Python logo

Programiranje 2

2. Želvja grafika

Vladimir Batagelj
Univerza v Ljubljani
FMF, matematika

različica: 4. marec 2014 / 03 : 16

Kazalo

Želvja grafika

Python uporablja za slikovni vmesnik sestav Tk, ki je dostopen s knjižnico tkinter. Ta vsebuje tudi podporo risanja. Ker je uporaba te knjižnice razmeroma zapletena, so za preprosta risanja dodali prijaznejšo knjižnico turtle. Leta 2006 je Gregor Lingl predstavil izpopolnjeno knjižnico xturtle, ki je osnova za izpopolnjeni `turtle` v Python 3.

Želvja grafika je ena izmed glavnih sestavin programskega jezika Logo, ki so ga razvili na MIT v drugi polovici šestdesetih let (Papert, Fuerzig). Logo se uporablja predvsem za zgodnje uvajanje otrok v programiranje. O želvji grafiki sta Abelson in diSessa napisala knjigo. Želvja grafika temelji na sledi, ki jo pri premikanju po ravnini za sabo pušča želvica.

Želvja grafika in Python 3

Pri interaktivnem delu z želvjo grafiko so (bile) v Python 3 težave. Rešitev je naslednja (povzeta po bytes):

V `c:\Python31` kliknemo z desno tipko na `pythonw` in izberemo možnost `Create Shortcut`. Sistem ustvari bližnjico `pythonw - Shortcut`. Preimenujemo jo (desni klik, `Rename`) v npr. `IDLE (turtle)`. Nato desno kliknemo na to datoteko in izberemo `Properties`. Polje `Target` spremenimo tako, da vsebuje

```
C:\Python31\pythonw.exe "C:\Python31\Lib\idlelib\idle.pyw" -n
```

Pri zagonu `IDLE (turtle)` se sedaj izpiše

```
  IDLE 3.1      ==== No Subprocess ====
```

Če želimo, da se bližnjica pojavi med drugimi možnosti za Python v seznamu programov, jo prestavimo (`move`) v področje (Vista)

```
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Python 3.1
```

oziroma (XP)

```
C:/Documents and settings/All Users/Start Menu/Programs/Python 3.1
```

Izbor ukazov – premiki in risanje

`forward(`d`)`	naprej za d
`fd(`d`)`	
`backward(`d`)`	nazaj za d
`bk(`d`)`	tudi `back(`d`)`
`right(`a`)`	desno za kot a
`rt(`a`)`	`"logo"`: 0 – S, 90 – V; `"standard"`: 0 – V, 90 – S
`left(`a`)`	levo za kot a
`lt(`a`)`	
`setposition(`xy`)`	premik na točko xy
`setpos(`xy`)`	tudi `goto(`xy`)`
`setx(`x`)`	sprememba koordinate x
`sety(`y`)`	sprememba koordinate y
`setheading(`a`)`	usmeri želvo v smeri (kot) a
`seth(`a`)`	
`home()`	v koordinatno izhodišče
`circle(`$r, \ldots$`)`	krog s polmerom r
`dot(`d, c`)`	pika premera d barve c
$i =$`stamp()`	odtis
`clearstamp(`i`)`	
`clearstamps(`n`)`	
`undo()`	prekliči zadnji želvji ukaz
`=speed(`s`)`	spremeni hitrost želvice 0 - 10, 1-počasi 10-najhitreje, 0-kar se da hitro

Izbor ukazov – stanje in koti

$xy =$`position()`	koordinate želvice
`pos()`	
$a =$`towards(`xy`)`	kot želvice proti točki
$x =$`xcor()`	koordinata x
$y =$`ycor()`	koordinata y
$a =$`heading()`	smer želvice
$d =$`distance(`xy`)`	oddaljenost želvice do točke
`degrees()`	koti se merijo v stopinjah
`radians()`	koti se merijo v radijanih
`showturtle()`	želvica postane vidna
`st()`	
`hideturtle()`	želvica se skrije
`ht()`	
$p =$`isvisible()`	ali je želvica vidna?
`shape(`N`)`	prikaz želve N: `"turtle"`, `"classic"`, `"arrow"`, `"circle"`, `"square"`, `"triangle"`, uporabniške
`resizemode(`m`)`	m: `"auto"`, `"user"`, `"noresize"`
`shapesize(`w, l, o`)`	raztega in debelina obrisa
`turtlesize()`	
`tilt(`a`)`	nagib – zasuk slike želve
`settiltangle(`a`)`	
$a =$`tiltangle()`	

