CAPITOLUL IV AMENAJĂRILE INTERIOARE

În acest capitol vom învăța:

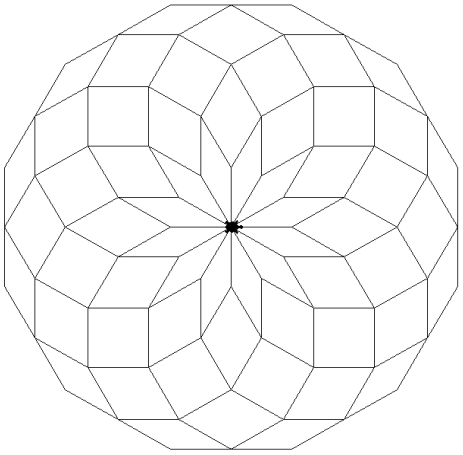
- Să realizăm figuri atractive prin utilizarea culorilor
- Să aplicăm instrucțiunile for, while și if în rezolvarea unor probleme concrete
- Să utilizăm mecanismul recursivității pentru scrierea unor soluții mai compacte

4.1 Ouă încondeiate

Ema a decis să amenajeze o camera în care să existe din mai multe țări câte o amintire. Din România a decis să ia ouă încondeiate. Iată câteva modele care i-au plăcut.

Vom încerca în continuare să realizăm fiecare model.

4.1.1 Dodecagoane



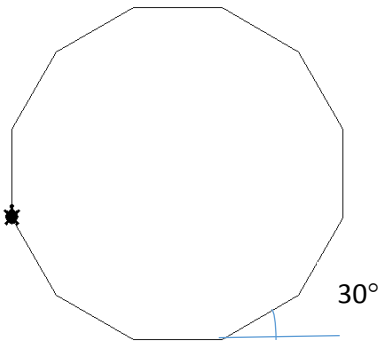
Acest model Ema l-a obținut din mai multe dodecagoane.

PUȚINĂ MATEMATICĂ

Teorema Suma unghiurilor unui poligon cu n laturi este (n-2)*180°.

Făcând calculele conform teoremei prezentate anterior obținem 10*180/12=150 grade – măsura unghiului. Soluția pentru desenarea unui dodecagon este:

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
ema.lt(90)
for i in range(12):
    ema.forward(100)
    ema.rt(30)
```



Pentru desenarea figurii vom repeta de 12 ori secvența care desenează un singur dodecagon. După fiecare desenare a unui dodecagon, broscuța se rotește spre dreapta cu 30 de grade.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
for i in range(12):
    for j in range(12):
        ema.forward(80)
        ema.rt(30)
    ema.rt(30)
```

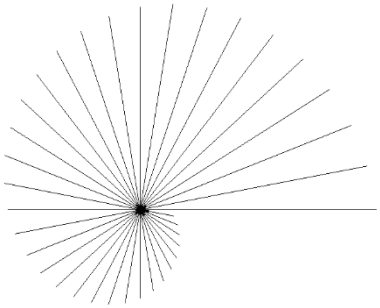
Încercați singuri

Definiți o funcție dodecagon cu un parametru latura și apelați funcția astfel încât să obțineți figura.

4.1.2 Spirala

Un alt model care i-a plăcut Emei este următorul:

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
for i in range(50,420,10):
    ema.fd(i)
    ema.pu()
    ema.bk(i)
    ema.pd()
    ema.rt(10)
```



PUȚINĂ CULOARE

FUNCȚII REFERITOARE LA SETAREA CULORILOR

FUNCȚIA	EFECT
Funcția colormode are două forme: 1) colormode(nr) 2) colormode()	1) Setează modul cum va fi exprimată combinația RGB. Nr poate să ia două valori: 1 sau 255. În cazul în care nr este 1 concentrația de culoare este exprimată prin numere cuprinse între 0 și 1. Apelul funcției color va fi de forma ema.color(0.5,0.5,0.3). Dacă nr este 255, concentrația de culoare este

	exprimată prin numere cuprinse între 0 și 255. În acest caz apelul funcției color va fi de forma ema.color(70,100,200). 2) Funcția colormode() returnează valoarea lui nr.
Funcția color are mai multe forme: 1) color(numeculoare) sau color(r,g,b) 2) color(numeculoare1, numeculoare2) sau color((r1,g1,b1), (r2,g2,b2)) 3) color()	1) Setează aceeași culoare pentru desenare și umplere. Exemple: import turtle ema=turtle.Turtle() ema.color("red") ema.color("#ff0000") 2) Setează culori diferite pentru desenare și umplere. Prima culoare este culoarea folosită pentru desenare și a doua culoare este culoarea folosită pentru umplere. Exemple: import turtle ema=turtle.Turtle() ema.color("red","blue") ema.color("#ff0000","#0000ff") 3) Returnează culoarea de desenare și de umplere. Returnează aceleași valori ca funcțiile pencolor() și fillcolor().
Funcția pencolor are următoarele forme: 1) pencolor(numeculoare) 2) pencolor(r,g,b) 3) pencolor((r,g,b)) 4) pencolor()	Primele trei forme ale funcției setează culoarea de desenare. Ultima formă returnează culoarea de desenare. Pentru combinația r,g,b valorile depind de setările făcute de funcția colormode. Exemplu: import turtle ema=turtle.Turtle() turtle.colormode(255) ema.pencolor(30,80,30) ema.fd(100) ema.pencolor("#ff0000") ema.fd(100)
Funcția fillcolor() are următoarele forme: 1) fillcolor(numeculoare) 2) fillcolor(r,g,b) 3) fillcolor((r,g,b)) 4) fillcolor()	Primele trei forme setează culoarea de umplere. Ultima formă returnează culoarea de umplere. Pentru combinația r,g,b valorile depind de setările făcute de funcția colormode. Exemplu: import turtle ema=turtle.Turtle() ema.shape("turtle") turtle.colormode(255) ema.fillcolor(30,80,30) ema.begin_fill() ema.circle(100) ema.end_fill()

1) begin_fill 2) end_fill	După ce am setat culoarea de umplere, înainte de a umple o formă trebuie să apelăm funcția begin_fill() iar după ce am terminat de umplut forma apelăm end_fill(). Exemplu: import turtle ema=turtle.Turtle() ema.shape("turtle") turtle.colormode(255) ema.fillcolor(30,80,30) ema.begin_fill() ema.circle(100) ema.end_fill()
Funcția bgcolor are două forme: 1) bgcolor(culoare) sau 2) bgcolor()	1) Setează culoarea de fundal import turtle turtle.bgcolor("#ff0000") 2) Returnează culoarea de fundal import turtle turtle.bgcolor()

Stabilirea grosimii liniei – este realizată de funcțiile pensize și width

FUNCȚIA	EFECT
Funcția pensize are două forme: 1) pensize(nr) sau 2) pensize()	1) Setează grosimea liniei de desenare la valoarea nr, unde nr este un număr pozitiv import turtle ema=turtle.Turtle() ema.pensize(5) 2) Returnează grosimea liniei de desenare import turtle ema=turtle.Turtle() ema.pensize() 5

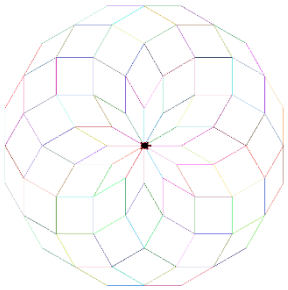
Funcția width are un efect identic cu funcția pensize.

Generarea numerelor aleatorii

Pentru a genera numere aleatorii în Python folosim funcțiile randrange și randint care au sintaxa următoare:
1) randrange(n) – generează numere aleatorii mai mici decât n.
2) randrange(a,b,pas) – generează numere aleatorii din intervalul [a,b) utilizând pasul pas.
3) randint(a,b) - generează numere întregi aleatorii din intervalul [a,b].

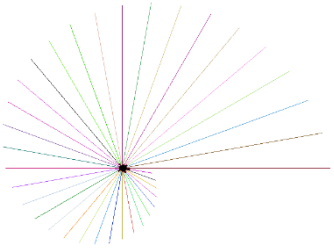
Exemplu:
```
import random
number = random.randrange(10,20,1)
print(number)
```

Primul model în culori



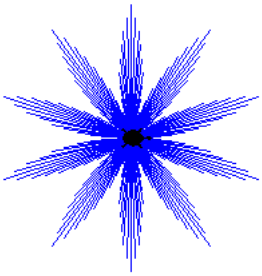
```
import turtle
import random
ema=turtle.Turtle()
ema.shape("turtle")
turtle.colormode(255)
for i in range(12):
    for j in range(12):
        ema.pencolor(random.randint(0,255),random.randint(0,255),random.randint(0,255))
        ema.forward(80)
        ema.rt(30)
    ema.rt(30)
```

Al doilea model în culori



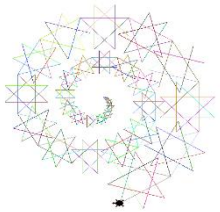
```
import turtle
import random
ema=turtle.Turtle()
ema.shape("turtle")
turtle.colormode(255)
for i in range(50,410,10):
    ema.pencolor(random.randint(0,255),random.randint(0,255),random.randint(0,255))
    ema.fd(i)
    ema.pu()
    ema.bk(i)
    ema.pd()
    ema.rt(10)
```

4.1.3 Ace de gheață



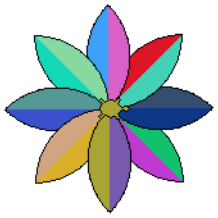
```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
turtle.colormode(255)
ema.pencolor(0,0,255)
def petala():
    for i in range(10,100,10):
        ema.fd(i)
        ema.bk(i)
        ema.rt(2)
    for i in range(100,10,-10):
        ema.fd(i)
        ema.bk(i)
        ema.rt(2)

