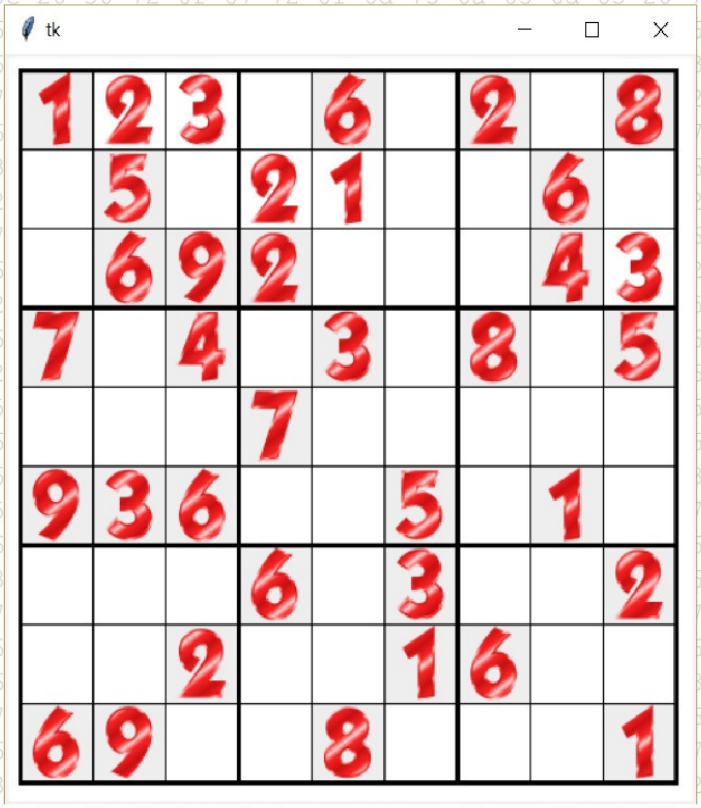
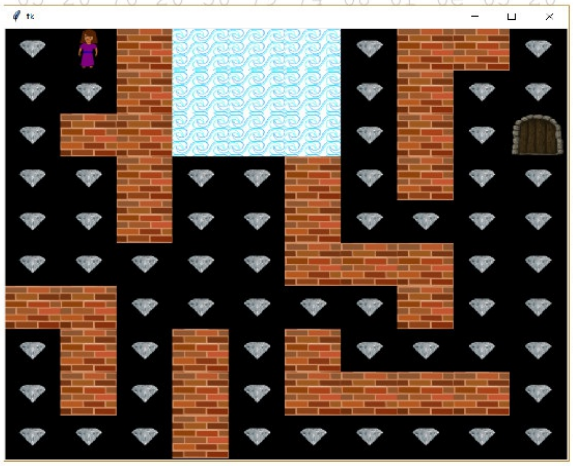
Programujeme v Pythone

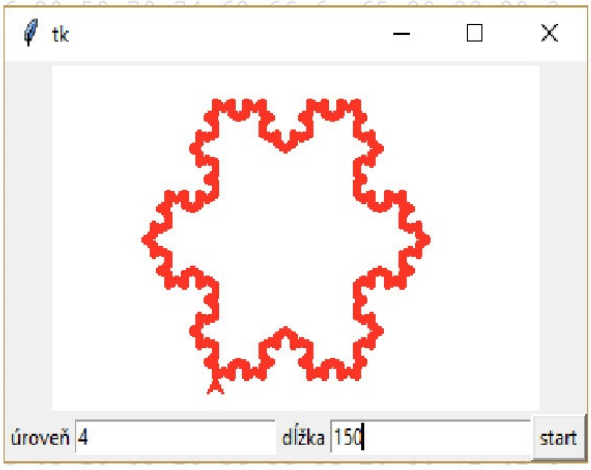
učebnica informatiky pre stredné školy

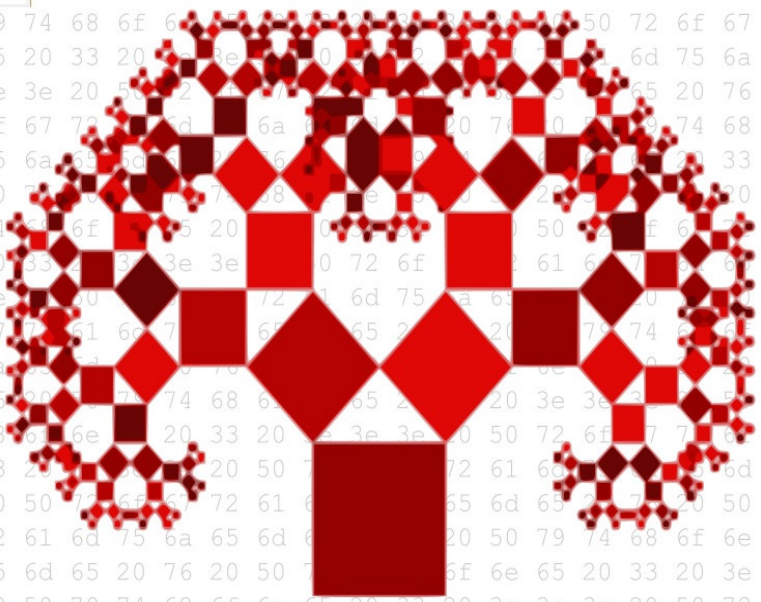
3. diel

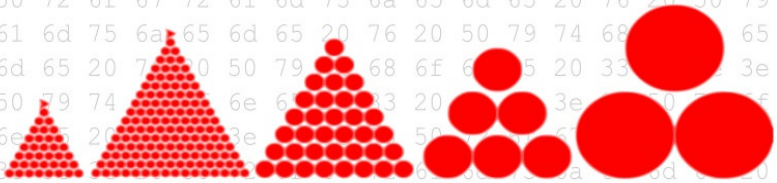
Peter Kučera, Jaroslav Výbošťok











www.programujemevpythone.sk

Programujeme v Pythone 3

učebnica informatiky pre stredné školy

Autori © Mgr. Peter Kučera, Mgr. Jaroslav Výbošťok

Design © Mgr. Peter Kučera

Jazyková korektúra: Mgr. Katarína Kučerová

Prvé vydanie, 2018

Verzia číslo: 20181216

Vydavateľ: Mgr. Peter Kučera

Ukážka z e-knihy

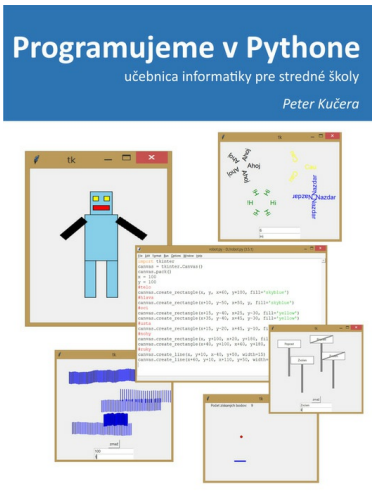
učebnicu a ďalšie materiály si môžete zakúpiť aj priamo na stránkach autora:
http://www.programujemevpythone.sk/ a https://www.facebook.com/programujemevpythone

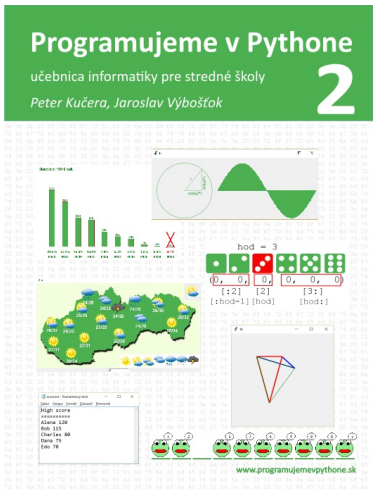
ISBN 978-80-570-0502-5 (pdf)

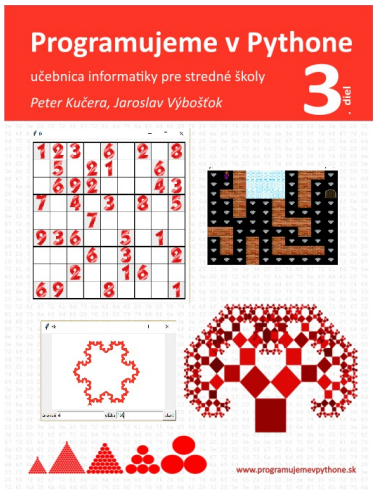
ISBN 978-80-570-0503-2 (epub)

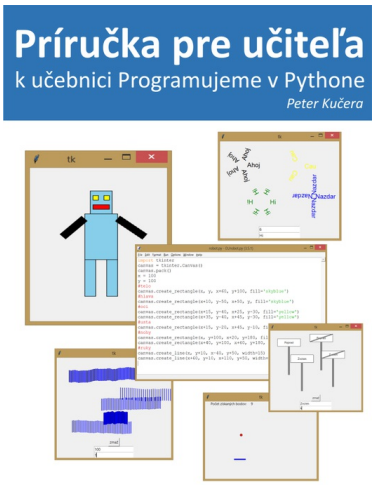
ISBN 978-80-570-0504-9 (mobi)

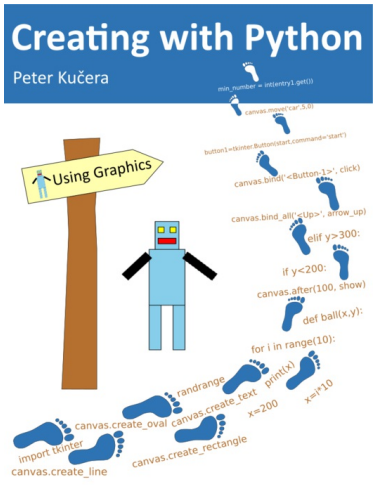
Z NAŠEJ PONUKY

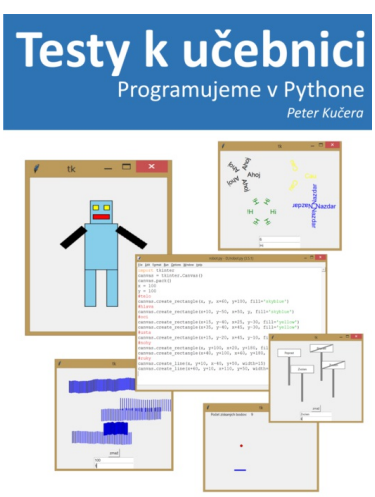


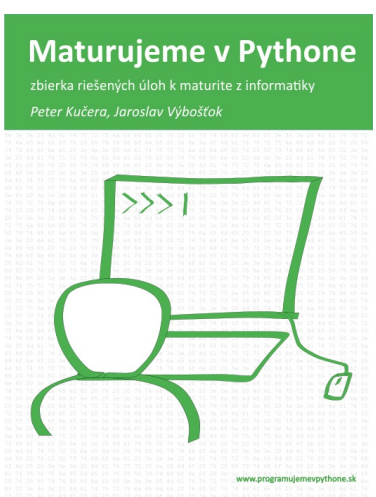


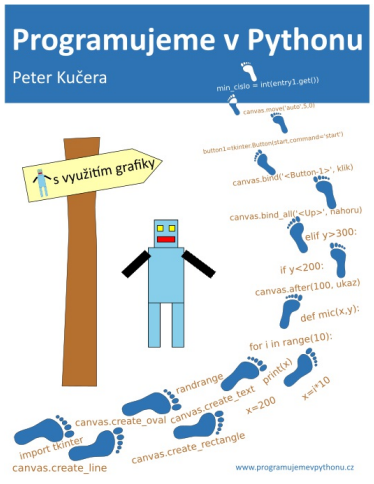












www.programujemevpythone.sk
www.facebook.com/programujemevpythone

Obsah

Úvod

Učebnica Programujeme v Pythone sa niesla v modrom dizajne a jej druhý diel v zelenom. Ešte chýbala tretia – červená zložka, aby bolo farebné spektrum RGB kompletné. Tretí diel, ktorého úvod práve čítate, sa nesie v červených farbách. Venuje sa témam, ktoré už prevyšujú cieľové požiadavky na maturitné skúšky. Učebnica je vhodná nielen pre všetkých študentov, ktorí si programovanie obľúbili a chcú sa mú hlbšie venovať aj v budúcnosti, ale aj pre tých, ktorí chcú byť vybavení silným nástrojom na riešenie problémov, ktoré dnešná digitálna doba prináša.

Aj obsah tohto dielu sme starostlivo namiešali a využili sme pri tom naše dlhoročné skúsenosti z vyučovania programovania na strednej škole.

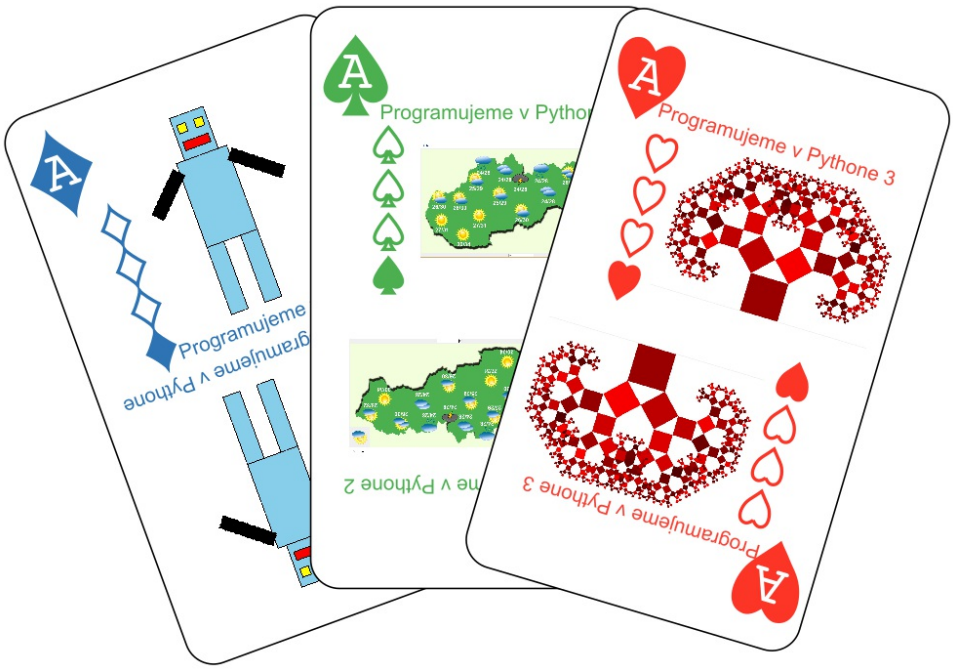
V učebnici nájdete množstvo praktických a riešených príkladov, úlohy na precvičenie, ale aj otázky, ktoré vás majú nabádať na premýšľanie, objavovanie súvislostí, diskusiu v skupine, experimentovanie, ale aj hľadanie chýb a intuitívne hľadanie optimálneho riešenia.

Naučíte sa vytvárať a používať zložitejšie štruktúry, programovať algoritmy s využitím rekurzie, ktorá je veľmi silným, ale elegantným nástrojom, a tiež si môžete otestovať svoje hranice pri používaní a tvorení grafových algoritmov. Veríme, že vás zaujme aj objektové programovanie.

Ak ste sa k tretiemu dielu dostali len náhodou a chýbajú vám základy, nebojte sa rekurzívne vrátiť k predchádzajúcej úrovni. Nekonečného zacyklenia sa obávať nemusíte, 1. diel Programujeme v Pythone rieši aj triviálne prípady :)

Pri programovaní s touto učebnicou vám prajeme veľa príjemných chvíľ, zábavy, úspechov, ale aj zopár chybových hlášok, pri ktorých sa trochu potrápite, no hlavne sa naučíte ešte viac ...

autori



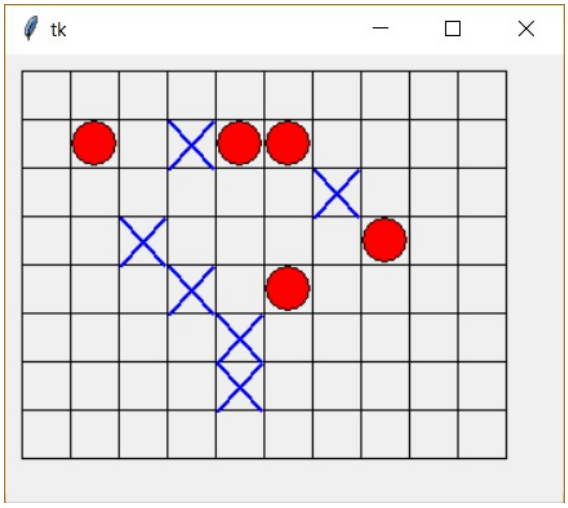
Textové súbory a obrázky k úlohám si môžete stiahnuť z:

...

1 Zoznam zoznamov (dvojrozmerné pole)

1.1 Piškvorky

Nasledujúci program je prípravou na hru Piškvorky. