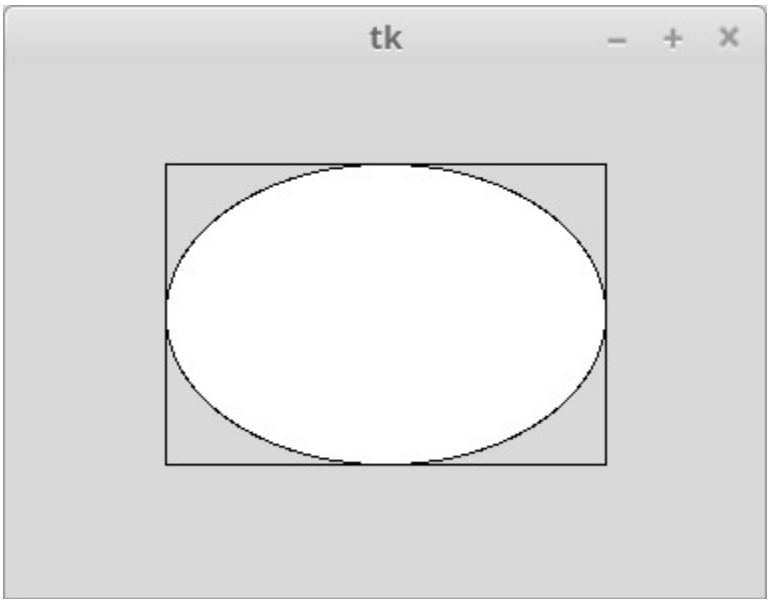
Zdroj: Andrej Blaho, http://python.input.sk

Kreslenie elíps

Keď už vieme kresliť obdĺžniky, potom prechod k elipsám je pre `tkinter` veľmi jednoduchý: napíšeme program s kreslením obdĺžnikov a potom namiesto `create_rectangle` zapíšeme `create_oval` - namiesto obdĺžnika sa presne na tom istom mieste nakreslí (vpísaná) elipsa - vlastne sa „zaoblia“ rohy obdĺžnika. Ukážme to na príklade, v ktorom nakreslíme obdĺžnik a potom na tom istom mieste (s tými istými súradnicami) nakreslíme elipsu:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

canvas.create_rectangle(80, 50, 300, 200)
canvas.create_oval(80, 50, 300, 200, fill='white')
```

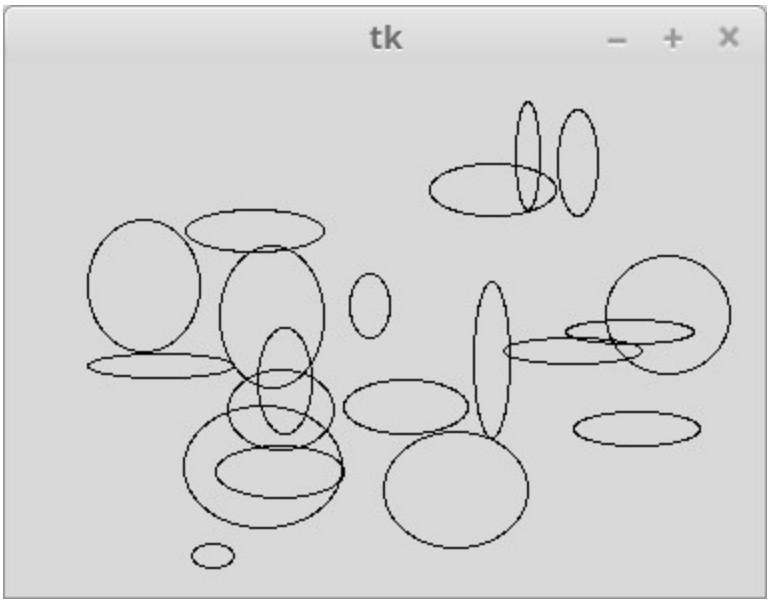
Nakreslenú elipsu sme zafarbili na bielo, aby ju bolo lepšie vidieť:



Otestujme kreslenie elíps na programe, v ktorom sme kreslili náhodne veľké obdĺžniky na náhodné pozície. Namiesto `create_rectangle` zapíšeme `create_oval`:

```
import tkinter
import random

canvas = tkinter.Canvas()
canvas.pack()

for i in range(20):
    sirka = random.randint(10, 80)
    vyska = random.randint(10, 80)
    x = random.randint(10, 370 - sirka)
    y = random.randint(10, 260 - vyska)
    canvas.create_oval(x, y, x + sirka, y + vyska)
```

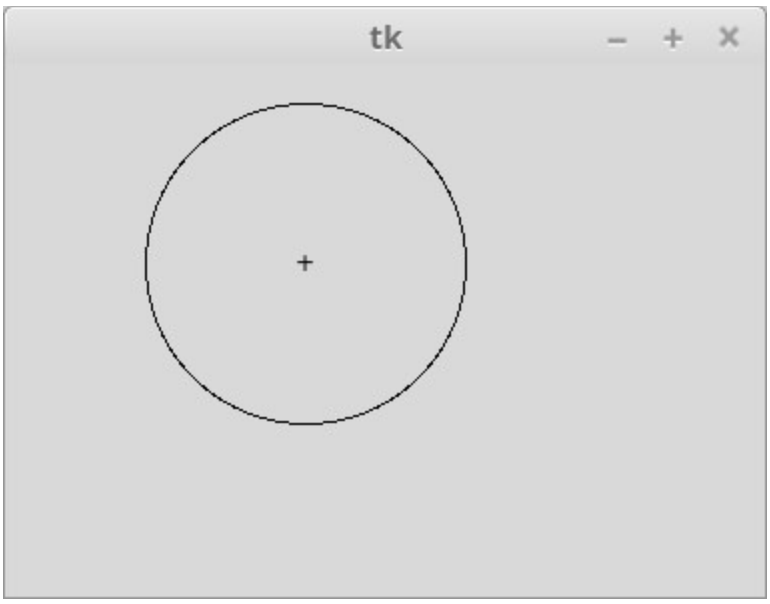
Program nakreslí 20 rôzne veľkých elíps:



Zrejme ste si uvedomili, že elipsa, ktorá vznikne zo štvorca (má rovnakú šírku a výšku), je **kružnica**. Hoci kružnice sa často kreslia tak, že zadáme súradnice stredu `(x, y)` a polomer `r`. Potom sa kružnica kreslí takto jednoducho:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

x, y = 150, 100
r = 80
canvas.create_oval(x-r, y-r, x+r, y+r)

canvas.create_text(x, y, text='+')
```

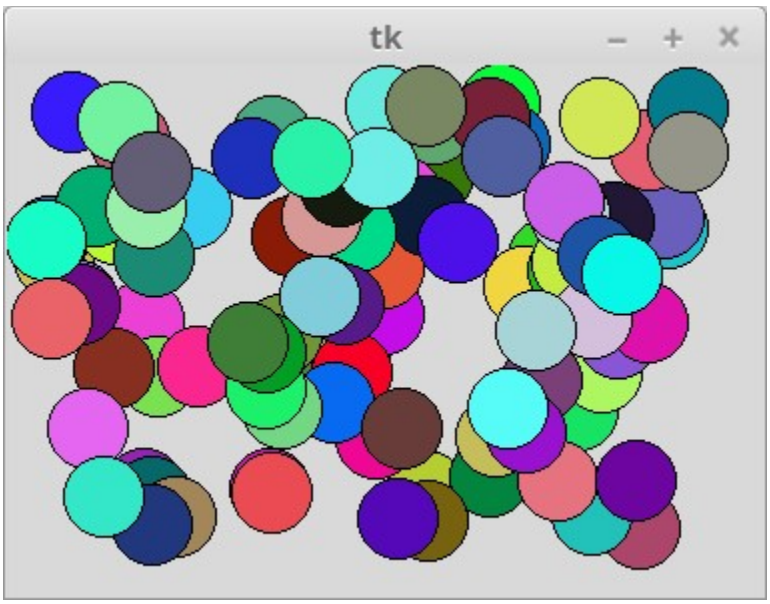
Na pozíciu stredu sme dokreslili krížik:



V ďalšom programe na náhodné pozície nakreslíme `100` rôznofarebných kruhov s polomerom `r=20`:

```
import tkinter
import random

canvas = tkinter.Canvas()
canvas.pack()

for i in range(100):
    x = random.randint(20, 350)
    y = random.randint(20, 240)
    r = 20
    fill = f'#{random.randrange(256**3):06x}'
    canvas.create_oval(x-r, y-r, x+r, y+r, fill=fill)
```

Všimnite si premennú `fill`, do ktorej priradíme náhodnú farbu. Vďaka tomu pri volaní príkazu `create_oval` zapisujeme `fill=fill` aby sme pomenovanému parametru `fill` priradili hodnotu premennej `fill`. Po spustení dostaneme:



Kreslenie úsečiek a lomených čiar

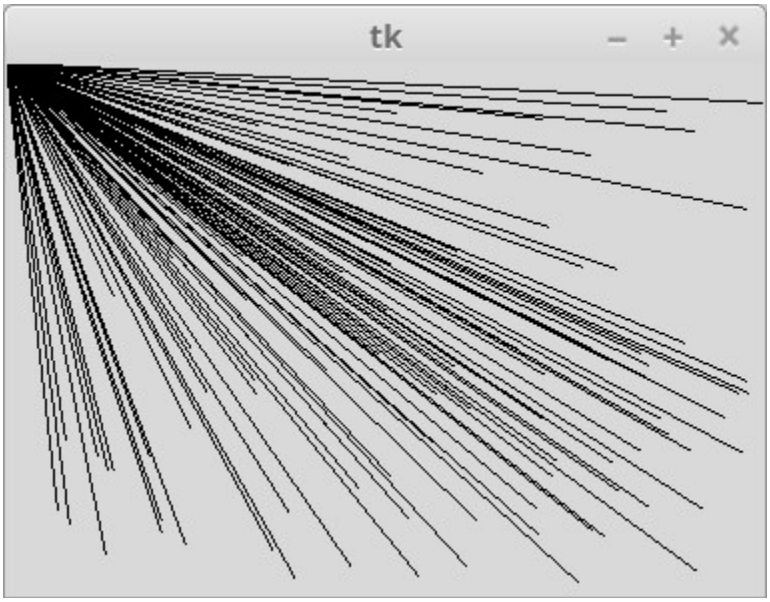
Ďalším grafickým príkazom kreslíme lomené čiary, t.j. čiary, ktoré sa skladajú z nadväzujúcich úsečiek. Tvar príkazu je:

```
canvas.create_line(x1, y1, x2, y2, x3, y3, ...)
```

Parametrom je postupnosť súradníc, ktoré tvoria lomenú čiaru. Táto postupnosť musí obsahovať aspoň 2 body - vtedy sa nakreslí jedna úsečka. Napíšme program, ktorý nakreslí `100` úsečiek, ktoré majú prvý bod v `(0, 0)` a druhý je náhodný v ploche:

```
import tkinter
import random

canvas = tkinter.Canvas()
canvas.pack()

for i in range(100):
    x = random.randint(20, 380)
    y = random.randint(20, 260)
    canvas.create_line(0, 0, x, y)
```

po spustení:



Doteraz sme používali takéto paralelné priradenie:

```
x, y = 100, 150
```

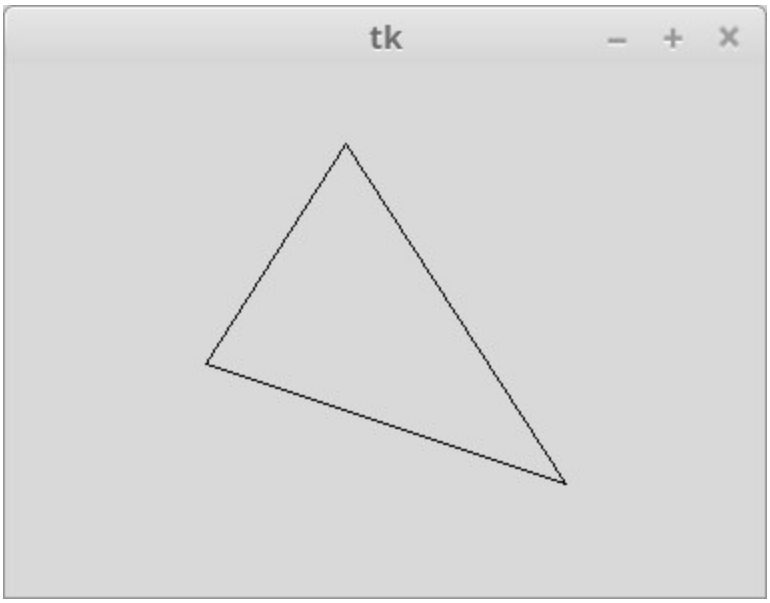
Do dvoch premenných `x` a `y` sa priradia nejaké dve hodnoty. Lenže Python ponúka ešte aj takýto variant priradenia:

```
a = 100, 150
```

V tomto prípade sa do premennej `a` priradí **dvojica** celých čísel. Takéto dvojice môžeme potom použiť ako parametre do grafických príkazov. Napríklad zadefinujeme tri dvojice čísel, teda súradnice troch bodov `a`, `b`, `c` a potom nakreslíme trojuholník pomocou troch úsečiek:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

a = 100, 150
b = 280, 210
c = 170, 40

canvas.create_line(a, b)
canvas.create_line(b, c)
canvas.create_line(c, a)
```

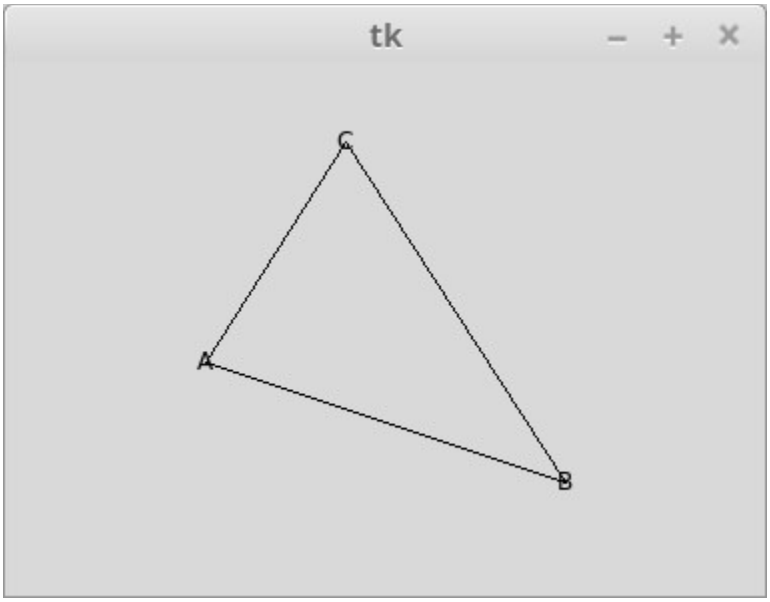
a vyzerá to takto:



Tri volania `create_line` môže spojiť do jedného a ešte k tomu pridáme aj pomenovanie týchto vrcholov reťazcami `'A'`, `'B'`, `'C'`:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

a = 100, 150
b = 280, 210
c = 170, 40

canvas.create_line(a, b, c, a)

canvas.create_text(a, text='A')
canvas.create_text(b, text='B')
canvas.create_text(c, text='C')
```

Všimnite si, že aj v grafických príkazoch `create_text` sme použili premenné, ktoré sú dvojicami celých čísel. Program nakreslí:



Body na kružnici

Už vieme kresliť kružnicu `(x0, y0)` s polomerom `r` napríklad takto:

```
canvas.create_oval(x - r, y - r, x + r, y + r)
```

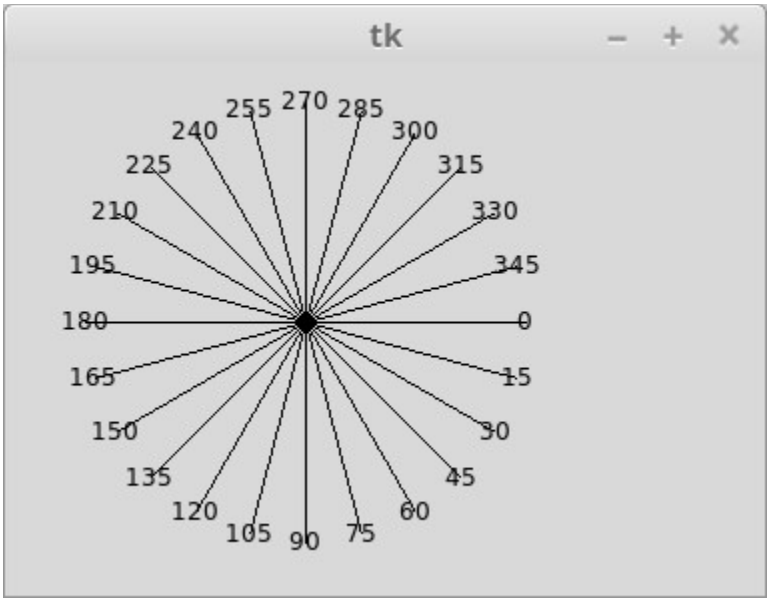
Lenže často riešime úlohy, v ktorých potrebujeme nie celú kružnicu, ale len niekoľko bodov na jej obvode. Využijeme goniometrické funkcie `sin` a `cos`. Potom každý bod na kružnici môžeme zapísať takýmto vzorcom:

```
x = x0 + r * cos(uhol)
y = y0 + r * sin(uhol)
```

kde `uhol` je zrejme nejaké číslo od 0 do 360 (nemusí byť celé) a (`x0`, `y0`) sú súradnice stredu kružnice. Zapíšme program, ktorý nakreslí lúče medzi stredom a bodmi na kružnici:

```
import tkinter
from math import sin, cos, radians

canvas = tkinter.Canvas()
canvas.pack()

x0, y0 = 150, 130
r = 110

for uhol in range(0, 360, 15):
    x = x0 + r * cos(radians(uhol))
    y = y0 + r * sin(radians(uhol))
    canvas.create_line(x0, y0, x, y)
    canvas.create_text(x, y, text=uhol)
```

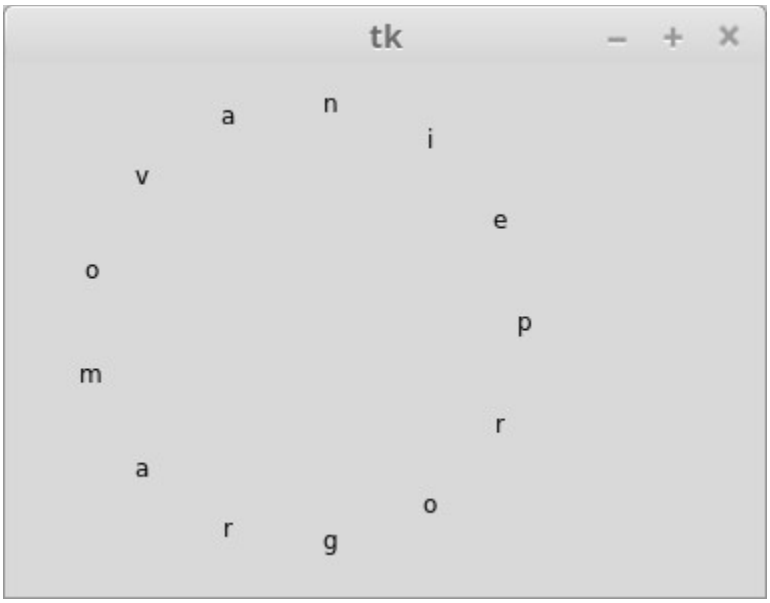
Program nakreslí 24 úsečiek, ktorých druhé konce sú rovnomerne rozmiestnené po obvode kružnice (po 15 stupňoch). Ku každej tejto úsečke sme pripísali aj jej prislúchajúci uhol:



Ešte ukážeme využitie tejto idey na vypisovanie nejakého textu po jednotlivých znakoch tak, že ich rovnomerne rozložíme po obvode kružnice. V tomto prípade nebudeme kresliť úsečky, vypíšeme len znaky textu. Keďže text vieme rozobrať na znaky len pomocou for-cyklu (napríklad `for znak in text`), premennú `uhol` budeme musieť zväčšovať priraďovaním v tele cyklu:

```
import tkinter
from math import sin, cos, radians

canvas = tkinter.Canvas()
canvas.pack()

x0, y0 = 150, 130
r = 110

text = input('zadaj text: ')
n = len(text)
uhol, posun = 0, 360/n

for znak in text:
    x = x0 + r * cos(radians(uhol))
    y = y0 + r * sin(radians(uhol))
    canvas.create_text(x, y, text=znak)
    uhol += posun
```

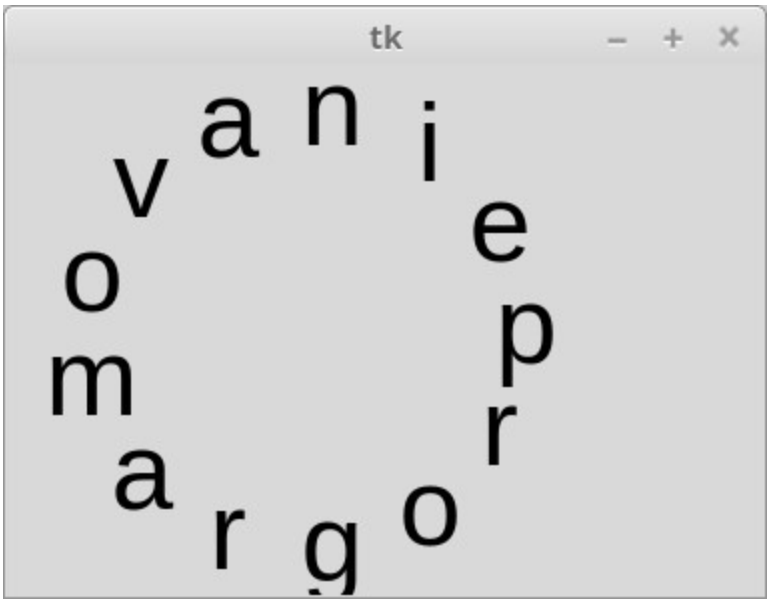
Použili sme tu štandardnú funkciu `len`, ktorá pre zadaný reťazec vráti počet znakov (tzv. dĺžka znakového reťazca). Keď zadáme vstupný reťazec `'programovanie'`, dostaneme:



Tu by sa oplatilo vedieť vypísať písmená tohto textu trochu väčším fontom. Na to slúži pomenovaný parameter `font`. Keď riadok s výpisom textu nahradíme:

```
canvas.create_text(x, y, text=znak, font='arial 35')
```

Program teraz na výpis textu použije font `'Arial'` a veľkosť znakov bude `35` (v rôznych operačných systémoch môže tento parameter dávať trochu rozdielne výsledky). Dostávame:

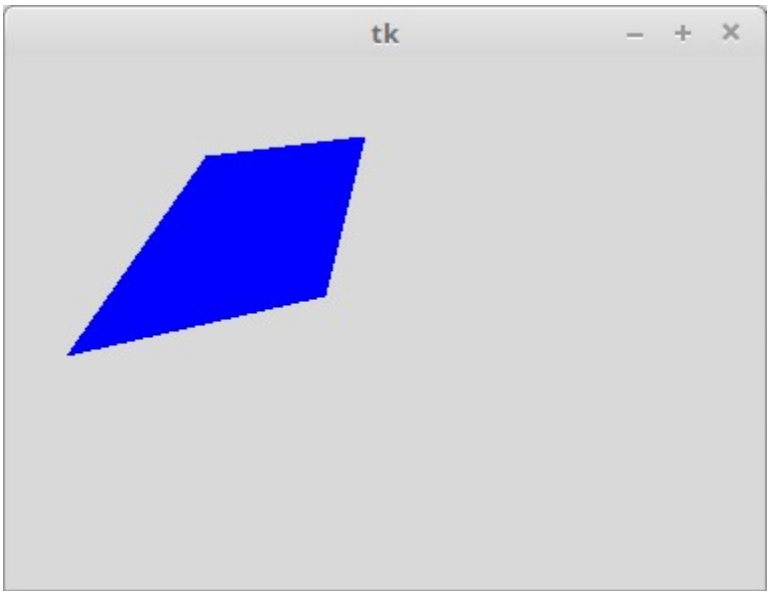


Kreslenie polygónov

Polygónom voláme oblasť grafickej plochy, ktorá je ohraničená zadanou lomenou čiarou (aspoň s tromi vrcholmi) a vyplní sa nejakou farbou. Body zadávame podobne ako pre `create_line`. Zrejme, keby sme zadali len dva body, bolo by to asi málo. Táto oblasť bude zafarbená čiernou farbou, prípadne ju môžeme zmeniť pomocou pomenovaného parametra `fill`, napríklad takto:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

a = (100, 50)
b = (30, 150)
c = (160, 120)
d = (180, 40)
canvas.create_polygon(a, b, c, d, fill='blue')
```

a vyzerá to takto:



Dostali by sme rovnaký polygón, aj keby sme zapísali:

```
canvas.create_polygon(100, 50, 30, 150, 160, 120, 180, 40, fill='blue')
```

alebo:

```
canvas.create_polygon((100, 50), (30, 150), (160, 120), (180, 40), fill='blue')
```

Všimnite si, že nakreslená oblasť (polygón) nemá nakreslený obrys. Ak by sme ho chceli vidieť, zadali by sme aj pomenovaný parameter `outline`.

Nasledovný program nakreslí `n` rovnostranných trojuholníkov, pričom všetky majú rovnakú veľkosť strany `a`. Trojuholníky budú v ploche umiestnené náhodne, t.j. nielen ich poloha bude náhodná, ale aj ich otočenie. okrem toho každý z nich zafarbíme náhodnou farbou. Pri kreslení náhodného trojuholníka použijeme takúto ideu:

1. zvolíme náhodné súradnice prvého vrcholu (pre istotu nie úplne blízko okrajov grafickej plochy)

2. zvolíme náhodný uhol otočenia trojuholníka

3. vypočítame druhý vrchol, ako bod na kružnici so stredom prvého vrcholu a s polomerom veľkosti strany trojuholníka

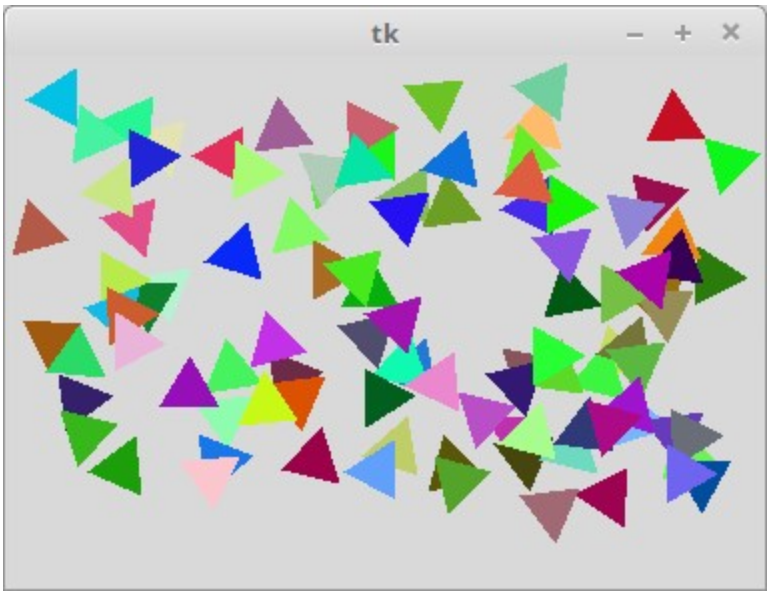
4. tretí vrchol trojuholníka leží na tej istej kružnici ako druhý vrchol, ale je o 60 stupňov otočený vľavo (alebo vpravo) oproti druhému vrcholu

Teraz máme istotu, že tieto tri vrcholy tvoria rovnostranný trojuholník so stranou `a` (je to rovnoramenný trojuholník, v ktorom je medzi ramenami uhol 60 stupňov).

Všimnite si, ako sme v tomto programe vyriešili importy zo všetkých troch modulov:

```
from tkinter import Canvas
from random import randint, randrange,
from math import sin, cos, radians

canvas = Canvas()
canvas.pack()

a = 30
n = 100

for i in range(n):
    x1 = randint(a, 380 - a)
    y1 = randint(a, 260 - a)
    uhol = randint(0, 360)
    x2 = x1 + a * cos(radians(uhol))
    y2 = y1 + a * sin(radians(uhol))
    uhol = uhol - 60
    x3 = x1 + a * cos(radians(uhol))
    y3 = y1 + a * sin(radians(uhol))
    farba = f'#{randrange(256**3):06x}'
    canvas.create_polygon(x1, y1, x2, y2, x3, y3, fill=farba)
```

Po spustení dostávame:



Grafický objekt obrázok

Aby sme do plochy mohli nakresliť nejaký obrázok, musíme najprv vytvoriť **obrázkový objekt** (pomocou `tkinter.PhotoImage()` prečítať obrázok zo súboru) a až tento poslať ako parameter do príkazu na kreslenie obrázkov `canvas.create_image()`.

Obrázkový objekt vytvoríme špeciálnym príkazom:

```
premenna = tkinter.PhotoImage(file='meno suboru')
```

v ktorom `meno suboru` je súbor s obrázkom vo formáte **png** alebo **gif**. Takýto obrázkový objekt môžeme potom vykresliť do grafickej plochy ľubovoľný počet-krát.

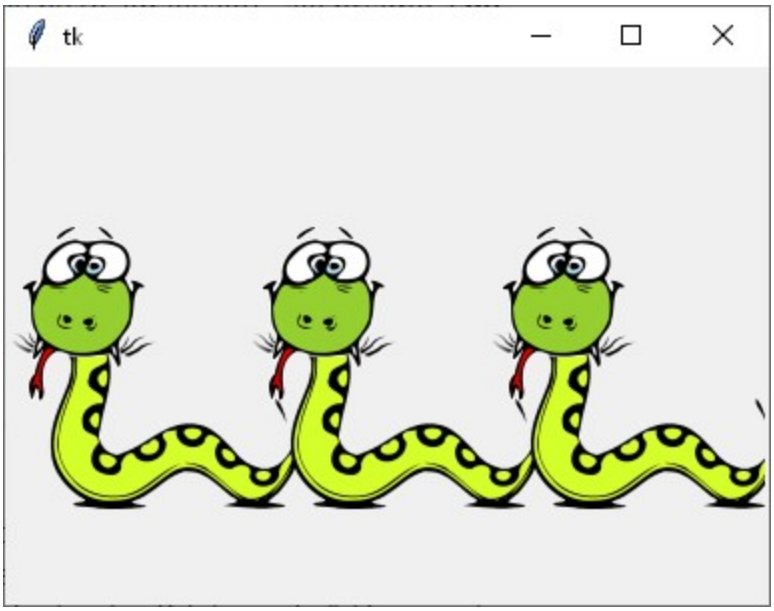
Samotná funkcia `canvas.create_image()` na vykreslenie obrázka má tri parametre: prvé dva sú súradnice stredu vykresľovaného obrázka a ďalší pomenovaný parameter určuje obrázkový objekt. Príkaz má tvar:

```
canvas.create_image(x, y, image=premenna)
```

V nasledovnom príklade sme si zvolili obrázok pyton.png a vykreslili sme ho niekoľkokrát vedľa seba:

```
import tkinter

canvas = tkinter.Canvas()
canvas.pack()

obr = tkinter.PhotoImage(file='pyton.png')
for x in range(80, 380, 120):
    canvas.create_image(x, 150, image=obr)
```

Po spustení dostávame:



Parametre grafickej plochy

Pri vytváraní grafickej plochy (pomocou `tkinter.Canvas()`) môžeme nastaviť veľkosť plochy ale aj farbu pozadia grafickej plochy. Môžeme uviesť tieto parametre:

- `bg` = nastavuje farbu pozadia (z anglického „background“)
- `width` = nastavuje šírku grafickej plochy
- `height` = výšku plochy

Napríklad:

```
canvas = tkinter.Canvas(bg='white', width=400, height=200)
```

Vytvorí bielu grafickú plochu, ktorá má šírku 400 a výšku 200.

Zhrnutie parametrov grafických príkazov

texty

```
canvas.create_text(x, y, ...)        # súradnica jedného bodu
```

- `text` = vypisovaný text
- `font` = písmo a veľkosť
    - buď `'meno veľkosť'` pre jednoslovné meno fontu
    - alebo `('meno', veľkosť)`
- `fill` = farba textu
- `angle` = uhol otočenia v stupňoch

- `anchor` = ukotvenie (pozícia `(x, y)`)
    - jedno z `'center'`, `'nw'`, `'n'`, `'ne'`, `'e'`, `'se'`, `'s'`, `'sw'`, `'w'`

obdĺžniky

```
canvas.create_rectangle(x, y, x, y, ...)     # súradnice dvoch bodov
```

- `width` = hrúbka obrysu
    - hodnota `0` označuje bez obrysu
- `outline` = farba obrysu
    - hodnota `''` označuje bez obrysu
- `fill` = farba výplne
    - hodnota `''` označuje bez výplne

elipsy

```
canvas.create_oval(x, y, x, y, ...)       # súradnice dvoch bodov
```

- `width` = hrúbka obrysu
    - hodnota `0` označuje bez obrysu
- `outline` = farba obrysu
    - hodnota `''` označuje bez obrysu
- `fill` = farba výplne
    - hodnota `''` označuje bez výplne

lomené čiary

```
canvas.create_line(x, y, x, y, x, y, x, y, ...)      # súradnice aspoň dvoch bodov
```

- `width` = hrúbka čiary
- `fill` = farba čiary
- `arrow` = šípka na konci čiary
    - jedno z `'first'`, `'last'`, `'both'`

polygóny

```
canvas.create_polygon(x, y, x, y, x, y, x, y, ...)       # súradnice aspoň dvoch bodov
```

- `width` = hrúbka obrysu
    - hodnota `0` označuje bez obrysu
- `outline` = farba obrysu

    - hodnota `''` označuje bez obrysu
- `fill` = farba výplne
    - hodnota `''` označuje bez výplne

© Copyright 2019-2020, Andrej Blaho.
Naposledy aktualizované 24. mar. 2020.
Vytvorené pomocou Sphinx 2.4.4.


This work is licensed under a Creative Commons Attribution 4.0 International License.