

Obsah

[1 Úvod](#)

[2 Inštalácia jazyka Python a spustenie prostredia IDLE](#)

[3 Grafické príkazy](#)

[3.1 Vytvorenie grafickej plochy](#)

[3.2 Súradnice bodov a kreslenie čiar](#)

[3.3 Kreslenie obdĺžnikov](#)

[3.4 Kreslenie elips](#)

[3.5 Písanie textu do grafickej plochy](#)

[4 Premenné a náhodné hodnoty](#)

[4.1 Náhodné čísla a príkazový režim \(shell\)](#)

[4.2 Premenné a náhodná farba](#)

[4.3 Kreslenie obrázkov na náhodnom mieste](#)

[5 Opakovanie časti programu - for cyklus](#)

[5.1 Opakujeme vykonávanie príkazov](#)

[5.2 For cyklom kreslíme obrázky s pravidelnosťou](#)

[6 Vytvárame podprogramy](#)

[7 Úlohy na opakovanie I](#)

[8 Klikanie myšou a ovládanie klávesnicou](#)

[8.1 Reakcia na ľavé tlačidlo myši](#)

[8.2 Reakcia na pravé tlačidlo myši](#)

[8.3 Reakcia na stláčanie klávesnice](#)

[9 Podmienené príkazy](#)

[10 Časovač](#)

[11 Tlačidlá a vstupné pole](#)

[12 Posúvanie objektov canvasu](#)

[13 Vytvárame jednoduché hry](#)

[13.1 Testovač pozornosti](#)

[13.2 Chytanie loptičiek](#)

[13.3 Hľadaj percento](#)

[14 Úlohy na opakovanie II](#)

[15 Pracujeme s textom](#)

[16 Test - Preteky rytierov](#)

[17 Poznámky](#)

[17.1 Používanie príkazu import](#)

[17.2 Náhodné farby](#)

[17.3 Zadávanie vstupu - entry](#)

[18 Použitá literatúra](#)

1 Úvod

Niečo o jazyku Python

Autorom jazyka Python je Guido van Rossum - počítačový programátor, ktorý sa narodil a vyrástol v Holandsku. Python vznikol v roku 1991 a názov získal podľa Monty Pythonovho Lietajúceho cirkusu (autor bol jeho fanúšikom). V súčasnosti je Python veľmi populárny a jeho popularita ešte rastie. Je to moderný programovací jazyk, ktorý podporuje rôzne programovacie paradigmy. Python je freeware a open source. Používa ho CERN, Google, Facebook, YouTube, Mozilla a iní. Beží na rôznych platformách, napr. Linux, Windows, Mac. Python sa učia študenti na mnohých špičkových univerzitách ako svoj úvodný programovací jazyk (MIT, Berkeley, ...).

Komu je určená táto učebnica?

Učebnica je určená pre študentov strednej školy pre predmet informatika, je koncipovaná aj pre samoukov. Tiež môže pomôcť učiteľom, ktorí sa rozhodujú pre vyučovanie jazyka Python.

Podrobnejšie o učebnici

Podľa Štátneho vzdelávacieho programu (ŠVP) pre gymnáziá programovanie tvorí jednu z piatich oblastí ŠVP - Algoritmické riešenie problémov. V ŠVP sa zámerne neurčuje konkrétny softvér a programovací jazyk, ktorý má škola vyučovať. Výber programovacieho jazyka je v kompetencii školy a učiteľa. V našom školskom prostredí má zatiaľ najväčšiu tradíciu jazyk Pascal, resp. objektový Pascal v prostrediach Delphi a Lazarus. Dnes je už tento jazyk starý a v praxi sa takmer nikde nepoužíva. Učebnica vychádza z našich niekoľkoročných skúseností z vyučovania programovania na gymnáziu v jazykoch Pascal, Object Pascal a z ročnej skúsenosti v jazyku Python. Aj u nás v škole sme riešili neaktuálnosť Delphi a Lazarusu a aj vďaka inšpiráciám od RNDr. Andreja Blaha PhD. sme prešli na jazyk Python. Kedže na tento jazyk v tom čase neexistovala vhodná učebnica, rozhodli sme sa ju napísať. Bude nás tešiť, keď pomôžeme aj pomocou tejto učebnice, aj pomocou vzdelávania v Klube učiteľov informatiky spopularizovať tento jazyk aj na slovenských stredných školách.

Učebnica reflekтуje obsahový a výkonový štandard ŠVP. Obsahuje kapitoly, ktoré podľa nás majú zvládnúť aj študenti, ktorí sa nebudú pripravovať na maturitu z informatiky. Obsah učebnice pokrýva základný kurz programovania v rozsahu približne 33 vyučovacích hodín, pričom niektoré z úloh odporúčame zadávať na domácu prípravu. Prvých 16 vyučovacích hodín je odčlenených siedmou kapitolou - Úlohy na opakovanie I. Ďalších 16 hodín končí štrnásťou kapitolou - Úlohy na opakovanie II. Kapitola 15 - pracujeme s textom je doplňujúca kapitola, ktorá je vhodná pre šikovnejšie skupiny alebo šikovnejších študentov v skupine. Nie však pre svoju náročnosť, ale len z časového hľadiska.

Učebnica je názorná, obsahuje množstvo praktických úloh, využíva grafické prostredie knižnice tkinter. Študenti sa na začiatku naučia kresliť grafické útvary, používať cyklus a vetvenie, ovládať program klávesnicou a myšou. V závere budú tvoriť animácie a jednoduché hry. V závere učebnice je jeden záverečný test z nášho školského prostredia.

Dávame do pozornosti

Stránku <https://www.facebook.com/programujemevpython/>, kde môžete písť otázky alebo odozvy na učebnicu.

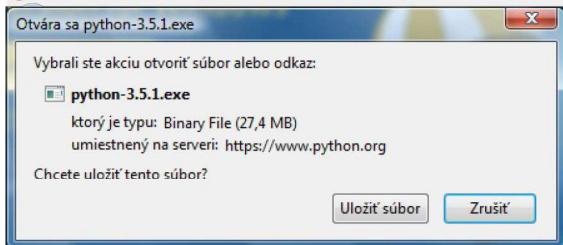
Učiteľom dávame do pozornosti aj Klub učiteľov informatiky v Bratislave na stránke: <http://www.1sg.sk/itklub>

Prajeme vám príjemné chvíle pri učení sa Pythonu :)
autor

2 Inštalácia jazyka Python a spustenie prostredia IDLE

Na stránke [www.python.org](https://www.python.org/downloads/) si stiahneme Python 3.5.1 (číslo verzie) vhodný pre náš počítač. Ak máme 64-bitovú verziu operačného systému, stiahneme si Python pre 64-bitový operačný systém. Dôležité je, aby sme mali nainštalovanú verziu začínajúcú číslom 3.

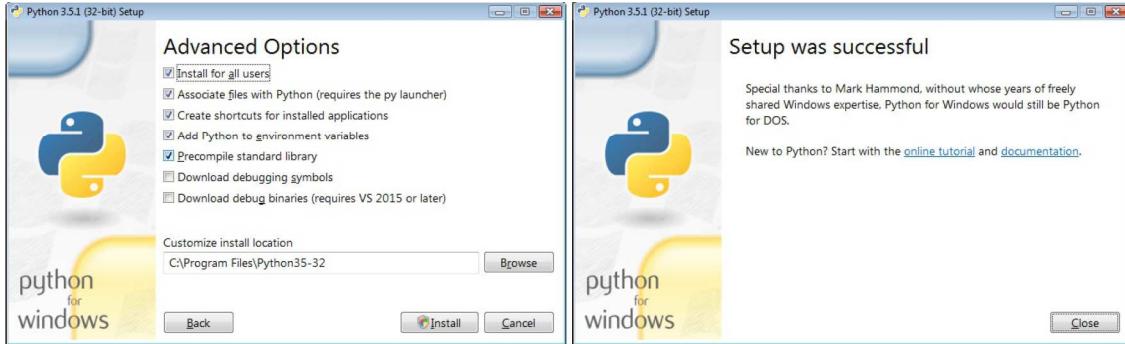
The screenshot shows the Python Software Foundation website. At the top, there's a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation is a search bar with a magnifying glass icon and a 'GO' button. To the right of the search bar are 'Socialize' and 'Sign In' buttons. The main content area has a large Python logo and the word 'python™'. Below it, a blue bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. A prominent section titled 'Download the latest version for Windows' features two yellow buttons: 'Download Python 3.5.1' and 'Download Python 2.7.11'. Below these buttons is a note about the difference between Python 2 and 3. Further down, there are links for different operating systems: Windows, Linux/UNIX, Mac OS X, and Other. A note about testing development versions is also present. To the right of the text is a cartoon illustration of two boxes with yellow and white striped parachutes falling from the sky. Below this section, there's a link for a specific release: 'Looking for a specific release? Python releases by version number:' followed by a table listing Python 3.4.4 and Python 3.5.1 with their respective release dates (December 21, 2015), download links, and release notes. A Firefox browser status bar at the bottom indicates: 'Otvára sa python-3.5.1.exe' and 'Zvolit, aké údaje sa odosielajú'.



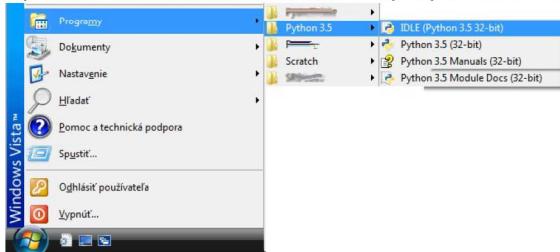
Zaklikneme Customize installation a tiež poslednú možnosť Add Python 3.5 PATH



V okne Advanced Options potvrdíme možnosť Install for all users.



Python 3.5.1 máme nainštalovaný. Spúšťame ho ikonou pre IDLE (Python 3.5)



Po spustení sa otvorí okno s názvom Python Shell. V tomto okne sa dajú priamo písť príkazy a po potvrdení enterom sa automaticky vykonajú. Je to interaktívne prostredie, označujeme ho aj názvom príkazový režim alebo len shell. My budeme najprv zapisovať program v programovacom režime. To znamená, že v okne shell vyberieme z hlavného menu File a v ňom ponuku New File. Otvorilo sa nám nové okno, kde zapíšeme program. Program uložíme. Ak ho chceme spustiť, stlačíme F5. Po spustení sa nám do shell-u napiše informácia, ktorý program sme spustili, a tiež sa reštartujú všetky nastavenia Pythonu v príkazovom režime.

3 Grafické príkazy

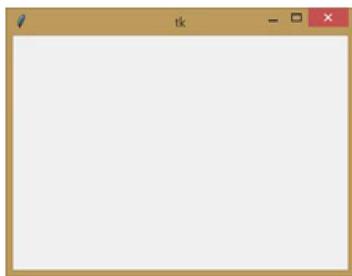
3.1 Vytvorenie grafickej plochy

Máme spustené prostredie IDLE a vytvorený nový súbor - sme v programovacom režime. Postup spustenia a vytvorenia súboru sme si ukázali v predchádzajúcej kapitole. Keď chceme pracovať s grafickou plochou, musíme najprv do programu naimportovať modul na prácu s grafickou plochou (je to nejaký program, v ktorom už niekto naprogramoval príkazy, ktoré budeme používať). My budeme na prácu s grafikou používať modul `tkinter`. Tento modul naimportujeme na začiatku programu zápisom `import tkinter`.

Po naimportovaní modulu vytvoríme grafickú plochu (plátno), do ktorej budeme neskôr kresliť `canvas = tkinter.Canvas()`. Príkaz `canvas.pack()` zabezpečí zobrazenie nového okna aj s vytvorenou grafickou plochou. Celý program zatial vyzerá takto:

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

Ked spustíme tento program (F5), vytvorí sa okno s grafickou plochou. V okne zatial nič nevidíme, lebo sme doň nič nenakreslili. Okno sa zatvorí klasicky .

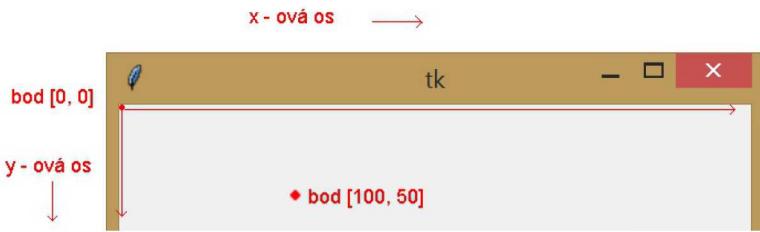


3.2 Súradnice bodov a kreslenie čiar

Do grafickej plochy môžeme kresliť napríklad čiary, obdĺžniky, elipsy. Pri kreslení potrebujeme príkazom zadávať súradnice bodov.



Každý bod má dve súradnice: x-ovú a y-ovú. X-ová súradnica určuje vodorovnú pozíciu a y-ová určuje zvislú pozíciu. Ale pozor, y-ová súradnica rastie smerom dole (nie ako sme zvyknutí v matematike, kde rastie smerom hore). Súradnice zapisujeme ako dvojicu čísel, pričom x-ová súradnica je vždy v tejto dvojici prvá a druhá je y-ová súradnica.



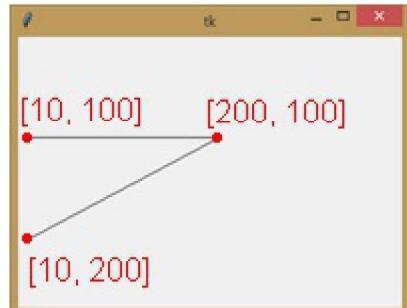
Otázky:

1. Aké súradnice bude mať bod, ktorý je o 20 bodov vľavo od bodu so súradnicami [100, 50]?
2. Aké súradnice bude mať bod, ktorý je o 20 bodov vpravo od bodu so súradnicami [100, 50]?
3. Aké súradnice bude mať bod, ktorý je o 20 bodov hore od bodu so súradnicami [100, 50]?
4. Aké súradnice bude mať bod, ktorý je o 20 bodov dole od bodu so súradnicami [100, 50]?
5. Čo majú spoločné všetky body, ktoré sú vľavo alebo vpravo od bodu [100, 50]?

Príkazom `canvas.create_line(10, 100, 200, 100)` nakreslíme čiaru, ktorá začína v bode so súradnicami [10, 100] a končí v bode [200, 100]. Týmto príkazom môžeme kresliť aj čiary, ktoré prechádzajú cez viacero bodov. Jednotlivé súradnice bodov pridávame do príkazu. Napríklad: `canvas.create_line(10, 100, 200, 100, 10, 200)` k pôvodnej čiare sme pridali ďalší bod so súradnicou [10, 200]. Po spustení programu:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_line(10, 100, 200, 100, 10, 200)
```

budeme vidieť čiaru, ktorá začína v bode [10, 100], pokračuje do bodu [200, 100] a končí v bode [10, 200].



Úloha:

- 1 Upravte tento program tak, aby dokreslil tieto dve čiary do trojuholníka.

Pri kreslení môžeme čiarám nastaviť aj hrúbku, aj farbu.

```
canvas.create_line(10, 100, 200, 100, width=5)
```

Nakreslí čiaru s hrúbkou 5 bodov.

```
canvas.create_line(10, 100, 200, 100, fill='red')
```

Nakreslí červenú čiaru.

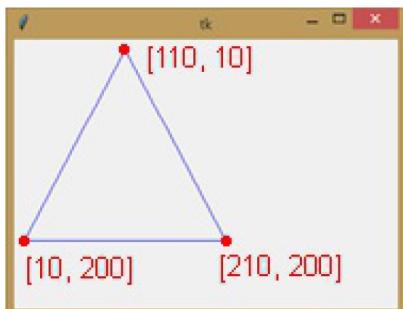
```
canvas.create_line(10, 100, 200, 100, fill='red', width=5)
```

alebo

```
canvas.create_line(10, 100, 200, 100, width=5, fill='red')
```

nakreslí červenú čiaru s hrúbkou 5 bodov. Hrúbku nastavujeme parametrom `width` a uvedieme mu číselnú hodnotu. Zafarbenie nastavujeme parametrom `fill` a v apostrofoch zapíšeme názov farby (môžeme použiť rôzne farby, napríklad: `'white', 'black', 'red', 'blue', 'yellow', 'green', 'maroon', 'orange', 'gray', 'skyblue', 'violet', 'fuchsia', 'olive'`).

Znak ' môžeme napísť na SK klávesnici stlačením Ctrl+Alt+P. Krok späť urobíme stlačením Ctrl+Z



Tento trojuholník môžeme nakresliť rôznymi postupmi:

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_line(110, 10, 10, 200, fill='blue')  
canvas.create_line(10, 200, 210, 200, fill='blue')  
canvas.create_line(210, 200, 110, 10, fill='blue')
```

alebo

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_line(110, 10, 10, 200, 210, 200, 110, 10, fill='blue')
```

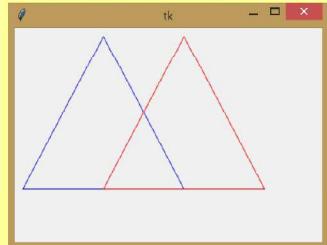
Úlohy:

2

Zamyslite sa. Môžeme trojuholník v predchádzajúcim programe nakresliť ešte inými spôsobmi?

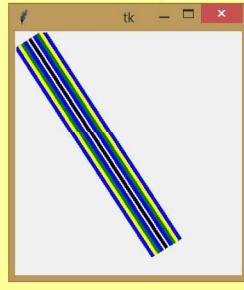
3

Nakreslite rovnaký trojuholník červenou farbou, posunutý podľa ukážky na obrázku.



4

Zmenou farby a hrúbky čiary nakreslite takýto obrázok:



5 Pomocou čiar nakreslite písmená L, T, H, Z.

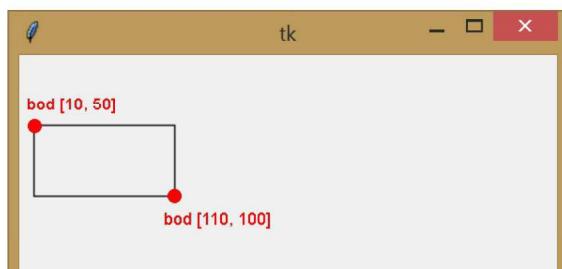
6 Pomocou čiar nakreslite obdĺžnik.

3.3 Kreslenie obdĺžnikov

Otázky:

6. Obdĺžnik už vieme nakresliť pomocou príkazu na kreslenie čiar. Ak kreslíme obdĺžnik pomocou čiar, najmenej koľko bodov (súradníc týchto bodov) musíme v príkaze na kreslenie čiar použiť?
7. Vieme jednoznačne určiť obdĺžnik aj pomocou menšieho počtu bodov? Najmenej koľko ich potrebujeme? Svoju odpoveď zdôvodnite.

Príkaz `canvas.create_rectangle(10, 50, 110, 100)` nakreslí obdĺžnik, ktorý je zadaný pomocou bodu so súradnicou [10, 50] a [110, 100].



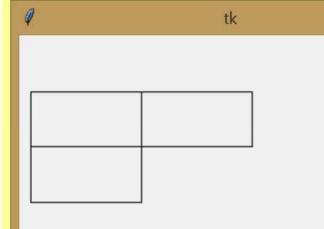
Otázky:

8. Akú šírku v bodoch má obdĺžnik na obrázku?
9. Akú výšku v bodoch má obdĺžnik na obrázku?
10. Môžeme tento istý obdĺžnik nakresliť aj týmto príkazom
`canvas.create_rectangle(110, 100, 10, 50)`?
11. Môžeme tento istý obdĺžnik (na tom istom mieste) nakresliť ešte pomocou iných súradníc?
12. Čo sa nakreslí po zadaní tohto príkazu `canvas.create_rectangle(10, 50, 110, 50)`?
13. Čo sa nakreslí po zadaní tohto príkazu `canvas.create_rectangle(10, 50, 10, 50)`?
14. Akú výšku a šírku majú predchádzajúce dva obdĺžniky?

Úlohy:

7 Nakreslite hneď vpravo vedľa obdĺžnika `canvas.create_rectangle(10, 50, 110, 100)` rovnako veľký obdĺžnik, ktorý sa ho bude dotýkať.

8 Nakreslite hneď pod ľavým obdĺžnikom (z predchádzajúcej úlohy) rovnako veľký obdĺžnik.



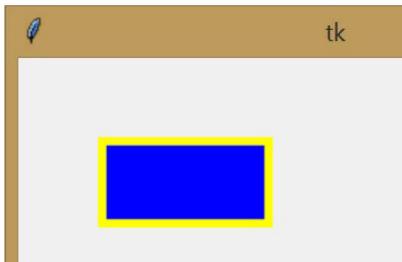
Otázky:

15. Musíme pri zadávaní príkazov na kreslenie obdĺžnikov v úlohách číslo 7 a 8 vymýšľať alebo vypočítavať súradnice všetkých bodov?
16. Majú nové body, ktoré zadávame, niečo spoločné s bodmi, ktoré sme už zadali?

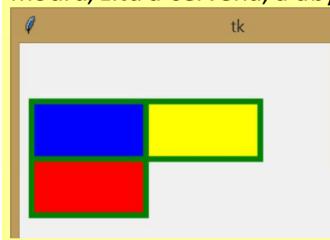
Aj pri kreslení obdĺžnika môžeme použiť parameter na nastavenie farby rovnakým spôsobom ako pri nastavení farby pri kreslení čiar. Parameter `fill` pri kreslení obdĺžnika nastavuje vnútornú farbu, čiže rovnako výplň kresleného útvaru. Neovplyvňuje ale farbu obrysу obdĺžnika, ako by sme si mohli myslieť. Na nastavenie farby strán obdĺžnika môžeme použiť parameter `outline`. Čiže príkaz:

