Lecția 7 - 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

Să ne amintim!

👉 Ce este o bibliotecă?

- În Python, o **bibliotecă** = un set de funcții gata făcute, pe care le putem folosi în loc să scriem noi totul de la zero.
- **Exemple:**
  - `turtle` → pentru a desena.
  - `freegames` → pentru a face jocuri mai ușor.

👉 Ce este o funcție?

- O funcție este ca o rețetă. Îi dai un nume și instrucțiuni, și apoi o poți folosi de câte ori vrei.
- **Exemplu:** `print("Salut!")` → folosește funcția `print`.

Să ne amintim!

👉 Ce este o listă?

- O **listă** este ca o **cutie cu compartimente numerotate** unde putem păstra diferite lucruri în ordine.
- Sintaxa de bază:

main.py

```
lista_goala = []
numere = [1, 2, 3, 4, 5]
culori = ['roșu', 'albastru', 'verde']
mixt = [1, 'text', 3.14, True]
```

👉 Ce este un dicționar?

- Un **dicționar** este ca un **catalog** sau o **enciclopedie** - pentru fiecare **cheie** (cuvânt) avem o **valoare** (definiție).
- Sintaxa de bază:

main.py

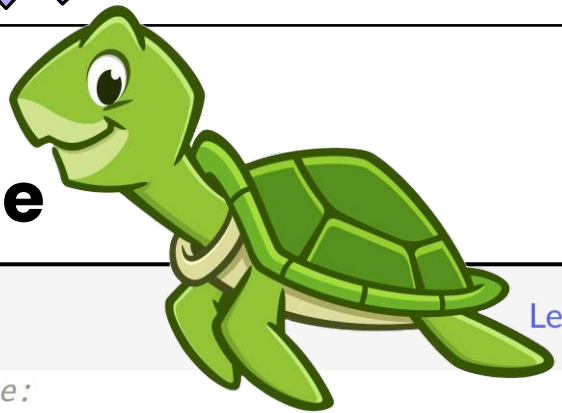
```
dictionar_gol = {}
persoana = {'nume':'Ana','varsta':12,'culoare_favorita':'albastru'}
```

Să ne amintim!

👉 **Cum funcționează evenimentele cu mouse-ul?**

- În Python putem spune: „când cineva face click, rulează această funcție”.
- Asta se face cu: `onscreenclick(tap)`.
- Adică → la fiecare click, se apelează funcția `tap`.

Funcții Turtle



main.py + Lecția 7 - Python Games

```
#Funcții importante:

setup(width, height, x, y) #creează fereastra de desen
hideturtle() #ascunde broasca țestoasă (vedem doar desenul)
tracer(False) #desenează instant (fără animație)
onscreenclick(funcție) #apelează o funcție când facem clic pe ecran
up() #ridică creionul (mută fără a desena)
goto(x, y) #merge la coordonatele specificate
dot(mărime, culoare) #desenează un cerc plin
update() #actualizează ecranul
bgcolor(culoare) #setează culoarea de fundal
```

INSTRUCȚIUNI DE CONTROL

INSTRUCŢIUNE ALTERNATIVĂ

- if
- if...else
- if...elif...else

INSTRUCŢIUNE REPETITIVĂ

- while
- for
- continue
- break
- pass

Instrucțiunea FOR

- **FOR** este ca un robot care repetă acțiuni! Îi spui "Fă asta de X ori" și el o face perfect!
- Imaginează-ți că îi spui prietenului tău: "Bate la ușă de 5 ori!" - el va bate: TOC, TOC, TOC, TOC, TOC!

Structura de bază FOR:

```
for variabilă in secvență:
    # bloc de cod care se execută de fiecare dată
```

Explicații:

- `variabilă`: primește pe rând fiecare element din secvență.
- `secvență`: poate fi o listă, un tuplu, un șir de caractere, un `range()` sau orice obiect iterabil.
- **bloc de cod**: codul indentat după `for` se execută pentru fiecare element.