for i in range(10):
    petala()
```

4.1.4 Steluțe



```
import turtle
import random
ema=turtle.Turtle()
ema.shape("turtle")
turtle.colormode(255)
def stea(latura):
        for i in range(8):
            ema.pencolor(random.randint(0,255),random.randint(0,255),random.randint(0,255))
            ema.fd(latura)
            ema.rt(135)

for i in range(4,124,4):
   stea(i)
   ema.fd(i)
   ema.rt(30)
```

4.1.5 Crizantema

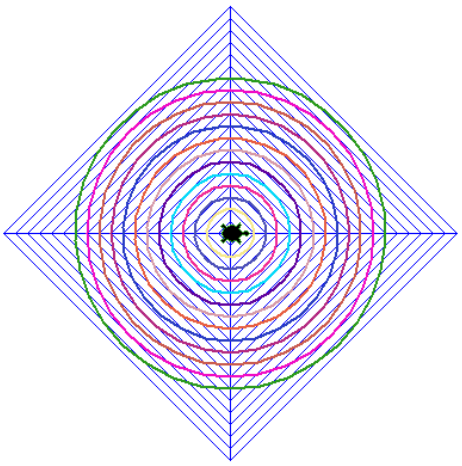


```
import turtle
import random
turtle.colormode(255)
ema=turtle.Turtle()
ema.shape("turtle")
def petala():
   for i in range(2):
      ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
      ema.begin_fill()
      for j in range(90):
         ema.fd(1)
         ema.rt(1)
      ema.end_fill()
      ema.rt(90)

def crizantema():
   for i in range(8):
      petala()
      ema.left(45)

crizantema()
```

Încercați singuri

Modificați programul astfel încât fiecare petală să aibă altă culoare.

4.1.6 Pătrate și cercuri



Descompunem problema în subprobleme.

Remarcăm că avem două figuri geometrice de bază: cercul și pătratul. Pătratul este format din patru triunghiuri dreptunghice isoscele.

Vom defini o funcție triunghi care va desena triunghiul dreptunghic și o funcție pătrat care va apela de patru ori funcția triunghi.

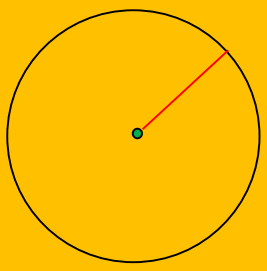
Pentru desenarea cercului avem nevoie de câteva cunoștințe noi.

PUȚINĂ MATEMATICĂ

CERCUL

Cercul este mulțimea tuturor punctelor din plan, egal depărtate de un punct fix numit centru. Distanța comună este denumită *raza cercului*.

În figura din dreapta punctul verde reprezintă centrul cercului iar linia roșie reprezintă raza cercului.



În Python, desenarea unui cerc este realizată de funcția CIRCLE.

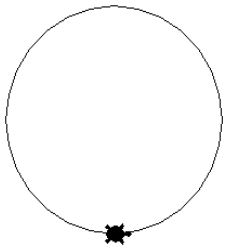
Sintaxa funcției este CIRCLE(rază, măsură, pași) unde rază reprezintă lungimea razei, măsură este măsura unghiului la centru pentru care se va desena arcul corespunzător. Dacă parametrul măsură nu există se va desena întregul cerc. Dacă parametrul măsură este pozitiv, desenarea se va face în sens invers acelor de ceasornic. Dacă parametrul măsură este negativ, desenarea arcului de cerc se va face în sensul acelor de ceasornic.

Cum un cerc este aproximat prin poligonul regulat înscris, pași reprezintă numărul de pași de utilizat. Dacă acest parametru nu este dat el va fi calculat automat. Parametrul pași poate fi utilizat pentru a desena poligoane regulate.

Exemplul 1

Desenarea unui cerc cu raza 100.

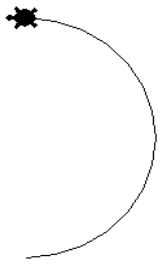
```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
ema.circle(100)
```



Exemplul 2

Desenarea unui semicerc cu raza 100, în sens invers acelor de ceasornic.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
ema.circle(100,180)
```



Exemplul 3

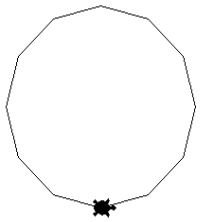
Desenarea unui semicerc cu raza 100, în sensul acelor de ceasornic.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
ema.circle(100,-180)
```

Exemplul 4

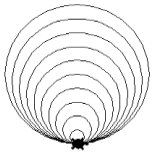
Desenarea unui poligon cu 12 laturi.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
ema.circle(100,360,12)
```



Exemplul 5
Secvența de instrucțiuni de mai jos va genera figura din dreapta.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
i=10
while i<=100:
   ema.circle(i)
   i=i+10
```



Raza cercului variază și ia pe rând valorile 10, 20, …, 100.

INSTRUCȚIUNEA WHILE

Instrucțiunea WHILE este o instrucțiune repetitivă. Ea este numită instrucțiune repetitivă condiționată anterior. Sintaxa instrucțiunii este:

```
while expresie:
        instrucțiuni
```

Efectul instrucțiunii este următorul: se evaluează expresia. Dacă expresia este adevărată se execută instrucțiunile după care se evaluează din nou expresia. Se continuă astfel până când expresia devine falsă. În momentul în care expresia devine falsă se trece la următoarea instrucțiune situată în program după instrucțiunea while.

Proiectarea oricărui program trebuie să asigure ieșirea din bucla while. Forma unei instrucțiuni while infinite este while True:

Observație importantă

În cadrul instrucțiunii while poate să apară ramura else. Instrucțiunile de pe ramura else se execută după ce s-au executat toate instrucțiunile din corpul instrucțiunii while, în cazul în care nu există o instrucțiune break în corpul instrucțiunii while.

Pentru a înțelege cum lucrează instrucțiunea while cu else executați programul de mai jos:

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
i=10
while i<=100:
   ema.circle(i)
   i=i+10
else:
   print("La iesirea din bucla valoarea lui i este ",i)
```

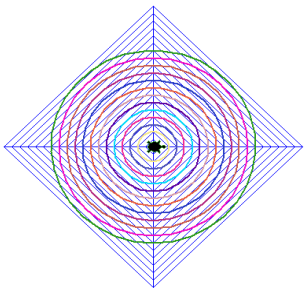
ECHIVALENȚA INSTRUCȚIUNILOR REPETITIVE

```
for contor in range(a,b,pas):
        instrucțiuni
```

⇔

```
contor=a
while contor<=b-pas:
        instructiuni
        contor=contor+pas
```

Iată soluția problemei inițiale:



```
import turtle
import random
import math
ema=turtle.Turtle()
ema.shape("turtle")
ema.pencolor("blue")

def triunghi(latura):
    ema.fd(latura)
    ema.lt(135)
    ema.fd(math.sqrt(2*latura*latura))
    ema.lt(135)
    ema.fd(latura)
    ema.lt(90)

def patrat(latura):
    for k in range(4):
        triunghi(i)
        ema.rt(90)

for i in range(10,200,10):
   patrat(i)

ema.pensize(2)
turtle.colormode(255)
for i in range(20, 140, 10):
    ema.pencolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.right(90)
    ema.pu()
    ema.fd(i)
    ema.pd()
    ema.right(270)
    ema.circle(i)
    ema.penup()
    ema.home()
```

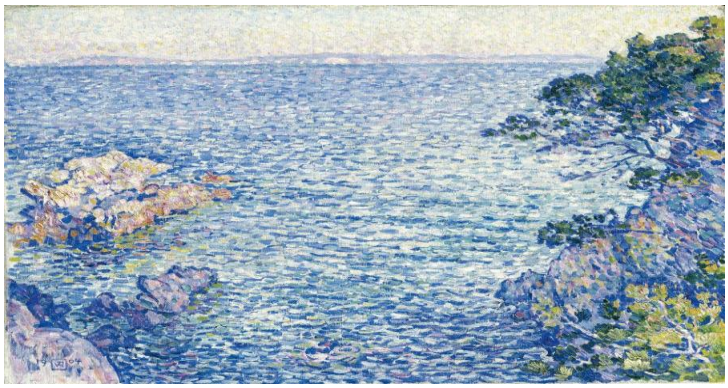
Încercați singuri

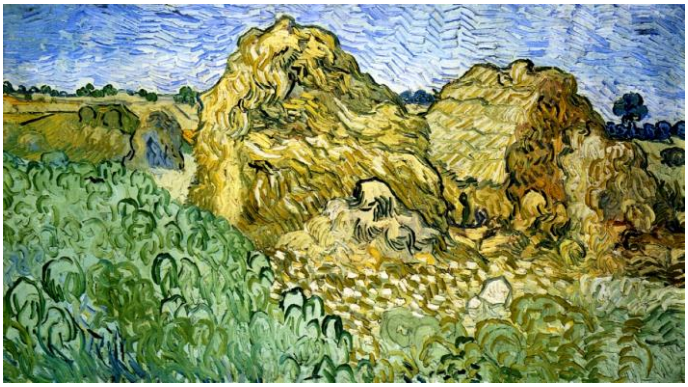
1) Realizați în Python modelul de bijuterie alăturat

și apoi încercați să îl printați la o imprimantă 3D.

2) Să pictăm ca Van Gogh

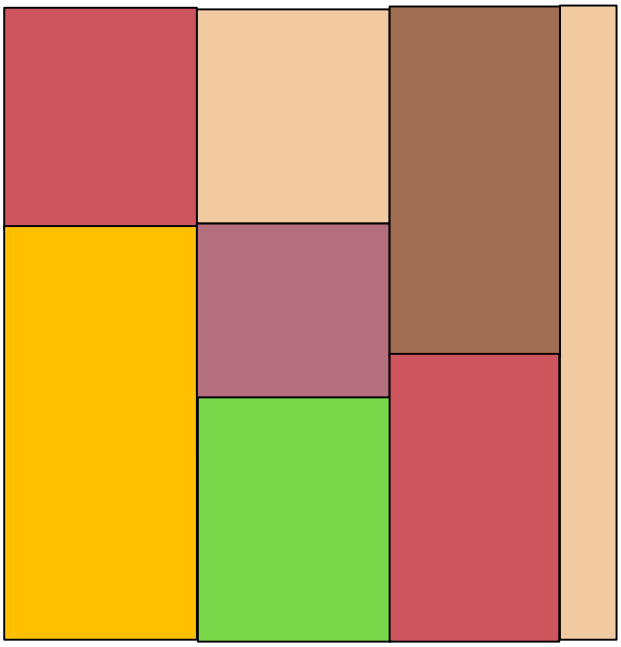
Vincent Van Gogh este astăzi considerat unul din cei mai mari pictori ai tuturor timpurilor.
El s-a impus în artă prin stilul său pointilist adică prin pictura cu puncte.
Analizați imaginile prezentate mai jos și încercați să scrieți în Python mici programe care să realizeze picturi imitând stilul lui Van Gogh.





4.2 Covorul

Ema a ales pentru covor modelul de mai jos.



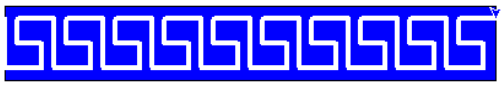
Pentru realizarea modelului remarcăm faptul că avem mai multe dreptunghiuri de diferite dimensiuni. Vom scrie o funcție **dreptunghi** pe care o vom apela cu diferiți parametri.

```
import turtle
import random
turtle.colormode(255)
ema=turtle.Turtle()
ema.shape("turtle")

def dreptunghi(lungime, latime):
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    for i in range(2):
        ema.fd(lungime)
        ema.left(90)
        ema.fd(latime)
        ema.left(90)
    ema.end_fill()


def model():
   ema.lt(90)
   dreptunghi(150,70)
   ema.fd(150)
   dreptunghi(100,70)
   ema.bk(150)
```

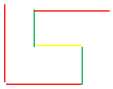
```
    ema.rt(90)
    ema.fd(70)
    ema.lt(90)
    dreptunghi(50,70)
    ema.fd(50)
    dreptunghi(100,70)
    ema.fd(100)
    dreptunghi(100,70)
    ema.bk(150)
    ema.rt(90)
    ema.fd(70)
    ema.lt(90)
    dreptunghi(125,70)
    ema.fd(125)
    dreptunghi(125,70)
    ema.bk(125)
    ema.rt(90)
    ema.fd(40)
    ema.lt(90)
    dreptunghi(250,40)

model()
```

4.3 Model din Grecia

Emei i-ar plăcea să aibă un ștergar cu un model tradițional din Grecia. Ștergarul ar trebui să arate ca în imaginea de mai jos:



Pentru realizarea modelului remarcăm că avem un dreptunghi în interiorul căruia se repetă de mai multe ori același model, modelul din dreapta.



Dimensiunile utilizate sunt, respectând culorile: 40, 20, 25.

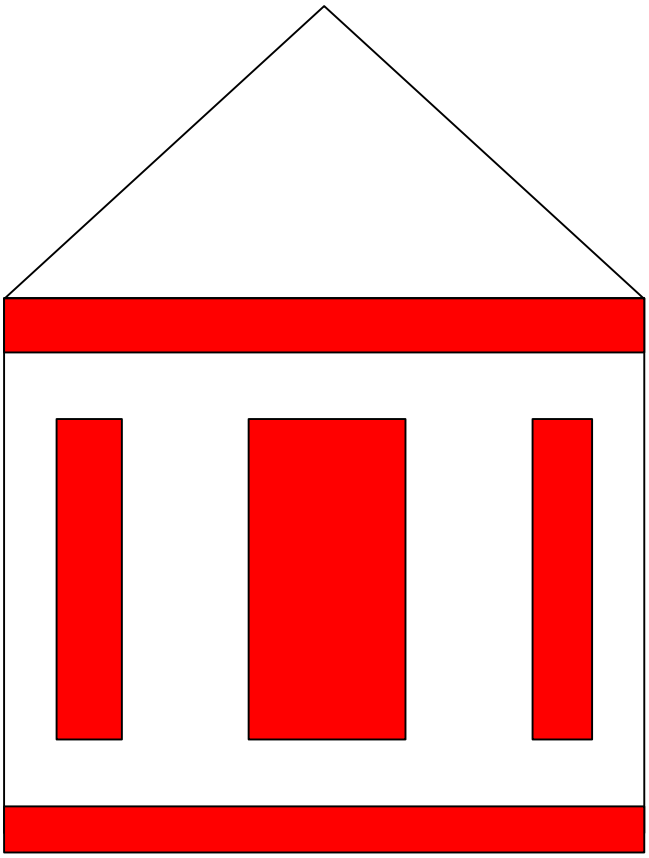
Soluție:

```
import turtle
ema=turtle.Turtle()
turtle.colormode(255)
ema.fillcolor(0, 0, 255)

def dreptunghi(lungime, latime):
    ema.begin_fill()
    for i in range(2):
        ema.fd(lungime)
        ema.lt(90)
        ema.fd(latime)
        ema.lt(90)
    ema.end_fill()

def desen():
   ema.fd(40)
   ema.lt(90)
   ema.fd(30)
   ema.lt(90)
   ema.fd(20)
   ema.lt(90)
   ema.fd(20)
   ema.rt(90)
   ema.fd(20)
   ema.rt(90)
   ema.fd(30)
   ema.rt(90)

def model():
   dreptunghi(400,60)
   ema.pencolor(255,255,255)
   ema.pensize(4)
   ema.lt(90)
   ema.pu()
   ema.fd(50)
```

```
    ema.pd()
    ema.rt(180)
    for i in range(10):
        desen()
model()
```

4.4 Model din Turcia

Emei i-ar plăcea în mod deosebit intrarea în camera cu modele de pe Pământ să se facă printr-o draperie care să conțină imaginea unui cort.



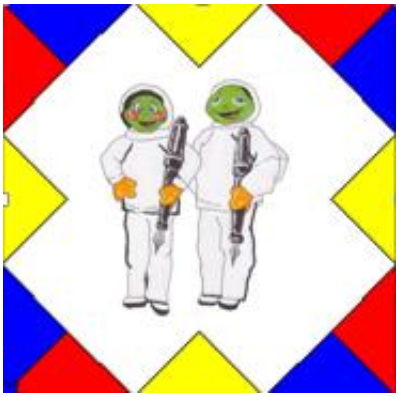
Remarcăm că avem mai multe dreptunghiuri, unul alb și celelalte colorate. Pentru a realiza acest cort vom folosi o funcție dreptunghi care are un parametru suplimentar ce ne va permite să distingem între dreptunghi alb sau colorat. Pentru dreptunghiuri folosim următoarele dimensiuni: pentru dreptunghiul mare 400 300, pentru dreptunghiurile orizontale de la bază și vârf 40 400, pentru dreptunghiurile verticale 140 40 respectiv 140 80.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def dreptunghi(lungime, latime, q):
      turtle.colormode(255)
      ema.fillcolor(255,0,0)
      if q==1:
         ema.begin_fill()
      for i in range(2):
         ema.fd(lungime)
```

```
        ema.lt(90)
        ema.fd(latime)
        ema.lt(90)
    if q==1:
        ema.end_fill()

def triunghi(latura):
    ema.rt(30)
    for i in range(3):
        ema.fd(latura)
        ema.rt(120)

def cort():
    dreptunghi(400,300,0)
    dreptunghi(400,40,1)
    ema.pu()
    ema.lt(90)
    ema.fd(260)
    ema.pd()
    ema.rt(90)
    dreptunghi(400,40,1)
    ema.pu()
    ema.lt(90)
    ema.fd(40)
    ema.pd()
    triunghi(400)
    ema.lt(30)
    ema.pu()
    ema.bk(220)
    ema.rt(90)
    ema.fd(80)
    ema.lt(90)
    ema.pd()
    dreptunghi(140,40,1)
    ema.pu()
    ema.rt(90)
    ema.fd(160)
    ema.lt(90)
    ema.pd()
    dreptunghi(140,80,1)
    ema.pu()
    ema.rt(90)
    ema.fd(120)
    ema.lt(90)
    ema.pd()
    dreptunghi(140,40,1)

cort()
```

4.5 Amintiri, dragi amintiri

Ema dorește o ramă foto cum este cea de mai jos.



Remarcăm că secvența formată din triunghiurile dreptunghice roșu, galben și albastru se repetă de patru ori. Pentru rezolvarea problemei definim o funcție triunghi cu un parametru q care va lua, la apel, una dintre valorile 1, 2 sau 3. Pentru q=1 se va seta culoarea de umplere roșu, pentru q=2 se va seta culoarea de umplere galben iar pentru q=3 se va seta culoarea de umplere albastru. Apoi definim o funcție linie care va apela de trei ori funcția triunghi. Mai jos este prezentată funcția triunghi în două variante, prima utilizând instrucțiuni if imbricate și a doua utilizând tot instrucțiuni if imbricate dar utilizand prescurtarea elif pentru else if.

Soluția 1

```
def triunghi(latura, q):
    ema.begin_fill()
    if q==1:
        ema.fillcolor("red")
    else:
        if q==2:
            ema.fillcolor("yellow")
        else:
            ema.fillcolor("blue")
    ema.fd(math.sqrt(2*latura*latura))
    ema.lt(135)
    ema.fd(latura)
    ema.lt(90)
    ema.fd(latura)
    ema.lt(135)
    ema.end_fill()
```

Soluția 2

```
def triunghi(latura, q):
    ema.begin_fill()
    if q==1:
        ema.fillcolor("red")
    elif q==2:
        ema.fillcolor("yellow")
    else:
        ema.fillcolor("blue")
    ema.fd(math.sqrt(2*latura*latura))
    ema.lt(135)
    ema.fd(latura)
    ema.lt(90)
    ema.fd(latura)
    ema.lt(135)
    ema.end_fill()
```

În cazul soluției 2 am folosit în loc de else if, elif.