```
canvas.create_rectangle(50, 50, 150, 100, fill='blue', outline='yellow', width=5)
```

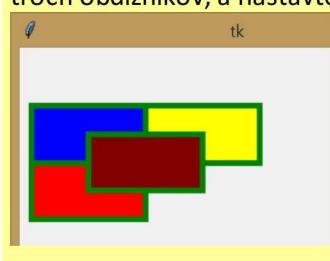
nakreslí obdĺžnik vyplnený modrou farbou so žltým obrysom, ktorý má hrúbku 5 bodov. Pripomeňme si, že tieto parametre môžu byť v rôznom poradí.

**Úlohy:**

- 9** Strom obdĺžnikom z úlohy č. 8 nastavte parametre tak, aby jednotlivé obdĺžniky mali tieto výplne: modrú, žltú a červenú, a aby farba ich obrysú bola zelená a obrys mal hrúbku 5 bodov.

**10**

- Doplňte do tohto obrázku štvrtý, rovnako veľký obdĺžnik, ktorého stred je v spoločnom bode všetkých troch obdĺžnikov, a nastavte mu hnedú výplň.

**Otázky:**

17. Čo sa stane, keď parametre na nastavenie farby pomiešame medzi súradnice bodov
`canvas.create_rectangle(outline='yellow', 50, 50, 150, 100, fill='blue')?`
18. V úlohe číslo 10 sme nakreslili štyri obdĺžniky štyrmi príkazmi. Záleží na poradí týchto príkazov? Čo sa stane, keď hnedý obdĺžnik budeme kresliť ako prvý? Poradie ktorých príkazov môžeme zmeniť, aby sa nám nakreslil rovnaký obrázok?
19. Čo sa stane, keď parameter `outline` nastavíme takto: `outline=''`?

20. Čo sa stane, keď parameter fill nastavíme takto: `fill=''`?
21. Čo sa stane, keď parameter fill a súčasne parameter outline nastavíme takto: `fill='', outline=''`?
22. Koľko obdĺžnikov budeme vidieť po spustení týchto príkazov?
 - a) `canvas.create_rectangle(120, 100, 170, 150)`
`canvas.create_rectangle(100, 100, 150, 150)`
`canvas.create_rectangle(150, 100, 200, 150)`
`canvas.create_rectangle(100, 150, 150, 200)`
 - b) `canvas.create_rectangle(120, 100, 170, 150, fill='white')`
`canvas.create_rectangle(100, 100, 150, 150, fill='')`
`canvas.create_rectangle(150, 100, 200, 150, fill='green')`
`canvas.create_rectangle(100, 150, 150, 200, fill='', outline='')`

Úlohy:

11 Nakreslite vlajku: a) Poľska, b) Francúzska, c) Nemecka, d) Maďarska, e) Švajčiarska, f) Izraela.

12 Iba dvomi príkazmi nakreslite vlajku Lotyšska.



13 Nakreslite vlajku Grécka.



14 Dano chcel nakresliť vlajku Francúzska a napísal tento program. Nájdite chybu v tomto programe.

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_rectangle(10,50,110,300, fill='blue', outline='')  
canvas.create_rectangle(60,50,160,300, fill='white', outline='')  
canvas.create_rectangle(110,50,210,300, fill='red', outline='')
```

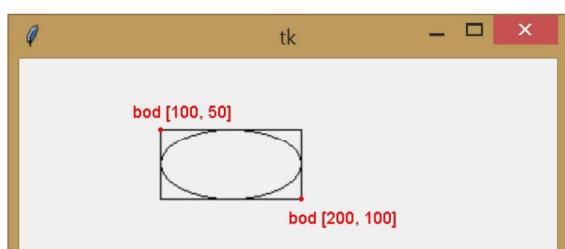
3.4 Kreslenie elíps

Už vieme, že príkaz `canvas.create_rectangle(100, 50, 200, 100)` nakreslí obdĺžnik, ktorého ľavý horný bod má súradnicu [100, 50] a pravý dolný bod má súradnicu [200, 100]. Ak použijeme rovnaké súradnice, ale zmeníme príkaz `canvas.create_rectangle` na `canvas.create_oval`, nakreslí sa elipsa.

Tento program s rovnakými súradnicami, ale rôznymi príkazmi:

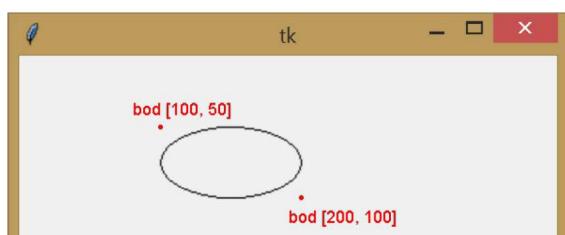
```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(100, 50, 200, 100)
canvas.create_oval(100, 50, 200, 100)
```

nakreslí aj obdĺžnik, aj elipsu. Ako vidíme na obrázku, súradnice bodov, ktoré určujú elipsu (ovál), sa nenachádzajú na elipse, ale mimo nej. Tieto body sú súradnicami obdĺžnika, v ktorom sa nakreslí vpísaná elipsa (čiže najväčšia možná elipsa, ktorá sa zmestí do tohto pomyselného obdĺžnika).



Program len s príkazom na kreslenie elipsy nakreslí len samotnú elipsu:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_oval(100, 50, 200, 100)
```



Pri kreslení elipsy nám zo začiatku môže pri vymýšľaní súradníc pomôcť, keď si najprv nakreslíme správny obdĺžnik (v ňom si predstavujeme najväčšiu elipsu) a až potom, keď máme správne súradnice, zmeníme tento príkaz na elipsu.

Otázky:

23. Vieme nakresliť obdĺžnik jedným príkazom. Potrebujeme špeciálny príkaz na kreslenie štvorca? Podľa čoho rozpoznáme, že súradnice v príkaze pre obdĺžnik nakreslia štvorec?
24. Môžeme pomocou príkazu `canvas.create_oval` kresliť kruhy? Potrebujeme špeciálny príkaz na kreslenie kruhov? Podľa čoho rozpoznáme, že súradnice v príkaze pre ovál nakreslia kruh?

V jednom programe môžeme kresliť aj viacero obrázkov vedľa seba. Ak nám nestačí veľkosť canvasu, alebo chceme zmeniť jeho farbu pozadia, môžeme tak urobiť už na začiatku programu pri vytváraní canvasu. Parametrom `height` nastavíme výšku, parametrom `width` nastavíme šírku a farbu pozadia nastavíme parametrom `bg`.

```
import tkinter  
canvas = tkinter.Canvas(bg='white', width=800, height=600)  
canvas.pack()
```

Úlohy:

15 K elipse `canvas.create_oval(100, 50, 200, 100)` nakreslite tesne vedľa nej rovnako veľkú elipsu.

16 Pre obe elipsy nastavte rôzne farby výplne, obrysov a tiež hrúbky čiar.

17 Nakreslite vlajku Japonska.

18 Nakreslite snehuliaka.

19 Nakreslite symetrickú tvár s očami, nosom a ústami:



20 Doplňte k tvári okuliare.

21 Nakreslite vlajku červeného polmesiaca:



22 Nakreslite postavičku s klobúkom:

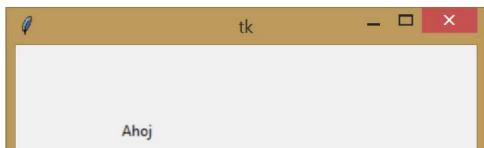


23 Nakreslite farebný terč.

3.5 Písanie textu do grafickej plochy

Príkaz `canvas.create_text(100, 70, text='Ahoj')` napíše na súradnicu [100, 70] text Ahoj. Prvé dva parametre tohto príkazu sú súradnice stredu vypisovaného textu. Text sa napíše tak, aby aj v zvislom, aj vo vodorovnom smere boli súradnice presne vycentrované v strede napísaného textu. Parametrom text zadávame samotný text, ktorý chceme napísat.

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_text(100, 70, text='Ahoj')
```



V príkaze môžeme použiť aj parameter `font`, ktorým určíme typ, veľkosť a rez písma. Napríklad `font='Arial 70 bold'`. Takto môžeme zadávať len jednoslovne názvy fontov. Viacslovný názov sa zadáva komplikovanejším spôsobom.

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_text(100, 70, text='Ahoj')  
canvas.create_text(200, 50, text='Python', font='Arial 70 bold')
```

Program napíše tieto texty:



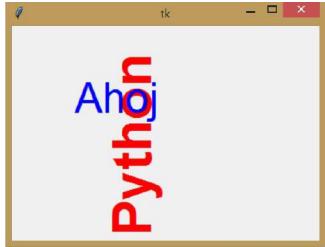
Farbu výplne písma určujeme parametrom `fill`, rovnako ako v príkazoch, ktoré už poznáme. Písmo môžeme aj otáčať parametrom `angle` (nefunguje to na počítačoch Mac).

```
canvas.create_text(100, 70, text='Ahoj', fill='blue')  
canvas.create_text(200, 50, text='Python', font='Arial 70 bold', fill='red')
```



```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
canvas.create_text(150, 150, text='Python', font='Arial 50 bold', fill='red',  
angle=90)  
canvas.create_text(130, 90, text='Ahoj', fill='blue', font='Arial 40')
```

Nápis Python sme otočili o 90 stupňov. Najprv sa nakreslí tento nápis a potom sa modrou farbou napíše Ahoj. Pri nastavení uhla sa text otáča okolo zadanej súradnice, čiže presného stredu textu (fiktívneho opísaného obdĺžnika textu). Otáčanie určuje uhol v protismere hodinových ručičiek (ako sme zvyknutí na matematike). Uhol 180 stupňov nám text otočí dole hlavou. Ak chceme text otáčať v smere hodinových ručičiek, môžeme použiť záporný uhol.



Úlohy:

24

Nakreslite tieto dopravné značky:



25

Nakreslite tieto dopravné značky:



4 Premenné a náhodné hodnoty

4.1 Náhodné čísla a príkazový režim (shell)

Na prácu s náhodnými číslami potrebujeme používať modul `random`. Importujeme ho rovnako ako modul `tkinter`, čiže na začiatku programu, a zapíšeme to takto: `import random`. Teraz môžeme používať príkazy `random.randint()`, `random.randrange()`.

```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()

canvas.create_text(100, 100, text=random.randint(10, 20))
canvas.create_text(150, 100, text=random.randint(5, 8))
```

Tento program vypíše dve náhodné celé čísla. Jedno bude na súradnici [100, 100] a bude z intervalu <10, 20> a druhé bude na súradnici [150, 100] a bude z intervalu <5, 8>.

Otázky:

1. Ktoré čísla bude program vypisovať na súradnici [150, 100]?
2. Zistite, ktoré čísla bude vypisovať program na súradnici [150, 100], ak príkaz `random.randint(5, 8)` nahradíme príkazom `random.randrange(5)`.
3. Ktoré čísla bude vypisovať, ak vo výpise použijeme príkaz:
 - a) `random.randrange(2)`
 - b) `random.randrange(1)`
4. Aký je rozdiel medzi príkazmi `random.randrange(3, 5)` a `random.randint(3, 5)`?

Príkazy môžeme písanie aj priamo v príkazovom režime (v okne Python Shell). Stačí v príkazovom riadku napísanie príkazu a po stlačení enter sa nám zadaný príkaz ihneď vykoná. Práve v príkazovom režime si môžeme vyskúšať, ktoré náhodné čísla bude počítač ťrebovať. Ak sme nespúšťali žiadeden program, v ktorom importujeme modul na náhodné čísla, môžeme modul importovať aj priamo z príkazového riadku. Ukážka príkazov, ktoré sme zadávali do príkazového riadku:

```
>>> import random
>>> random.randint(10, 15)
14
>>>
```

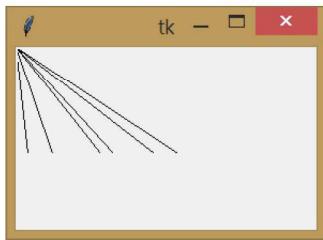
Vyskúšajte, čo bude kresliť tento program:

```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_line(0, 0, random.randrange(200), 100)
```

Aby sme lepšie videli, kde sa kreslia tieto čiary, ktoré majú v jednej zo súradníci náhodné číslo, môžeme do spusteného programu (už je v ňom vytvorené okno s grafickou plochou a importovaný modul random) zadávať aj príkazy priamo v shelli (príkazovom režime). Hneď po potvrdení príkazu budeme vidieť, čo sa udeje v grafickej ploche.

```
>>> canvas.create_line(0, 0, random.randrange(200), 100)
>>> canvas.create_line(0, 0, random.randrange(200), 100)
```

```
>>> canvas.create_line(0, 0, random.randrange(200), 100)
```



Úlohy:

1

Vypíšte do grafickej plochy vedľa seba šesť náhodných čísel od 1 do 49. Myslite si, že môžeme takýto program použiť na žrebovanie čísel v lotérii? Svoju odpoveď zdôvodnite.

4.2 Premenné a náhodná farba

Tento program nám nakreslí náhodný obdĺžnik, ktorý má ľavý horný bod na súradnici [50, 100].

```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(50, 100, 50+random.randint(30, 100),
                      100+random.randint(30, 100))
```

Ako by sme zabezpečili, aby sa na súradnici [50, 100] nakreslil náhodný štvorec? Čo musí spĺňať obdĺžnik, aby bol štvorcovom? Musí mať rovnakú aj šírku, aj výšku. Príkaz `random.randint()` nám ale pri každom použití vyžrebuje náhodnú hodnotu. Občas sa nám môže stať, že hodnota bude rovnaká a bude to štvorec, ale často vyžrebuje rôzne čísla. Potrebujeme číslo vyžrebovať len raz, zapamätať si ho a potom ho použiť na miestach, kde ho používame. Údaje si môžeme pamätať v premennej. Premenné poznáte z matematiky a pomocou nich ste si označovali hodnotu, ktorá sa mohla meniť, často napríklad ako x .

My si v programe vytvoríme premennú `velkost`, vyžrebujeme náhodné číslo a zapamätáme si ho do tejto premennej a potom túto zapamätanú hodnotu použijeme na miestach, kde to potrebujeme.

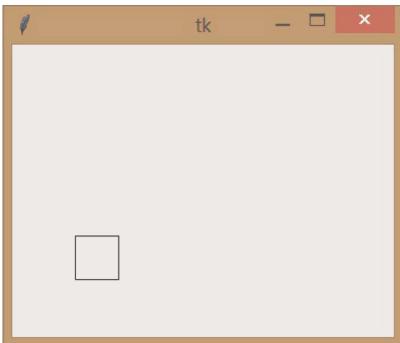
```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()
velkost = random.randint(30, 100)
canvas.create_rectangle(50, 100, 50+velkost, 100+velkost)
```

Do premennej si môžeme zapamätať aj súradnice, kde kreslíme štvorec, a potom pri kreslení použiť iba premenné.

```
x = 50
y = 100
velkost = random.randint(30, 100)
canvas.create_rectangle(x, y, x+velkost, y+velkost)
```

Ak doplníme žrebovanie náhodného čísla aj do premennej x a y , budeme kresliť náhodný štvorec na náhodnom mieste.

```
x = random.randrange(200)
y = random.randrange(200)
velkost = random.randint(30, 100)
canvas.create_rectangle(x, y, x+velkost, y+velkost)
```

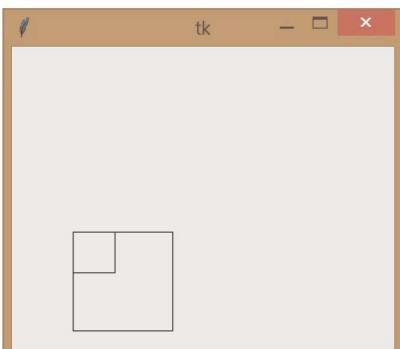


Ak sme už program spustili a napíšeme do príkazového riadku meno premennej a stlačíme enter, Python nám napiše aktuálnu hodnotu premennej.

```
>>> x  
61  
>>> y  
186  
>>> velkost  
42
```

V príkazovom riadku môžeme aj nastavovať hodnoty premenných.

```
>>> velkost = 100  
>>> canvas.create_rectangle(x, y, x+velkost, y+velkost)  
2  
>>>
```



S premennými môžeme vykonávať aj jednoduché matematické operácie – plus, mínus, násobenie, delenie.

```
>>> a = 5  
>>> b = 10  
>>> c = a+b  
>>> c  
15  
>>> d = c-2*a  
>>> d  
5  
>>> d/2  
2.5
```

A tiež môžeme spraviť výpočet so samotnou premenou a výsledok si zapamätať do tej istej premennej, čiže zmeniť jej hodnotu podľa výpočtu.

```
>>> a = 5  
>>> a = a+1  
>>> a  
6  
>>>
```

V samotnom programe (programovací režim) nestačí napísať len samotné meno premennej. Python po spustení programu **nebude** vedieť, čo má robiť s premennou. Na rozdiel od príkazového režimu musíme na vypísanie nejakej hodnoty do okna shell použiť príkaz `print()`. Napríklad: `print(a)` vypíše hodnotu premennej `a`. Pomocou príkazu `print()` môžeme vypisovať aj rôzne výpočty napr.: `print(a*3+5), print(6+2*4)` alebo aj text (podobne ako v príkaze `create_text print('Ahoj')`).

Príkazom `random.choice()` môžeme zo zadanej postupnosti vyberať náhodný prvok. Tento príkaz môžeme použiť na vyžrebovanie náhodnej farby z vymenovaných farieb. Vyžrebovanou farbou vyplníme obdĺžnik z predchádzajúceho príkladu. Aby sme nemali veľmi dlhé zápisy príkazov, vyžrebovanú farbu si zapamätáme do premennej. Tu máme výsledný program:

```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()
x = random.randrange(200)
y = random.randrange(200)
velkost = random.randint(30, 100)
farba = random.choice(['red', 'yellow', 'blue'])
canvas.create_rectangle(x, y, x+velkost, y+velkost, fill=farba)
```

Úlohy:

- 2** Upravte program tak, aby sme vo štvorci videli aj vyžrebovanú veľkosť štvorca.
- 3** Vytvorte program, ktorý bude kresliť náhodnú čiaru s náhodnou hrúbkou. Informáciu o dĺžke čiary a jej hrúbke vypíše aj do shellu.
- 4** Vytvorte program, ktorý bude na náhodnom mieste kresliť vodorovnú čiaru s dĺžkou 50 bodov. Informáciu o mieste kreslenia vypíše aj do shellu.

4.3 Kreslenie obrázkov na náhodnom mieste

Doposiaľ sme kreslili obrázky na pevne určenom mieste. Všetky zadané súradnice boli konkrétnie čísla. Na náhodnom mieste vieme kresliť jednoduché útvary. Na náhodnom mieste môžeme kresliť aj komplikovanejšie obrázky, ak si premyslíme relatívne súradnice bodov. To znamená, že si v obrázku určíme jeden z bodov ako základný a všetky ďalšie súradnice určujeme relatívne od tohto základného bodu. To, či je ďalší bod vľavo alebo vpravo od tohto bodu, znamená, či je x-ová súradnica menšia alebo väčšia. Čiže či od súradnice základného bodu niečo odčítame alebo k nemu niečo pričítame. A podobne je to aj na y-ovej súradnici.

V programe si môžeme písť aj rôzne poznámky, ktoré si Python nebude všímať. Takému textu hovoríme komentáre. Komentár vždy začína znakom `#` a končí na konci riadku. Niekedy pri hľadaní chyby potrebujeme, aby program niektoré riadky nevykonával. Vtedy tiež môžeme na začiatku týchto riadkov použiť `#.`. Ak je riadkov viac, označíme ich a v menu vyberieme Format / Comment Out Region, čo doplní na riadky `##`.

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(100, 100, 160, 200, fill='skyblue') #telo
canvas.create_rectangle(110, 50, 150, 100, fill='skyblue') #hlava
#oci
canvas.create_rectangle(115, 60, 125, 70, fill='yellow')
canvas.create_rectangle(135, 60, 145, 70, fill='yellow')

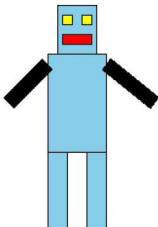
canvas.create_rectangle(115, 80, 145, 90, fill='red') #usta
#nohy
```

```

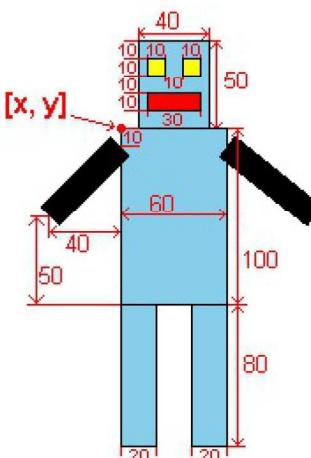
canvas.create_rectangle(100, 200, 120, 280, fill='skyblue')
canvas.create_rectangle(140, 200, 160, 280, fill='skyblue')
#ruky
canvas.create_line(100,110,60,150, width=15)
canvas.create_line(160,110,210,150, width=15)

```

Tento program nakreslí tohto robota.



Ak by sme chceli rovnakého robota kresliť na rôznych miestach, určíme si jeden bod ako základný. Napríklad si môžeme ako základný bod určiť ľavý horný bod tela obdĺžnika, ten si všeobecne označíme ako bod so súradnicou $[x, y]$ a všetky ďalšie body budú mať súradnicu podľa toho, ako ďaleko sa nachádzajú od tohto bodu. Na obrázku vidíme označený základný bod a veľkosti jednotlivých častí obrázku.



Obdĺžnik trupu robota môžeme určiť pomocou základného bodu a pravého dolného bodu obdĺžnika trupu. Tento bod je na x-ovej osi o 60 bodov vpravo (pretože obdĺžnik má šírku 60 bodov) a na y-ovej osi o 100 bodov dole (pretože má určenú výšku 100 bodov). Kedže základný bod má všeobecnú súradnicu $[x, y]$, pravý dolný roh trupu bude mať súradnicu $[x+60, y+100]$. Trup teda nakreslíme príkazom

```
canvas.create_rectangle(x, y, x+60, y+100).
```

Obdĺžnik hlavy môžeme nakresliť pomocou ľavého horného bodu hlavy a pravého dolného bodu hlavy. Ľavý horný bod hlavy je vzdialenosť na x-ovej súradnici o 10 bodov vpravo a na y-ovej súradnici o 50 bodov hore od základného bodu. Je teda na súradnici $[x+10, y-50]$. Pravý dolný bod hlavy je vzdialenosť na x-ovej osi o 50 bodov vpravo a y-ovú súradnicu má rovnakú ako základný bod, jeho súradnice sú teda $[x+50, y]$ a hlavu nakreslíme príkazom `canvas.create_rectangle(x+10, y-50, x+50, y)`.

Celý program, ktorý kreslí tohto robota na relatívnom mieste, môžeme zapísť takto:

```

import tkinter
canvas = tkinter.Canvas()
canvas.pack()
x = 100
y = 100
#telo
canvas.create_rectangle(x, y, x+60, y+100, fill='skyblue')
#hlava
canvas.create_rectangle(x+10, y-50, x+50, y, fill='skyblue')

```

```

#oci
canvas.create_rectangle(x+15, y-40, x+25, y-30, fill='yellow')
canvas.create_rectangle(x+35, y-40, x+45, y-30, fill='yellow')
#usta
canvas.create_rectangle(x+15, y-20, x+45, y-10, fill='red')
#nohy
canvas.create_rectangle(x, y+100, x+20, y+180, fill='skyblue')
canvas.create_rectangle(x+40, y+100, x+60, y+180, fill='skyblue')
#ruky
canvas.create_line(x, y+10, x-40, y+50, width=15)
canvas.create_line(x+60, y+10, x+110, y+50, width=15)

```

Ak nastavíme hodnotu premennej x a y na náhodnú hodnotu, bude sa robot kresliť na náhodnom mieste.

Úlohy:

- 5** Dotvorte robota tak, aby mal nakreslené uši, antény na hlave a na trupe nakreslené meno.
- 6** Nakreslite túto značku na náhodnom mieste tak, aby maximálna povolená rýchlosť bola náhodná.

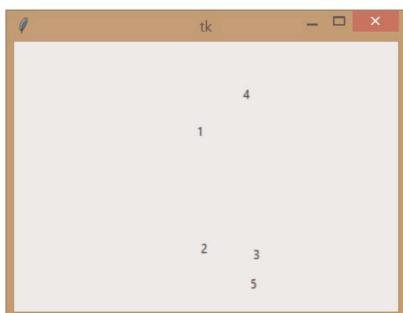


5 Opakovanie časti programu - for cyklus

5.1 Opakujeme vykonávanie príkazov

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_text(randrange(400), randrange(300), text= '1')
canvas.create_text(randrange(400), randrange(300), text= '2')
canvas.create_text(randrange(400), randrange(300), text= '3')
canvas.create_text(randrange(400), randrange(300), text= '4')
canvas.create_text(randrange(400), randrange(300), text= '5')
```

Všimnite si iný spôsob importovania knižnice `random`. Keď modul `random` importujeme zápisom `from random import *`, nemusíme pri žrebovaní písť celý zápis `random.randrange(400)`, ale stačí zapísat iba `randrange(400)`. Tento zápis znamená: z modulu `random` importuj všetko.

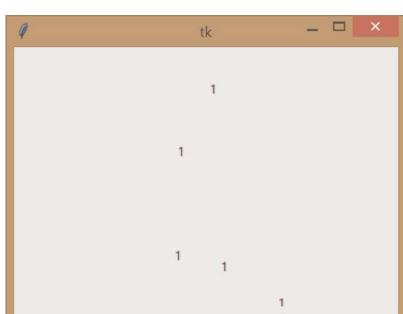


Tento program postupne vypíše čísla 1, 2, 3, 4 a 5 na náhodnom mieste grafickej plochy. Ak by sme chceli vypísať čísla až do 10, stačí ešte viackrát skopírovať príkaz na vypísanie textu a postupne zmeniť vypisované čísla na 6, 7, 8, 9 a 10. Predstavme si, že by sme takto chceli napísat až 50 čísel, to už by bol nás program veľmi dlhý a neustále by sme kopírovali príkazy. Je tu aj jednoduchšie riešenie. Ak chceme viackrát opakovať nejaký príkaz alebo postupnosť príkazov, môžeme použiť `for` cyklus. Program zapíšeme takto:

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

for i in range(1,6):
    canvas.create_text(randrange(400), randrange(300), text='1')
```

Teraz bude program päťkrát opakovať príkaz `create_text` a zatiaľ zakaždým vypíše 1. Zápisom `for i in range(1, 6):` sme zapísali, že nasledujúca časť programu sa bude opakovať a postupne si budeme v premennej `i` pamätať čísla uvedené v zátvorke, čiže 1 až 5. Aj keď je tam zapísané 1 až 6, pre hodnotu 6 sa už telo cyklu nezopakuje. Takto bude vyzeráť naše okno:



Ked' nahradíme zápis `text='1'` zápisom `text=i`, bude sa nám namiesto čísla 1 vypisovať hodnota v premennej `i`, čiže postupne čísla 1 až 5.

```
for i in range(10,16):
    canvas.create_text(randrange(400), randrange(300), text=i)
```

Otázky:

1. Ktoré čísla bude vypisovať vyššie uvedený program, v ktorom sme zmenili rozsah na `range(10, 16)`?
2. Čo bude robiť program, ak namiesto `text=i` napíšeme `text='1'`?
3. Zistite, ako treba upraviť rozsah pre predchádzajúci program, aby vypisoval:
 - a) čísla od 15 do 19 vrátane,
 - b) štyri čísla od čísla 23.
4. Zistite, ktoré čísla vypíše program, ak mu zadáme takýto rozsah:
 - a) `range(50, 60)`
 - b) `range(15, 16)`
 - c) `range(10, 10)`
 - d) `range(15, 10)`
 - e) `range(5)`
 - f) `range(3)`
 - g) `range(-10, -5)`
 - h) `range(-10, 3)`
5. Čo bude robiť tento program?

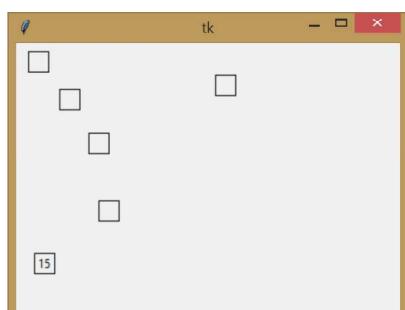
```
for i in range(20,25):
    print(i*10)
```

Porovnajte, v čom sa líšia nasledujúce dva programy:

```
for i in range(10,16):
    x = randint(0, 300)
    y = randint(0, 300)
    canvas.create_rectangle(x-10, y-10, x+10, y+10)
    canvas.create_text(x, y, text=i)
```

```
for i in range(10,16):
    x = randint(0, 300)
    y = randint(0, 300)
    canvas.create_rectangle(x-10, y-10, x+10, y+10)
    canvas.create_text(x, y, text=i)
```

Pri zápise for cyklu si musíme dávať veľký pozor na odsunutie začiatku riadku – medzery na začiatku. Vo for cykle sa opakujú len príkazy, ktoré sú rovnako odsunuté. Ak je niektorý riadok s menším počtom medzier, alebo text začína hned' na začiatku riadku, tento príkaz už program zaraduje mimo cyklus, a teda sa nebude opakovať. Druhý program nám už neopakuje posledný riadok. Počítač rozumie zápisu tak, že tento riadok je už mimo cyklus. Preto sa nám vykreslí päť štvorčekov, cyklus skončí a až potom sa raz vykoná príkaz `create_text` a vypíše nám aktuálnu hodnotu premennej `i`, teda číslo 15.



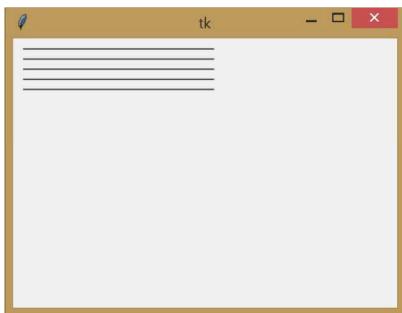
5.2 For cyklom kreslíme obrázky s pravidelnosťou

Zistite, čo nakreslí tento program:

```
y = 0
for i in range(1,6):
    y = y + 10
    canvas.create_line(10, y, 200, y)
```

Premennej `y` ešte pred cyklom nastavíme hodnotu na nulu. Potom sa bude opakovať cyklus postupne pre `i` s hodnotami 1, 2, 3, 4 a 5. Zakaždým sa v programe zvýší hodnota premennej `y` o desať, čiže v `y` budú postupne hodnoty 10, 20, 30, 40, 50 a pre tieto hodnoty sa zakaždým nakreslí vodorovná čiara. Táto čiara má rovnakú y-ovú súradnicu, ako je práve hodnota premennej `y`, a čiara začína na x-ovej súradnici 10 a končí na x-vej súradnici 200. Ak si za premenné postupne dosadzujeme hodnoty, ktoré budú mať, v skutočnosti sa nakreslia čiary s týmito hodnotami súradníc.

```
canvas.create_line(10, 10, 200, 10)
canvas.create_line(10, 20, 200, 20)
canvas.create_line(10, 30, 200, 30)
canvas.create_line(10, 40, 200, 40)
canvas.create_line(10, 50, 200, 50)
```



Rovnaký efekt – vykreslenie rovnakej notovej osnovy – získame aj nasledujúcimi dvomi zápismi. Porozmýšľajte a zdôvodnite, prečo aj tieto programy nakreslia to isté.

```
for i in range(1,6):
    canvas.create_line(10, i*10, 200, i*10)
```

```
for i in range(10,60, 10):
    canvas.create_line(10, i, 200, i)
```

V tomto programe sme v zápisе `range` použili tri čísla. Tretím číslom určíme, akým krokom sa má hodnota premennej `i` meniť. Ak ju neuvedieme ako doposiaľ, automaticky je krok nastavený na 1.

For cyklus opakuje príkazy veľmi rýchlo, my teda vidíme hned' výsledok. Ak by sme chceli vykreslovanie spomaliť, môžeme v cykle použiť príkaz `canvas.after(1000)`. Tento príkaz pozdrží vykonávanie programu na čas uvedený v zátvorke. Čas uvádzame v milisekundách, čiže `canvas.after(1000)` čaká 1 sekundu. K tomuto príkazu musíme ešte použiť príkaz `canvas.update()`, ktorý zabezpečí, aby sa aj medzi čakaniami aktualizovalo grafické plátno. Ak by sme ho nepoužili, program by pracoval pomalšie (s čakaním) a grafické plátno by sa nám prekreslilo až po skončení for cyklu. Hotový program zapíšeme takto:

```
for i in range(1,6):
    canvas.create_line(10, i*10, 200, i*10)
    canvas.update()
    canvas.after(1000)
```

Do programu doplníme ešte príkaz `print()`, aby sme v príkazovom režime (v okne shell) videli jednotlivé hodnoty `y`-ových súradníc čiar.

```

for i in range(1,6):
    print(i*10)
    canvas.create_line(10, i*10, 200, i*10)
    canvas.update()
    canvas.after(1000)

```

Úlohy:

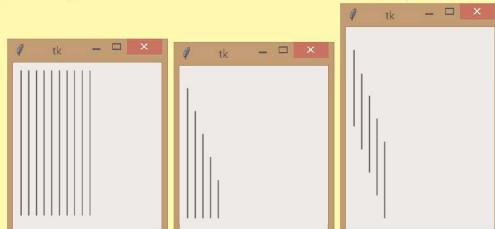
1

Experimentujte s pôvodným programom a skúste nakresliť tieto obrázky:



2

For cyklom nakreslite tieto obrázky:



3

Skúste odhadnúť, čo nakreslí nasledujúci program:

```

x = 0
for i in range(1,11):
    x = x+20
    canvas.create_rectangle(x, 10, x+15, 20)

```

Experimentujte s číslami v programe a zistite, ako zmenia nakreslený obrázok a pozorovanie zdôvodnite.

Čo sa stane, ak v riadku `x = 0` zadáme iné číslo?

Čo sa stane, ak v riadku `x = x+20` zadáme namiesto 20 iné číslo?

Čo sa stane, ak v príkaze `rectangle` zmeníme 15 na iné číslo?

4

Skúste vymyslieť iný zápis programu, ktorý nakreslí rovnaký obrázok ako tento program:

```

x = 0
for i in range(1,11):
    x = x + 20
    canvas.create_rectangle(x, 10, x+15, 20)

```

5

Adam chcel nakresliť takýto obrázok:



a vytvoril tento program:

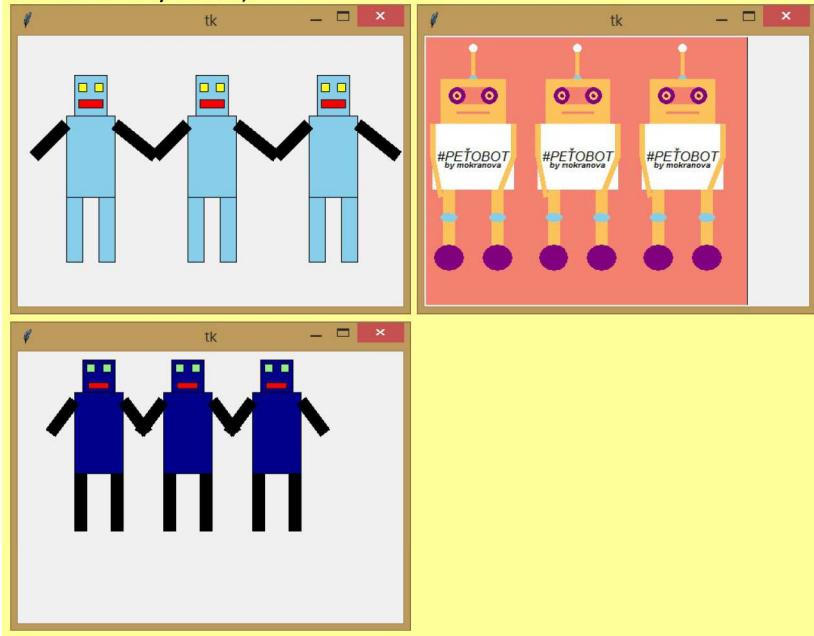
```

for i in range(10, 16):
    y = 0
    canvas.create_rectangle(100, y, 200, y+10)

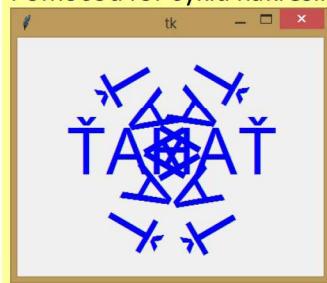
```

Nájdite v tomto programe chybu a opravte ho tak, aby nakreslil obrázok, ktorý chcel Adam pôvodne nakresliť.

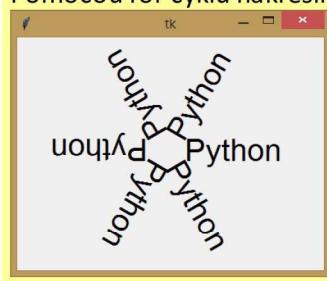
6 Pomocou for cyklu nakreslite vedľa seba tri roboty (z predchádzajúcej kapitoly alebo vlastný modifikovaný robot).



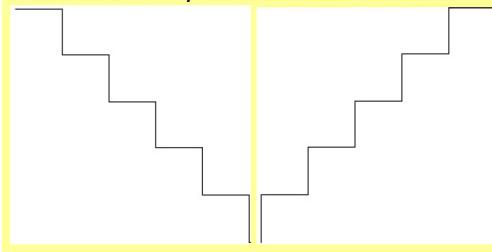
7 Pomocou for cyklu nakreslite otočený nápis ŤAHAŤ:



8 Pomocou for cyklu nakreslite otočený nápis Python:



9 Pomocou for cyklu nakreslite schodisko:



10 Pomocou for cyklu napište do okna shell:
a) všetky párne čísla od 20 do 40,
b) všetky nepárne čísla od 20 do 40.

11

Pomocou for cyklu napište do okna shell tieto čísla:

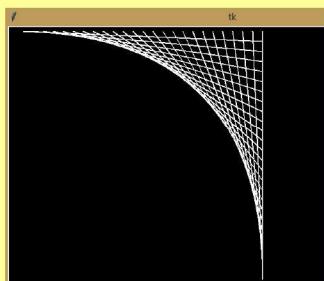
- a) 1, 2, 4, 8, 16, 32, 64, 128, 256, 512,
- b) 3, 2, 1, 0, -1, -2, -3.

12

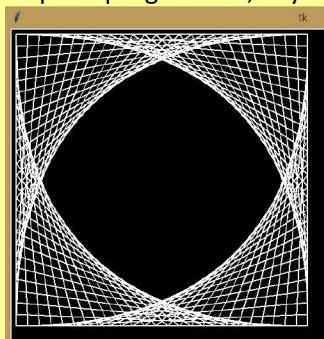
Tento program nakreslí obrázok v ukážke:

```
from random import *
import tkinter
canvas=tkinter.Canvas(bg='black',width=1000,height=800)
canvas.pack()

x=10
y=10
for i in range(1,26):
    x=x+20
    y=y+20
    canvas.create_line(x,10,510,y,fill='white',width=2)
    canvas.update()
    canvas.after(100)
```



Doplňte program tak, aby nakreslil aj zvyšné časti tohto celého obrázku:



13

Nakreslite tieto vzory (bez použitia príkazu create_text):

- a) 
- b) 

14

Na olympiáde v roku 2014 mali naši športovci na oblečení vzory z čičmanských dreveníc. Nájdite na internete takéto vzory a niektorý z nich nakreslite pomocou for cyklu.

6 Vytvárame podprogramy

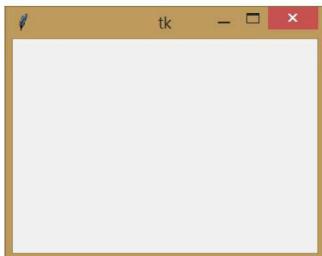
Pri kreslení komplikovanejších obrázkov sa nás program stáva neprehľadným, pretože obsahuje veľa príkazov. Sprehľadniť program nám pomáha používanie komentárov. Ďalší spôsob je použitie podprogramu. Je to menší kúsok programu, ktorý pomenujeme, a keď ho chceme použiť, len napišeme v programe meno tohto podprogramu. V podstate si takto vytvoríme vlastné príkazy, ktoré môžeme používať v našom programe. Odborne sa takýmto podprogramom v Pythone hovorí funkcie (v iných programovacích jazykoch sa môžete stretnúť aj s názvom procedúry alebo funkcie, alebo oboje). My teraz nakreslíme dom a vytvoríme k nemu aj funkciu (príkaz) dom.

