CAPITOLUL II BIBLIOTECA TKINTER

În acest capitol vom învăța:

- Să creăm o fereastră;
- Să adăugăm în fereastră, utilizând funcțiile corespunzătoare, linii, dreptunghiuri, poligoane, arce, elipse;
- Să aplicăm funcțiile pentru desenarea figurilor geometrice în realizarea anumitor desene;
- Să realizăm animații simple.

Coco și Ema au găsit ajutoare în încercarea lor de a desena o racheta și alte lucruri necesare călătoriei. Ei au aflat de biblioteca TKINTER și au decis să o folosească. Este vorba oare de un loc cu multe cărți unde cei doi pot citi? Destul de asemănător, în Python, o bibliotecă este o colecție de componente create de alți programatori iar TKINTER este o bibliotecă Python foartă populară pentru crearea de interfețe utilizator.

2.1 Crearea unei ferestre

Coco creează o fereastră așa cum au multe aplicații de pe calculatorul pe care îl folosește.

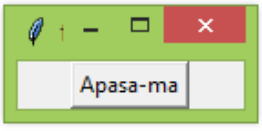
```
from tkinter import *
fereastra=Tk()
```

Această instrucțiune creează o fereastră. 'fereastra' este o variabilă ce conține un obiect din clasa Tk. Ne putem gandi la acest obiect ca la un pachet care conține tot ce ne trebuie pentru a construi ferestre și pentru a adăuga componente la acestea. Iar clasa Tk este o descriere a acestor pachete.

2.2 Adăugarea unui buton în fereastră

O fereastră goală nu îl ajută pe Coco foarte mult așa că el continuă să experimenteze ce poate face folosind TKINTER. Mai intai adaugă un buton.

```
from tkinter import *
fereastra=Tk()
btn=Button(fereastra, text="Apasa-ma")
btn.pack()
```



Acest text apare pe buton.

Pune butonul în fereastră.

2.3 Apelarea unei funcții la apăsarea pe un buton

Remarcăm faptul că la apăsarea butonului în aplicația anterioară nu se întâmplă nimic. Pentru a afișa un mesaj la apăsarea butonului procedăm ca mai jos.

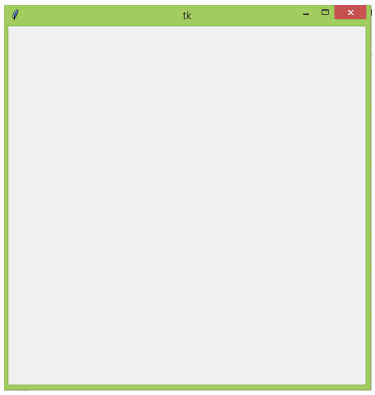
```
def salut():
  print('Salutari de pe PLANETA PROGRAMATORILOR')

from tkinter import *
fereastra=Tk()
btn=Button(fereastra, text="Apasa-ma", command=salut)
btn.pack()
```

Coco a aflat și câteva mici secrete. Btn este o variabilă care reprezintă un alt obiect care conține date și funcții și care îl ajută să construiască și să lucreze cu butoane. Pack este o funcție care definitivează plasarea butonului în fereastră. Button este o altă funcție specială care construiește butoane dacă îi spune în ce fereastră trebuie plasat butonul, ce să scrie pe el și ce trebuie să se întâmple atunci când butonul este apăsat. Funcția aceasta specială se numește constructor ceea ce nu îl miră pe Coco pentru că ea chiar construiește obiecte. Despre constructori vom afla mai multe în ultimul capitol al acestei cărți.

2.4 Crearea unei planșe pentru desen

Lui Coco chiar îi place să deseneze așa că se bucură când află că își poate construi o planșă de desenat.

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra, width=500, height=500)
plansa.pack()
```



Lățimea planșei

Înălțimea planșei

Planșa lui Coco are 500 de pixeli lățime și tot atât în înălțime iar pentru afișarea ei se folosește aceeași funcție pack ca și la buton deci nu are de ținut minte prea mult.

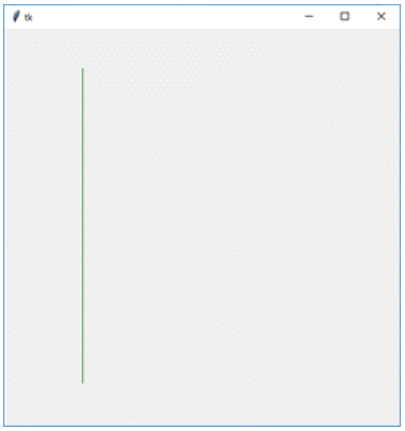
Planșa lui Coco este încă goală iar el este nerăbdător să se apuce de desenat.

2.5 Desenarea liniilor

Ema tocmai a aflat că obiectul *plansa*, în afară de *pack*, mai conține o funcție create_line cu ajutorul căreia poți desena o linie sau mai multe.

Împreună cu Coco, se pune pe încercări pentru a afla tot ce poate face ea. Pe canvas-ul gol construit deja de Coco, ei deseneaza un segment specificand punctul de inceput și pe cel de sfârșit.

```
from tkinter import *
fereastra = Tk()
plansa = Canvas(fereastra, width=500, height=500)
plansa.pack()
plansa.create_line(100, 50, 100, 450, fill="green")
```



În mintea lui Coco apare o intrebare: Cum se poate ca numerele 100, 50 să reprezinte punctul de început și 100, 450 să reprezinte punctul de sfârșit? Lucrurile se lămuresc în momentul în care Ema îi desenează sistemul de coordonate și îi arată că punctul de coordonate (0,0) este în colțul stânga sus. 100 reprezintă distanța de la punct la axa Ox si 50 reprezintă distanța de la punct la axa Oy. "green" nu ridică semne de întrebare deoarece Coco cunoaște limba engleză și își dă seama că, datorită lui, linia este desenată în culoarea verde.

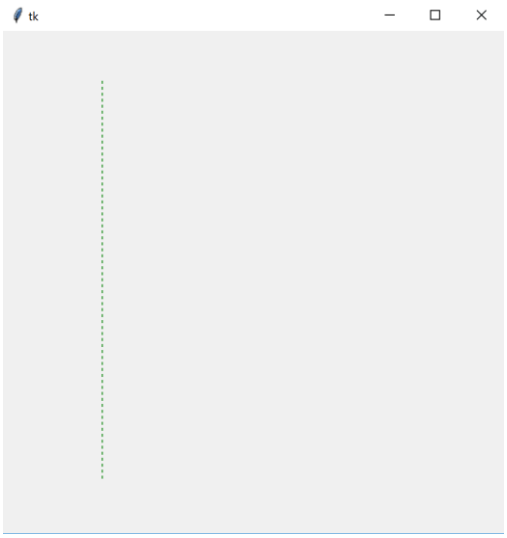
Lucrurile merg așa de bine încât cei doi decid să deseneze un zigzag.

```
from tkinter import *
fereastra = Tk()
plansa = Canvas(fereastra, width=500, height=500)
plansa.pack()
plansa.create_line(100, 50, 140, 450, 180, 50, 220, 450,
260, 50, 300, 450, 340, 50, fill="green")
```

Asta a fost foarte simplu, a trebuit doar să enumere toate punctele din figură în ordinea în care apar (100, 50), (140, 450), (180, 50), (220, 450), (260, 50), (300, 450), (340, 50).

Nu au nici un motiv să se oprească așa că decid să deseneze cu linii întrerupte.

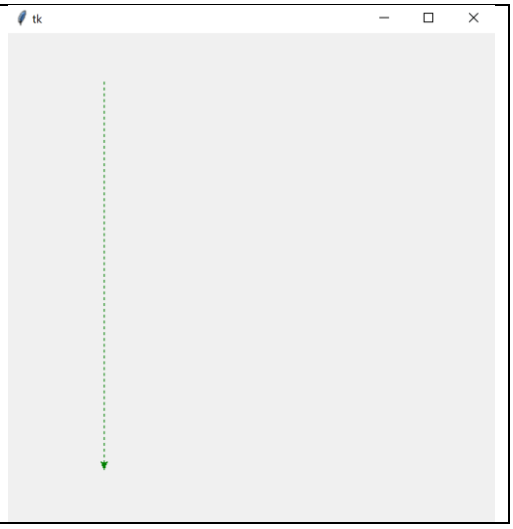
```
from tkinter import *
fereastra = Tk()
plansa = Canvas(fereastra, width=500, height=500)
plansa.pack()
plansa.create_line(100, 50, 100, 450, dash = (4, 3),
fill="green")
```



Un mic secret al celor doi: dash = (4, 3) determină ca la desenarea liniei, după fiecare 4 puncte colorate să urmeze 3 puncte goale.

După care, adaugă un capăt în formă de săgeată la segment.

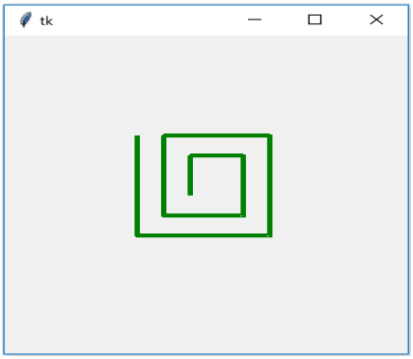
```
from tkinter import *
fereastra = Tk()
plansa = Canvas(fereastra, width=500, height=500)
plansa.pack()
plansa.create_line(100, 50, 100, 450, dash = (4, 3),
fill="green", arrow="last")
```



Le-a plăcut așa de mult încât au lăsat și câteva exerciții pentru cititori.

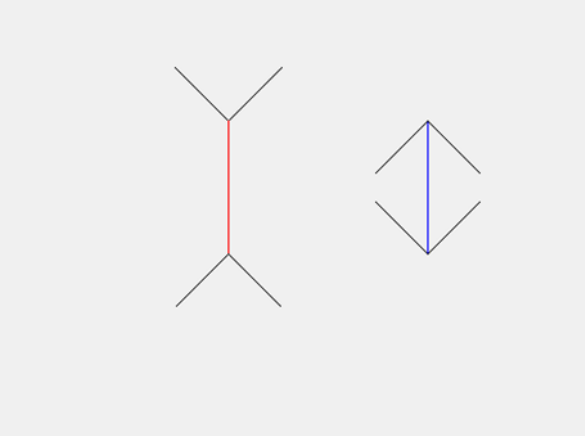
Încercați singuri

1. Desenați o figură cum este cea de mai jos.



Dimensiunea initială a segmentului desenat este de 100 de pixeli, aceasta scăzând treptat cu câte 20 de pixeli. Linia a fost desenată îngroșat cu ajutorul unui parametru suplimentar width=4 indicând o grosime a liniei de 4 pixeli.

2. Care dintre segmentele de dreaptă de mai jos este mai lung, cel roșu sau cel albastru? Analizați codul sursă pentru a afla răspunsul.

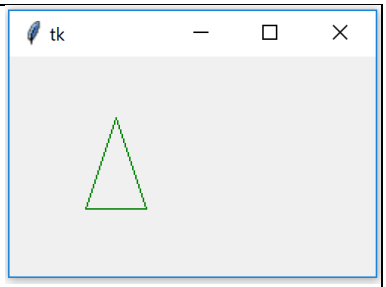


```
from tkinter import *
import random
import time
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
plansa.create_line(150,150,150,250,fill="red")
plansa.create_line(110,110,150,150)
plansa.create_line(190,110,150,150)
plansa.create_line(150,250,110,290)
plansa.create_line(150,250,190,290)
plansa.create_line(300,150,300,250,fill="blue")
plansa.create_line(300,150,260,190)
plansa.create_line(300,150,340,190)
plansa.create_line(300,250,260,210)
plansa.create_line(300,250,340,210)
```

Foarte multe triunghiuri

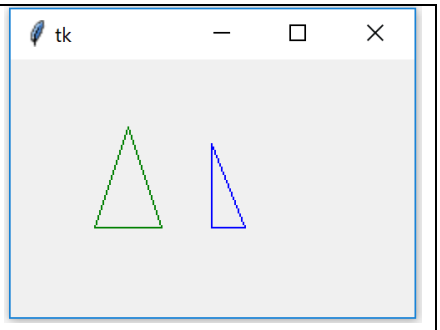
Știm să desenăm linii dar varful navei noastre spațiale se aseamană cu un triunghi. Oare îl putem desena? Nu ar trebui să fie prea greu, avem de desenat trei linii.

```
from tkinter import *

fereastra=Tk()

plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

liniaAB = plansa.create_line(50, 100, 70, 40, fill="green")
liniaBC = plansa.create_line(70, 40, 90, 100, fill="green")
liniaCA = plansa.create_line(90, 100, 50, 100, fill="green")
```



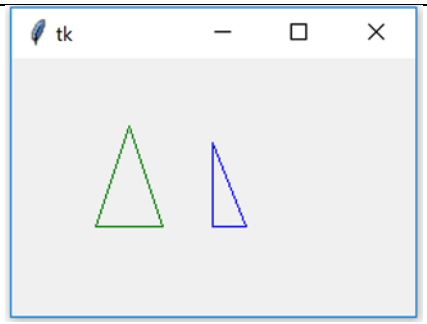
Vârful rachetei noastre arată foarte bine dar mai trebui să desenăm triunghiuri pentru aripioarele rachetei. Mai putem face unul? Sigur.

```
from tkinter import *

fereastra=Tk()

plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

liniaAB = plansa.create_line(50, 100, 70, 40, fill="green")
liniaBC = plansa.create_line(70, 40, 90, 100, fill="green")
liniaCA = plansa.create_line(90, 100, 50, 100, fill="green")

liniaEF = plansa.create_line(120, 100, 120, 50, fill="blue")
liniaFG = plansa.create_line(120, 50, 140, 100, fill="blue")
liniaGE = plansa.create_line(140, 100, 120, 100, fill="blue")
```



- Este gata și o aripioară a rachetei dar ce nu îmi place este că trebuie să repetăm tot timpul cele trei funcții de trasare a liniilor pentru fiecare triunghi. Am putea face ceva?
- O, da! Am aflat că există funcții, grupuri de instrucțiuni care au un nume și care pot fi utilizate menționând numai numele lor. Cred că putem grupa cele trei instrucțiuni de desenare a unui triunghi într-o funcție.
- Și cum o să-i spunem?
- Chiar deseneaza_triunghi.