```
if conditie1:
        instructiuni1
elif conditie2:
        instructiuni2
else:
        instructiuni
```

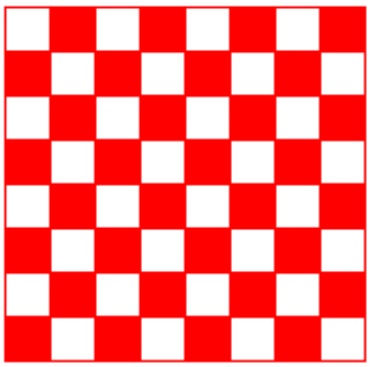
O soluție presupunând cateta triunghiului dreptunghic 50 este următoarea:

```
import turtle
import math
ema=turtle.Turtle()
ema.shape("turtle")

def triunghi(latura,q):
    ema.begin_fill()
    if q==1:
        ema.fillcolor("red")
    else:
        if q==2:
            ema.fillcolor("yellow")
        else:
            ema.fillcolor("blue")
    ema.fd(math.sqrt(2*latura*latura))
    ema.lt(135)
    ema.fd(latura)
    ema.lt(90)
    ema.fd(latura)
    ema.lt(135)
    ema.end_fill()

def linie():
    for i in range(1,4,1):
        triunghi(50,i)
        ema.fd(math.sqrt(2*50*50))

for k in range(4):
    linie()
    ema.lt(90)
```

Observație importantă: În limbajul Python nu există instrucțiunea switch existentă în limbaje precum C++ sau JavaScript.

4.6 Tabla de șah

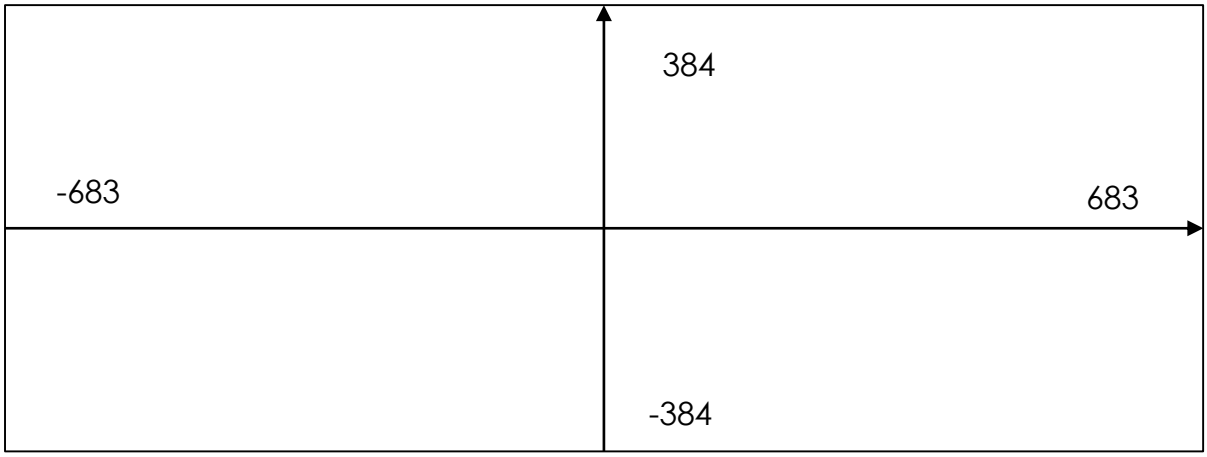
Ema își dorește o tablă de șah cum este cea din modelul de mai jos.



Pentru realizarea soluției avem nevoie de câteva cunoștințe noi legate de poziționarea pe ecran.

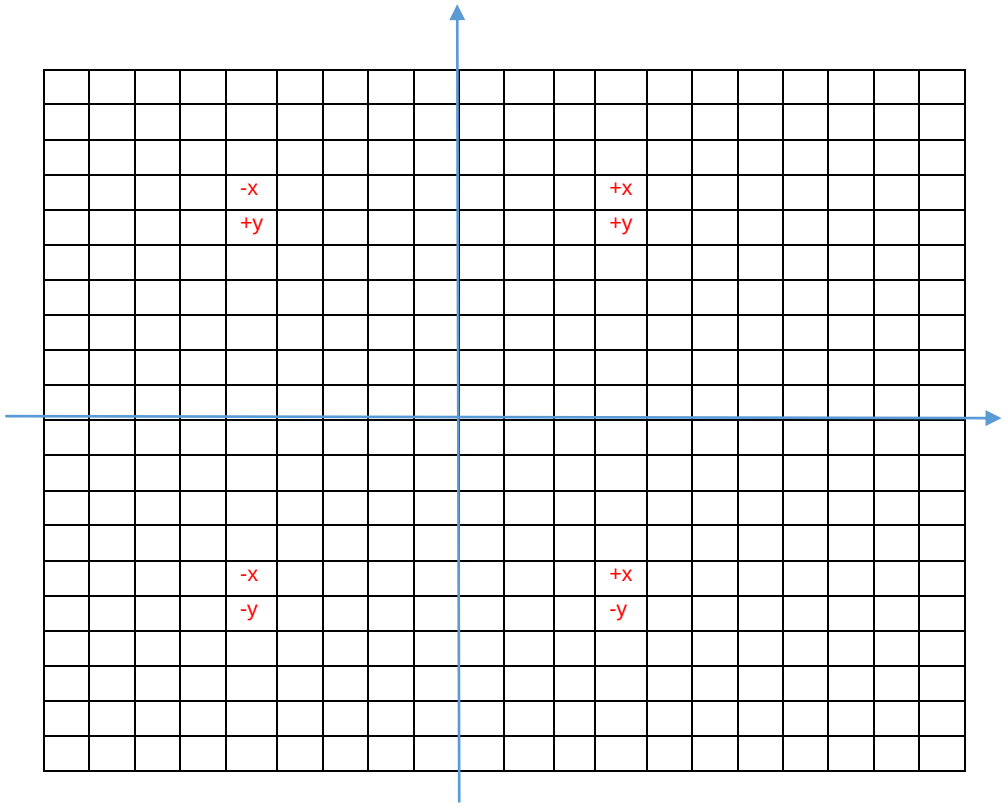
Python folosește sistemul de coordonate xOy, cunoscut de la algebră, pentru poziționarea pe ecran.

Dacă presupunem fereastra de dimensiuni 1366x768 pixeli, ea este organizată ca în imagine:



La intersecția celor două axe se găsește punctul de coordonate (0, 0).

Ne imaginăm ecranul împărțit în linii și coloane ca în imaginea de mai jos:



Coordonata x reprezintă numărul coloanelor la stânga sau la dreapta punctului de coordonate (0, 0).

Coordonata y reprezintă numărul rândurilor, în sus sau în jos, raportat la punctul de coordonate (0, 0).

Pentru a determina dimensiunile ecranului folosim funcțiile window_width() și window_height().

```
>>> import turtle
>>> turtle.window_width()
1366
>>> turtle.window_height()
705
>>>
```

Funcții pentru poziționarea pe ecran

Funcția	Efect
setx(nr)	Setează coordonata x a broscuței la valoarea nr, unde nr este un număr întreg sau real.
sety(nr)	Setează coordonata y a broscuței la valoarea nr, unde nr este un număr întreg sau real.
goto(nr1, nr2) sau setpos(nr1, nr2) sau setposition(nr1, nr2)	Mută cursorul broscuță în poziția (nr1, nr2) unde nr1, nr2 sunt numere întregi sau reale.
position()	Afișează coordonatele curente ale broscuței

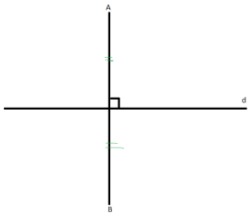
sau pos()	
setheading(nr_grade) prescurtată seth(nr_grade)	Setează capul broscuței într-o anumită direcție stabilită de nr_grade.
home()	Aduce broscuța în poziția (0,0)

Soluție:

Vom considera tabla de șah cu latura 160 iar patratul mic cu latura 20.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def patrat(LATURA,q):
    if q==1:
        ema.color(1,0,0)
        ema.begin_fill()
    for i in range(4):
        ema.fd(LATURA)
        ema.left(90)
    if q==1:
        ema.end_fill()


def tabla_de_sah():
    patrat(160,0)
    for j in range (0, 4, 1):
        for i in range (0, 4, 1):
            ema.penup()
            ema.setpos(i*40, j*40)
            ema.pendown()
            patrat(20,1)
    for j in range(1, 9, 2):
        for i in range (1,9,2):
            ema.penup()
            ema.setpos(i*20, j*20)
            ema.pendown()
            patrat(20,1)

tabla_de_sah()
```

Puțină matematică – simetria față de o dreaptă

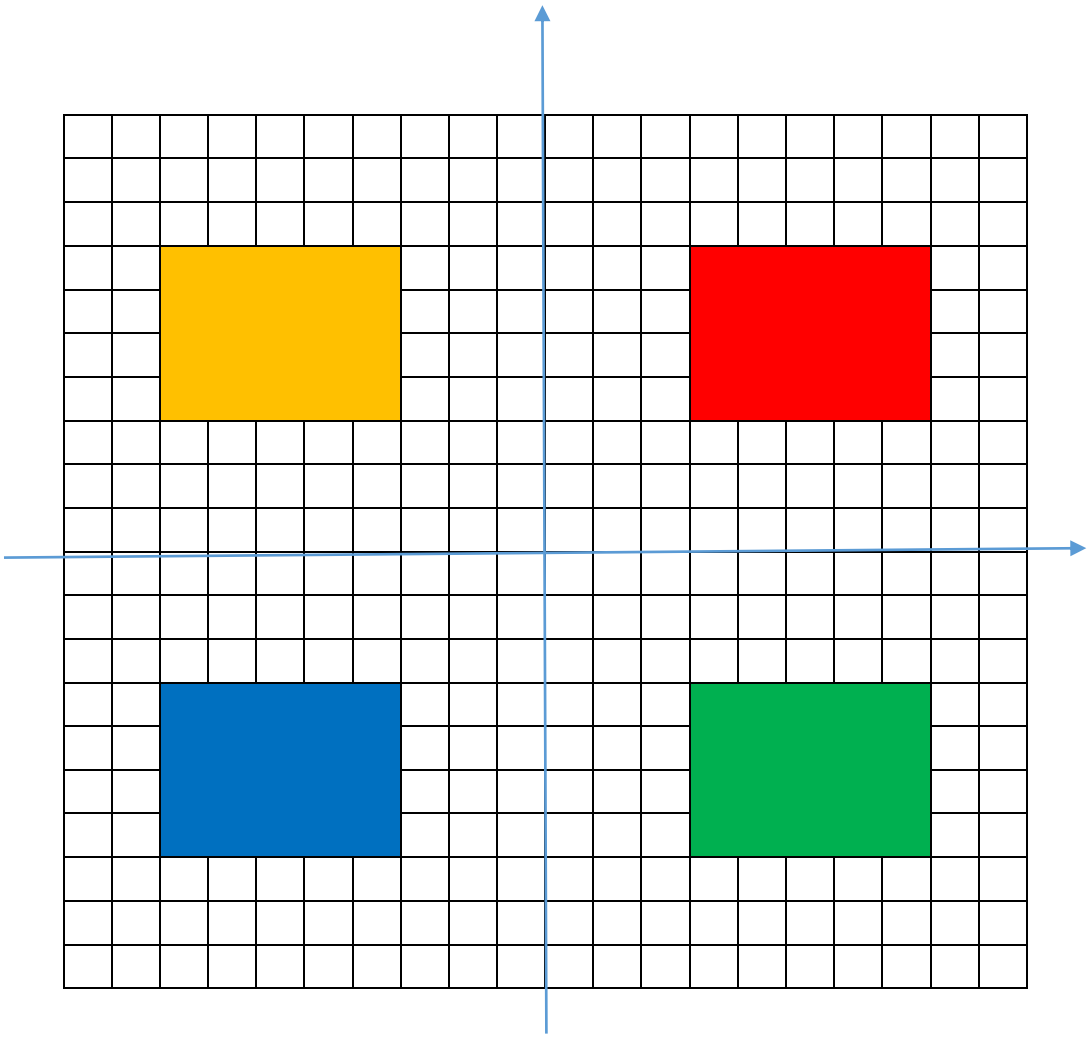


Definiție: Două puncte A și B se numesc simetrice față de dreapta d, dacă dreapta d este mediatoarea segementului AB.

Observație: Dacă două puncte sunt simetrice în raport cu o dreaptă, atunci fiecare dintre ele este simetricul celuilalt față de dreapta dată.

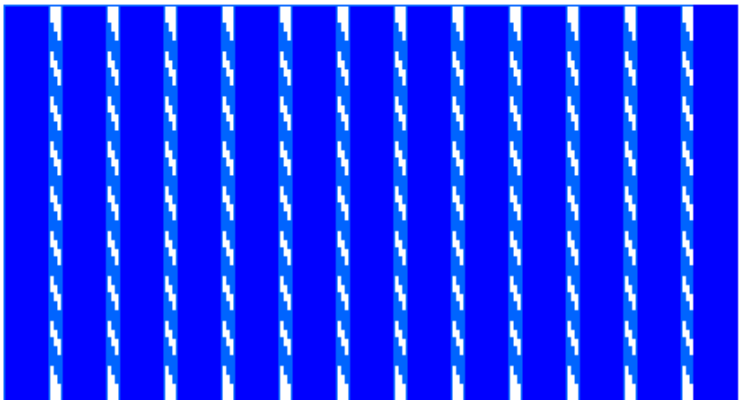
Încercați singuri

Realizați dreptunghiurile de mai jos folosind funcții pentru poziționarea pe ecran. Coordonatele dreptunghiului roșu sunt: stânga sus 100, 150, dreapta jos: 180, 100. Calculați singuri coordonatele celorlalte dreptunghiuri astfel încât să fie simetrice atât față de Ox cât și față de Oy.



4.7 Tapetul

Ema dorește tapetul din imagine.

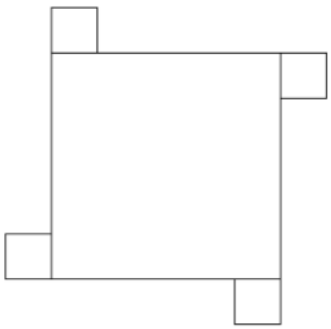


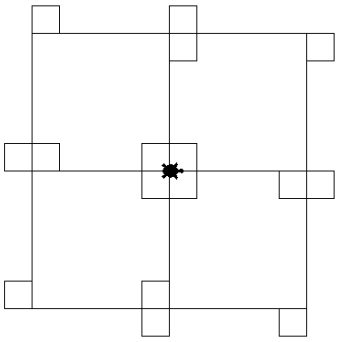
```
import turtle
import random
ema=turtle.Turtle()
turtle.colormode(255)
ema.pencolor(0,100,255)
def model():
    ema.pensize(2)
    for i in range(8):
        ema.pu()
        ema.rt(90)
        ema.fd(2)
        ema.lt(90)
        ema.fd(20)
        ema.pd()
        ema.fd(15)
        ema.pu()
        ema.rt(90)
        ema.fd(2)
        ema.lt(90)
        ema.bk(20)
        ema.pd()
        ema.fd(15)
        ema.pu()
        ema.rt(90)
        ema.fd(2)
        ema.lt(90)
        ema.bk(20)
        ema.pd()
        ema.fd(15)
        ema.pu()
        ema.lt(90)
        ema.fd(6)
        ema.rt(90)
```

```
def dreptunghi(lungime,latime,q):
    if q==1:
        ema.fillcolor(0,0,255)
        ema.begin_fill()
    for i in range(2):
        ema.fd(lungime)
        ema.rt(90)
        ema.fd(latime)
        ema.rt(90)
    if q==1:
        ema.end_fill()
    if q==2:
        model()

for i in range(12):
    ema.pensize(1)
    ema.pu()
    ema.home()
    ema.goto(32*i,0)
    ema.pd()
    ema.lt(90)
    dreptunghi(220,25,1)
    ema.rt(90)
    ema.fd(25)
    ema.lt(90)
    dreptunghi(220,7,2)
ema.goto(32*12,0)
dreptunghi(220,25,1)
```

4.8 Gresia

Ema ar dori gresie din bucăți de forma celor din imaginea de mai jos. Ajutați-o pe Ema să genereze bucăți de gresie alăturate. Observam că figura este formată din pătrate de laturi diferite. Presupunem că latura pătratului mare are lungimea 150 iar latura pătratului mic are lungimea 30.





Soluție:

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def patrat(latura):
    for i in range(4):
        ema.fd(latura)
        ema.rt(90)

def figura_de_baza(latura1):
    for i in range(4):
        ema.fd(latura1)
        patrat(30)
        ema.rt(90)

for j in range(4):
    figura_de_baza(150)
    ema.rt(90)
```

Încercați singuri

1. Adăugați culori modelului
2. Realizați alte modalități de aranjare a gresiei

Încercați să rulați programul de mai jos.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def patrat(latura):
    for i in range(4):
        ema.fd(latura)
        ema.rt(90)

patrat(150)
while True:
    for j in range(4):
        for i in range(4):
            ema.fd(150)
            patrat(30)
            ema.rt(90)
        ema.rt(90)
    ema.fd(300)
```

Ce ati remarcat? Broscuța desenează la infinit plăci de gresie.

De ce? Deoarece avem o instrucțiune while True: care reprezintă o buclă infinită. Pentru a opri programul după desenarea unui anumit număr de plăci vom folosi instrucțiunea break.

Instrucțiunea break

Instrucțiunea break realizează ieșirea forțată dintr-o instrucțiune repetitivă (for sau while). Instrucțiunea break este utilizată, de obicei, pentru ieșirea dintr-o buclă infinită.

Pentru a rezolva problema noastră, vom folosi un contor care va număra câte plăci de gresie au fost desenate. În momentul în care contorul a ajuns la 8, întrerupem bucla infinită.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def patrat(latura):
    for i in range(4):
        ema.fd(latura)
        ema.rt(90)

patrat(150)
k=0
while True:
    for j in range(4):
        k=k+1
        for i in range(4):
            ema.fd(150)
            patrat(30)
            ema.rt(90)
        ema.rt(90)
```

```
    if k==8:
        break
    ema.fd(300)
```

Observație importantă

În cazul în care avem mai multe instrucțiuni repetitive imbricate, instrucțiunea break va realiza ieșirea forțată din instrucțiunea repetitivă în corpul căreia se află nu și din instrucțiunile repetitive care o conțin.

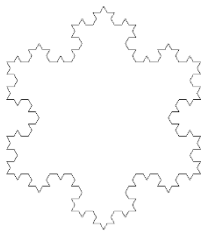
Exemplu:

```
for i in range(3):
    for j in range(3):
        if j==1:
            break
        print(i,'*',j,'=',i*j)
```

După execuția programului, pe ecran se va afișa rezultatul de mai jos.

```
0 * 0 = 0
1 * 0 = 0
2 * 0 = 0
>>>
```

4.9 Perdeaua



Ema și-ar dori ceva care să îi amintească de iernile de pe Pământ. S-a orientat pentru perdea la un model cu fulgi de nea cum este cel alăturat.

Acest model reprezintă curba lui Koch.

Curba lui Koch

Un matematician suedez, Niels Fabian Helge von Koch a definit o construcție matematică cunoscută sub numele de "Curba lui Koch". Această figură se construiește astfel: se pornește de la un triunghi echilateral și apoi pe fiecare latură se construiește un triunghi echilateral.

Pentru a rezolva această problemă vom studia noțiunea de recursivitate.

Recursivitate în Python

Recursivitatea reprezintă o tehnică de programare de o importanță deosebită. Noțiunea de recursivitate din programare este derivată în mod natural din noțiunea matematică.

În matematică, o noțiune este definită recursiv dacă în cadrul definiției apare însăși noțiunea care se definește.

Conceptul de recursivitate se referă la faptul că o problemă complicată poate fi descrisă în termenii unei versiuni mai simple a ei însăşi, ceea ce conduce la posibilitatea de descriere (în limbaj recursiv) într-o formă foarte compactă a unor probleme foarte complexe.

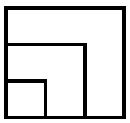
În Python, recursivitatea este exprimată cu ajutorul funcțiilor, deoarece ele se identifică printr-un nume care este apoi specificat în apeluri.

O funcție este recursivă dacă apelul său apare când funcția este activă (lansată în execuție).

Proiectarea oricărei funcții recursive trebuie să asigure ieșirea din recursivitate. De aceea, apelul recursiv este întotdeauna inclus într-o instrucțiune de decizie (IF) sau într-o instrucțiune repetitivă.

Exemplul 1

Să se scrie o funcție recursivă Python care să realizeze desenul din dreapta:



```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
def patrat_rec(latura):
    if latura<=30:
        for i in range(4):
            ema.fd(latura)
            ema.rt(90)
        patrat_rec(latura+10)
ema.lt(90)
patrat_rec(10)
```

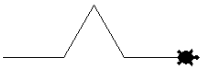
Apelul funcției recursive se realizează prin patrat_rec(10)

Să revenim la problema noastră. Definim funcția Koch astfel:

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")

def Koch(n, latura):
    if n<1:
        if latura>0:
            ema.fd(latura)
            return
    Koch(n-1,latura//3)
    ema.lt(60)
    Koch(n-1, latura//3)
    ema.rt(120)
    Koch(n-1,latura//3)
    ema.lt(60)
    Koch(n-1,latura//3)
```

La apelul Koch(0,200) se va genera figura din dreapta.

La apelul Koch(1,200) se va genera figura din dreapta.



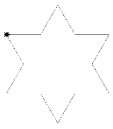
La apelul Koch(2,200) se va genera figura din dreapta

Definim funcția fulg astfel:

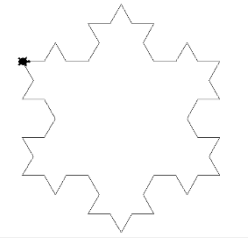
```
def fulg(n,latura):
    Koch(n,latura)
    ema.rt(120)
    Koch(n,latura)
    ema.rt(120)
    Koch(n,latura)
    ema.rt(120)
```

La apelul fulg(0,390) se va genera figura din dreapta.

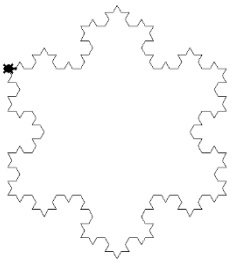
La apelul fulg(1,390) se va genera figura din dreapta.



La apelul fulg(2,390) se va genera figura din dreapta.

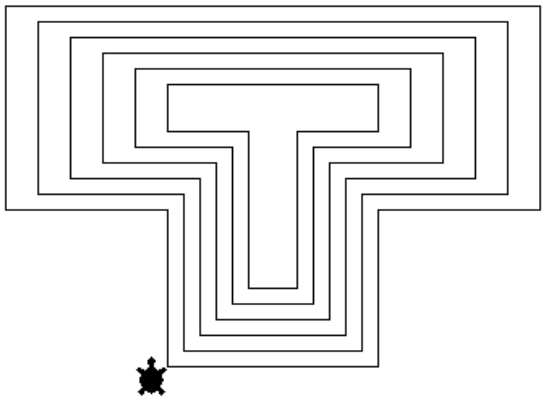


La apelul fulg(3,390) se va genera figura din dreapta.



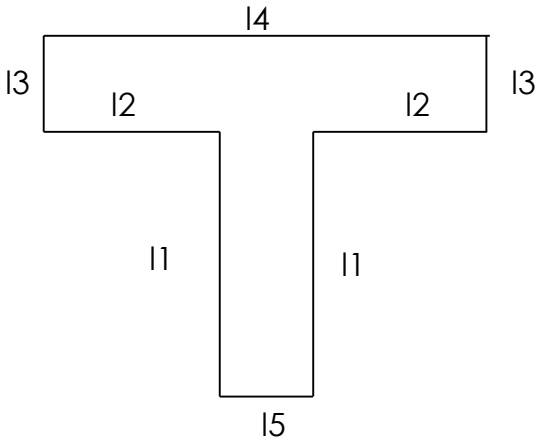
4.10 Unele îmbunătățiri ale lui Coco

Coco s-a gândit să își scrie numele mai artistic pe exteriorul rachetei. A văzut o literă scrisă ca mai jos, i-a plăcut și a încercat să o deseneze. A rugat-o pe Ema să îl ajute.



Iată soluția pentru figura de mai sus:

Pasul 1 Scriem funcția pentru desenarea unei singure litere T. Dimensiunile sunt prezentate în imaginea de mai jos.



```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
def t(l1,l2,l3,l4,l5):
    ema.fd(l1)
    ema.lt(90)
    ema.fd(l2)
    ema.rt(90)
    ema.fd(l3)
    ema.rt(90)
    ema.fd(l4)
    ema.rt(90)
    ema.fd(l3)
    ema.rt(90)
    ema.fd(l2)
    ema.lt(90)
    ema.fd(l1)
    ema.rt(90)
    ema.fd(l5)
    ema.rt(90)

ema.lt(90)
t(100,50,30,130,30)
```

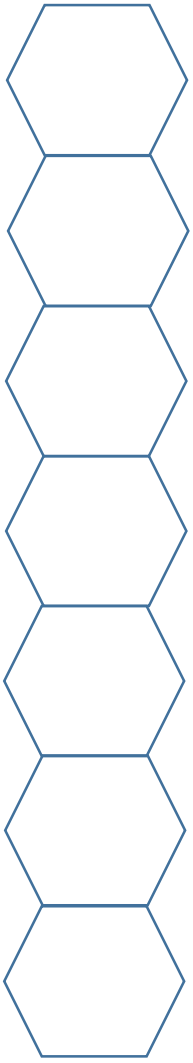
În urma execuției programului din partea stângă, pe ecran ne apare o singură literă T.

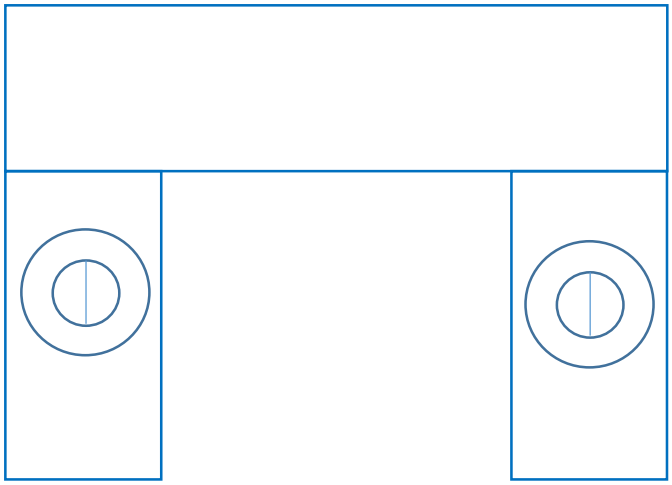
Pasul 2 Definim recursiv funcția multiple_t care apelează funcția t. Adăugăm, de asemenea, un parametru n care va indica numărul literelor T care urmează a fi desenate.

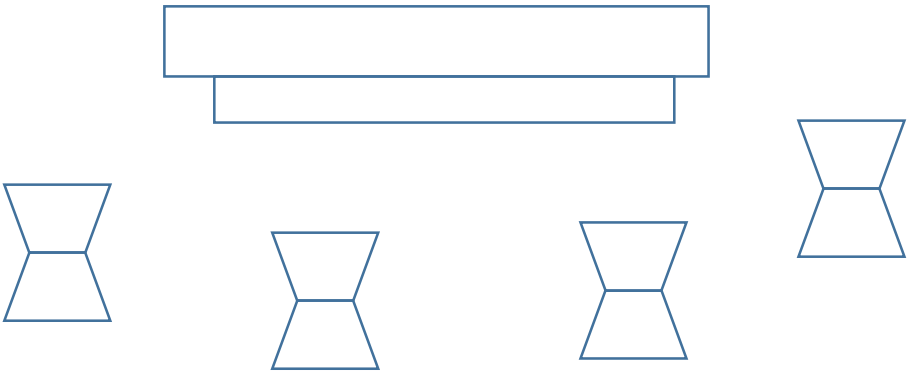
```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
def t(l1,l2,l3,l4,l5):
    ema.fd(l1)
    ema.lt(90)
    ema.fd(l2)
    ema.rt(90)
    ema.fd(l3)
    ema.rt(90)
    ema.fd(l4)
    ema.rt(90)
    ema.fd(l3)
    ema.rt(90)
    ema.fd(l2)
    ema.lt(90)
    ema.fd(l1)
    ema.rt(90)
    ema.fd(l5)
    ema.rt(90)

def multiple_t(l1,l2,l3,l4,l5,n):
    if n>0:
        t(l1,l2,l3,l4,l5)
        ema.pu()
        ema.bk(10)
        ema.lt(90)
        ema.fd(10)
        ema.pd()
        ema.rt(90)
        multiple_t (l1,l2+10,l3+20,l4+40,l5+20,n-1)
ema.lt(90)
multiple_t(100,50,30,130,30,6)
```

Încercați singuri

1. Desenați numele COCO adaptând pentru literele C și O modelul folosit pentru litera T.
2. Ajutați-o pe Ema să realizeze Coloana Infinitului, Poarta Sărutului și Masa Tăcerii după modelele de mai jos:

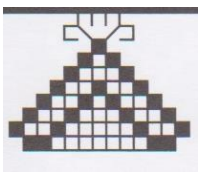


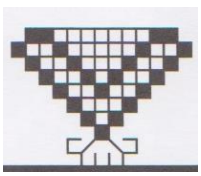


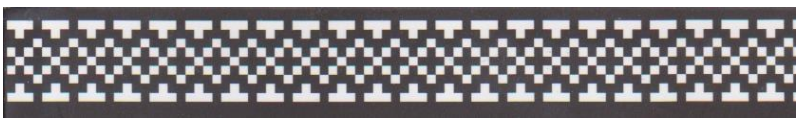


3. Identificați elemente specifice altor țări și încercați să le realizați în Python.

4. Realizați modelele tradiționale românești prezentate mai jos.

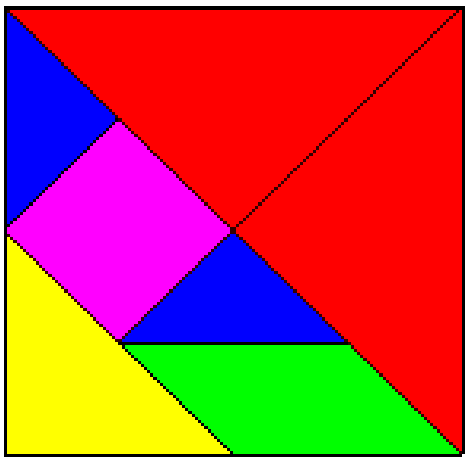






4.11 Jocul Tangram

Ema dorește să ia de pe Pământ o amintire de origine japoneză, jocul Tangram. O aranjare a figurilor jocului Tangram dorește să apară pe faianța din baie.



Jocul Tangram – descriere

Pătratul magic sau Jocul celor şapte figuri este o tehnică de lucru de origine japoneză care constă în combinarea a şapte figuri geometrice (numite și tanuri), respectiv 2 triunghiuri mari, 1 triunghi mijlociu, 2 triunghiuri mici, un pătrat şi un paralelogram, provenite din împărțirea unui pătrat mare, după reguli simple, dar stricte. Cu ajutorul acestor figuri geometrice pot fi realizate imagini stilizate ale unor obiecte reale.

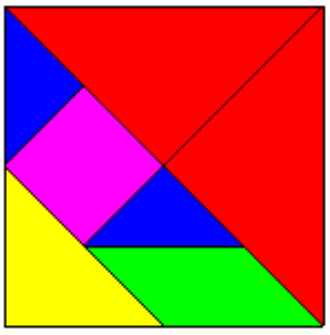
Regulile ce trebuie respectate în construcția imaginilor, prin această tehnică, sunt următoarele:

- se folosesc toate figurile geometrice ale pătratului în figura creată;

- acestea au voie să se atingă, dar nu să se suprapună.

Efectuarea calculelor matematice

l



Vom nota cu l latura pătratului. Toate calculele matematice le vom face relativ la l.

Calculele matematice pentru triunghiul dreptunghic mare

Aplicând teorema lui Pitagora calculam lungimea diagonalei. Obținem $l^2+l^2=d^2$, adică $2l^2=d^2$. Din ultima relație rezultă $d=l\sqrt{2}$. Triunghiul dreptunghic mare are catetele egale cu jumătatea diagonalelor, adică $l\frac{\sqrt{2}}{2}$. Efectuând simplificarea obținem lungimile catetelor $l/\sqrt{2}$.

Calculele matematice pentru triunghiul dreptunghic mediu

Triunghiul dreptunghic mediu are catetele egale cu l/2 iar ipotenuza este linie mijlocie în triunghiul dreptunghic format deci egală cu jumătatea diagonalei, adică $l\frac{\sqrt{2}}{2}$ (după simplificare se obține $l/\sqrt{2}$).

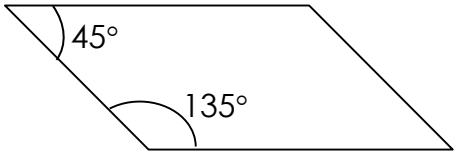
Calculele matematice pentru triunghiul dreptunghic mic

Triunghiul dreptunghic mic are ipotenuza egală cu l/2 iar catetele sunt egale cu un sfert din diagonale, adică $l\frac{\sqrt{2}}{4}$ $(\text{sau } l/(2\sqrt{2}))$.

Calculele matematice pentru pătrat

Pătratul are latura egală cu un sfert din lungimea diagonalelor, adică $l\frac{\sqrt{2}}{4}$ $(\text{sau } l/(2\sqrt{2}))$.

Calculele matematice pentru paralelogram

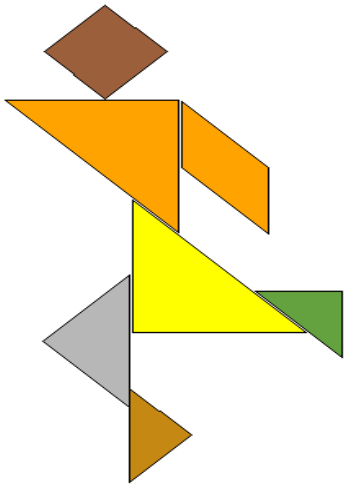


Deoarece triunghiul dreptunghic mediu (cel galben) este isoscel, unghiurile lui ascuțite au 45°.

Rezultă că unghiul obtuz al paralelogramului are 135°. Cum suma unghiurilor unui paralelogram este 360°, obținem pentru unghiurile ascuțite valoarea 45°.

Laturile paralelogramului sunt egale cu ipotenuza triunghiului dreptunghic mic și cu jumătate din ipotenuza triunghiului dreptunghic mediu, adică l/2 respectiv $l/(2\sqrt{2})$.

Ema a decis să pună pe faianță figura de mai jos.



În soluția prezentată mai jos am adăugat funcțiilor un parametru q care poate să primească două valori: 1 sau 2. Pentru q=1 se desenează figura orientată spre stânga, pentru q=2 se desenează figura orientată spre dreapta.

DESENARE TRIUNGHI DREPTUNGHIC MIC

```
import turtle
import random
import math
ema=turtle.Turtle()
ema.shape("turtle")
def triunghi_mic(latura,q):
    turtle.colormode(255)
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    ema.fd(latura/2)
    if q==1:
        ema.lt(135)
        ema.fd((latura/math.sqrt(2))/2)
        ema.lt(90)
        ema.fd((latura/math.sqrt(2))/2)
        ema.lt(135)
    else:
        ema.rt(135)
        ema.fd((latura/math.sqrt(2))/2)
        ema.rt(90)
        ema.fd((latura/math.sqrt(2))/2)
        ema.rt(135)
    ema.end_fill()
```

DESENARE TRIUNGHI DREPTUNGHIC MEDIU

```
def triunghi_mediu(latura,q):
    turtle.colormode(255)
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    ema.fd(latura/math.sqrt(2))
    if q==1:
        ema.lt(135)
        ema.fd(latura/2)
        ema.lt(90)
        ema.fd(latura/2)
        ema.lt(135)
    else:
        ema.rt(135)
        ema.fd(latura/2)
        ema.rt(90)
        ema.fd(latura/2)
        ema.rt(135)
    ema.end_fill()
```

DESENARE TRIUNGHI DREPTUNGHIC MARE

```
def triunghi_mare(latura,q):
    turtle.colormode(255)
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    ema.fd(latura)
    if q==1:
        ema.lt(135)
        ema.fd(latura/math.sqrt(2))
        ema.lt(90)
        ema.fd(latura/math.sqrt(2))
        ema.lt(135)
    else:
        ema.rt(135)
        ema.fd(latura/math.sqrt(2))
        ema.rt(90)
        ema.fd(latura/math.sqrt(2))
        ema.rt(135)
    ema.end_fill()
```

DESENARE PĂTRAT

```
def patrat(latura,q):
    turtle.colormode(255)
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    if q==1:
        for i in range(4):
            ema.fd(latura/(2*math.sqrt(2)))
            ema.lt(90)
    else:
        for i in range(4):
            ema.fd(latura/(2*math.sqrt(2)))
            ema.rt(90)
    ema.end_fill()
```

DESENARE PARALELOGRAM

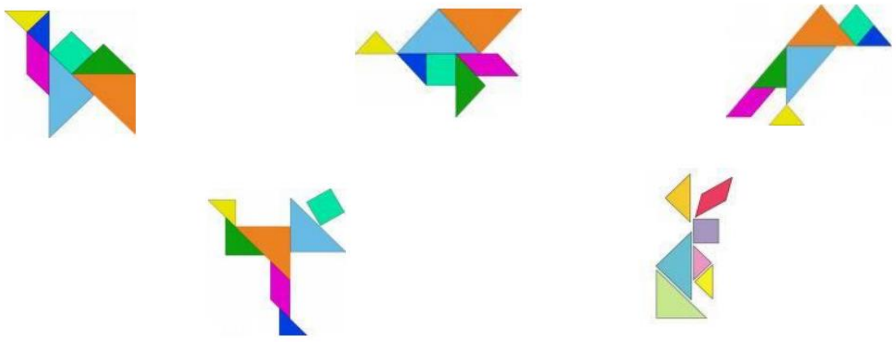
```
def paralelogram(latura,q):
    turtle.colormode(255)
    ema.pencolor("blue")
    ema.fillcolor(random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    ema.begin_fill()
    if q==1:
        for i in range(2):
            ema.fd(latura/(2*math.sqrt(2)))
            ema.lt(45)
            ema.fd(latura/2)
            ema.lt(135)
    else:
        for i in range(2):
            ema.fd(latura/(2*math.sqrt(2)))
            ema.rt(45)
            ema.fd(latura/2)
            ema.rt(135)
    ema.end_fill()
```

FUNCȚIA TANGRAM

```
def tangram():
    ema.lt(90)
    triunghi_mic(150,2)
    ema.pu()
    ema.goto(0,60)
    ema.pd()
    triunghi_mediu(150,1)
    ema.pu()
    ema.goto(108,120)
    ema.pd()
    ema.lt(45)
    triunghi_mare(150,1)
    ema.pu()
    ema.goto(130,100)
    ema.pd()
    triunghi_mic(150,2)
    ema.pu()
    ema.goto(30,200)
    ema.pd()
    triunghi_mare(150,2)
    ema.rt(45)
    ema.pu()
    ema.goto(32,305)
    ema.pd()
    ema.rt(180)
    paralelogram(150,1)
    ema.pu()
    ema.goto(-15,307)
    ema.pd()
    ema.lt(135)
    patrat(150,1)

tangram()
```

Încercați singuri

Realizați în Python următoarele figuri folosind piese ale jocului Tangram.



Despre complexitatea temporală a programelor

În informatică, nu este suficient un program să funcționeze corect, el trebuie să îndeplinească niște criterii de performanță.

Un criteriu de performanță foarte important este timpul de execuție.

Remarcați în scrierea codului pentru jocul Tangram că am adăugat un parametru suplimentar q fiecărei funcții care desenează o figură geometrică. Dacă q=1, se desenează figura spre stânga. Dacă q=2, se desenează figura spre dreapta. Am fi putut rezolva problema și fără q? Da, folosind instrucțiunile de poziționare pe ecran. De ce nu am ales această variantă? Pentru că orice apel de funcție suplimentar necesită timp iar programul nu mai este la fel de rapid.

4.12 Introducere în programarea orientată obiect

Aplicația 1 - Rulați în fereastra Shell instrucțiunile de mai jos

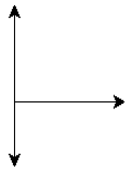
```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun
Type "copyright", "credits" or "licens
>>> import turtle
>>> ema=turtle.Turtle()
>>> ema.fd(100)
>>> coco=turtle.Turtle()
>>> coco.rt(90)
>>> coco.fd(50)
>>> alan=turtle.Turtle()
>>> alan.lt(90)
>>> alan.fd(75)
>>>
```

ema este un obiect, o instanță a clasei Turtle.

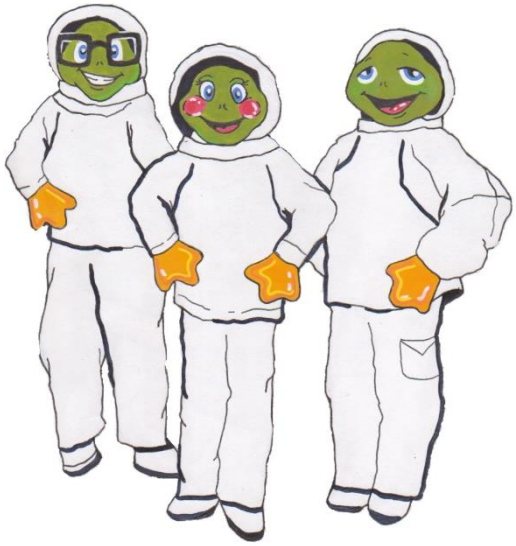
coco este un obiect, o instanță a clasei Turtle.

alan este un obiect, o instanță a clasei Turtle.

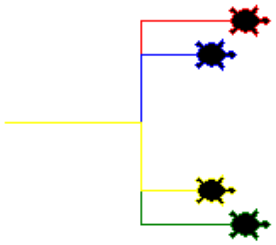
În urma execuției acestor instrucțiuni pe ecran ne apare imaginea de mai jos.



Sunt surprins. Tocmai am aflat că suntem obiecte.



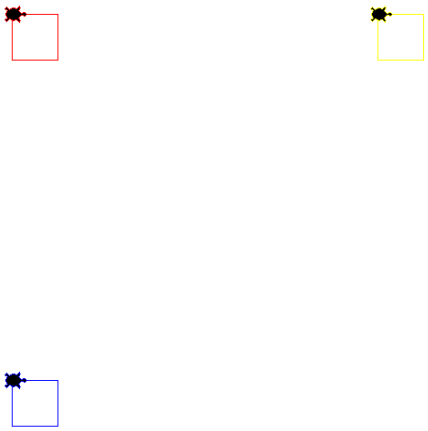
Aplicația 2 – rulați programul de mai jos. Încercați să mai adăugați o broscuță



```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
coco=turtle.Turtle()
coco.shape("turtle")
grace=turtle.Turtle()
grace.shape("turtle")
alan=turtle.Turtle()
alan.shape("turtle")
ema.pencolor("red")
ema.fd(80)
ema.lt(90)
ema.fd(60)
ema.rt(90)
ema.fd(60)
coco.pencolor("blue")
coco.fd(80)
coco.lt(90)
coco.fd(40)
coco.rt(90)
coco.fd(40)
grace.pencolor("green")
grace.fd(80)
grace.rt(90)
grace.fd(60)
grace.lt(90)
grace.fd(60)
alan.pencolor("yellow")
alan.fd(80)
alan.rt(90)
alan.fd(40)
alan.lt(90)
alan.fd(40)
```

Aplicația 3 - Rulați programul de mai jos. Imaginați-vă că broscuțele sunt firele de execuție ale unui program.

```
import turtle
ema=turtle.Turtle()
ema.shape("turtle")
coco=turtle.Turtle()
coco.shape("turtle")
grace=turtle.Turtle()
grace.shape("turtle")
ema.pu()
ema.goto(-200,200)
ema.pd()
ema.pencolor("red")
coco.pu()
coco.goto(200,200)
coco.pd()
coco.pencolor("yellow")
grace.pu()
grace.goto(-200,-200)
grace.pd()
grace.pencolor("blue")
for i in range(4):
    ema.fd(50)
    coco.fd(50)
    grace.fd(50)
    ema.rt(90)
    coco.rt(90)
    grace.rt(90)
```



BIBLIOGRAFIE

1. Jason R. Briggs, Python pour les kids, editions Eyrolles, 2013,
2. Paul Barry, David Griffiths, Head First Programming, O'Reilly
3. Jim Muller – "The Great Logo Adventure: Discovering Logo on and Off the Computer", editura Doone Publications
4. Stan Munson – *"TerrapinLogo tutorial"*, ISBN 109876543
5. https://docs.python.org/3/library/turtle.html

Această carte a fost înregistrată la Biblioteca Națională cu ISBN 978-973-0-30239-4.