Izbor ukazov – pero

`pendown()`	spusti pero
`pd()`	tudi `down()`
`penup()`	dvigni pero
`pu()`	tudi `up()`
`pensize(`d`)`	velikost peresa; tudi `width()`
t =`pen()`	vse lastnosti peresa
`isdown()`	ali je pero spuščeno
`color(`c_1, c_2`)`	$\equiv$ (`pencolor(`c_1`)`, `fillcolor(`c_2`)`)
=`pencolor(`c`)`	barva peresa
=`fillcolor(`c`)`	barva zapolnjevanja
`filling()`	zapolnjevanje ?
`begin_fill()`	začetek zapolnjevanja (barvanja notranjosti) lika
`end_fill()`	konec zapolnjevanja lika
`reset()`	počisti zaslon in začni nanovo
`clear()`	zbriši sledi
`write(`z`,...)`	napiše niz znakov z; `move= True/False`, `align= "left"/"center"/"right"`, `font=` (ime,velikost,oblika)

Izbor ukazov – dogodki

`onclick()`	
`onrelease()`	
`ondrag()`	
`begin_poly()`	začetek beleženja mnogokotnika
`end_poly()`	konec beleženja mnogokotnika
M =`get_poly()`	zadnji zabeleženi mnogokotnik
z =`Turtle()`	ustvari novo želvo z
z =`getturtle()`	vrne brezimno želvo
`getpen()`	
`clone(z)`	ustvari dvojnika želve z
`getscreen()`	
`setundobuffer()`	
`undobufferentries()`	
`delay()`	
`tracer()`	
`update()`	
`listen()`	
`onkey()`	
`onscreenclick()`	
`onclick()`	
`ontimer()`	

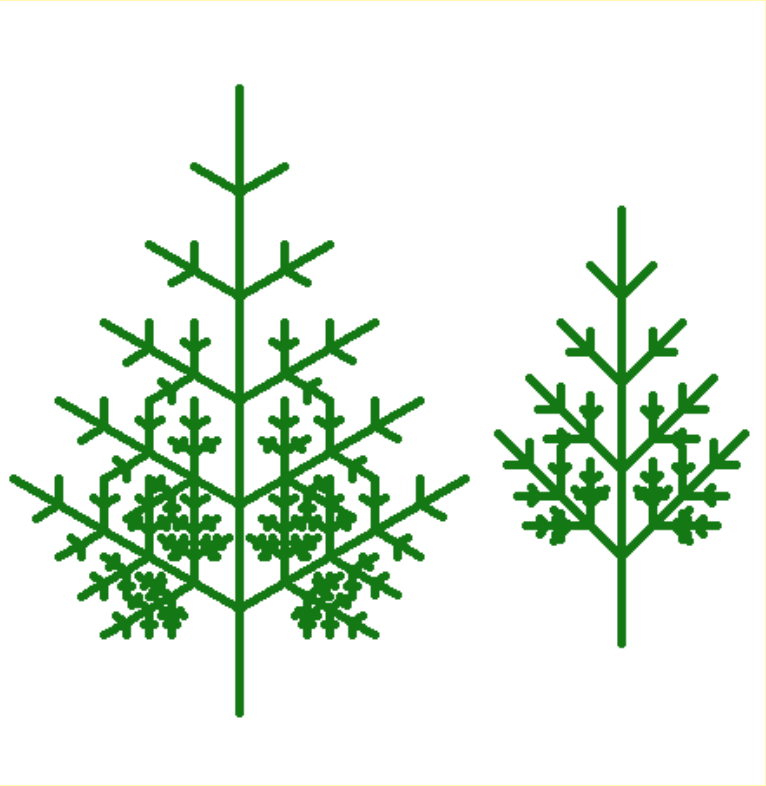
Izbor ukazov – okno

`bgcolor(`c`)`	barva ozadja
`bgpic(`P`)`	slika v ozadju
`clearscreen()`	pobriši zaslon
`clear()`	
`resetscreen()`	začni nanovo
`reset()`	
`screensize(`w, h, c`)`	želvje okno naj bo široko w, visoko h in barve c
`setworldcoordinates(`P`)`	svet: $P = (llx, lly, urx, ury)$
`mode(`m`)`	m: `"logo"`, `"standard"`
`colormode(`m`)`	način zapisa barv RGB: 1.0 ali 255
`getcanvas()`	
s =`getshapes()`	seznam želvjih oblik
`register_shape(`N, M`)`	dodaj novo obliko želve M z imenom N
`addshape()`	ali sličico v zapisu GIF
s =`turtles()`	seznam želv
d =`window_height()`	višina okna
d =`window_width()`	širina okna
`bye()`	končaj z risanjem, zapri okno
`exitonclick()`	ob kliku na okno ga zapri
`setup()`	
`title(`N`)`	N postavi za naslov okna

Primeri

```
>>> from turtle import *
>>> reset()
>>> shape()
'classic'
>>> shape("turtle")
>>> fd(100); rt(90)
>>> ena = getturtle(); ena.fd(50)
>>> dva = Turtle(); dva.rt(120)
>>> dva.pencolor("blue"); dva.shape("circle")
>>> dva.fd(100)
>>> bye()
```


Primer – Smreka

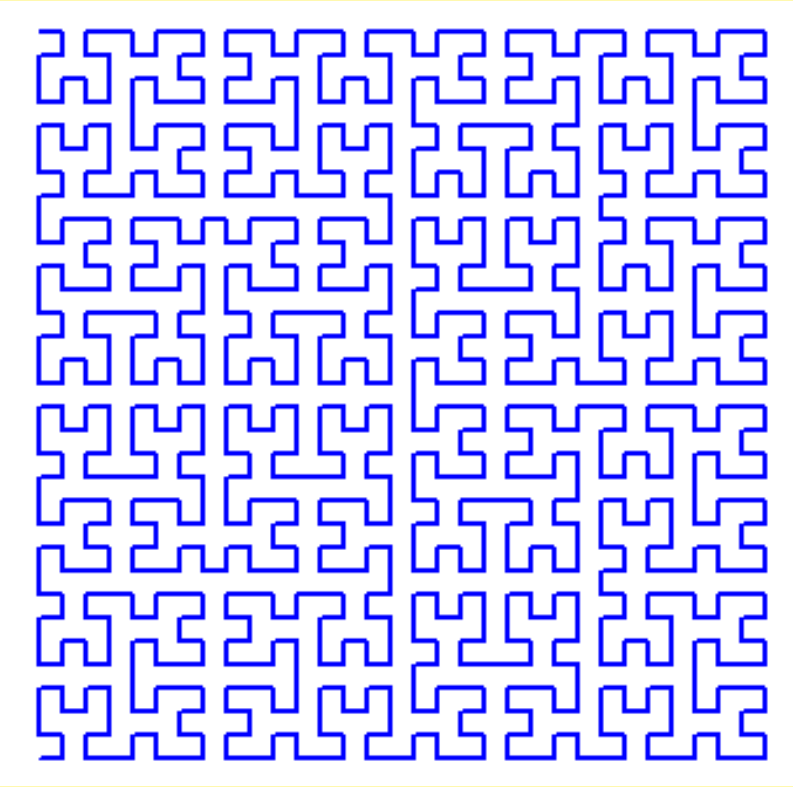


datoteka `smreka.py`

```
from turtle import *

def smreka(d,a,n):
  if n > 0:
    fd(d); rt(a); smreka(d/2,a,n-1)
    lt(a); smreka(d,a,n-1)
    lt(a); smreka(d/2,a,n-1)
    rt(a); pu(); bk(d); pd()

def NovoLeto():
  reset(); speed(0); ht(); seth(90)
  colormode(255);
  pencolor((20,120,20)); pensize(5)
  pu(); setpos(-80,-190)
  pd(); smreka(60,60,6)
  pu(); setpos(140,-150)
  pd(); smreka(50,45,5)
  exitonclick()

NovoLeto()
```

```
>>> import sys
>>> wdir = r'c:\users\Batagelj\test\Python\turtle'
>>> sys.path.append(wdir)
>>> import smreka
```