```
from tkinter import *

fereastra=Tk()

plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

def deseneaza_triunghi(ax, ay, bx, by, cx, cy, culoare):
    plansa.create_line(ax, ay, bx, by, fill=culoare)
    plansa.create_line(bx, by, cx, cy, fill=culoare)
    plansa.create_line(cx, cy, ax, ay, fill=culoare)

deseneaza_triunghi(50, 100, 70, 40, 90, 100, "green")
deseneaza_triunghi(120, 100, 120, 50, 140, 100, "blue")
```



- Este mult mai simplu așa. Nu mai trebuie sa scriem create_line și nici să repetăm coordonatele vârfurilor. Este de ajuns să apelăm deseneaza_triunghi și să menționam coordonatele vârfurilor și culoarea. Este și mult mai ușor de înțeles.
- Mi-a venit o idee, ce ar fi dacă în interiorul triunghiului nostru vom desena un alt triunghi? Putem să alegem cele trei vârfuri ale triunghiului din interior ca fiind mijloacele laturilor triunghiului nostru initial. Hai să incercăm.
- Să le explicăm și copiilor cum am procedat. Mai întâi am folosit variabilele ax și ay pentru a reprezenta coordonatele varfului A. Puteam folosi in continuare numere dar cu ajutorul variabilelor ne putem da seama mai bine despre cum este organizat programul. Acesta devine

din ce în ce mai complicat și este cam greu de înțeles folosind numai numere. La fel, bx și by reprezintă coordonatele vârfului B iar cx și cy reprezintă coordonatele vârfului C.

```
ax = 40
ay = 120
bx = 100
by = 60
cx = 160
cy = 120
```

După care am desenat conturul triunghiului inițial. Este puțin mai mare față de triunghiurile anterioare pentru că va conține un alt triunghi în interior.

```
deseneaza_triunghi(ax, ay, bx, by, cx, cy, "green")
```

În continuare, calculăm mijlocul laturii AB pe care îl vom nota cu M. Coordonatele lui sunt mx și my.

```
mx = (ax + bx)/2
my = (ay + by)/2
```

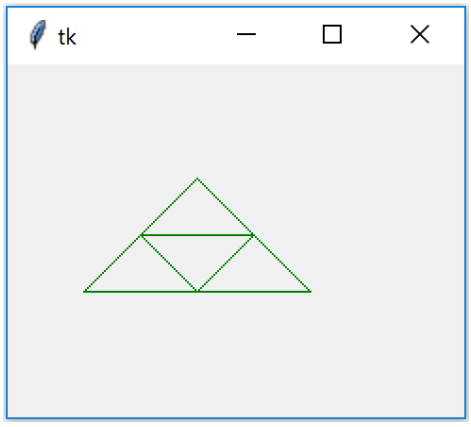
Similar, N și P sunt mijloacele lui BC și CA iar coordonatele lor sunt nx, ny și px, py.

```
nx = (bx + cx)/2
ny = (by + cy)/2
px = (cx + ax)/2
py = (cy + ay)/2
```

Tot ce ne mai rămâne acum este să desenăm triunghiul interior

```
deseneaza_triunghi(mx, my, nx, ny, px, py, "green")
```

Și suntem gata!



Iată întregul program:

```
from tkinter import *
fereastra=Tk()

plansa=Canvas(fereastra,width=600, height=600)
```

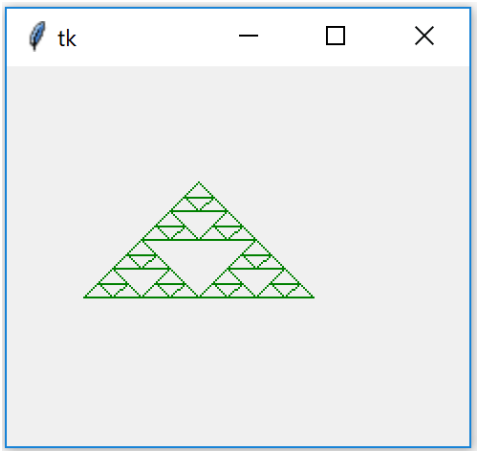
```
plansa.pack()

def deseneaza_triunghi(ax, ay, bx, by, cx, cy, culoare):
    plansa.create_line(ax, ay, bx, by, fill=culoare)
    plansa.create_line(bx, by, cx, cy, fill=culoare)
    plansa.create_line(cx, cy, ax, ay, fill=culoare)

ax = 40
ay = 120
bx = 100
by = 60
cx = 160
cy = 120

deseneaza_triunghi(ax, ay, bx, by, cx, cy, "green")
mx = (ax + bx)/2
my = (ay + by)/2
nx = (bx + cx)/2
ny = (by + cy)/2
px = (cx + ax)/2
py = (cy + ay)/2
deseneaza_triunghi(mx, my, nx, ny, px, py, "green")
```

- Dar este super, Coco! Putem să desenăm in continuare și să facem triunghiuri din ce în ce mai mici în interiorul triunghiurilor mari. Vom obține un model foarte frumos.
- O da! Va trebui însă să ne definim o nouă funcție pentru că este greu sa repetam codul pentru fiecare triunghi mai mic pe care îl desenăm în interiorul unui triunghi mai mare. Vom numi funcția repeta_triunghi_in_triunghi și ea va desena triunghiul MNP după care va repeta procedeul în cazul triunghiurilor AMP, MBN și NCP. Vom folosi și un parametru adancime care ne va spune cât de adânc trebuie să mergem desenând triunghiuri în triunghiuri. Atunci când adâncimea devine 0 ne vom opri.
- Planul este foarte bun Coco. Uite, am și scris funcția la care te gândești tu.

```
def repeta_triunghi_in_triunghi(ax, ay, bx, by, cx, cy, culoare, adancime):
    mx = (ax + bx)/2
    my = (ay + by)/2
    nx = (bx + cx)/2
    ny = (by + cy)/2
    px = (cx + ax)/2
    py = (cy + ay)/2

