă vrei să alegi o **bomboană la întâmplare**, fără să te uiți, folosești `choice()`.
- Calculatorul alege o **culoare la întâmplare** din listă.



Funcția `random()`

- Aceasta generează un **număr aleator între 0 și 1**.



Dicționare

Ce este un dicționar?

- Un **dicționar** în Python este ca un **dicționar adevărat**, unde ai **cuvinte** și **definiții**.
- În Python, însă, cuvintele se numesc **chei**, iar definițiile se numesc **valori**.
- Practic, un dicționar **leagă o cheie de valoarea ei**, ca să poți găsi rapid informația dorită.

Exemplu:

```
persoana = {'nume': 'Ana', 'varsta': 25, 'oras': 'București'}
```

Aici:

- Cheile sunt: 'nume', 'varsta', 'oras'
- Valorile sunt: 'Ana', 25, 'București'

Cum îl putem folosi:

```
print(person['nume'])    # Va afișa: Ana
print(person['varsta'])  # Va afișa: 25
print(person['oras'])    # Va afișa: București
```



Despre jocul Pong

Astăzi vom învăța cum să facem un joc simplu, dar distractiv: **Pong**.
Este un joc în doi jucători unde fiecare controlează o „paletă” și trebuie să lovească mingea înapoi, fără să o lase să treacă.

- ❖ Jucătorul 1 folosește tastele **W** (sus) și **S** (jos).
- ❖ Jucătorul 2 folosește tastele **I** (sus) și **K** (jos).
Primul care ratează mingea pierde.



Programarea jocului

```
from random import choice, random
from turtle import *
from freegames import vector
```



- ❖ Importăm funcțiile și bibliotecile necesare.
- ❖ * înseamnă că putem folosi toate funcțiile din `turtle` direct.

```
def value():
    """Randomly generate value between (-5, -3) or (3, 5)."""
    return (3 + random() * 2) * choice([1, -1])
```

Ce face:



- Creează o **valoare aleatoare** între **-5 și -3** sau între **3 și 5**.
- Este folosită pentru a stabili **viteza inițială** a mingii (în ce direcție și cât de repede pornește).

Programarea jocului

Cum funcționează:

1. `random()` → generează un număr între 0 și 1 (ex: 0.35).
2. `random() * 2` → între 0 și 2 (ex: 0.7).
3. `3 + ...` → între 3 și 5 (ex: 3.7).
4. `choice([1, -1])` → alege aleator între 1 și -1.
 - Dacă alege 1, valoarea e pozitivă → mingea merge la dreapta/jos.
 - Dacă alege -1, valoarea e negativă → mingea merge la stânga/sus.

```
ball = vector(0, 0)
aim = vector(value(), value())
state = {1: 0, 2: 0}
```

Ce fac:

- `ball = vector(0, 0)` → mingea începe **din centru** (0,0).
- `aim = vector(value(), value())` → direcția mingii este stabilită la întâmplare, cu ajutorul funcției `value()`.
- `state = {1: 0, 2: 0}` → un **dicționar** care reține poziția paletelor:
 - 1 = paleta jucătorului 1 (stânga), la început în poziția 0.
 - 2 = paleta jucătorului 2 (dreapta), tot la poziția 0.

Programarea jocului

```
def move(player, change):  
    """Move player position by change."""  
    state[player] += change
```



Ce face:

- Mută paleta unui jucător **în sus sau în jos**.

Cum:

- `player` → spune ce jucător mutăm (1 sau 2).
- `change` → cât de mult să mutăm paleta (+20 pentru sus, -20 pentru jos).
- `state[player] += change` → adună schimbarea la poziția actuală.

👉 Exemplu: Dacă `state[1] = 0` și apelăm `move(1, 20)`, atunci `state[1]` devine 20.
Paleta jucătorului 1 urcă.



Programarea jocului

```
def rectangle(x, y, width, height):  
    """Draw rectangle at (x, y) with given width and height."""  
    up()  
    goto(x, y)  
    down()  
    begin_fill()  
    for count in range(2):  
        forward(width)  
        left(90)  
        forward(height)  
        left(90)  
    end_fill()
```

Programarea jocului

Ce face:

- Desenează un **dreptunghi plin** la poziția (x, y) .
- Este folosit pentru a desena **paletele** celor doi jucători.

Cum:

1. `up()` → ridică „creionul” (nu desenează).
2. `goto(x, y)` → mută cursorul la poziția (x, y) .
3. `down()` → pune „creionul jos” și începe desenul.
4. `begin_fill() ... end_fill()` → umple dreptunghiul cu culoare.
5. Bucla `for count in range(2)` → trasează laturile dreptunghiului:
 - `forward(width)` → lățime.
 - `left(90)` → întoarce cursorul.
 - `forward(height)` → înălțime.
 - și tot așa.

Programarea jocului

```
def draw():  
    """Draw game and move pong ball."""  
    clear()  
    rectangle(-200, state[1], 10, 50)  
    rectangle(190, state[2], 10, 50)
```



Ce face:

- Este **motorul jocului** – desenează totul și mișcă mingea.

Pașii:



1. `clear()` → șterge tot ce era desenat înainte.
2. `rectangle(-200, state[1], 10, 50)`
 - Aici desenăm **paleta jucătorului 1**.
 - Funcția `rectangle(x, y, width, height)` știe să deseneze un dreptunghi plin.
 - `x = -200` → înseamnă că paleta este **în partea stângă** a ecranului.
 - `y = state[1]` → poziția verticală a paletei depinde de **starea jucătorului 1** (cât a mutat-o cu tastele).
 - `width = 10` → paleta are **10 pixeli lățime**.
 - `height = 50` → paleta are **50 pixeli înălțime**.

Programarea jocului

3. `rectangle(190, state[2], 10, 50)`

- Același lucru, dar pentru **paleta jucătorului 2**.
- `x = 190` → paleta este **în partea dreaptă** a ecranului.
- `y = state[2]` → poziția verticală depinde de mișcările jucătorului 2.
- `width = 10, height = 50` → are aceeași dimensiune ca paleta jucătorului 1.



```
ball.move(aim)
x = ball.x
y = ball.y
up()
goto(x, y)
dot(10)
update()
```

- `ball.move(aim)` → mută mingea conform direcției stocate în `aim`.
- `dot(10)` → desenează mingea ca un cerc mic.
- `update()` → actualizează ecranul.

Programarea jocului

```
if y < -200 or y > 200:  
    aim.y = -aim.y
```

- ❖ Dacă mingea depășește sus ($y > 200$) sau jos ($y < -200$), atunci își inversează direcția verticală (y).

```
if x < -185:  
    low = state[1]  
    high = state[1] + 50  
    if low <= y <= high:  
        aim.x = -aim.x  
    else:  
        return
```

- Dacă mingea ajunge la stânga ($x < -185$):
 - Verificăm dacă y (înălțimea mingii) este între low și $high$ (zona paletelor).
 - Dacă da → mingea sare înapoi ($aim.x = -aim.x$).
 - Dacă nu → jucătorul 1 a ratat → jocul se oprește ($return$).

Programarea jocului



```
if x > 185:  
    low = state[2]  
    high = state[2] + 50  
    if low <= y <= high:  
        aim.x = -aim.x  
    else:  
        return
```



- Aceeași logică, dar pentru paleta jucătorului 2.

```
ontimer(draw, 50)
```



- Spune jocului: „reapelează funcția `draw()` după 50 milisecunde”.
- Astfel, animația se repetă mereu.

Programarea jocului



```
setup(420, 420, 370, 0)
hideturtle()
tracer(False)
listen()
onkey(lambda: move(1, 40), 'w')
onkey(lambda: move(1, -40), 's')
onkey(lambda: move(2, 40), 'i')
onkey(lambda: move(2, -40), 'k')
draw()
done()
```

Ce face:



- `setup(420, 420, 370, 0)` → creează fereastra de joc.
- `hideturtle()` → ascunde cursorul „țestoasă”.
- `tracer(False)` → face ca desenarea să fie instantanee.
- `listen()` → pregătește programul să asculte tastele.
- `onkey(...)` → leagă tastele la mișcările paletelor.
- `draw()` → pornește desenarea inițială.
- `done()` → menține fereastra deschisă.
- `lambda` este o **funcție mică, fără nume** → Fără `lambda`, programul ar face mișcarea **imediat**, nu când vrem noi.