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
  
canvas.create_rectangle(100, 100, 150, 150)  
canvas.create_line(100, 100, 125, 50, 150, 100)
```

Program nakreslí z obdĺžnika a lomenej čiary dom so strechou. V ďalšej ukážke sme definovali našu vlastnú funkciu s názvom `dom`. Funkciu definujeme slovíčkom `def`, za ktorým nasleduje názov funkcie, potom zátvorky a na konci dvojbodka (rovnako ako vo `for` cykle).

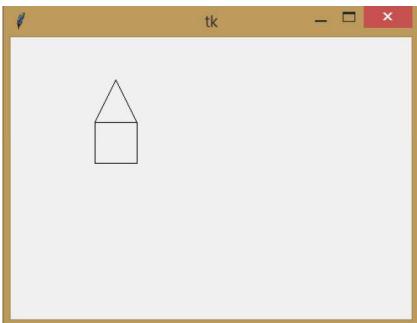
```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
  
def dom():  
    canvas.create_rectangle(100, 100, 150, 150)  
    canvas.create_line(100, 100, 125, 50, 150, 100)
```

Odsunutie príkazov od ľavého okraja pomáha Pythonu pochopiť, čo všetko patrí k našej funkcií. Rovnako sme odsunutie používali pri zápisе `for` cyklu. Tento program po spustení ešte nebude nič robiť, lebo my sme len definovali, čo bude vykonávať naša funkcia (náš príkaz), ale ešte sme ju nikde nepoužili (nespustili, nevykonali).



Ak chceme vykonať túto funkciu, napišeme v programe `dom()` (na mieste programu, kde sa má funkcia vykonať).

```
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
  
def dom():  
    canvas.create_rectangle(100, 100, 150, 150)  
    canvas.create_line(100, 100, 125, 50, 150, 100)  
  
dom()
```



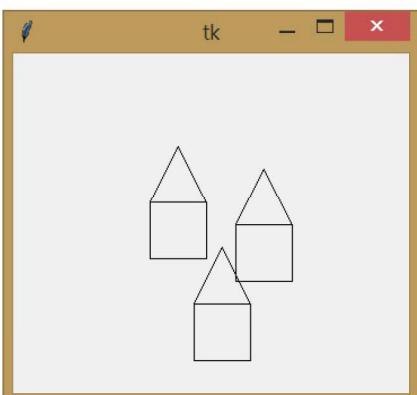
Tento program už aj spustí funkciu `dom()`. Zatiaľ veľmi nevidíme význam definovania vlastného príkazu, ale zmeňme funkciu `dom` tak, aby kreslila dom na náhodnom mieste.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def dom():
    x = randint(100, 300)
    y = randint(100, 300)
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)

dom()
dom()
dom()
```

V tomto programe už vykonáme funkciu `dom()` trikrát, a teda sa nám nakreslia tri domy na náhodnom mieste.

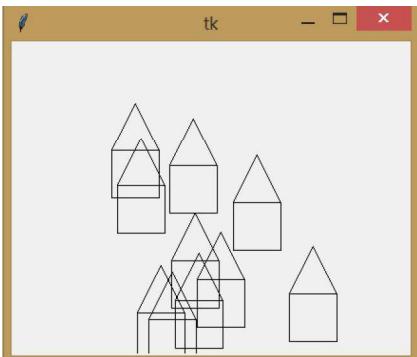


Ak by sme použili `for` cyklus, môžeme ľahko nakresliť aj 10 domov.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def dom():
    x = randint(100, 300)
    y = randint(100, 300)
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)

for i in range(1, 11):
    dom()
```

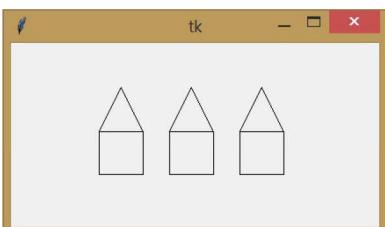


Funkcia sa dá upraviť aj tak, aby sme jej mohli pri použití zadať miesto, kde chceme nakresliť dom. Podobne ako pri používaní funkcie `create_rectangle` číslami v zátvorke ovplyvňujeme miesto nakreslenia. Takýmto číslam (alebo aj iným údajom), ktoré ovplyvňujú funkciu, hovoríme **parametre**. Pri definovaní funkcie s parametrom uvedieme do zátvorky názvy jednotlivých parametrov oddelené čiarkou (ak ich chceme mať viac ako jeden). Tieto názvy môžeme používať v našej funkcií (v podstate sú to akési premenné, ktoré sa dajú použiť len v tejto funkcií) a majú tú hodnotu, ktorú sme zadali pri použití funkcie.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def dom(x, y):
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)

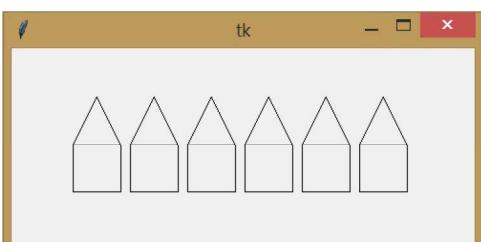
dom(100, 100)
dom(180, 100)
dom(260, 100)
```



```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def dom(x, y):
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)

for i in range(1, 7):
    dom(i*60, 100)
```



Úlohy:

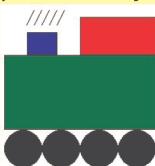
1

Vytvorte funkciu vagón, ktorá na zadanom mieste nakreslí vagón. Funkciu využite vo for cykle, aby ste nakreslili vlak zložený z vagónov.



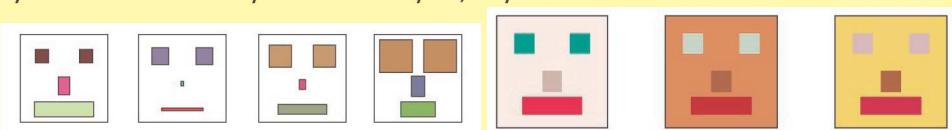
2

Vytvorte funkciu, ktorá na zadanom mieste nakreslí rušeň. Nakreslite rušeň na začiatok vlaku v predchádzajúcej úlohe.



3

Vytvorte funkciu, ktorá na zadanom mieste nakreslí tvár štvorcového robota, ktorý má oči, nos a ústa náhodnej veľkosti a náhodnej farby (zo zoznamu vymenovaných farieb) a zároveň je jeho tvár symetrická. Funkciu využite vo for cykle, aby ste nakreslili vedľa seba niekoľko tvári.



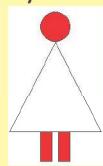
4

Vytvorte funkciu auto, ktorá na zadanom mieste nakreslí takéto auto:



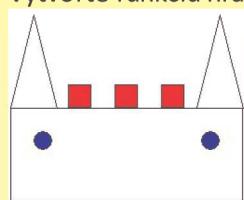
5

Vytvorte funkciu postavička, ktorá na zadanom mieste nakreslí postavičku:



6

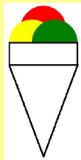
Vytvorte funkciu hrad, ktorá na zadanom mieste nakreslí hrad:



7 Úlohy na opakovanie I

1

Vytvorte program, ktorý nakreslí na obrazovku kornútok so zmrzlinou. V kornútku budú tri kopčeky zmrzliny – jahodová, kiwi a banánová. Kopčeky vyfarbrite príslušnou farbou.



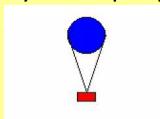
2

Vytvorte podprogram, ktorý na zadanom mieste nakreslí strom s čerešňou:



3

Vytvorte podprogram, ktorý na zadanom mieste nakreslí balón s červeným košom:



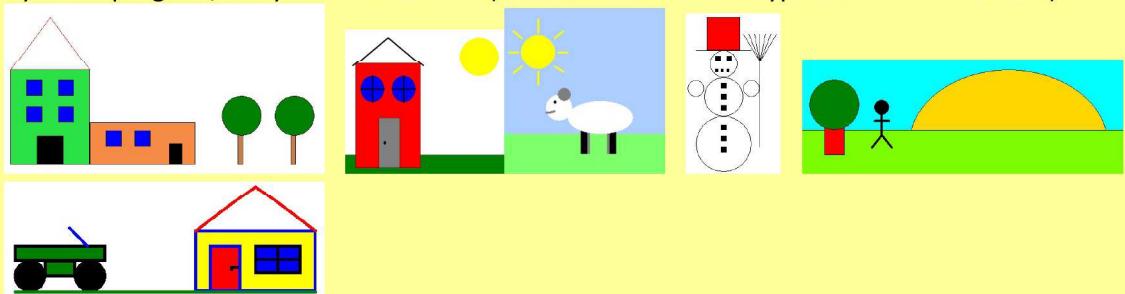
4

Vytvorte podprogram, ktorý nakreslí náhodný čiarový kód a do okna shell vypíše hrúbky jednotlivých čiar:



5

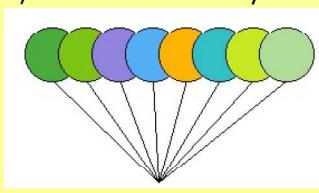
Vytvorte program, ktorý nakreslí obrázok (niektoré časti môžete vypínať náhodnou farbou):



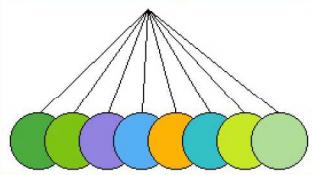
6

Nakreslite:

a) osem rôznofarebných balónov (farby sa môžu aj opakovať),

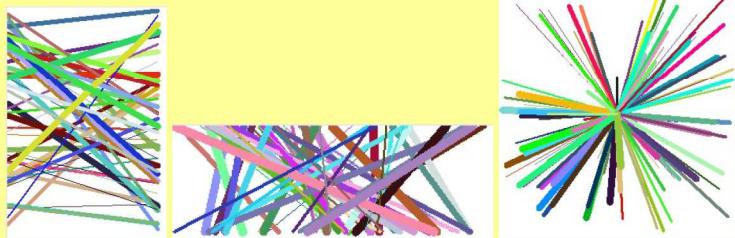


b) rôznofarebný kolotoč (farby sa môžu aj opakovať).



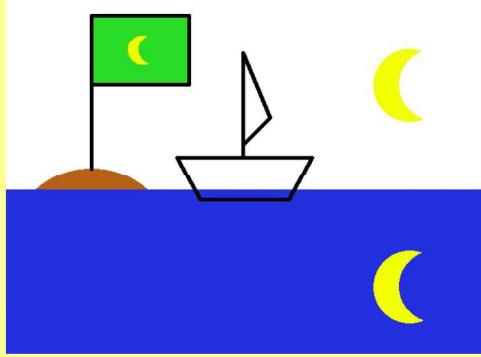
7

Vytvorte program, ktorý nakreslí nasledovné obrázky:



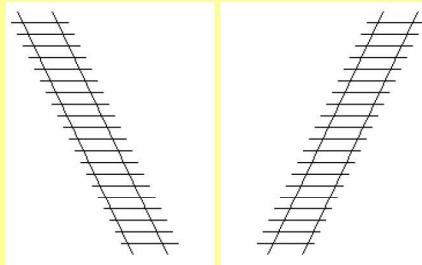
8

Vytvorte program, ktorý nakreslí obrázok (Ostrov s vlajkou): Úlohu naprogramujte tak, aby mesiac bol nakreslený v náhodnej výške nad hladinou a jeho odraz na hladine bol symetrický.



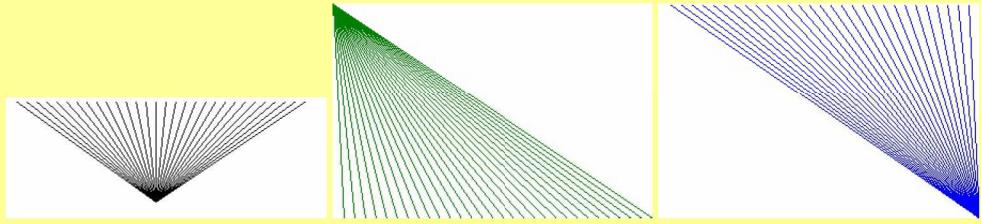
9

Nakreslite rebrík:



10

Nakreslite nasledovné obrazce:

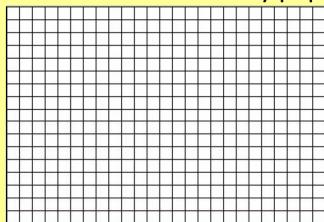


11

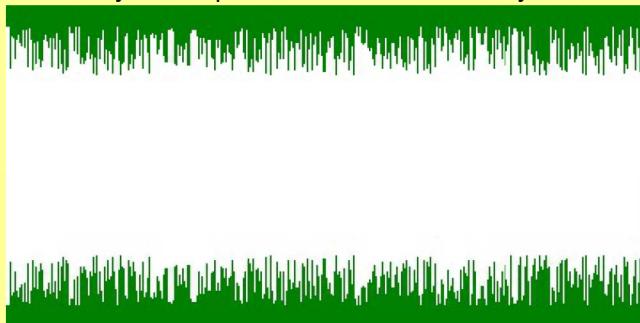
Vytvorte program, ktorý na obrazovku nakreslí 20 kružníc. Stredy kružníc sa nachádzajú na vodorovnej priamke. Farba každej kružnice je náhodná. Polomer kružnice je náhodný v rozsahu od 50 do 100 bodov. Šírka pera, ktorou sú kružnice nakreslené, je 3.

**12**

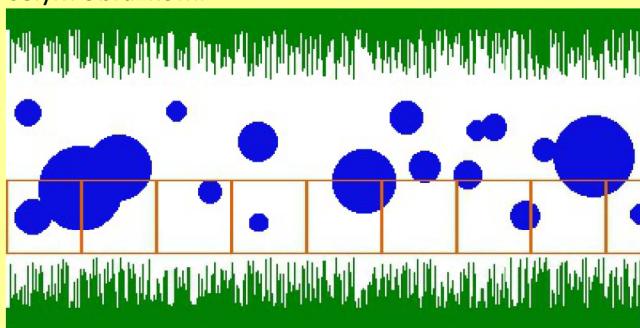
Nakreslite štvorčekový papier:

**13**

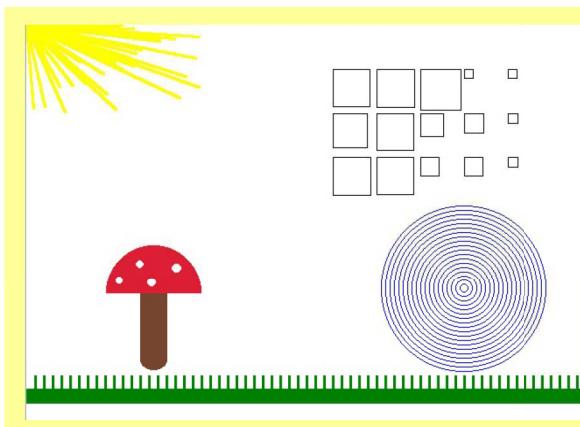
Vytvorte program, ktorý nakreslí jaskyňu s kvapľami. Šírka každého kvapľa je jeden bod a dĺžka je náhodná, no minimálne 20 bodov a maximálne 70 bodov. Ako vidieť na obrázku, v našej jaskyni sú kvaple symetrické podľa vodorovnej osi, ktorá prechádza stredom jaskyne. Kvaple sa nachádzajú tesne pri hornom a dolnom okraji obrázku a zaberajú šírku celého obrázku.

**14**

Doplňte do programu podprogram, ktorý nakreslí v jaskyni mláčky vody. Každá mláka má tvar kruhu, je nakreslená modrou farbou a polomer má v rozsahu od 20 do 100 bodov. Žiadna z mláčok sa nemôže dotýkať kvapľov. Počet mláčok je náhodný. Aby sme mohli jaskyňou prechádzať, dokreslite do nej vodorovný rebrík. Rebrík je hnedý a počet stupienkov je 10. Rebrík prechádza celým obrázkom.



Záhradka – úlohy 15 až 19



15

Vytvorte program, ktorý nakreslí muchotrávku:



16

Doplňte do programu funkciu, ktorá pod muchotrávkou nakreslí zelenou farbou trávu:



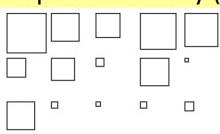
17

Doplňte do programu príkazy, ktoré nakreslia 40 náhodných slnečných lúčov. Slnečné lúče sú žltej farby a vychádzajú z ľavého horného rohu obrazovky, nekreslia sa však cez celý canvas, ale iba do vami navrhnutej vzdialenosť (viď obrázok).



18

Doplňte do programu funkciu, ktorá zakreslí do pravej hornej časti obrázku mapu zastavaných pozemkov. Každý pozemok má tvar štvorca a veľkosť strany každého pozemku je náhodná, no maximálne 50 bodov. Mapa má tvar obdĺžnika. V jednom riadku mapy je päť pozemkov a na mape sú tri riadky (viď obrázok).



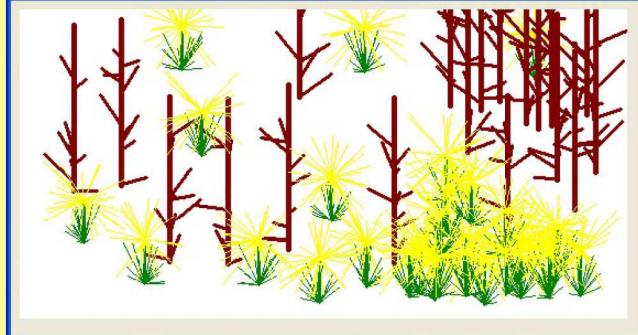
19

Doplňte do programu funkciu, ktorá nakreslí do obrázku modré jazierko. Jazierko je nakreslené dvadsiatimi sústrednými kružnicami modrej farby (viď výsledný obrázok programu v úvode tejto sady úloh).



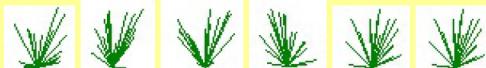
Krajinka - úlohy 20 až 21

Vytvorte program Krajinka podľa zadania v úlohách 20 až 21.

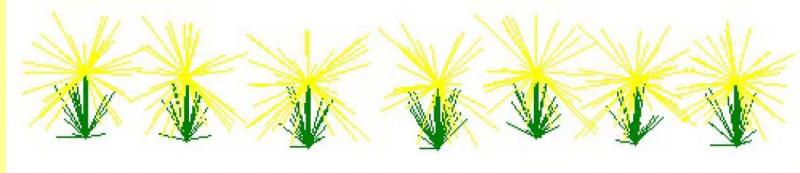


20

Na náhodných miestach sa kreslí kvet. Květ sa skladá z troch častí (trs trávy, stonka a kvet). Trs trávy obsahuje dvadsať náhodných stebiel. Každé steblo je od koreňa vzdialé na x-ovej súradnici maximálne 20 bodov vľavo alebo vpravo. Na y-ovej súradnici je od koreňa vzdialé maximálne 40 bodov a vždy je umiestnené smerom hore od koreňa. Ukážka náhodných trsov:

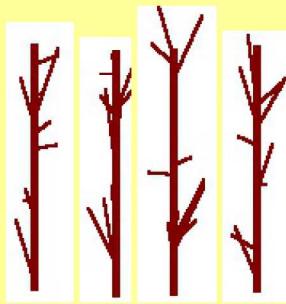


Stonka má dĺžku 50 bodov. Květ je žltý a obsahuje 40 čiar, ktoré vychádzajú zo stredu kvetu a sú náhodné. Každá čiara kvetu je v smere x-ovej a y-ovej osi najďalej 40 bodov od stredu kvetu. Ukážka náhodných kvetov:



21

Na náhodných miestach sa kreslí strom. Kmeň stromu je vysoký 150 bodov a má hrúbkou 5 bodov. Strom má 10 náhodných konárov s hrúbkou 2 body. Všetky konáre smerujú hore. Ukážka náhodných stromov:



8 Klikanie myšou a ovládanie klávesnicou

8.1 Reakcia na ľavé tlačidlo myši

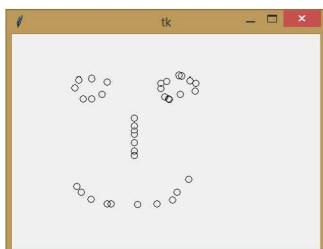
Pri používaní programov sme zvyknutí ovládať ich myšou kliknutím na rôzne prvky v programe. Pri interakcii používateľa s programom vznikajú takzvané udalosti, na ktoré môžeme v programe reagovať. Príkazom bind môžeme v programe nastaviť, na ktoré udalosti chceme reagovať a ktorú funkciu chceme v tejto situácii vykonáť. Udalosti majú presne definovaný názov. Napríklad udalosť kliknutia ľavým tlačidlom myši má názov <Button-1>. Teraz vytvoríme program, ktorý bude na mieste kliknutia ľavým tlačidlom myši kresliť krúžok.

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

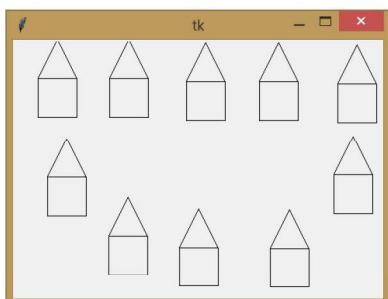
def kruzok(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_oval(x-5, y-5, x+5, y+5)

canvas.bind('<Button-1>', kruzok)
```

Vytvorili sme si novú funkciu `kruzok`. Príkazom `canvas.bind('<Button-1>', kruzok)` sme nastavili, že na udalosť <Button-1> (čiže kliknutie ľavým tlačidlom myši) budeme reagovať vykonaním funkcie `kruzok`. Udalosť automaticky posielá tejto funkcií aj informácie o súradničach myši a my sme si tieto získané informácie vo funkcií `kruzok` označili parametrom s názvom súradnice. Parameter `suradnice` obsahuje aj x-ovú aj y-ovú súradnicu. K jednotlivým súradničiam sa dostaneme tak, že zapíšeme `suradnice.x` alebo `suradnice.y`. Ako vidíte, v našej funkcií `kruzok` sme si tieto jednotlivé súradnice zapamätali do lokálnych premenných `x` a `y`, ktoré už ďalej bežne používame vo funkcií.



Nasledujúci program bude na mieste kliknutia kresliť domček.



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

def dom(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_rectangle(x, y, x+50, y+50)
```

```

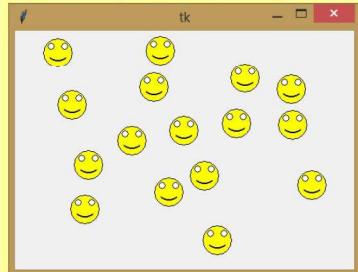
    canvas.create_line(x, y, x+25, y-50, x+50, y)
    canvas.bind('<Button-1>', dom)

```

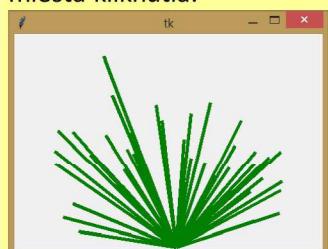
Úlohy:

1 Vytvorte program, v ktorom budeme kliknutím myši kresliť krížik. Samotný stred prekríženia bude presne v mieste kliknutia.

2 Vytvorte program, ktorý bude kresliť v mieste kliknutia smajlíka.

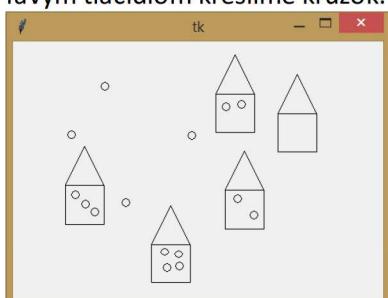


3 Vytvorte program, ktorý bude kresliť hrubé zelené čiary vychádzajúce zo stredu v spodnej časti okna do miesta kliknutia.



8.2 Reakcia na pravé tlačidlo myši

Pravé tlačidlo myši vyvolá udalosť s názvom '`<Button-3>`'. Ked' túto udalosť zviažeme príkazom `bind` s nejakou funkciou, môžeme na ňu reagovať. Nasledujúci program kreslí pri stlačení pravého tlačidla dom a ľavým tlačidlom kreslíme krúžok.



```

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

def kruzok(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.create_oval(x-5, y-5, x+5, y+5)

def dom(suradnice):
    x = suradnice.x

```

```

y = suradnice.y
canvas.create_rectangle(x, y, x+50, y+50)
canvas.create_line(x, y, x+25, y-50, x+50, y)

canvas.bind('<Button-1>', kruzok)
canvas.bind('<Button-3>', dom)

```

V programe môžeme reagovať aj na pohyb myši. Pohyb myši bez zatlačeného tlačidla vyvolá udalosť s názvom '`<Motion>`'. Pohyb myši so zatlačeným ľavým tlačidlom vyvolá udalosť s názvom '`<B1-Motion>`' a pravé tlačidlo s pohybom '`<B3-Motion>`'. Občas môžeme využiť aj udalosť '`<ButtonRelease-1>`' (uvolnenie zatlačeného ľavého tlačidla) alebo '`<Double-Button-1>`' (dvojklik).

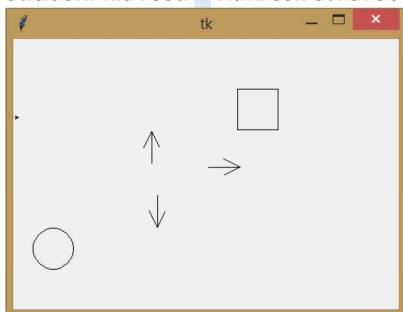
8.3 Reakcia na stláčanie klávesnice

Na udalosti stlačenia klávesu reagujeme príkazom `canvas.bind_all('kláves', funkcia)`. Kláves je konkrétny znak na klávesnici, napr. '`a`' alebo '`d`', alebo názov ovládacieho klávesu. Napríklad môžeme používať tieto klávesy:

- '`<Up>`' – šípka hore,
- '`<Down>`' – šípka dole,
- '`<Right>`' – šípka vpravo,
- '`<Left>`' – šípka vľavo,
- '`<space>`' – medzera,
- '`<Return>`' – enter (pozor, je aj udalosť '`<Enter>`', ale tá označuje niečo iné).

Parameter `funkcia` je názov funkcie, ktorá sa vykoná. Volaná funkcia však musí mať jeden parameter, napríklad `event` (udalosť), v ktorom sa funkcií pošlú podrobnejšie informácie o udalosti. Rovnako ako pri kliknutí myši aj pri stlačení klávesu sa posielajú v parametri súradnice aktuálnej pozície myši. Aj keď s týmto parametrom nepracujeme v našej funkcií, musíme ho zadať pri definovaní funkcie.

Nasledujúci príklad pri stlačení šípky vpravo, hore a dole nakreslí na náhodnom mieste prislúchajúcu šípkou a pri stlačení klávesu `s` nakreslí štvorec a pri stlačení klávesu `k` kružnicu.



```

import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def stvorec(event):
    x = randrange(300)
    y = randrange(250)
    canvas.create_rectangle(x, y, x+50, y+50)

def kruh(event):
    x = randrange(300)
    y = randrange(250)
    canvas.create_oval(x, y, x+50, y+50)

def sipka_hore(event):

```

```

x = randrange(300)
y = randrange(250)
canvas.create_line(x-10, y+20, x, y, x+10, y+20)
canvas.create_line(x, y, x, y+40)

def sipka_dole(event):
    x = randrange(300)
    y = randrange(250)
    canvas.create_line(x-10, y-20, x, y, x+10, y-20)
    canvas.create_line(x, y, x, y-40)

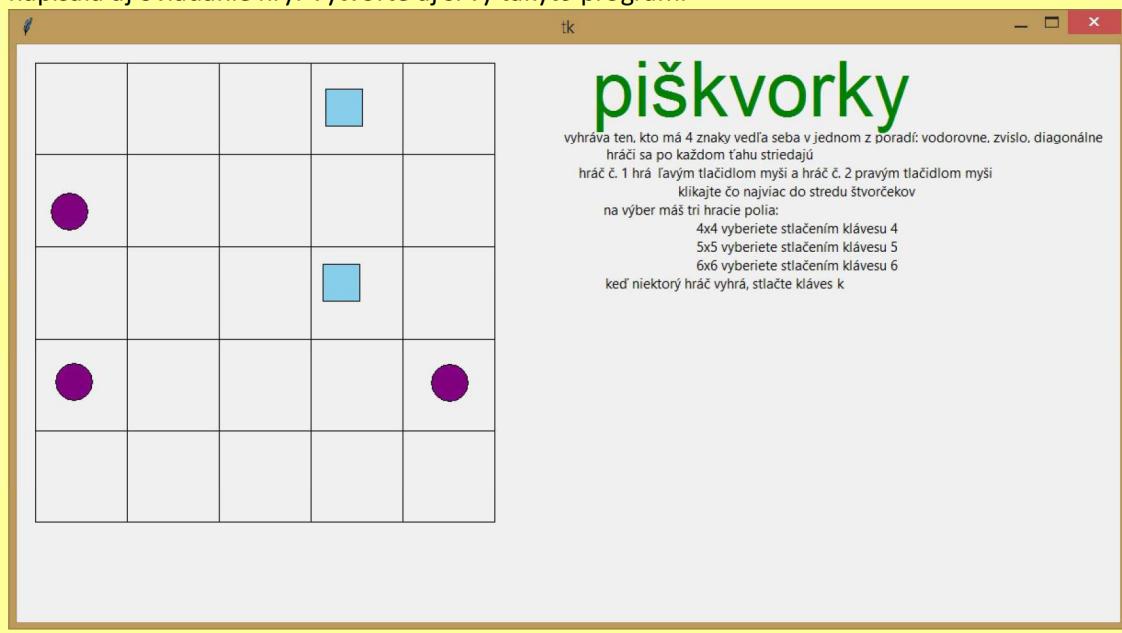
def sipka_vpravo(event):
    x = randrange(300)
    y = randrange(250)
    canvas.create_line(x-20, y-10, x, y, x-20, y+10)
    canvas.create_line(x, y, x-40, y)

canvas.bind_all('s', stvorec)
canvas.bind_all('k', kruh)
canvas.bind_all('<Up>', sipka_hore)
canvas.bind_all('<Down>', sipka_dole)
canvas.bind_all('<Right>', sipka_vpravo)

```

Úlohy:

- 4** Vymyslite a vytvorte program, ktorý reaguje na klávesnicu a myš. K jednotlivým reakciám vytvorte funkcie a vymyslite niečo zaujímavé.
- 5** Katka sa rozhodla, že si vytvorí program, v ktorom sa môže hrať piškvorky a nemusí používať papier. Zatiaľ vie naprogramovať len kreslenie plochy a útvarov. V budúcnosti si naprogramuje piškvorky, kde útvary "doskočia" presne do stredu polička a kde počítač bude kontrolovať výhru hráča. V programe napísala aj ovládanie hry. Vytvorte aj si vy takýto program.



Zatiaľ sme vytvárali programy, v ktorých sme každý zo stlačených klávesov zviazali s inou funkciou. Sú situácie, keď chceme udalosť stlačenia ktoréhoľvek klávesu zviazať s jednou funkciou. Takáto udalosť sa označuje názvom '`<Key>`'. V zavolanej funkcií, ktorá reaguje na túto udalosť, môžeme cez vstupný parameter a jeho zložku (atribút) `.keysum` zistiť aj konkrétny stlačený kláves. Nasledujúci program pri stlačení klávesu napíše tento kláves na náhodné miesto `canvas`-u a vypíše ho aj do okna shell.

```
import tkinter
```

```
from random import *
canvas = tkinter.Canvas(width= '300', height= '300')
canvas.pack()

def pis(event):
    print(event.keysym)
    x = randrange(300)
    y = randrange(300)
    canvas.create_text(x, y, text=event.keysym)

canvas.bind_all('<Key>', pis)
```

Úlohy:

6

V predchádzajúcej ukážke sme sa k informácii o stlačenom klávese dostali cez zložku (atribút) `.keysym` vstupného parametra, ktorý sme si pomenovali `event`. Čiže sme k nemu pristupovali s označením `event.keysym`. Zistite, čo sa zmení, keď budeme k nemu pristupovať cez zápis `event.char` alebo `event.keycode`. Skúste napísanie tieto informácie príkazom `print` aj do okna shell.

9 Podmienené príkazy

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def nahodny_stvorec():
    x = randrange(300)
    y = randrange(300)
    a = randint(20, 80)
    canvas.create_rectangle(x, y, x+a, y+a)

for i in range(1, 21):
    nahodny_stvorec()
    canvas.update()
    canvas.after(500)
```

Otázky:

- Čo bude robiť tento program?

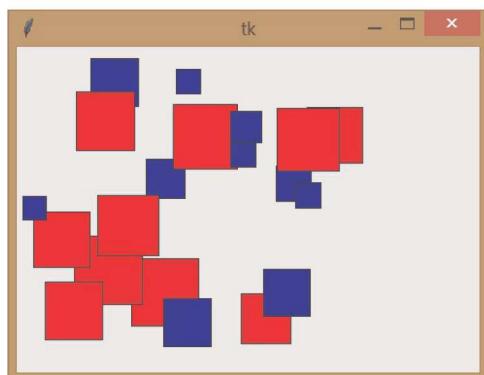
Predstavme si, že chceme upraviť program tak, aby väčšie štvorce kreslil červenou farbou a menšie modrou. V takýchto situáciách musíme v programe zapísť podmienku. Isté príkazy sa vykonajú iba pri splnenej podmienke a isté príkazy sa naopak vykonajú, keď podmienka nie je splnená. Napríklad: určili by sme si, že štvorce s veľkosťou nad 50 px budú červené a menšie budú modré. Nás podmienený príkaz by mohol znieť – ak veľkosť je väčšia ako 50, tak kresli červený štvorec, inak kresli modrý štvorec.

Zapišeme to takto:

```
if a>50:
    canvas.create_rectangle(x, y, x+a, y+a, fill='red')
else:
    canvas.create_rectangle(x, y, x+a, y+a, fill='blue')
```

Namiesto `a>50` môžu byť aj iné podmienky (podľa toho, čo chceme, aby bolo splnené). Môžeme používať aj takéto zápisy:

```
a<50      podmienka je splnená, ak a je menšie ako 50,
a<=50     podmienka je splnená, ak a je menšie alebo rovné ako 50,
a>=50     podmienka je splnená, ak a je väčšie alebo rovné ako 50,
a==50     podmienka je splnená, ak a je rovné 50,
a!=50     podmienka je splnená, ak a nerovná sa 50,
50 < a < 60  podmienka je splnená, ak a je medzi 50 a 60 (okrem 50 a 60)
50 <= a <= 60 podmienka je splnená, ak a je medzi 50 a 60 (vrátane 50 a 60)
```



A tu je celý program:

```

import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def nahodny_stvorec():
    x = randrange(300)
    y = randrange(300)
    a = randint(20,80)
    if a>50:
        canvas.create_rectangle(x, y, x+a, y+a, fill='red')
    else:
        canvas.create_rectangle(x, y, x+a, y+a, fill='blue')

for i in range(1,21):
    nahodny_stvorec()
    canvas.update()
    canvas.after(500)

```

Všimnite si, že za príkazom `if` a podmienkou nasleduje dvojbodka (rovnako ako vo `for` cykle alebo za definovaním funkcie). Dvojbodka nasleduje aj za príkazom `else`, za ktorým zapisujeme príkazy, ktoré sa majú vykonať v prípade, že podmienka **nie je** splnená. Aj v zápise vetvenia je dôležité odsunúť príkazy. Všetky odsunuté príkazy patria pod `if`, respektívne pod `else` vetvu. Ak by sme niektorý riadok neodsunuli, program by ich vykonával, ako keby neboli v `if` alebo `else`, resp. by ohlásil chybu odsadenia.

Upravme funkciu `nahodny_stvorec` takto:

```

def nahodny_stvorec():
    x = randrange(300)
    y = randrange(300)
    a = randint(20,80)
    if 40<a<60:
        canvas.create_rectangle(x, y, x+a, y+a, fill='red')
        canvas.create_text(x+10, y+10, text=a)
    else:
        canvas.create_rectangle(x, y, x+a, y+a, fill='blue')
        canvas.create_text(x+10, y+10, text=a)

```

Otzáky:

2. Čo bude robiť tento program?
3. Ako by sme mohli tento program zjednodušiť?

Úlohy:

- 1** Vytvorte program, v ktorom sa najprv v strede obrazovky nakreslí čiara. Ak klikneme myšou nad túto čiaru, bude program kresliť modrý štvorec, ak klikneme pod túto čiaru, bude program kresliť žltý kruh.
- 2** Vyskúšajte v predchádzajúcej úlohe vyniechať zápis s celou časťou `else`: aj príkazmi vo vetve `else`. Bude fungovať aj tento neúplný `if`?

V podmienenom príkaze môžeme zadávať aj viacero podmienok a spájame ich logickým operátorom `and` a `or`. Keď chceme mať splnené všetky podmienky súčasne, použijeme operátor `and`. Operátor `or` použijeme, keď stačí splnenie aspoň jednej z podmienok.

```

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

```

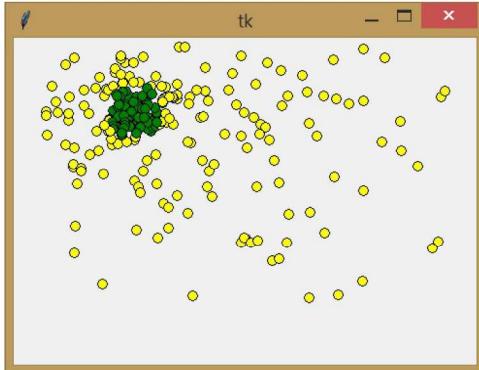
```

def kruzok(suradnice):
    x = suradnice.x
    y = suradnice.y
    if 100<x<150 and 50<y<100:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='green')
    else:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='yellow')

canvas.bind('<Button-1>', kruzok)

```

Program bude kresliť zelené krúžky iba v prípade, že x je medzi 100 a 150 a súčasne platí, že y je medzi 50 a 100. Ak budeme veľakrát klikáť na canvas, budeme vidieť, kde sa kreslia zelené a kde žlté krúžky.



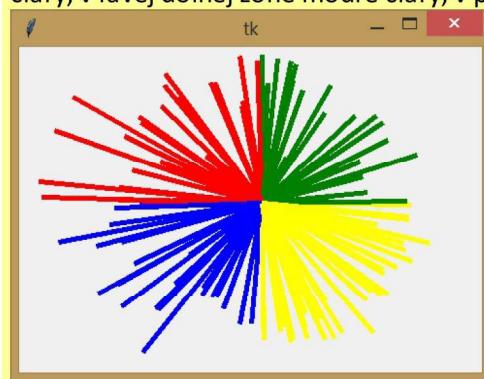
Otázky:

4. Čo bude robiť tento program, ak zmeníme operátor `and` na operátor `or`?
5. Čo bude robiť tento program, ak zmeníme podmienku takto: `x<150 or y>150`?

Úlohy:

3

Vytvorte program, ktorý zo stredu obrazovky nakreslí hrubú čiaru do miesta kliknutia myši. Čiaru kreslí farbou podľa miesta kliknutia. Obrazovka je rozdelená na štyri zóny. V ľavej hornej zóne kreslí červené čiary, v ľavej dolnej zóne modré čiary, v pravej hornej zóne zelené čiary a v pravej dolnej žlté čiary.



4

Vytvorte program, ktorý na mieste kliknutia myši kreslí farebné krúžky tak, že z nich vznikne trojpruhová vlajka niektoréj z krajín: Holandsko, Maďarsko, Litva, Nemecko, Rakúsko.

```

import tkinter
from random import *
canvas = tkinter.Canvas(width= '300', height= '300')
canvas.pack()

def pis(event):

```

```

x = randrange(300)
y = randrange(300)
pismeno = event.char
if pismeno=='a':
    canvas.create_text(x, y, text='ahoj')
if pismeno=='b':
    canvas.create_text(x, y, text='bye')
if pismeno=='c':
    canvas.create_text(x, y, text='čau')
if pismeno=='d':
    canvas.create_text(x, y, text='dobrý deň')
if pismeno=='n':
    canvas.create_text(x, y, text='nazdar')

canvas.bind_all('<Key>', pis)

```

Predchádzajúci program nám vypíše na náhodnom mieste obrazovky pozdrav podľa toho, ktorý kláves sme stlačili. V programe nemáme pozdrav na každé písmeno. Ak stlačíme písmeno, ku ktorému nemáme pozdrav, nič sa neudeje.

Otázky:

- Chceli sme, aby nám program v prípade, že k nejakému písmenu nemá pozdrav, napísal: "K tomuto písmenu nepoznám pozdrav." A tak sme program upravili takto:

```

import tkinter
from random import *
canvas = tkinter.Canvas(width= '300', height= '300')
canvas.pack()

def pis(event):
    x = randrange(300)
    y = randrange(300)
    pismeno = event.char
    if pismeno=='a':
        canvas.create_text(x, y, text='ahoj')
    if pismeno=='b':
        canvas.create_text(x, y, text='bye')
    if pismeno=='c':
        canvas.create_text(x, y, text='čau')
    if pismeno=='d':
        canvas.create_text(x, y, text='dobrý deň')
    if pismeno=='n':
        canvas.create_text(x, y, text='nazdar')
    else:
        canvas.create_text(x, y, text='K tomuto písmenu nemám pozdrav.')

canvas.bind_all('<Key>', pis)

```

Prečo nebude fungovať tento program podľa našej predstavy?

Aby program fungoval podľa našej predstavy, museli by sme ho zapísať približne v takomto znení: ak je stlačené písmeno a, napiš ahoj, v iných prípadoch zisti, či je stlačené písmeno b. Ak áno, napiš bye a v iných prípadoch (nebolo stlačené ani a ani b) napiš, že nemám pozdrav. V programe v prvom if-e vytvoríme vetvu else (pre všetky ostatné situácie) a v tejto vetve sa spýtame ďalším (vnoreným a odsunutým) if-om, či je stlačené b. V prípade, že nie je, vykoná sa else vetva tohto vnoreného if-u. Takto by sme sa mohli v každej ďalšej else vetve pýtať ďalším vnoreným if-om na ďalšie písmeno, a tak postupne vylúčiť všetky, na ktoré máme pozdrav. Pozrite si skrátenú ukážku programu:

```

if pismeno=='a':
    canvas.create_text(x, y, text='ahoj')

```

```

else:
    if pismeno=='b':
        canvas.create_text(x, y, text='bye')
    else:
        canvas.create_text(x, y, text='K tomuto písmenu nemám pozdrav.')

```

V takýchto situáciách je jednoduchšie spojiť vetvu `else` a nasledujúci vnorený `if` do skráteného zápisu `elif`, za ktorým rovno nasleduje podmienka, ktorá by bola v ďalšom vnorenom `if`-e. V nasledujúcej ukážke vidíme, ako elegantne môžeme zapísť celý program pomocou `elif` a tiež si uvedomíme, že teraz už funguje podľa našej pôvodnej predstavy.