    deseneaza_triunghi(mx, my, nx, ny, px, py, culoare)
    adancime = adancime - 1
    if adancime > 0:
        repeta_triunghi_in_triunghi(ax, ay, mx, my, px, py, culoare, adancime)
        repeta_triunghi_in_triunghi(mx, my, bx, by, nx, ny, culoare, adancime)
        repeta_triunghi_in_triunghi(nx, ny, cx, cy, px, py, culoare, adancime)
```

Daca o apelăm cu o adâncime de 3 vom obține un desen foarte frumos.

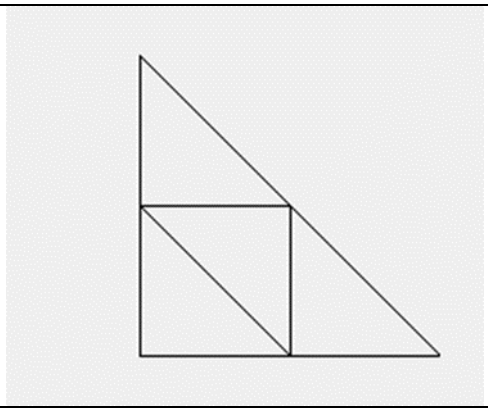
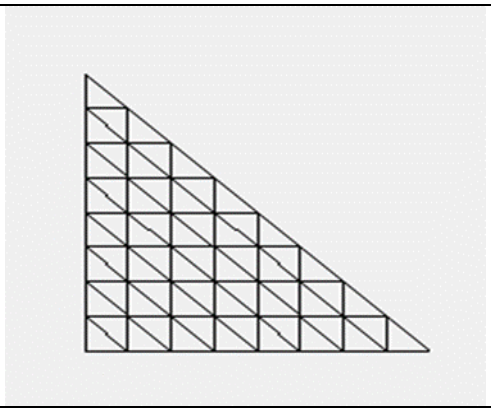
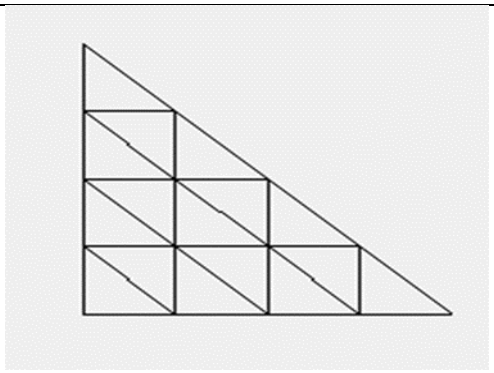
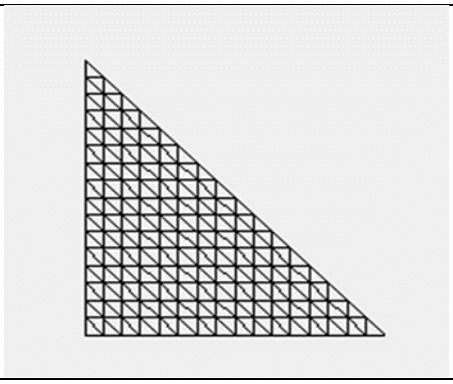
```
repeta_triunghi_in_triunghi(ax, ay, bx, by, cx, cy, "green", 3)
```



Întregul program este următorul:

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

def deseneaza_triunghi(ax, ay, bx, by, cx, cy, culoare):
    plansa.create_line(ax, ay, bx, by, fill=culoare)
    plansa.create_line(bx, by, cx, cy, fill=culoare)
    plansa.create_line(cx, cy, ax, ay, fill=culoare)

ax = 40
ay = 120
bx = 100
by = 60
cx = 160
cy = 120
deseneaza_triunghi(ax, ay, bx, by, cx, cy, "green")

def repeta_triunghi_in_triunghi(ax, ay, bx, by, cx, cy, culoare, adancime):
    mx = (ax + bx)/2
    my = (ay + by)/2
    nx = (bx + cx)/2
    ny = (by + cy)/2
    px = (cx + ax)/2
    py = (cy + ay)/2
    deseneaza_triunghi(mx, my, nx, ny, px, py, "green")
    adancime = adancime - 1
    if adancime > 0:
        repeta_triunghi_in_triunghi(ax, ay, mx, my, px, py, culoare, adancime)
        repeta_triunghi_in_triunghi(mx, my, bx, by, nx, ny, culoare, adancime)
        repeta_triunghi_in_triunghi(nx, ny, cx, cy, px, py, culoare, adancime)

repeta_triunghi_in_triunghi(ax, ay, bx, by, cx, cy, "green", 3)
```

Pentru prietenii noștri curioși care doresc să exerseze am pregătit următoarele figuri pe care îi rugăm să le reproducă. Un mic secret, cele patru figuri sunt obținute plecând de la același triunghi și modificând parametrul adâncime.

adancime=1	adancime=3
	
adancime=2	**adancime=4**
	

Cu cât avem o adâncime mai mare, cu atât desenăm mai multe triunghiuri în interiorul altor triunghiuri acestea devenind din ce în ce mai mici. Și mai este o diferență pe care unii dintre voi au observat-o. Se desenează și în interiorul triunghiului din mijloc așa că programul trebuie modificat puțin.

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

def deseneaza_triunghi(ax, ay, bx, by, cx, cy, culoare):
    plansa.create_line(ax, ay, bx, by, fill=culoare)
    plansa.create_line(bx, by, cx, cy, fill=culoare)
    plansa.create_line(cx, cy, ax, ay, fill=culoare)

ax = 40
ay = 40
bx = 40
by = 200
cx = 200
cy = 200
deseneaza_triunghi(ax, ay, bx, by, cx, cy, "black")
def repeta_triunghi_in_triunghi_plus_centru(ax, ay, bx, by, cx, cy, culoare, adancime):
```

```
    mx = (ax + bx)/2
    my = (ay + by)/2
    nx = (bx + cx)/2
    ny = (by + cy)/2
    px = (cx + ax)/2
    py = (cy + ay)/2

    deseneaza_triunghi(mx, my, nx, ny, px, py, culoare)
    adancime = adancime - 1
    if adancime > 0:
        repeta_triunghi_in_triunghi_plus_centru(ax, ay, mx, my, px, py, culoare, adancime)
        repeta_triunghi_in_triunghi_plus_centru(mx, my, bx, by, nx, ny, culoare, adancime)
        repeta_triunghi_in_triunghi_plus_centru(nx, ny, cx, cy, px, py, culoare, adancime)
        repeta_triunghi_in_triunghi_plus_centru(mx, my, nx, ny, px, py, culoare, adancime)

repeta_triunghi_in_triunghi_plus_centru(ax, ay, bx, by, cx, cy, "black", 4)
```

Funcția noastră se numește acum repeta_triunghi_in_triunghi_plus_centru pentru ca vom desena și în triunghiul din centru MNP. Se observă și apelul suplimentar care face acest lucru, este ultima linie din funcția repeta_triunghi_in_triunghi_plus_centru.

```
repeta_triunghi_in_triunghi_plus_centru(mx, my, nx, ny, px, py, culoare, adancime)
```

Uuf, desenele noastre sunt foarte frumoase dar sunt monocrome, am desenat numai cu verde sau numai cu negru. Mi-ar plăcea să alegem culorile la întâmplare.

Cred ca putem rezolva asta. Biblioteca random din Python conține o funcție choice care alege pentru noi la întâmplare o culoare dintr-o listă.

Vom defini o lista de culori

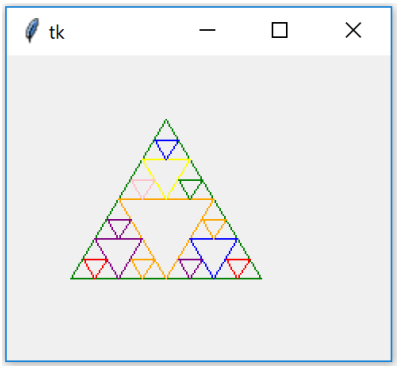
```
lista_culori = ['red','green','blue','yellow','pink','orange','purple']
```

și vom alege din ea o culoare la întâmplare cu ajutorul lui choice

```
culoare = choice(lista_culori)
```

Noua noastră funcție se va numi repeta_triunghi_in_triunghi_multicolor și nu va mai avea parametrul culoare pentru că aceasta se alege la întâmplare din lista noastră de culori.

Am terminat modificările și ce colorat este desenul nostru!



Programul complet în cazul in care doriți să încercați și voi este:

```
from tkinter import *
from random import *

fereastra=Tk()

plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()

def deseneaza_triunghi(ax, ay, bx, by, cx, cy, culoare):
    plansa.create_line(ax, ay, bx, by, fill=culoare)
    plansa.create_line(bx, by, cx, cy, fill=culoare)
    plansa.create_line(cx, cy, ax, ay, fill=culoare)

ax = 40
ay = 140
bx = 100
by = 40
cx = 160
cy = 140

deseneaza_triunghi(ax, ay, bx, by, cx, cy, "green")

def repeta_triunghi_in_triunghi_multicolor(ax, ay, bx, by, cx, cy, adancime):
    mx = (ax + bx)/2
    my = (ay + by)/2
    nx = (bx + cx)/2
    ny = (by + cy)/2
    px = (cx + ax)/2
    py = (cy + ay)/2

    lista_culori = ['red','green','blue','yellow','pink','orange','purple']
    culoare = choice(lista_culori)

    deseneaza_triunghi(mx, my, nx, ny, px, py, culoare)

    adancime = adancime - 1

    if adancime > 0:
        repeta_triunghi_in_triunghi_multicolor(ax, ay, mx, my, px, py, adancime)
        repeta_triunghi_in_triunghi_multicolor(mx, my, bx, by, nx, ny, adancime)
        repeta_triunghi_in_triunghi_multicolor(nx, ny, cx, cy, px, py, adancime)

repeta_triunghi_in_triunghi_multicolor(ax, ay, bx, by, cx, cy, 3)
```

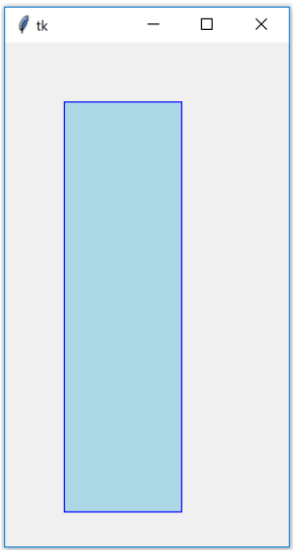
Identificarea obiectelor pe planșă

Dacă avem mai multe obiecte grafice pe o planșă cum le putem identifica? Toate funcțiile pentru desenarea obiectelor grafice returnează un identificator care este un număr. Mai jos este un exemplu. Se observă că se returnează valorile 1,2,3. Dacă am adăuga încă un obiect pe planșă, acesta ar avea identificatorul 4.

```
Python 3.5.2 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> from tkinter import *
>>> fereastra=Tk()
>>> plansa=Canvas(fereastra,width=600, height=600)
>>> plansa.pack()
>>> plansa.create_line(50, 100, 70, 40, fill="green")
1
>>> plansa.create_line(70, 40, 90, 100, fill="green")
2
>>> plansa.create_line(90, 100, 50, 100, fill="green")
3
>>>
```

2.6 Desenarea dreptunghiurilor

- Cât de mult mă bucur că am reușit să desenăm vârful rachetei noastre și aripioarele ei. Avem și un model foarte interesant cu triunghiuri să o decorăm!
- Să nu uităm că nu am terminat, mai avem corpul rachetei și pentru el va trebui să desenăm un dreptunghi. Am făcut câteva cercetări și am aflat că putem folosi funcția create_rectangle. Privește, am reușit deja să îl desenez.

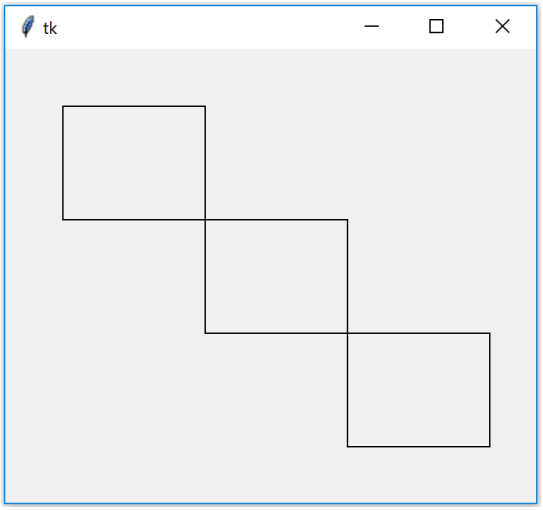


- Îți voi arăta și cum am procedat, nu a fost un program foarte complicat, a fost chiar simplu.

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
plansa.create_rectangle(50, 50, 150, 400, fill="light blue", outline="blue")
```

- Primele două numere 50, 50 din apelul lui create_rectangle sunt coordonatele colțului stânga sus al dreptunghiului iar numerele care urmeaza 150 și 400 sunt coordonatele colțului dreapta jos. Avem un dreptunghi cu lungimea 100 (150 – 50) și lățimea 350 (400 - 50). Culoarea albastru deschis a dreptunghiului o obținem cu ajutorul lui fill="light blue" iar chenarul albastru este desenat pentru că am folosit outline="blue".

E așa de ușor! Tocmai am desenat și eu ceva.



- Avem acum o scară cu ajutorul căreia putem urca în rachetă. Nici nu a fost așa de greu de făcut.

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=400, height=400)
plansa.pack()
lungime=100
latime=80
for i in range(0,3):
   plansa.create_rectangle(40+lungime*i,40+latime*i,40+lungime*(i+1),40+latime*(i+1))
```

- Mai întâi am stabilit dimensiunea treptelor, acestea au lungimea de 100 și lățimea de 80. Apoi, pentru că aveam de desenat 3 trepte am folosit comanda:

```
for i in range(0,3)
```

- Dacă ar fi să o transpunem în limba română, ea s-ar exprima ca “repetă pentru i în mulțimea 0, 1 și 2”. Avem 3 numere 0, 1 și 2, pentru fiecare din ele construindu-se câte o treaptă.

- Cum ai făcut ca treptele să se potrivească atât de bine, una în continuarea celeilate?

Ei bine, am făcut câteva calcule. Să îți arăt.

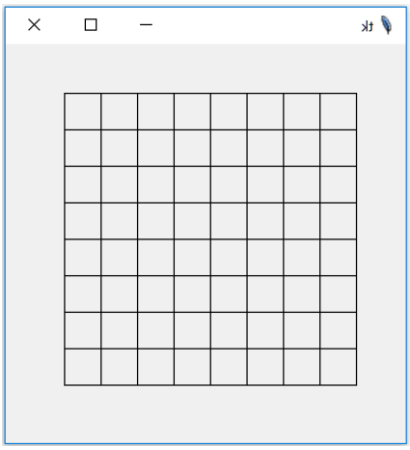
		Colțul stânga sus	Colțul dreapta jos
Treapta 1	i=0	X = 40+100*0 = 40 Y = 40+80*0 = 40	X = 40+100*1 = 140 Y = 40+80*1 = 120
Treapta 2	i=1	X = 40+100*1 = 140 Y = 40+80*1 = 120	X = 40+100*2 = 240 Y = 40+80*2 = 200
Treapta 3	i=2	X = 40+100*2 = 240 Y = 40+80*2 = 200	X = 40+100*3 = 340 Y = 40+80*3 = 280

- Cred că observi acum că treapta 1 și treapta 2 au un colț comun la fel cum au și treapta 2 și treapta 3. Așa am făcut ca treptele să vină una în continuarea celeilalte, ele se ating într-un colț.

- Este grozav! Mai am totuși o întrebare: cum ai obținut mulțimea 0, 1 și 2 de numere?

- Mulțimea aceasta de numere am obținut-o cu ajutorul apelului funcției range(0, 3). Această funcție ne dă un șir de numere întregi începând cu 0 și oprindu-se înainte să ajungă la 3, deci 0, 1 și 2. Poți să încerci și tu să adaugi o nouă treaptă la scară folosind range(0, 4). Mulțimea de numere pe care ne-o dă apelul lui range în acest caz este 0, 1, 2 și 3.

Am înțeles. Cred că am să desenez o tablă de șah. Îmi place să joc șah în timpul călătoriilor. Tabla de șah este formată din pătrate organizate în 8 linii și 8 coloane așa că am să desenez 64 de dreptunghiuri la care lungimea este egală cu lățimea.



Programul este foarte asemănător cu cel făcut de tine.