Primer – Hilbertova krivulja

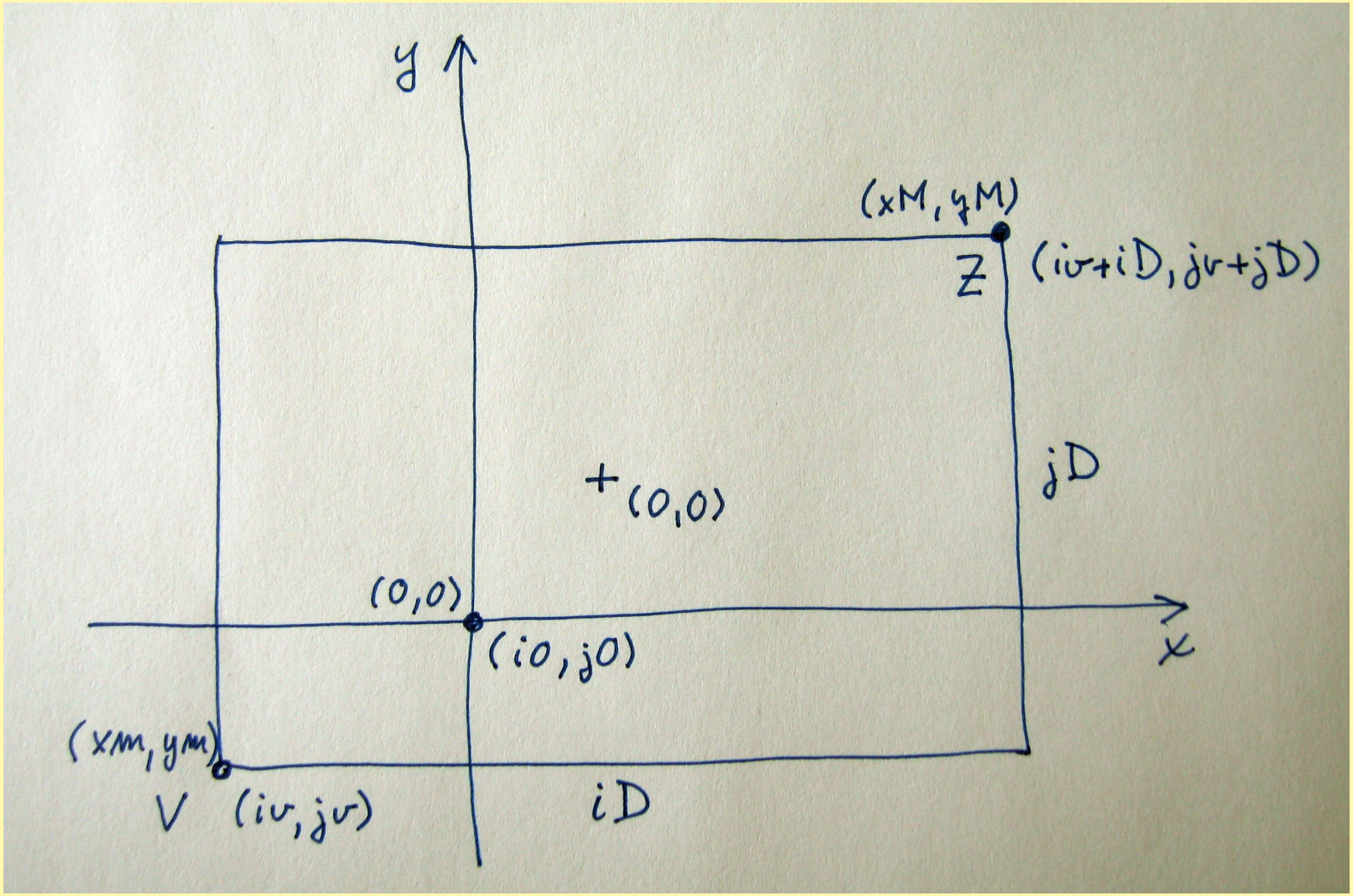


datoteka `Hilbert.py`

```
from turtle import *

def Hilbert(n,a,h):
  if n==0: return
  rt(a); Hilbert(n-1,-a,h); fd(h)
  lt(a); Hilbert(n-1,a,h); fd(h)
  Hilbert(n-1,a,h); lt(a); fd(h)
  Hilbert(n-1,-a,h); rt(a)

def RisHi():
  reset(); speed(0); ht()
  pu(); setpos(-160,-160); pd()
  seth(90); pensize(2)
  colormode(255); pencolor("blue")
  Hilbert(5,90,10)
  exitonclick()

RisHi()
```

```
>>> import sys
>>> wdir = r'c:\users\Batagelj\test\Python\turtle'
>>> sys.path.append(wdir)
>>> import Hilbert
```