```

import tkinter
from random import *
canvas = tkinter.Canvas(width= '300', height= '300')
canvas.pack()

def pis(event):
    x = randrange(300)
    y = randrange(300)
    pismeno = event.char
    if pismeno=='a':
        canvas.create_text(x, y, text='ahoj')
    elif pismeno=='b':
        canvas.create_text(x, y, text='bye')
    elif pismeno=='c':
        canvas.create_text(x, y, text='čau')
    elif pismeno=='d':
        canvas.create_text(x, y, text='dobrý deň')
    elif pismeno=='n':
        canvas.create_text(x, y, text='nazdar')
    else:
        canvas.create_text(x, y, text='K tomuto písmenu nemám pozdrav.')

canvas.bind_all('<Key>', pis)

```

Ked' už sme spomenuli elegantný zápis programu, môžeme ho ešte trochu vylepšíť. To, čo chceme napísať, si zapamätáme do premennej `oznam` a len v závere príkazom `create_text` napíšeme text, ktorý je v premennej `oznam`. Iste si spomíname, že to robíme podobne, ako ked' sme si pamäタali farbu, ktorou sme niečo kreslili.

```

import tkinter
from random import *
canvas = tkinter.Canvas(width= '300', height= '300')
canvas.pack()

def pis(event):
    x = randrange(300)
    y = randrange(300)
    pismeno = event.char
    if pismeno=='a':
        oznam='ahoj'
    elif pismeno=='b':
        oznam='bye'
    elif pismeno=='c':
        oznam='čau'
    elif pismeno=='d':
        oznam='dobrý deň'
    elif pismeno=='n':
        oznam='nazdar'
    else:
        oznam='K tomuto písmenu nemám pozdrav.'
    canvas.create_text(x, y, text=oznam)

canvas.bind_all('<Key>', pis)

```

Úlohy:

5 Doplňte do programu ďalšie pozdravy.

6 K jednotlivým pozdravom doplňte aj ďalšie nastavenia - napríklad farbu pozdravu alebo veľkosť písma.

10 Časovač

V predchádzajúcej kapitole sme na mieste kliknutia kreslili farebné krúžky podľa nejakej podmienky. Na nakreslenie väčšieho množstva krúžkov sme museli veľakrát kliknúť myšou. Toto zatiaľ vieme vyriešiť použitím for cyklu – naraz nakreslíme viacero krúžkov na náhodnej pozícii.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def kruzok():
    x = randrange(450)
    y = randrange(320)
    if 100<x<150 or 50<y<100:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='green')
    else:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='yellow')

for i in range(1, 1000):
    kruzok()
```



Už vieme upraviť for cyklus, aby vykonával jednotlivé opakovania postupne s nejakým časovým pozdržaním. Využívali sme na to príkaz `canvas.after()` a `canvas.update()`. Takto upravíme for cyklus na postupné kreslenie.

```
for i in range(1, 1000):
    kruzok()
    canvas.update()
    canvas.after(10)
```

For cyklus používame v situáciách, keď vopred vieme odhadnúť počet opakovaní cyklu. Aj preto sa mu zvykne hovoriť cyklus s pevným počtom opakovaní. Sú však situácie, keď potrebujeme opakovať nejakú činnosť v pravidelných časových intervaloch a vopred nepoznáme presný počet opakovaní. Na riešenie takéhoto problému môžeme použiť takzvaný časovač (timer), ktorý v pravidelných časových intervaloch opakuje zadanú postupnosť príkazov. Na to sa využíva príkaz `canvas.after(čas, funkcia)`, ktorému zadávame okrem času v milisekundách aj meno funkcie, ktorá sa má vykonať po pozdržaní programu.

Náš program po úprave s využitím časovača:

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def kruzok():
    x = randrange(450)
    y = randrange(320)
```

```

if 100<x<150 or 50<y<100:
    canvas.create_oval(x-5, y-5, x+5, y+5, fill='green')
else:
    canvas.create_oval(x-5, y-5, x+5, y+5, fill='yellow')
canvas.after(10, kruzok)

kruzok()

```

Po spustení programu sa raz zavolá a spustí funkcia `kruzok`, ktorá nakreslí jeden krúžok, a naplánuje ďalšie spustenie funkcie `kruzok()` o 10 milisekúnd. Takto sme zabezpečili, že program sa bude opakovať nekonečne dlho. Všimnite si, že na rozdiel od for cyklu tu nepotrebujeme použiť príkaz `canvas.update()` (poznámka pre expertov - hoci to tak vyzerá, nejedná sa o rekurzívne volanie, ale naplánovanie volania funkcie).

V jednom programe môžeme použiť aj viacero časovačov a každý bude opakovať svoje príkazy (plánovať svoje ďalšie spustenia) nezávisle od druhého časovača.

```

import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

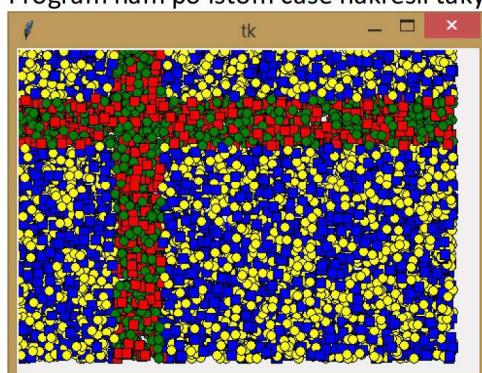
def kruzok():
    x = randrange(450)
    y = randrange(320)
    if 100<x<150 or 50<y<100:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='green')
    else:
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='yellow')
    canvas.after(10, kruzok)

def stvorec():
    x = randrange(450)
    y = randrange(320)
    if 100<x<150 or 50<y<100:
        canvas.create_rectangle(x-5, y-5, x+5, y+5, fill='red')
    else:
        canvas.create_rectangle(x-5, y-5, x+5, y+5, fill = 'blue')
    canvas.after(30, stvorec)

kruzok()
stvorec()

```

Program nám po istom čase nakreslí takýto obrázok:



Nasledujúci program postupne vykreslí pod sebou niekoľko loptičiek.

```

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

```

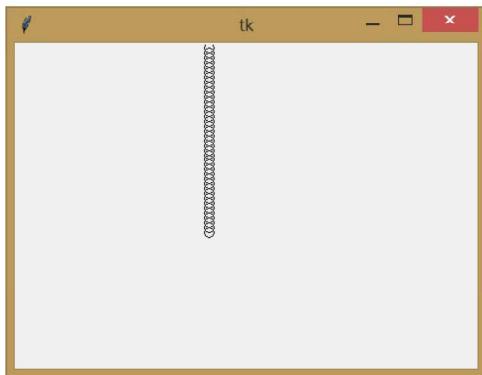
```

def lopticka():
    canvas.create_oval(x-5, y-5, x+5, y+5)
    global y
    y = y+5
    if y<200:
        canvas.after(100, lopticka)

x = 200
y = 5
lopticka()

```

Všimnite si, že v programe sme nastavili premennú `x` na hodnotu 200 a `y` na hodnotu 5. V týchto premenných si pamäťame súradnice, kde sa nakreslí prvá loptička. Po nastavení premenných spustíme funkciu `lopticka`. Táto funkcia nakreslí loptičku (krúžok) na súradničach podľa hodnôt premenných `x` a `y`. Ďalšiu loptičku chceme nakresliť o 5 bodov nižšie. Preto zvýšime `y` o 5, čiže zapíšeme `y = y+5`. Kedže sme vo funkcií `lopticka`, nemôžeme tu bežne meniť hodnotu premennej, ktorá sa používa v hlavnom programe mimo našej funkcie. Premenné `y` aj `x` sú globálne premenné, a nie lokálne premenné vo funkcií, kde ich chceme meniť. V takejto situácii musíme použiť príkaz `global` a meno premennej, aby sme funkcií oznámili, že chceme pracovať s globálnou premennou. Teraz má `y` hodnotu 10. V ďalšom riadku sa pýtame, či `y` je menšie ako 200, a iba ak je táto podmienka splnená, naplánujeme ďalšie spustenie funkcie `lopticka` o 100 milisekúnd. Vidíme, že sme vytvorili časovač, ktorý neopakuje príkazy donekonečna, ale len kým je splnená podmienka. Po spustení programu budeme vidieť, ako sa kreslí viacero loptičiek pod sebou.



Príkazom `canvas.delete('all')` môžeme vymazať všetko, čo sme nakreslili. Ak tento príkaz napišeme na začiatok funkcie `lopticka`, funkcia zakaždým zmaže všetko nakreslené a my budeme vidieť iba poslednú nakreslenú loptičku. Toto je princíp animácie. Najprv zmažeme pôvodné obrázky a rovnaký útvar kreslíme postupne na posunutom mieste.

```

def lopticka():
    canvas.delete('all')
    canvas.create_oval(x-5, y-5, x+5, y+5)
    global y
    y = y+5
    if y<200:
        canvas.after(100, lopticka)

```

Otázky:

1. Čo bude robiť tento program, ak vymeníme poradie príkazov `create_oval` a `delete`?
2. Na ktorej súradnici sa nakreslí posledná loptička?
3. Čo bude robiť program, ak podmienku nahradíme takto: `y==200`?
4. Čo bude robiť program, ak budeme zvyšovať `y` o 6 (`y = y+6`) a podmienku nahradíme takto: `y==200`?

Úlohy:

1 Vytvorte program, v ktorom sa bude pohybovať loptička vodorovne od ľavého okraja k pravému okraju.

2 Vytvorte program, v ktorom bude loptička padať zhora zvislo nadol a keď príde k dolnému okraju, bude znova padať zhora nadol.

3 Vytvorte program, v ktorom sa bude pohybovať loptička šikmo nadol.

4 Odhadnite, čo bude robiť nasledovný program:

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

def lopticka():
    canvas.delete('all')
    canvas.create_oval(x-5, y-5, x+5, y+5)
    global y
    y = y+5
    if y>250:
        y = 5
    if pokracovat == 1:
        canvas.after(100, lopticka)

def stop(suradnice):
    global pokracovat
    pokracovat = 0

pokracovat = 1
x = 200
y = 5
lopticka()
canvas.bind('<Button-1>', stop)
```

5 Odhadnite, čo bude robiť nasledovný program:

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

def lopticka():
    canvas.delete('all')
    canvas.create_oval(x-5, y-5, x+5, y+5)
    global y
    y = y+5
    if y>250:
        y = 5
    if pokracovat == 1:
        canvas.after(100, lopticka)

def stop(suradnice):
    global pokracovat
    if pokracovat == 1:
        pokracovat = 0
    else:
        pokracovat = 1
        lopticka()
```

```

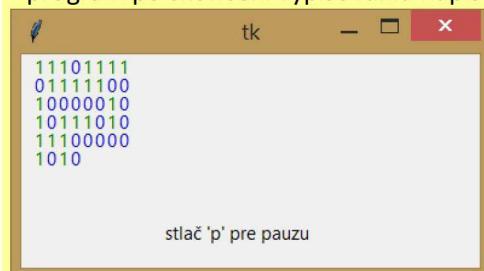
pokracovat = 1
x = 200
y = 5
lopticke()
canvas.bind('<Button-1>', stop)

```

6

Vytvorte program, ktorý bude postupne písati do canvasu 0 alebo 1 (náhodne si vyberá) a bude mať tieto vlastnosti:

- program začína písati z ľavého horného rohu v jednom riadku, až kým ho nezaplní,
- keď sa zaplní jeden riadok, písanie pokračuje na ďalšom riadku (môžete si určiť, koľko ich má byť v jednom riadku),
- 0 sa píše modrou farbou (resp. si vyberte farbu),
- 1 sa píše zelenou farbou (resp. si vyberte farbu),
- po stlačení klávesu p sa vypisovanie zastaví (pauza),
- opäťovným stlačením klávesu p sa zapne pokračovanie zastaveného vypisovania,
- keď sa zaplní celá obrazovka, program skončí,
- program po skončení vypisovania napiše počet napísaných jednotiek a počet nul.



7

Vytvorte program, ktorý bude mať tieto vlastnosti:

- z ľavého okraja sa pohybuje vo vodorovnom smere modrá gulička,
- z pravého okraja sa pohybuje vo vodorovnom smere k ľavej strane červená gulička,
- v každom kroku posunu si guličky (každá zvlášť) určia veľkosť posunu od 5 do 10 pixelov,
- animácia sa zastaví, keď sa loptičky stretnú,
- v mieste stretu sa niečo vypíše, napríklad BUM,
- animácia bude naprogramovaná v jednej funkcií (použite len jeden časovač).

11 Tlačidlá a vstupné pole

V okne programu môžeme vytvárať rôzne prvky, na ktoré sme zvyknutí pri používaní bežných programov. Teraz si ukážeme, ako vytvoríme tlačidlo, a ako mu nastavíme, čo má robiť po kliknutí.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def stvorcek(x, y, info):
    canvas.create_rectangle(x-10, y-10, x+10, y+10)
    canvas.create_text(x, y, text=info)

def button1_klik():
    stvorcek(randrange(300), randrange(200), randrange(100))

button1 = tkinter.Button(text='kresli stvorcek', command=button1_klik)
button1.pack()
```

Príkazom `tkinter.Button` vytvárame tlačidlo. V príkaze nastavujeme text zobrazený na tlačidle a príkaz, ktorý sa má vykonať po kliknutí na tlačidlo. Do premennej `button1` sme uložili referenciu na nové tlačidlo. Zápis `button1.pack()` zabezpečí zobrazenie nového tlačidla. Ešte sme v programe vytvorili funkciu `button1_click`, ktorá je nastavená ako príkaz pre toto tlačidlo `command = button1_click`. V tejto funkcií zapíšeme, čo všetko sa má diať po stlačení tlačidla.

Ďalší program obsahuje už dve tlačidlá, kde druhé vykreslí 10 štvorčekov tesne vedľa seba, postupne s číslami 1 až 10.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def stvorcek(x, y, info):
    canvas.create_rectangle(x 10, y 10, x+10, y+10)
    canvas.create_text(x, y, text=info)

def button1_klik():
    stvorcek(randrange(300), randrange(200), randrange(100))

def button2_klik():
    for i in range (1, 11):
        stvorcek(i*20, 100, i)

button1 = tkinter.Button(text='kresli stvorcek', command=button1_klik)
button1.pack()

button2 = tkinter.Button(text='stvorceky', command=button2_klik)
button2.pack()
```

Okrem tlačidiel môžeme mať v programe aj prvok `Entry` (vstupné pole), ktorý slúži na zadávanie vstupu do programu.

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def stvorcek(x, y, info):
    canvas.create_rectangle(x-10, y-10, x+10, y+10)
```

```

    canvas.create_text(x, y, text=info)

def button1_klik():
    stvorcek(randrange(300), randrange(200), randrange(100))

def button2_klik():
    for i in range (1, 11):
        stvorcek(i*20, 100, i)

def button3_klik():
    stvorcek(randrange(300), randrange(200), entry1.get())

button1 = tkinter.Button(text='kresli štvorček', command=button1_klik)
button1.pack()

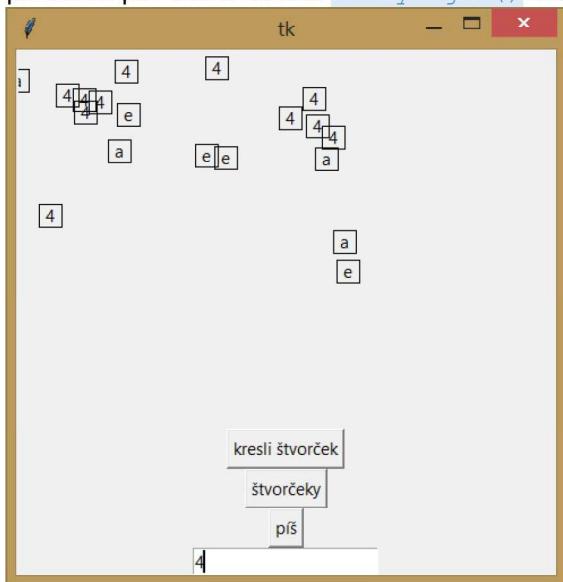
button2 = tkinter.Button(text='štvorčeky', command=button2_klik)
button2.pack()

button3 = tkinter.Button(text='piš', command=button3_klik)
button3.pack()

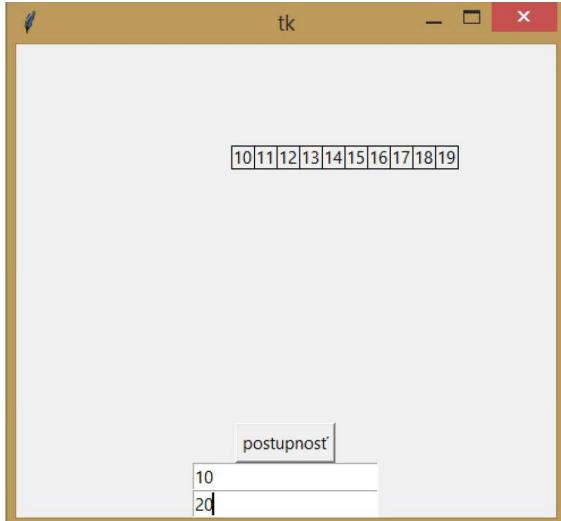
entry1 = tkinter.Entry()
entry1.pack()

```

Zápisom `entry1 = tkinter.Entry()` sme vytvorili okienko, do ktorého môžeme zadávať text (niekedy ho nazývame aj vstupné alebo textové pole). Referenciu na toto vstupné okienko sme uložili do premennej `entry1`. Tiež ho potrebujeme zobraziť príkazom `pack()`. Pri kreslení štvorčeka, po kliknutí na tlačidlo 3 si pozrieme pomocou funkcie `entry1.get()` aktuálne údaje v našom `entry1` a nakreslíme ich do štvorčeka.



Nasledujúci program nám vykreslí vedľa seba postupnosť štvorčekov podľa zadaných informácií vo vstupných poliach `entry1` a `entry2`. Do `entry1` zadávame, od ktorého čísla chceme opakovať cyklus, a do druhého zadávame, po ktoré číslo chceme opakovať cyklus (teda okrem neho samotného). Ešte sme museli v programe použiť funkciu `int()`, ktorá premení číslo v textovej podobe na číselnú hodnotu, s ktorou môžeme pracovať ako s bežným číslom.



```

import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def stvorcek(x, y, info):
    canvas.create_rectangle(x-10, y-10, x+10, y+10)
    canvas.create_text(x, y, text=info)

def button1_klik():
    od = int(entry1.get())
    do = int(entry2.get())
    for i in range (od, do):
        stvorcek(i*20, 100, i)

button1 = tkinter.Button(text='postupnosť', command=button1_klik)
button1.pack()

entry1 = tkinter.Entry()
entry1.pack()

entry2 = tkinter.Entry()
entry2.pack()

```

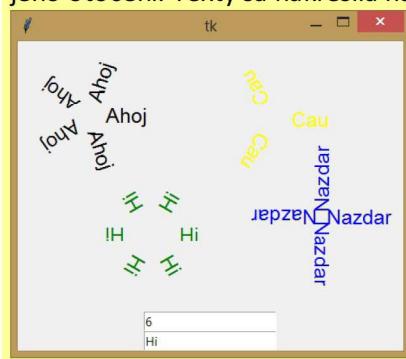
Úlohy:

1

Upravte program tak, aby sa postupnosť vždy kreslila od ľavého okraja obrazovky.

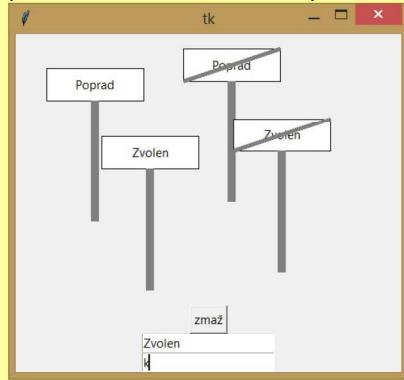
2

Vytvorte program, v ktorom môžeme kresliť otočené texty na mieste kliknutia myši. V programe sú dve súčiastky entry. Do jednej zadávame text, ktorý sa má písat do canvasu, a do druhej zadávame počet jeho otočení. Texty sa nakreslia náhodnou farbou z vopred určených farieb.

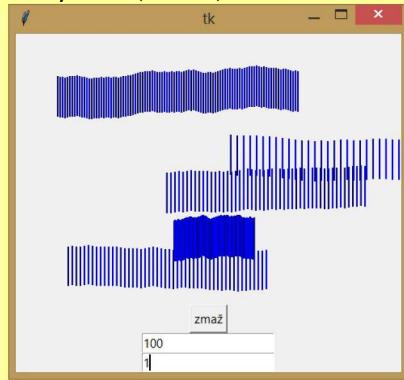


3 Doplňte do programu z úlohy 2 ďalšiu súčiastku entry, do ktorej môžeme zadať farbu, ktorou sa text nakreslí, a tlačidlo, ktoré zmaže obsah canvasu.