1. range(stop) - De la 0 la stop-1:

- **IMPORTANT: Python numără de la 0, nu de la 1!!!**

main.py

```
1  print("Numărătoarea pentru decolare:")
2  for secunda in range(5):
3      print(f"{secunda}... 🚀")
4  print("DECOLARE! 🎆")
```

Output:

```
Numărătoarea pentru decolare:
0... 🚀
1... 🚀
2... 🚀
3... 🚀
4... 🚀
DECOLARE! 🎆
```

2. range(start, stop)
- De la start la stop-1:

- **IMPORTANT: Python numără de la 0, nu de la 1!!!**

main.py

```
1 for exercitiu in range(3, 8):
2     print("Am făcut exercitiul", exercitiu)
```

Output:

```
Am făcut exercitiul 3
Am făcut exercitiul 4
Am făcut exercitiul 5
Am făcut exercitiul 6
Am făcut exercitiul 7
```

3. range(start, stop, step)
- Cu pași -

main.py

```
1  print("Numărăm din 2 în 2:")
2  for număr in range(0, 11, 2):  # 0, 2, 4, 6, 8, 10
3      print(număr)
```

Output:

```
Numărăm din 2 în 2:
0
2
4
6
8
10
```

main.py

```
1  print("Numărătoarea inversă:")
2  for număr in range(10, 0, -1):  # de la 10 la 1
3      print(f"{număr}... ⏰")
4  print("ZERO! Gata! ⚡")
```

Output:

```
Numărătoarea inversă:
10... ⏰
9... ⏰
8... ⏰
7... ⏰
6... ⏰
5... ⏰
4... ⏰
3... ⏰
2... ⏰
1... ⏰
ZERO! Gata! ⚡
```

Despre jocul Connect Four

Connect Four este un joc pentru doi jucători:

- **Jucătorul 1:** folosește discuri galbene
- **Jucătorul 2:** folosește discuri roșii
- **Obiectiv:** primul care conectează 4 discuri (vertical, orizontal sau diagonal) câștigă!

Programarea jocului

Pasul 1: Importurile

- Importăm toate funcțiile din biblioteca Turtle
- Importăm funcția `line` din FreeGames

```
from turtle import *
from freegames import line
```

Pasul 2: Variabilele globale

- `turns` este un dicționar care spune cine urmează:
  - dacă acum e rândul **roșului** → după aceea vine **galbenul**;
  - dacă acum e rândul **galbenului** → după aceea vine **roșul**.
- `state` este tot un dicționar care ține minte:
  - `player`: cine este jucătorul curent (începe galbenul);
  - `rows`: o listă cu 8 zerouri → fiecare element reprezintă o coloană și arată câte discuri sunt puse acolo.

```
turns = {'red': 'yellow', 'yellow': 'red'}
state = {'player': 'yellow', 'rows': [0] * 8}
```

Programarea jocului

Pasul 3: **Funcția grid() - Desenarea grilei**

- Colorează fundalul în **albastru deschis**.
- **Primul `for`**: desenează linii verticale pentru coloane la fiecare 50 de pixeli.
- **Al doilea `for` dublu**: desenează cercuri albe (golurile unde intră piesele).
- La final: `update()` → afișează totul instant, fără animație lentă.

```
def grid():
    bgcolor('light blue')
    for x in range(-150, 200, 50):
        line(x, -200, x, 200)
    for x in range(-175, 200, 50):
        for y in range(-175, 200, 50):
            up()
            goto(x, y)
            dot(40, 'white')
    update()
```

Programarea jocului

Pasul 4: Funcția tap() - Gestionarea clicurilor

Aceasta este partea „interactivă” a jocului:

- Primește coordonatele (`x`, `y`) unde s-a dat click.
- Află **jucătorul curent** (galben sau roșu).
- Calculează **coloana** unde s-a dat click:
  - `(x + 200) // 50` transformă poziția pe ecran într-un număr de coloană (0–7).
- Verifică câte discuri sunt deja în acea coloană (`count = rows[row]`).
- Calculează poziția exactă unde trebuie desenat cercul:
  - `x` pentru coloană,
  - `y` pentru cât de sus să fie cercul (depinde de `count`).
- Desenează un cerc (`dot(40, player)`) cu culoarea jucătorului.
- Actualizează lista `rows` (mai adaugă un disc acolo).
- Schimbă jucătorul (`state['player'] = turns[player]`).

```
def tap(x, y):
    player = state['player']
    rows = state['rows']
    row = int((x + 200) // 50)
    count = rows[row]
    x = ((x + 200) // 50) * 50 -
200 + 25
    y = count * 50 - 200 + 25
    up()
    goto(x, y)
    dot(40, player)
    update()
    rows[row] = count + 1
    state['player'] =
turns[player]
```

Programarea jocului

Ce înseamnă `[0] * 8`?

- Creează o listă cu 8 elemente, toate fiind 0
- Rezultatul: `[0, 0, 0, 0, 0, 0, 0, 0]`

Fiecare poziție din listă reprezintă:

- `rows[0]` = câte discuri sunt în coloana 0
- `rows[1]` = câte discuri sunt în coloana 1
- `rows[2]` = câte discuri sunt în coloana 2
- ...și tot așa până la `rows[7]`

```
'rows': [0] * 8
```

Programarea jocului

La început: Toate coloanele sunt goale

```
state = {'player': 'yellow', 'rows': [0, 0, 0, 0, 0, 0, 0, 0]}
```

După câteva mutări:

- Coloana 3 are **2 discuri** (unul galben jos, unul galben deasupra)
- Coloana 4 are **1 disc** (roșu)
- Toate celelalte coloane sunt **goale** (0 discuri)

```
# Jucătorul galben pune în coloana 3
# Jucătorul roșu pune în coloana 4
# Jucătorul galben pune din nou în coloana 3

state = {'player': 'red', 'rows': [0, 0, 0, 2, 1, 0, 0, 0]}
#                                  ^  ^  ^  ^  ^  ^  ^  ^
#                                  0  1  2  3  4  5  6  7
```

Programarea jocului

De ce +200?

- Grila noastră începe la x = -200 și se termină la x = +150
- Adunând 200, mutăm totul în teritoriul pozitiv pentru calcule mai ușoare
- Înainte: -200, -150, -100, -50, 0, 50, 100, 150
- După +200: 0, 50, 100, 150, 200, 250, 300, 350

```
x = ((x + 200) // 50) * 50 - 200 + 25
```

Ce face `//`?

- `//` este **împărțirea întreagă** (fără zecimale)
- Împărțim la 50 pentru că fiecare coloană are lățimea de 50 de pixeli

```
75//50 = 1 # Suntem în coloana 1
125//50 = 2 # Suntem în coloana 2
225//50 = 4 # Suntem în coloana 4
```

De ce înmulțim din nou cu 50?

- Pentru a ajunge la **marginea stângă** a coloanei
- Transformă numărul coloanei înapoi în coordonate

```
# Dacă suntem în coloana 2:
2 * 50 = 100  # x = 100 (marginea stângă a coloanei 2)
```

Programarea jocului

De ce scădem 200?

- Anulăm adunarea de la început
- Revenim în sistemul original de coordonate

```
100 - 200 = -100  # Poziția reală a marginii stângi
```

De ce +25?

- Pentru a **centra** discul în coloană
- Fiecare coloană are 50 pixeli, deci centrul e la 25 pixeli de la margine

```
-100 + 25 = -75  # Centrul coloanei!
```

Programarea jocului

- `count` = câte discuri sunt deja în coloană
- `count * 50` = înălțimea la care trebuie să pun discul
- `- 200` = ajustare pentru sistemul de coordonate (grila începe la y = -200)
- `+ 25` = centrarea discului pe înălțime

```
y = count * 50 - 200 + 25
```

Programarea jocului

Pasul 5: **Inițializarea jocului**

- Creează fereastra de 420x420 pixeli
- Ascunde broasca țestoasă
- Dezactivează animația
- Desenează grila inițială
- Activează detectarea clicurilor
- Pornește bucla principală

```
setup(420, 420, 370, 0)
hideturtle()
tracer(False)
grid()
onscreenclick(tap)
done()
```