Primer – Risanje funkcije



Primer – Risanje funkcije

Koordinate (x, y) ravnine $\mathbb{R}^2$ in zaslonske koordinate (i, j) so med seboj linearno povezane: $i = ax + b$ in $j = Ay + B$.

Določimo a in b; za A in B gre enako. Če vstavimo v $i = ax + b$ točki V in Z, dobimo $iv = a \cdot xm + b$ in $iv + iD = a \cdot xM + b$.

Rešitvi a in b tega sistema enačb sta

$$a = \frac{iD}{xM - xm} \quad \text{in} \quad b = iv - a \cdot xm = iv - \frac{iD \cdot xm}{xM - xm}$$

oziroma končno, če označimo a z ei (enota na i)

$$i = ei \cdot (x - xm) + iv \quad \text{in} \quad x = \frac{1}{ei}(i - iv) + xm$$

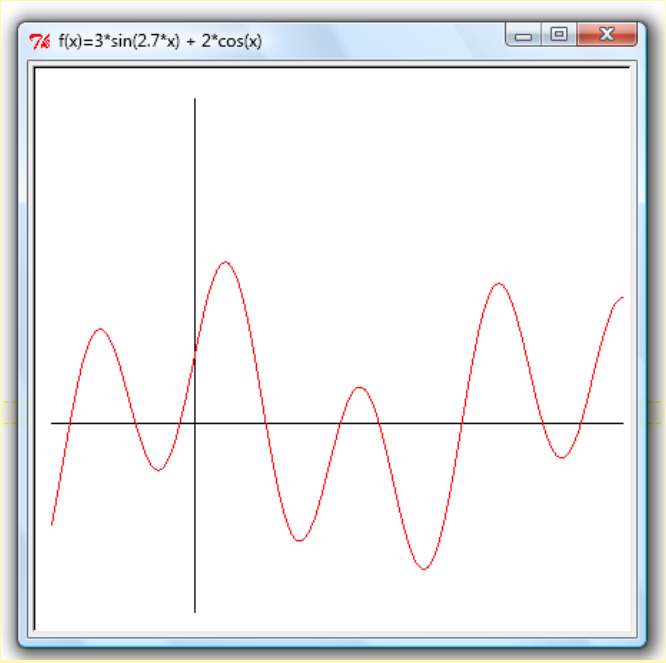
Podobno velja:

$$ej = \frac{jD}{yM - ym}, \quad j = ej \cdot (y - ym) + jv \quad \text{in} \quad y = \frac{1}{ej}(j - jv) + ym$$

Za zaslonski koordinati $(i0, j0)$ koordinatnega izhodišča dobimo

$$i0 = iv - ei \cdot xm \quad \text{in} \quad j0 = jv - ej \cdot ym$$

Primer – risanje funkcij



datoteka `funkcija.py`

```
from turtle import *
from math import *

mode("logo")
f = "3*sin(2.7*x) + 2*cos(x)"
iv = -200; jv = -180; iD = 400; jD = 360
xm = -2.5; xM = 7.5; ym = -5.5; yM = 9.5
ei = iD/(xM-xm); ej = jD/(yM-ym)
i0 = iv-ei*xm; j0 = jv-ej*ym

reset(); title("f(x)="+f)
ht(); pencolor("black")
pu(); setpos(iv,j0); seth(90); pd(); fd(iD)
pu(); setpos(i0,jv); seth( 0); pd(); fd(jD)
x = xm; j = round((eval(f)-ym)*ej)+jv
pu(); setpos(iv,j); pd(); pencolor("red")
for i in range(1,iD+1):
  x = i/ei + xm
  j = round((eval(f)-ym)*ej)+jv
  setpos(i+iv,j)
exitonclick()
```

```
>>> import sys
>>> wdir = r'c:\users\Batagelj\test\Python\turtle'
>>> sys.path.append(wdir)
>>> import funkcija
```