4 Vytvorte program, v ktorom môžeme kresliť na mieste kliknutia myši značku začiatku alebo konca obce/mesta. V programe sú dve súčiastky entry. Do jednej zadávame názov obce a do druhej zadávame, či sa má nakresliť značka začiatku alebo konca. Keď do druhého entry napišeme písmeno k, nakreslí sa značka konca obce - bude prečiarknutá. V programe bude aj tlačidlo, ktoré zmaže grafickú plochu. Značku nakreslite podľa obrázku:



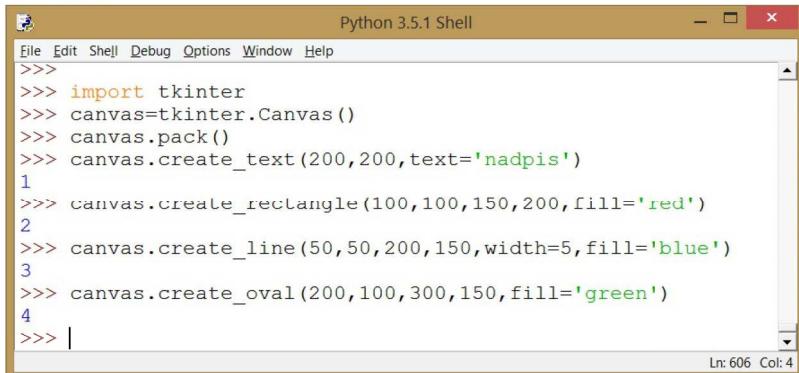
5 Vytvorte program, v ktorom môžeme kresliť na mieste kliknutia myši susediace zvislé čiary. V programe sú dve súčiastky entry. Do jednej zadávame počet čiar. Do druhej súčiastky entry zadávame veľkosť medzery medzi čiarami. Začiatok čiary je posunutý oproti predchádzajúcej čiare o náhodný posun z intervalu -1 až +1. Farba čiar je náhodne vybratá z vopred určených hodnôt. My sme použili farby blue1, blue2, blue3 a blue4. V programe bude aj tlačidlo, ktoré zmaže grafickú plochu.



6 Doplňte do programu ďalšiu súčiastku entry, ktorou môžeme určovať dĺžku čiar.

12 Posúvanie objektov canvasu

Už nám je dávno známe, že príkazy môžeme písat aj v okne shell. My sme takto postupne napísali príkazy na importovanie modulu tkinter a nakreslili sme text, rectangle, line a oval. Náš postup v okne shell vidíme na obrázku:



The screenshot shows the Python 3.5.1 Shell window. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The code area contains the following Python script:

```
>>> import tkinter
>>> canvas=tkinter.Canvas()
>>> canvas.pack()
>>> canvas.create_text(200,200,text='nadpis')
1
>>> canvas.create_rectangle(100,100,150,200,fill='red')
2
>>> canvas.create_line(50,50,200,150,width=5,fill='blue')
3
>>> canvas.create_oval(200,100,300,150,fill='green')
4
>>> |
```

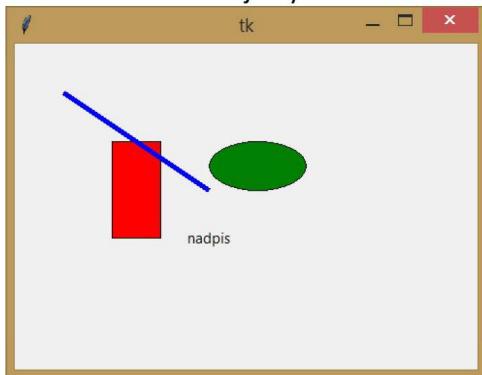
The status bar at the bottom right indicates Ln: 606 Col: 4.

Výsledok bol rovnaký, ako keby sme spustili tento program:

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

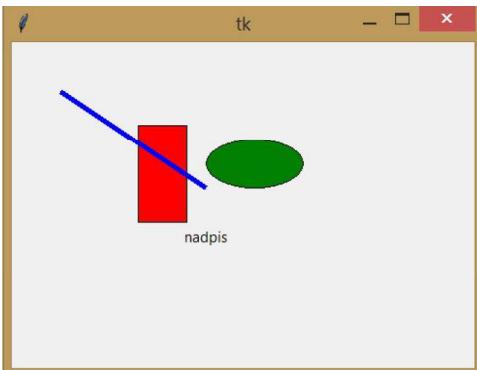
canvas.create_text(200,200,text='nadpis')
canvas.create_rectangle(100,100,150,200,fill='red')
canvas.create_line(50,50,200,150,width=5,fill='blue')
canvas.create_oval(200,100,300,150,fill='green')
```

Nakreslili sa tieto objekty:



Ešte raz si pozorne pozrime okno shell na prvom obrázku. Po nakreslení každého z útvarov sa na ďalší riadok napísalo číslo vytvoreného útvaru (objektu). Všetky veci, ktoré vytvoríme príkazmi začínajúcimi `canvas.create_`, majú po vytvorení priradené číslo. Vytvorené objekty `canvas`-u Python čísluje postupne od čísla 1. Tieto čísla môžeme použiť na rôzne zmeny objektu. Príkazom `canvas.move` môžeme objekty vytvorené v `canvas`-e posúvať. Napríklad `canvas.move(2, 30, -15)` posunie objekt s číslom 2 o +30 bodov na osi x a o -15 bodov na osi y (čiže vpravo a hore). V našom prípade posunieme červený obdĺžnik (má číslo 2).

```
>>> canvas.move(2, 30, -15)
>>>
```



Čísla vytvorených objektov si môžeme aj zapamätať do premennej a potom použiť pri posúvaní premennú, v ktorej si číslo útvaru pamätáme. Nasledujúci program si každý vytvorený objekt zapamäta do premennej a pri stlačení šípky vpravo zavolá funkciu, ktorá jednotlivé objekty posunie vpravo.

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

nadpis = canvas.create_text(200,200,text='nadpis')
obdlznik = canvas.create_rectangle(100,100,150,200,fill='red')
ciara = canvas.create_line(50,50,200,150,width=5,fill='blue')
oval = canvas.create_oval(200,100,300,150,fill='green')

def posun_vpravo(event):
    canvas.move(nadpis, 5, 0)
    canvas.move(obdlznik, 5, 0)
    canvas.move(ciara, 5, 0)
    canvas.move(oval, 5, 0)

canvas.bind_all('<Right>',posun_vpravo)
```

Kedže vytvorené objekty sa číslujú od 1 postupne, je zatiaľ zbytočné si nakreslené veci v predchádzajúcim programe pamätať. Číslovanie môžeme použiť aj na posúvanie objektov pomocou for cyklu. Predchádzajúci program môžeme zapísať aj takto:

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

canvas.create_text(200,200,text='nadpis')
canvas.create_rectangle(100,100,150,200,fill='red')
canvas.create_line(50,50,200,150,width=5,fill='blue')
canvas.create_oval(200,100,300,150,fill='green')

def posun_vpravo(event):
    for i in range(1,5):
        canvas.move(i,5,0)

canvas.bind_all('<Right>',posun_vpravo)
```

Ked' posúvame úplne všetky objekty, môžeme namiesto konkrétneho čísla použiť aj zápis `'all'`, `canvas.move('all',5,0)`. My sme sa s týmto zápisom už stretli v príkaze `canvas.delete('all')`. Teraz už asi tušíme, že ak by sme chceli zmazať len jeden objekt, môžeme do príkazu `canvas.delete('all')` zadať namiesto parametra `'all'` číslo konkrétneho objektu. Nasledujúci program posúva vpravo všetky nakreslené objekty a vľavo posunie jeden náhodne vybraný, ktorý má číslo medzi 1 až 5 (vrátane).

```
import tkinter
from random import *
canvas = tkinter.Canvas()
```

```

canvas.pack()

canvas.create_text(200,200,text='nadpis')
canvas.create_rectangle(100,100,150,200,fill='red')
canvas.create_line(50,50,200,150,width=5,fill='blue')
canvas.create_oval(200,100,300,150,fill='green')

def posun_vpravo(event):
    canvas.move('all',5,0)

def posun_vlavo(event):
    canvas.move(randint(1,4),-5,0)

canvas.bind_all('<Right>',posun_vpravo)
canvas.bind_all('<Left>',posun_vlavo)

```

Otázky:

- Čo bude robiť predchádzajúci program, ak v príkazovom režime vytvoríme ďalšie objekty? Budeme posúvať aj tie? Prečo?
- Aký je rozdiel v posúvaní cez parameter 'all' a posúvaním v cykle v prípade, že by sme z príkazového riadku vytvorili nové objekty?
- Čo sa stane, ak budeme posúvať alebo vymazávať objekt, ktorý sme už raz vymazali?

Keď sme si vytvorili obrázok z viacerých objektov, posúva sa ľažšie naraz celý obrázok. Museli by sme poznáť číslo prvého nakresleného objektu tohto obrázku a počet nakreslených častí. Za predpokladu, že sme ich kreslili hneď v poradí za sebou, môžeme takto obrázok posúvať cez cyklom od prvého čísla objektu až po číslo podľa počtu častí. Je tu však aj jednoduchšia možnosť. Všetky časti obrázku si môžeme označiť nejakou spoločnou značkou a potom môžeme posúvať objekty, ktoré majú spoločnú značku. Objekt si označujeme pri vytváraní parametrom `tags`, napríklad `canvas.create_oval(x,y,x+10,y+10, tags='auto')`. Potom môžeme pri posúvaní namiesto čísla objektu použiť jeho značku. Napríklad `canvas.move('auto', 5, 0)` značku môžeme použiť aj pri mazaní objektu. V nasledujúcom programe sme nakreslili auto a bicykel a jednotlivé časti týchto obrázkov sme si označkovali. Potom ich pomocou časovača posúvame každý zvlášť.

```

import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

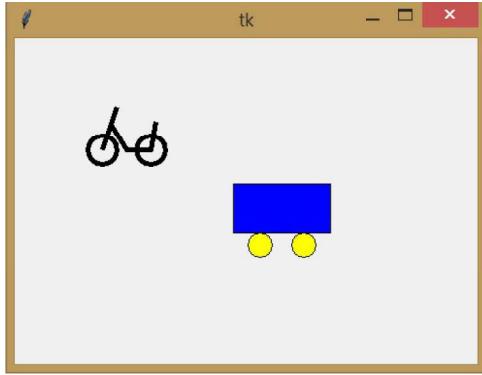
canvas.create_rectangle(100,150,200,200,fill='blue',tags='auto')
canvas.create_oval(115,200,140,225,fill='yellow',tags='auto')
canvas.create_oval(160,200,185,225,fill='yellow',tags='auto')

canvas.create_oval(200,100,230,130, fill='',width=5,
                  outline='black',tags='bicykel')
canvas.create_oval(250,100,280,130, fill='',width=5,
                  outline='black',tags='bicykel')
canvas.create_line(215,115,230,70, width=5,tags='bicykel')
canvas.create_line(225,90,240,115,265,115,270,85, width=5,tags='bicykel')

def posuvaj():
    canvas.move('bicykel',-5,0)
    canvas.move('auto',5,0)
    canvas.after(100,posuvaj)

posuvaj()

```



Bicykel aj auto po istom čase odídu z obrazovky. V takom prípade ich chceme ukázať na opačnom konci. Dá sa to riešiť rôznymi spôsobmi. Kedže to chceme vyriešiť pomocou nám už známych príkazov, najjednoduchšie bude počítať si v premennej pre každý obrázok jeho aktuálnu pozíciu. V prípade, že už bude mimo obrazovky, posunieme obrázok o šírku celej obrazovky na opačnú stranu.

```
import tkinter
from random import *
canvas = tkinter.Canvas(bg='white', width=600, height=250)
canvas.pack()

sirka=600

x_auto = 100
canvas.create_rectangle(100,150,200,200,fill='blue',tags='auto')
canvas.create_oval(115,200,140,225,fill='yellow',tags='auto')
canvas.create_oval(160,200,185,225,fill='yellow',tags='auto')

x_bicykel = 200
canvas.create_oval(200,100,230,130, fill='',width=5,outline='black',
                  tags='bicykel')
canvas.create_oval(250,100,280,130, fill='',width=5,outline='black',
                  tags='bicykel')
canvas.create_line(215,115,230,70, width=5,tags='bicykel')
canvas.create_line(225,90,240,115,265,115,270,85, width=5,tags='bicykel')


def posuvaj():
    global x_auto, x_bicykel
    x_bicykel = x_bicykel-5
    canvas.move('bicykel',-5,0)
    if x_bicykel<0:
        x_bicykel = x_bicykel + sirka
        canvas.move('bicykel',sirka,0)
    x_auto = x_auto+5
    canvas.move('auto',5,0)
    if x_auto>sirka:
        x_auto = x_auto - sirka
        canvas.move('auto',-sirka,0)
    print(x_auto) #informácie si môžeme vypísať aj do shellu
    canvas.after(100,posuvaj)

posuvaj()
```

13 Vytvárame jednoduché hry

13.1 Testovač pozornosti

Vytvorme hru (program), ktorá nám bude v pravidelných časových intervaloch vykresľovať na náhodnom mieste zelený štvorec, do ktorého má hráč kliknúť. Musí to stihnúť, kým štvorec nezmení miesto. Ak sa trafí do štvorca, získa 1 bod, ak sa netrafí do štvorca, naopak, jeden bod stratí. V hornej časti obrazovky hráč vidí počet získaných bodov.

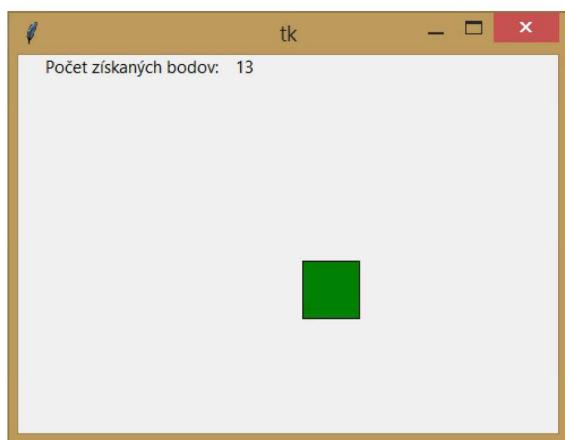
V tomto programe je riešenie:

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def timer1():
    canvas.delete('all')
    global sx, sy
    sx = randrange(300)
    sy = randint(20, 250)
    canvas.create_rectangle(sx, sy, sx+velkost, sy+velkost, fill='green')
    canvas.create_text(100, 10, text='Počet získaných bodov:')
    canvas.create_text(200, 10, text=pocet_bodov)
    canvas.after(500, timer1)

def klik(suradnice):
    global pocet_bodov
    x = suradnice.x
    y = suradnice.y
    if sx < x < sx+velkost and sy < y < sy+velkost:
        pocet_bodov = pocet_bodov + 1;
    else:
        pocet_bodov = pocet_bodov - 1;

pocet_bodov = 0
sx = 0
sy = 0
velkost = 50
timer1()
canvas.bind('<Button-1>', klik)
```

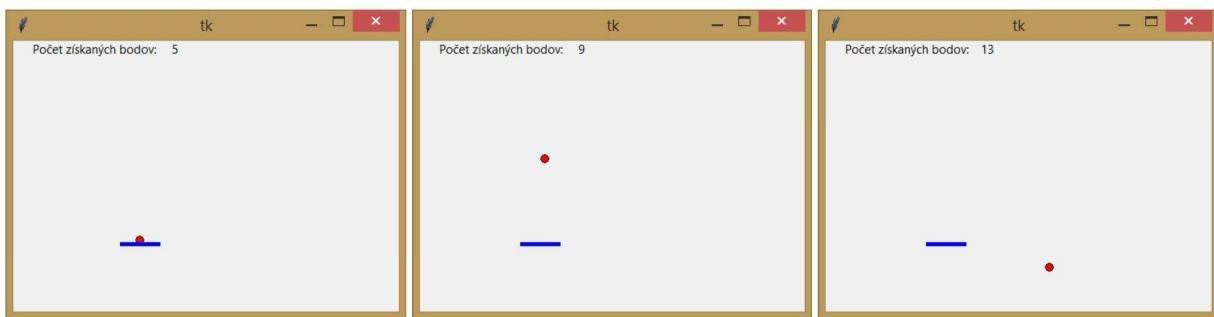


Úlohy:

- 1** Upravte program tak, aby sa zastavil, ak sme získali viac ako 10 bodov, a zagratoval nám k výborným reakciám.
- 2** Upravte program tak, aby sa zastavil, ak sme získali menej ako -10 bodov, a napísal nám, že je to s našimi reakciami trochu horšie.
- 3** Upravte program tak, aby niekedy kreslil červený štvorec, do ktorého nesmieme klikáť. Ak trafíme červený štvorec, program nám odpočíta 2 body. Zastaví sa takto upravený program pri -10 bodoch (ak sme riešili zároveň aj úlohu č. 2)?
- 4** Upravte program tak, aby sa podľa bodového stavu menila náročnosť – napríklad na istý počet získaných bodov sa štvorec premiestňuje častejšie, alebo sa kreslí menší, alebo častejšie sa kreslí červený štvorec.

13.2 Chytanie loptičiek

Vytvorme hru (program), v ktorej bude zhora obrazovky padať červená loptička. V spodnej časti obrazovky bude modrý kurzor – vodorovná čiara. Pohybom myši premiestňujeme náš modrý kurzor vľavo alebo vpravo. Cieľom hry je chytiť kurzorom padajúcu loptičku. Za chytenie loptičky získavame 1 bod. Keď loptičku chytíme alebo nechytíme, loptička začne znova padať zhora, ale na náhodnom mieste. Náš bodový stav sa vypisuje v hornej časti obrazovky.



V tomto programe je riešenie:

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def kresli_lopticku(x, y):
    canvas.create_oval(x-5, y-5,x+5, y+5, fill='red')

def kresli_kurzor(x, y):
    canvas.create_line(x, y, x+50, y, fill='blue', width=5)

def vypis_body(pocet_bodov):
    canvas.create_text(100, 10, text='Počet získaných bodov:')
    canvas.create_text(200, 10, text=pocet_bodov)

def timer1():
    canvas.delete('all')
    global lopticka_x, lopticka_y, pocet_bodov
    lopticka_y = lopticka_y + 5
    kresli_lopticku(lopticka_x, lopticka_y)
```

```

kresli_kurzor(kurzor_x, kurzor_y)
vypis_body(pocet_bodov)
if kurzor_x<lopticka_x<kurzor_x+50 and kurzor_y-10<lopticka_y<kurzor_y:
    pocet_bodov = pocet_bodov + 1
    lopticka_x = randrange(300)
    lopticka_y = 20
if lopticka_y>300:
    lopticka_x = randrange(300)
    lopticka_y = 20
canvas.after(10, timer1)

def posun_mysi(suradnice):
    global pocet_bodov, kurzor_x
    kurzor_x = suradnice.x
    canvas.delete('all')
    kresli_lopticku(lopticka_x, lopticka_y)
    kresli_kurzor(kurzor_x, kurzor_y)
    vypis_body(pocet_bodov)

pocet_bodov = 0
lopticka_x = randrange(300)
lopticka_y = 20
kurzor_x = 150
kurzor_y = 250

timer1()
canvas.bind('<Motion>', posun_mysi)

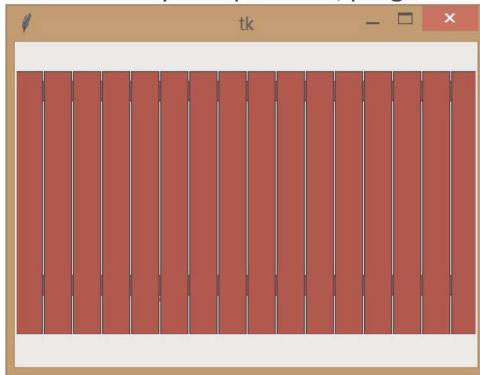
```

Úlohy:

- 5** Upravte program tak, aby nám odpočítal body, ak loptičku nechytíme.
- 6** Upravte program tak, aby sa podľa získaných bodov menila náročnosť – napríklad rýchlosť padania loptičky alebo šírka kurzora.
- 7** Upravte program tak, aby nám pri istom počte získaných bodov gratuloval k úspechu. Prípadne môže nakresliť veselého smajlíka.
- 8** Upravte program tak, aby sa pri nízkom alebo zápornom počte bodov zastavil a oznámil nám, že máme isté rezervy. Prípadne môže nakresliť smutného smajlíka.
- 9** Upravte program tak, aby sme využili na posun kurzora a loptičky príkaz `canvas.move`.

13.3 Hľadaj percento

Študenti radi získavajú percentá. Vytvorte im hru Hľadaj percento. Najprv sa na ploche nakreslí veľké červené %. Program sa pozdrží na 100 milisekúnd a hned' potom sa cez percento nakreslí hnedý latkový plot, ktorý percento prekryje. Môže sa stať, že niekedy kúsok percenta cez latky pretŕča. Úlohou hráča je kliknúť na miesto, za ktorým je percento skryté. Hráčovi program radí, či treba kliknúť vpravo alebo vľavo. Keď hráč klikne v blízkosti skrytého percenta, program mu oznamí, že vyhral.



V tomto programe je riešenie:

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()

def nakresli_percento(x,y):
    canvas.create_text(x, y, text='%', font='Arial 20', fill='red')

def nakresli_plot():
    canvas.create_rectangle(0, 40, 500, 60, fill='brown')
    canvas.create_rectangle(0, 240, 500, 260, fill='brown')
    x = 0
    for i in range(0, 20):
        canvas.create_rectangle(x, 30, x+28, 300, fill='brown')
        x = x + 30

def klik(suradnice):
    x = suradnice.x
    y = suradnice.y
    canvas.delete('all')
    nakresli_percento()
    nakresli_plot()
    if percento_x - 10 < x < percento_x + 10:
        canvas.create_text(200,10, text='Výborne, získal si percento')
    if x < percento_x - 10:
        canvas.create_text(200,10, text='Chod' viac vpravo')
    if percento_x + 10 < x:
        canvas.create_text(200,10, text='Chod' viac vľavo')

percento_x = randrange(500)
percento_y = randint(40, 280)
nakresli_percento()
canvas.update()
canvas.after(100)
nakresli_plot()

canvas.bind('<Button-1>', klik)
```

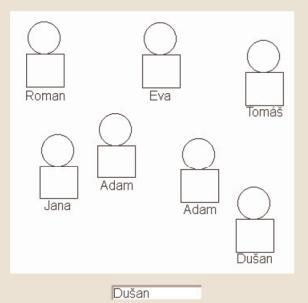
Úlohy:

- 10** Doplňte do programu tlačidlo, ktoré zapne novú hru – znova výzrebuje miesto nakreslenia percenta a všetko pripraví na štart hry.
- 11** Upravte program tak, že nám bude počítať počet pokusov hádania a priebežne ich bude vypisovať v hornej časti obrazovky.
- 12** Upravte program tak, že po vyčerpaní istého množstva pokusov nám napíše nejakú smutnú správu a už nám nedovolí ďalej hádať.