```
from tkinter import *
fereastra=Tk()

plansa=Canvas(fereastra, width=400, height=400)
plansa.pack()

lungime=30
latime=30

for i in range(0,8):
   for j in range(0,8):
      plansa.create_rectangle(40+lungime*i,40+latime*j,40+lungime*(i+1),40+latime*(j+1))
```

Cred că trebuie să le explicăm cititorilor de ce ai folosit formulele 40+lungime*i,40+latime*j pentru coordonatele vârfurilor stânga sus ale pătratelor.

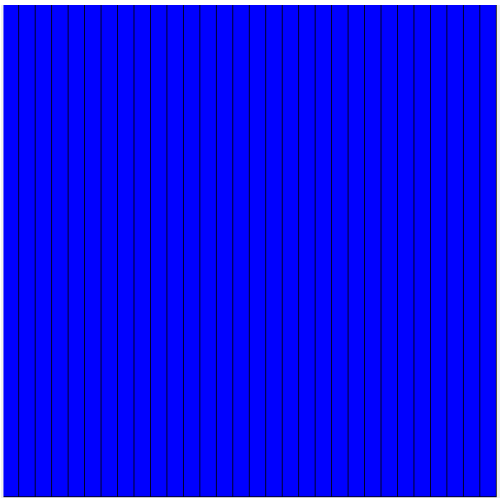
Nimic mai simplu. Avem 8 coloane de desenat pe care le numerotăm cu numerele 0, 1, 2 ... până la 7. Pe fiecare coloană avem 8 pătrate plasate pe 8 linii pe care le numerotăm de asemenea cu

numere de la 0 la 7. Să ne gândim la pătratul din coloana i = 4 și linia j = 3. Acest pătrat are în stanga lui alte 4 pătrate deci coordonata X a vârfului lui va fi 40 + lungime * 4 sau 40 + lungime * i. Deasupra lui sunt alte 3 patrate deci coordonata Y a varfului va fi 40+latime*3 sau 40+latime*j și am obținut formula noastră.

Aplicația 4 – Cortina care se închide

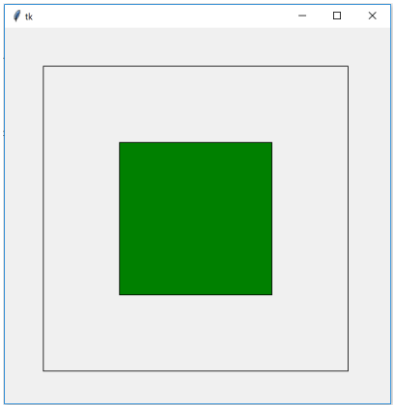
Ema are nevoie pentru clubul de teatru de o cortină. În aplicația de mai jos avem o cortină care se închide. Vă lăsăm să analizați singuri programul.

```
from tkinter import *
import time
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
for i in range(0,300,20):
    plansa.create_rectangle(i,0,i+20,600,fill='blue')
    time.sleep(0.05)
    plansa.update()
    plansa.create_rectangle(600-i-20,0,600-i,600,fill='blue')
    time.sleep(0.05)
    plansa.update()
```



Între timp, Coco și Ema își propun să deseneze un model ceva mai complicat.

- Ce ar fi să desenăm un pătrat în interiorul unui pătrat și apoi alte pătrate mai mici în interior? spune Ema
- Cred că trebuie să începi să desenezi ca să îmi arăți.
- Sigur! Voi începe prin a desena un pătrat în interiorul altui pătrat.



```
from tkinter import *

fereastra=Tk()
```

```
plansa=Canvas(fereastra,width=640, height=480)
plansa.pack()

def deseneaza_patrat(colt_stanga_x, colt_stanga_y, latura, culoare = None):
    plansa.create_rectangle(colt_stanga_x, colt_stanga_y, colt_stanga_x+latura,colt_stanga_y+latura,
fill=culoare)

def deseneaza_patrat_in_patrat(colt_stanga_x, colt_stanga_y, latura):
    deseneaza_patrat(colt_stanga_x, colt_stanga_y, latura)
    patrat_interior_colt_x = colt_stanga_x + latura // 4
    patrat_interior_colt_y = colt_stanga_y + latura // 4
    patrat_interior_latura = latura // 2
    deseneaza_patrat(patrat_interior_colt_x, patrat_interior_colt_y, patrat_interior_latura, "green")

deseneaza_patrat_in_patrat(50, 50, 400)
```

- Cred că am înțeles. Mai întâi ai construit o funcție deseneaza_patrat care primește ca argumente coordonatele colțului stânga sus al pătratului și lungimea laturii. Funcția are și un argument culoare care ne permite să alegem culoarea în care va fi desenat interiorul pătratului. Dar ce înseamnă cuvântul None folosit la acest parametru?

None arată că acest parametru este opțional. Dacă nu specificăm nici o culoare atunci pătratul va fi desenat gol. Folosim acest lucru în cadrul funcției deseneaza_patrat_in_patrat. Patratul exterior este desenat gol cu ajutorul apelului:

```
deseneaza_patrat(colt_stanga_x, colt_stanga_y, latura)
```

Mai apoi calculăm coordonatele colțului stânga sus ale pătratului interior.

```
patrat_interior_colt_x = colt_stanga_x + latura // 4
patrat_interior_colt_y = colt_stanga_y + latura // 4
```

și dimensiunea laturii pătratului interior ca fiind jumatate din latura pătratului inițial

```
patrat_interior_latura = latura // 2
```

după care desenăm pătratul interior colorat cu verde

```
deseneaza_patrat(patrat_interior_colt_x, patrat_interior_colt_y, patrat_interior_latura, "green")
```

Acum cred că stiu ce vrei să faci. Putem împărți pătratul inițial în patru pătrate și să repetăm procedeul pentru fiecare dintre ele.

2.6 Ștergerea unei figuri geometrice de pe planșă

Ema ar vrea pentru clubul ei de teatru o cortină care se deschide. Pentru realizarea cortinei va desena dreptunghiuri și le va șterge pe rând, pornind de la interior spre exterior. Ștergerea unei figuri geometrice este realizată de funcția delete care are sintaxa delete(id) unde id este id-ul figuri care se va sterge.

Aplicație – Cortina care se deschide

```
from tkinter import *
import time
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
dreptunghi_id=list()
for i in range(0,300,20):
    id1=plansa.create_rectangle(i,0,i+20,600,fill='blue')
    dreptunghi_id.append(id1)
    id2=plansa.create_rectangle(600-i-20,0,600-i,600,fill='blue')
    dreptunghi_id.append(id2)
for i in range(len(dreptunghi_id)-1,-1,-1):
    plansa.delete(dreptunghi_id[i])
    del dreptunghi_id[i]
    time.sleep(0.05)
    plansa.update()
```

Retinem id-urile dreptunghiurilor într-o listă.

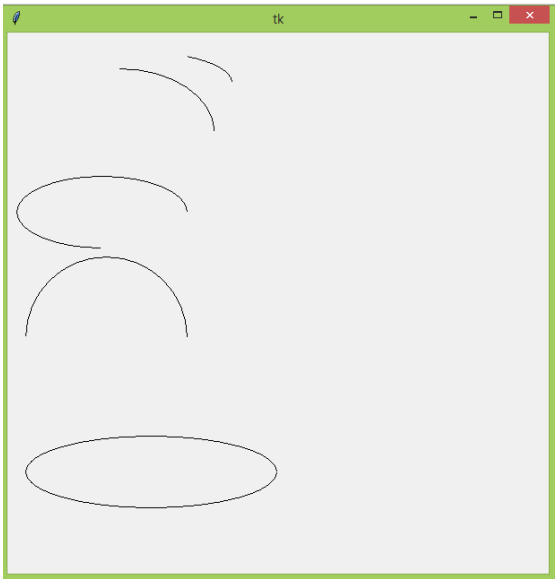
Parcurgem lista de la dreapta la stânga si ștergem obiectele în ordinea inversă creării.

2.7 Desenarea arcelor

Aplicație

Coco ar avea nevoie să deseneze o elipsă și face diferite încercări. Analizați singuri codul de mai jos.

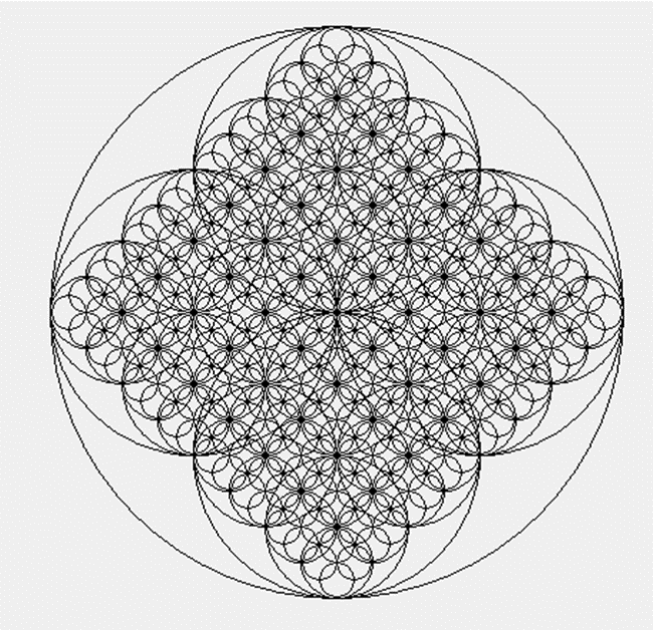
```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
plansa.create_arc(20,20,250,90,extent=55,style=ARC)
plansa.create_arc(20,40,230,180,extent=90,style=ARC)
plansa.create_arc(10,160,200,240,extent=270,style=ARC)
plansa.create_arc(20,250,200,430,extent=180,style=ARC)
plansa.create_arc(20,450,300,530,extent=359,style=ARC)
```



2.8 Desenarea unui oval

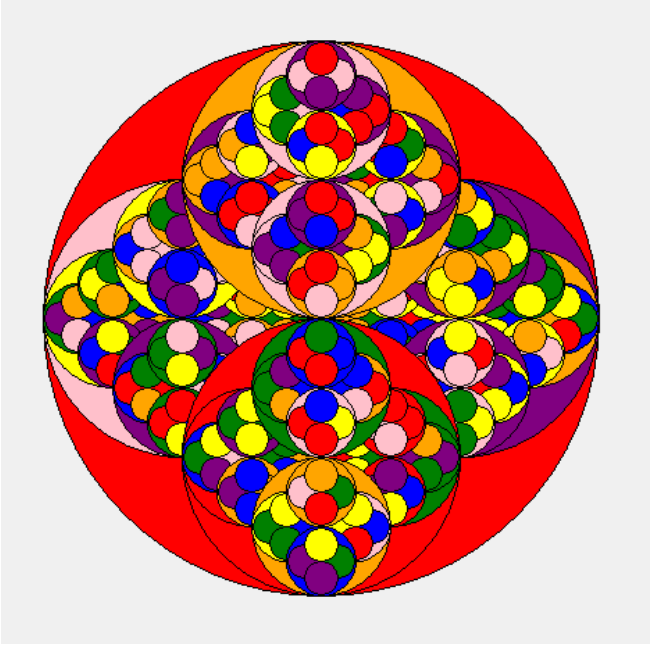
Coco ar vrea sa poată să deseneze niște cercuri, un model deosebit. Credeți că a reușit?

```
from tkinter import *
import time
fereastra=Tk()
plansa=Canvas(fereastra, width=600, height=600)
plansa.pack()
def cerc(x,y,raza):
   if raza>10:
         plansa.create_oval(x-raza,y-raza,x+raza,y+raza)
         time.sleep(0.1)
         fereastra.update()
         cerc(x-raza//2,y,raza/2)
         cerc(x+raza//2,y,raza//2)
         cerc(x,y+raza//2,raza//2)
         cerc(x,y-raza//2,raza//2)

x=300
y=300
raza=200
cerc(x,y,raza)
```



Puțină culoare – modificăm programul anterior astfel încât să fie cercurile colorate

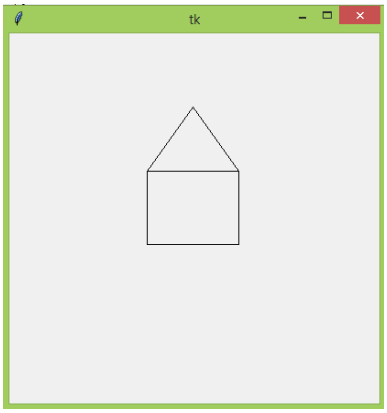
```
from tkinter import *
from random import *
import time
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
def cerc(x,y,raza):
    if raza>10:
        culoare=choice(['red','green','blue','yellow','pink','orange','purple'])
        plansa.create_oval(x-raza,y-raza,x+raza,y+raza,fill=culoare)
        time.sleep(0.1)
        fereastra.update()
        cerc(x-raza//2,y,raza//2)
        cerc(x+raza//2,y,raza//2)
        cerc(x,y+raza//2,raza//2)
        cerc(x,y-raza//2,raza//2)

x=300
y=300
raza=200
cerc(x,y,raza)
```



2.9 Desenarea poligoanelor

Coco și Ema ar dori să deseneze o casă care să le amintească de casa lor de pe Pământ.

Credeți că au reușit?



Soluție:

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=400, height=400)
plansa.pack()
plansa.create_rectangle(150,150,250,230)
plansa.create_polygon(150,150,200,80,250,150,fill="",outline="black")
```

2.10 Afișarea textelor

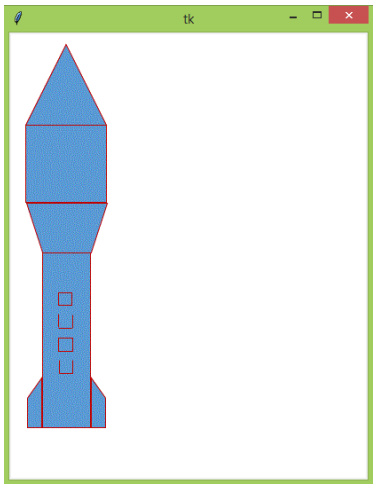
Coco și Ema doresc un mesaj de bun venit pentru musafirii lor. Care variantă vi se pare cea mai reușită.

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600,height=600)
plansa.pack()
plansa.create_text(280,100,text='Bine ati venit pe PLANETA PROGRAMATORILOR!')
plansa.create_text(280,130,text='Bine ati venit pe PLANETA PROGRAMATORILOR!',fill='red')
plansa.create_text(280,160,text='Bine ati venit pe PLANETA PROGRAMATORILOR!',fill='blue',font=('Times',12))
plansa.create_text(280,190,text='Bine ati venit pe PLANETA PROGRAMATORILOR!',fill='green',font=('Helvetica',14))
plansa.create_text(280,220,text='Bine ati venit pe PLANETA PROGRAMATORILOR!',fill='orange',font=('Courier',15))
```

2.11 Inserarea imaginilor

Aplicația 1

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra, width=400, height=500,bg='white')
plansa.pack()
imagine1=PhotoImage(file='c:\\doina\\racheta.gif')
plansa.create_image(0,0,anchor=NW,image=imagine1)
```

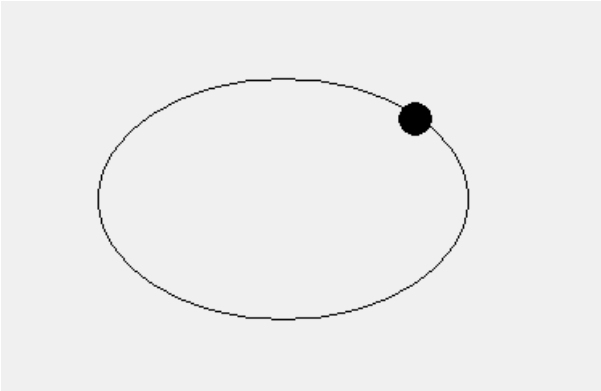


2.12 Crearea unei animații – mișcarea unei planete pe orbită

Coco începe să se joace, dorește o primă animație. Pentru aceasta el folosește funcția move.

Funcția move(id,x,y) mișcă obiectul care are identificatorul id cu x pixeli spre dreapta și y pixeli în jos.

```
from tkinter import *
import time
fereastra=Tk()
plansa=Canvas(fereastra, width=600, height=600)
plansa.pack()
plansa.create_oval(150,150,380,300)
plansa.create_oval(200,150,220,170,fill='black')
for x in range(10):
   plansa.move(2,5,-1)
   fereastra.update()
   time.sleep(0.05)
for x in range(10):
   plansa.move(2,5,1)
   fereastra.update()
   time.sleep(0.05)
for x in range(14):
   plansa.move(2,3,1)
   fereastra.update()
   time.sleep(0.05)
```



Încercați singuri

Continuați mișcarea planetei pe orbită astfel încât să acopere toată traiectoria.

2.13 Reacția la evenimente

Când este apăsată o tastă sau este mișcat mouse-ul spunem că a apărut un “eveniment”. Calculatoarele știu să detecteze evenimentul care a apărut și pot fi programate să reacționeze adecvat.

Pentru a identifica un eveniment, acesta are un nume în biblioteca Tkinter.

Evenimente declanșate de mouse	
<Button-1>	Clic pe butonul stâng al mouse-ului
<Button-3>	Clic pe butonul drept al mouse-ului
<B1-Motion>	Obiectul este mutat, cu butonul stâng al mouse-ului apăsat (utilizați B2 pentru butonul din mijloc, B3 pentru butonul drept). Poziția curentă a indicelui mouse-ului este furnizată în membrii x și y ai obiectului eveniment trecut la apel.
Evenimente declanșate de tastatură	
<Right>	Este apăsată tasta săgeată dreapta
<Left>	Este apăsată tasta săgeată stânga
<Up>	Este apăsată tasta săgeată în sus
<Down>	Este apăsată tasta săgeată în jos
<space>	Este apăsată bara de spațiu
<KeyPress-x>	Este apăsată tasta X

Aplicația 1 Planeta pe orbită – folosind tastele săgeți, mișcați planeta pe orbită

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra, width=600, height=600)
plansa.pack()
plansa.create_oval(150,150,380,300)
plansa.create_oval(200,150,220,170,fill='black')
def misca_planeta(eveniment):
    if eveniment.keysym=='Up':
        plansa.move(2,0,-2)
    elif eveniment.keysym=='Down':
        plansa.move(2,0,2)
    elif eveniment.keysym=='Left':
        plansa.move(2,-2,0)
    else:
        plansa.move(2,2,0)
plansa.bind_all('<KeyPress-Up>', misca_planeta)
plansa.bind_all('<KeyPress-Down>', misca_planeta)
plansa.bind_all('<KeyPress-Left>', misca_planeta)
plansa.bind_all('<KeyPress-Right>', misca_planeta)
```

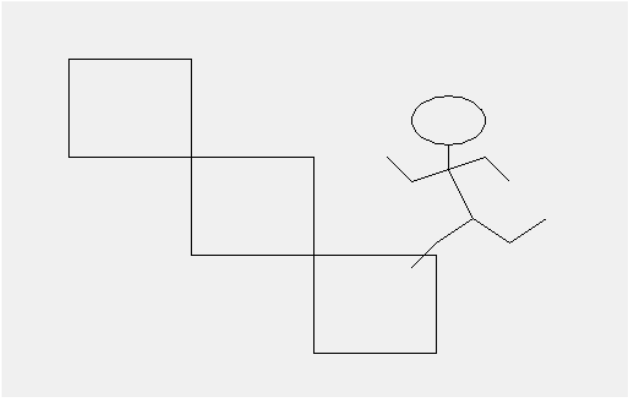
Planeta se mișcă în sus

Leagă tasta săgeată în sus de funcția misca_planeta

Aplicația 2 Schimbarea culorii planetei – folosind tastele p și n schimbați culoarea în portocaliu sau negru

```
from tkinter import *
import time
fereastra=Tk()
plansa=Canvas(fereastra, width=600, height=600)
plansa.pack()
traiectoria=plansa.create_oval(150,150,380,300)
planeta=plansa.create_oval(200,150,220,170,fill='black')

def planeta_neagra(event):
    plansa.itemconfig(planeta,fill='black')

def planeta_portocalie(event):
    plansa.itemconfig(planeta,fill='orange')

plansa.bind_all('<KeyPress-n>', planeta_neagra)
plansa.bind_all('<KeyPress-p>', planeta_portocalie)
```

Aplicația 3 - Folosind tastele săgeți mișcați omulețul astfel încât să urce scările.



Soluție:

```
from tkinter import *
fereastra=Tk()
plansa=Canvas(fereastra,width=600, height=600)
plansa.pack()
lungime=100
latime=80
for i in range(1,4,1):
    plansa.create_rectangle(40+lungime*i,40+latime*i,40+lungime*(i+1),40+latime*(i+1))
cap=plansa.create_oval(420,150,480,190)
gat=plansa.create_line(450,190,450,210)
maini=plansa.create_line(400,200,420,220,480,200,500,220)
corp=plansa.create_line(450,210,470,250)
picioare=plansa.create_line(420,290,440,270,470,250,500,270,530,250)
distanta=5
def miscare_omulet(event):
    if event.keysym=='Up':
        plansa.move(cap,0,-distanta)
        plansa.move(gat,0,-distanta)
        plansa.move(maini,0,-distanta)
        plansa.move(corp,0,-distanta)
        plansa.move(picioare,0,-distanta)
    elif event.keysym=='Down':
        plansa.move(cap,0,distanta)
        plansa.move(gat,0,distanta)
        plansa.move(maini,0,distanta)
        plansa.move(corp,0,distanta)
        plansa.move(picioare,0,distanta)
    elif event.keysym=='Left':
        plansa.move(cap,-distanta,0)
        plansa.move(gat,-distanta,0)
        plansa.move(maini,-distanta,0)
        plansa.move(corp,-distanta,0)
        plansa.move(picioare,-distanta,0)
    elif event.keysym=='Right':
        plansa.move(cap,distanta,0)
        plansa.move(gat,distanta,0)
        plansa.move(maini,distanta,0)
        plansa.move(corp,distanta,0)
        plansa.move(picioare,distanta,0)
plansa.bind_all('<Key>',miscare_omulet)
```