14 Úlohy na opakovanie II

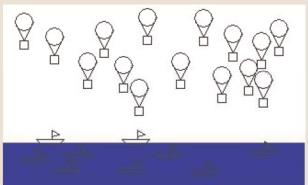
1

Vytvorte program – pomôcku pre učiteľa na vytvorenie zasadacieho poriadku. Do súčiastky Entry zadávame meno študenta, ktorého chceme usadiť. Kliknutím do plochy nakreslíme na mieste kliknutia tohto študenta aj s menom.



2

Vytvorte program, v ktorom si môžeme klikaním myši nakresliť lietajúce balóny na oblohe a plávajúce lodičky na mori. Program hneď po spustení nakreslí hladinu mora. Keď klikneme myšou nad morom, tak sa nakreslí balón. Ak klikneme na more, nakreslí sa lodička.



3

Vytvorte hru rovnaké kocky. V hre sa zobrazujú náhodné hodnoty dvoch hracích kociek (1 - 6). Hodnoty sa v pravidelných intervaloch menia. Úlohou hráča je zatlačiť tlačidlo „Rovnaké“, ak sú na oboch kockách rovnaké hodnoty. Ak stlačil tlačidlo v správnom okamihu, pripočítajú sa mu dva body. Keď sú hodnoty na kockách rôzne, hráč stratí jeden bod. V hornej časti obrazovky sa vypisuje aktuálny bodový stav.

4

Vytvorte program – „šetrič obrazovky“, ktorý:

- nám umožňuje pomocou súčiastky Entry zadávať ľubovoľný text,
- v pravidelných intervaloch zobrazuje zadaný text na náhodnom mieste a náhodnou farbou zo zadaných možností,
- zadaný text v každom ďalšom zobrazení nakreslí otočený s väčším uhlom,
- text vykresľuje postupne iba s uhlami otočenia od 10 do 90 a potom znova od začiatku s uhlom 10 až 90,
- pri kliknutí na plochu obsah obrazovky zmaže.

5

Vytvorte program Pokazený semafor. Na križovatke sa pokazil semafor (obsahuje červenú, žltú a zelenú farbu). Teraz na ňom v pravidelných časových intervaloch svietia rôzne svetlá takto:

- niekedy svietia všetky tri svetlá,
- niekedy svieti len zelené alebo len žlté, alebo len červené svetlo,
- niekedy svieti naraz červené a žlté svetlo.

Ukážka všetkých možností je na obrázkoch:



6

Vytvorte program Titulky do televízie, ktorý:

- bude v pravidelných intervaloch kresliť na obrazovke text zadaný v Entry,
- bude text postupne posúvať z pravého dolného okraja obrazovky k ľavému dolnému okraju,
- text po zmiznutí na ľavej strane opäť ukáže na pravej strane.

7

Vytvorte program Pyrotechnik.



Modrý káblik
červený káblik
žltý káblik
zelený káblik

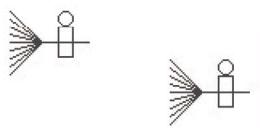
Veľkými číslami sa na ploche odpočítavajú sekundy (napr. od 60 do 0) a v časovom limite treba prestrihnúť správny káblik, aby nevybuchla bomba. Program má štyri tlačidlá – modrý, žltý, zelený, červený káblik. Kliknutie na tlačidlo znamená, že sme sa rozhodli prestrihnúť daný káblik.

Počítač po spustení náhodne vyberie správny káblik a zapamätá si ho. Hráč ho musí v časovom limite stlačiť (prestrihnúť).

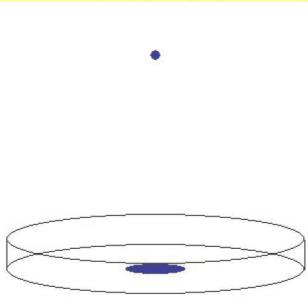
Sú dve možnosti riešenia – ak nestlačí včas správny káblik, napíše sa „Bomba vybuchla“ a nemôže stlačiť nič iné, alebo môže stláčať aj iné kabliky, kým bombu nezničí.

8

Vytvorte program, ktorý bude simulaovať pristávanie dvoch bosoriek vedľa seba. Pri pristávaní sa bosorky pohybujú smerom dole. V každom kroku animácie sa každá bosorka pohybuje náhodnou rýchlosťou od 0 do 10 bodov (v jednom kroku). Program nám v závere vypíše na obrazovku informáciu o poradí pristátia bosoriek.

**9**

Vytvorte program, v ktorom do Petriho misky postupne kvapkajú kvapky vody. Miska je na začiatku prázdna. Po padnutí kvapky do misky sa nakreslí mláčka, ktorá má rovnaký, len zmenšený tvar misky. Mláčka sa pri každej ďalšej kvapke zväčší. Ak už mláčka pokrýva celé dno misky, mláčka sa po spadnutí ďalšej kvapky nezväčší. V programe je aj tlačidlo, ktorým môžeme zastaviť kvapkanie, alebo ho zapnúť a všetko bude pokračovať. Program nám v hornej časti obrazovky vypisuje počet spadnutých kvapiek.



15 Pracujeme s textom

Zatiaľ sme najčastejšie pracovali s číslami. Čísla si vieme aj pamätať v premennej. Okrajovo sme sa už stretli aj s textovou informáciou. Vypisovali sme ju príkazom `canvas.create_text`. Ale aj keď sme si žrebovali náhodnú farbu zo zoznamu farieb a zapamätali sme si ju do nejakej premennej, v skutočnosti sme pracovali s textom, ktorý pomenúva nejakú farbu. To znamená, že si už vieme do premennej zapamätať aj nejaký text. V programovaní sa mu zvykne hovoriť aj **reťazec**, reťazec znakov alebo textový reťazec. S textovým reťazcom (stringom) sme sa stretli aj pri používaní komponentu `Entry`.

Mali by sme rozumieť nasledujúcemu programu:

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

slovo1 = 'veľké'
slovo2 = 'tajomstvo'
canvas.create_text(200, 100, text=slovo1)
canvas.create_text(200, 200, text=slovo2)
```

Program nakreslí na obrazovku pod seba dve slová - veľké a tajomstvo. Môže sa nám zdať, že si ich zatiaľ pamätáme v premenných zbytočne. Textové reťazce môžeme aj spájať operáciou `+`.

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

slovo1 = 'veľké'
slovo2 = 'tajomstvo'
spojene = slovo1 + slovo2
canvas.create_text(200, 100, text=spojene)
```

Už sme si ukázali, že nemusíme všetko písť a kresliť len do canvasu, ale môžeme informácie vypísať príkazom `print()` aj do okna shell (príkazového riadku).

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

slovo1 = 'veľké'
slovo2 = 'tajomstvo'
spojene = slovo1 + slovo2
print(spojene)
```

V tomto príklade vidíme, že canvas a knižnicu tkinter nemusíme použiť. Pôjde nám aj takýto program:

```
slovo1 = 'veľké'
slovo2 = 'tajomstvo'
spojene = slovo1 + slovo2      #3. riadok
print(spojene)
```

Otázky:

1. Čo sa stane, keď v programe zapíšeme namiesto `spojene = slovo1 + slovo2` toto:
`spojene = slovo2 + slovo1`?
2. Ako by ste zmenili zápis `spojene = slovo1 + slovo2`, keby ste chceli mať medzi slovami aj medzeru?
3. Čo sa stane, keď tretí riadok v predchádzajúcom programe zameníme za nasledujúci zápis?
 - a) `spojene = slovo1*2`

```
b) spojene = slovo1+' '*2+slovo2  
c) spojene = (slovo1+' ')*2+slovo2
```

4. Odhadnite, čo bude robiť nasledovný program.

```
from random import *  
ake = choice(('velké','malé','obrovské','drobné','smutné','veselé'))  
co = choice(('tajomstvo','prekvapenie','predsavzatie'))  
spojene = ake +' '+co  
print(spojene)
```

Z predchádzajúceho programu môžeme vytvoriť aj funkciu a doplníme program tak, aby nám vypísal viacero náhodne poskladaných viet.

```
from random import *  
  
def nahodna_veta():  
    kto = choice(('Kamarát','Spolužiak','Andrej','Roman'))  
    corobil = choice(('videl','prezradil','povedal','napísal','zistil',  
                      'nakreslil'))  
    ake = choice(('velké','malé','obrovské','drobné','smutné','veselé'))  
    co = choice(('tajomstvo','prekvapenie','predsavzatie'))  
    spojene = kto +' '+corobil+' '+ake+' '+co+'.'  
    print(spojene)  
  
for i in range(1,21):  
    nahodna_veta()
```

Ked' program spustíme, môžeme vidieť aj takéto vety:

```
Spolužiak prezradil velké prekvapenie.  
Andrej zistil malé predsavzatie.  
Andrej napísal drobné prekvapenie.  
Spolužiak zistil drobné prekvapenie.  
Kamarát nakreslil obrovské predsavzatie.  
Roman prezradil obrovské prekvapenie.  
Spolužiak nakreslil smutné predsavzatie.  
Roman nakreslil velké tajomstvo.  
Andrej povedal smutné predsavzatie.  
Andrej videl smutné prekvapenie.  
Kamarát napísal velké predsavzatie.  
Andrej videl smutné predsavzatie.  
Kamarát videl malé predsavzatie.  
Spolužiak zistil drobné prekvapenie.  
Roman videl drobné predsavzatie.  
Spolužiak zistil veselé tajomstvo.  
Kamarát povedal velké tajomstvo.  
Roman zistil obrovské tajomstvo.  
Kamarát videl velké tajomstvo.  
Spolužiak zistil obrovské tajomstvo.
```

Úlohy:

1

Rozšírite program na tvorbu náhodných viet. Doplňte do programu ďalšie slová, vytvorte viacslovné vety.

2

Ktoré slová z viet by sme mohli vyniechať, ak chceme, aby boli vety správne a mali zmysel? Upravte program tak, aby mali vety premenlivý počet slov a program sa náhodne rozhodol, či dané slovo z vety vynechá.

- 3** Upravte úlohu č. 1 alebo 2 tak, aby vo vetech boli aj ženské mená a podľa toho, či sa vybrało meno zo zoznamu mužských alebo ženských mien, sa vo vete prispôsobí sloveso do správneho rodu (napísal/napísala, zistil/zistila).
- 4** Vytvorte hru, v ktorej zhora canvasu postupne padá náhodne vyžrebované slovo (z postupnosti vopred napísaných slov). Postupnosť slov tvoria napríklad pravopisne správne a nesprávne slová. Úlohou hráča je klinutím myši chytať len pravopisne správne slová. Nesprávne slová musí hráč nechať spadnúť na spodok obrazovky. Za chytenie správneho slova získa hráč +1 bod, za chytenie nesprávneho slova stratí 2 body. Na obrazovke priebežne vidíme počet získaných bodov.
- 5** Doplňte úlohu č. 3 tak, aby nám hra pri bodovom zisku viac ako 10 bodov napísala, že nám gratuluje a že sme získali nejakú cenu. Tiež môže byť náhodná z vopred napísaného zoznamu. Napríklad: nový telefón, lístky do divadla, knihu s názvom, poukaz na doučovanie pravopisu.

16 Test - Preteky rytierov

V tomto teste si môžete preskúsať, čo ste sa naučili. Test je stavaný na 120 minút samostatnej práce. Počas riešenia testu nepoužívajte žiadne pomôcky, majte zapnuté jedine prostredie Python IDLE a v ňom tvorte nový program.

Text k úlohám č. 1 - 6

Vytvorte program, ktorý bude simulaovať preteky rytierov o dobytie hradu podľa nasledovného zadania. Snažte sa vytvoriť program čo najefektívnejšie, všade, kde je to možné a vhodné, použite for cyklus. V prípade, že neviete riešiť celú úlohu, vyriešte aspoň časť, ktorú viete.



1

V hornej časti grafickej plochy nakreslite hradby (cez celú šírku plochy).



2

V pravej časti hradieb nakreslite vstupnú bránu.



3

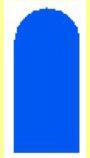
Kliknutím na hradby (okrem časti, kde je brána) sa nakreslí na mieste kliknutia okno so zadanou výškou. Výšku zadávame pomocou súčiastky Entry. Celé okno je nakreslené modrou farbou.



výška okna 30



výška okna 50



výška okna 70

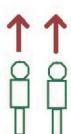
4

Upravte program tak, že po stlačení tlačidla Strom sa pred hradbami na náhodnom mieste nakreslí jeden zelený strom. Stromy sa nekreslia pred vstupom do hradu.



5

Upravte program tak, že bude simulaovať preteky dvoch rytierov bežiacich k hradu. Rytieri bežia smerom k bráne z pravého dolného rohu obrazovky. Na začiatku sú pri okraji obrazovky vedľa seba a potom sa začnú pohybovať smerom hore. Pri každom posune sa každý z nich posunie o náhodný krok s veľkosťou 0 až 10 bodov. Keď sa jeden z nich dostane k bráne, animácia zastane a na obrazovku sa vypíše číslo rytiera, ktorý vyhral.



6

Upravte program tak, že nám na začiatku umožní zadať tip na víťaza (vami zvolenou formou – napríklad tlačidlami, alebo pomocou Entry a pod.) a až potom odštartovať preteky. Po skončení pretekov program vyhodnotí, či sme tipovali správne.

KONIEC TESTU

17 Poznámky

17.1 Používanie príkazu import

Aký je rozdiel v zápisе import tkinter a from tkinter import *? V prvom prípade musíme písаť k príkazu skratku, z ktorého modulu pochádza. Čiže:

```
from tkinter import *
canvas = Canvas()
canvas.pack()
```

alebo:

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

17.2 Náhodné farby

V Pythone sa farba určuje zápisom v šestnáškovej sústave alebo textom s názvom farby.

```
canvas.create_oval(10,10,100,100, fill='#00FFFF')
canvas.create_oval(10,10,150,100, fill='blue')
```

Ak chceme miešať úplne náhodnú farbu, potrebujeme náhodné čísla naformátovať do tvaru v šestnáškovej sústave. Je vhodné spraviť si funkciu a používať to takto:

```
def rgb(r, g, b):
    return '#{:02x}{:02x}{:02x}'.format(r, g, b)

farba = rgb(100, 100, 200)
canvas.create_oval(10,10,100,100, fill=farba)
```

Samozrejme, nemusíme na to použiť premennú farba, ale môžeme to zapísаť priamo na mieste použitia. V prípade, že vyberáme náhodnú farbu z vymenovaných farieb - postupnosti (bud' názvom farby alebo zápisom v šestnáškovej sústave), môžeme použiť príkaz `random.choice()`.

```
farba = random.choice(('blue', 'red', 'yellow', '#00FFFF'))
canvas.create_oval(10,10,100,100, fill=farba)
```

17.3 Zadávanie vstupu - entry

Do súčiastky entry môžeme príkazom `insert` napísаť (vložiť) text. V nasledujúcom programe sme hned' po spustení programu vložili do `entry` (na začiatok, čiže pozíciu 0) text 'Sem zadaj svoje meno'. Po kliknutí na tlačidlo s textom `Privítaj` ma sa na náhodnom mieste napíše pozdrav Ahoj s vyplneným menom podľa `entry`.

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

def privitanie():
    meno = entry1.get()
```

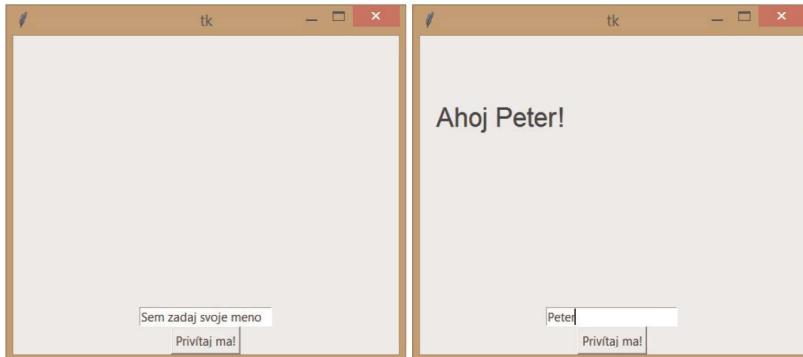
```

        canvas.create_text(100,100,text='Ahoj '+meno+'!', font='Arial 20')

entry1 = tkinter.Entry()
entry1.pack()
entry1.insert(0,'Sem zadaj svoje meno')

button1 = tkinter.Button(text='Privítaj ma!',command=privitanie)
button1.pack()

```



Príkaz `entry1.delete(0, tkinter.END)` vymaže obsah celého entry (od pozície 0 po koniec). Použili sme tento príkaz, aby sme napísané meno vymazali a znova tam vložili text 'Sem zadaj svoje meno'.

```

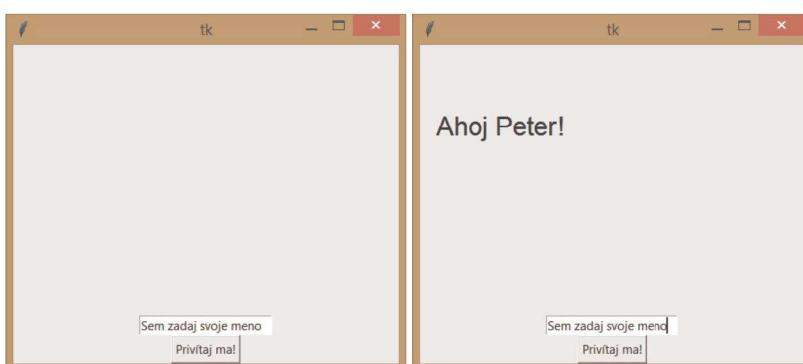
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

def privitanie():
    meno = entry1.get()
    canvas.create_text(100,100,text='Ahoj '+meno+'!', font='Arial 20')
    entry1.delete(0,tkinter.END)
    entry1.insert(0,'Sem zadaj svoje meno')

entry1 = tkinter.Entry()
entry1.pack()
entry1.insert(0,'Sem zadaj svoje meno')

button1 = tkinter.Button(text='Privítaj ma!',command=privitanie)
button1.pack()

```



18 Použitá literatúra

- [1] Blaho, A.: Informatika pre stredné školy – Programovanie v Delphi. Bratislava: SPN, 2006, ISBN 80 - 10 - 00421 - 9
- [2] Blaho, A.: <http://python.input.sk>, 2016
- [3] Bellušová, M., Varga, M., Zimanová, R.: Informatika pre stredné školy – Algoritmy s Pascalom. Bratislava: SPN, 2002, ISBN 80 - 08 - 03289 - 8
- [4] Hanulová, E., Kučera, P.: Maturita z informatiky. Tvorba zadanií. Bratislava: MPC, 2007, ISBN 978 - 80 - 7164 - 440 - 8

ISBN 978-80-972320-4-7 (pdf)



ISBN 978-80-972320-5-4 (epub)



ISBN 978-80-972320-6-1 (mobi)

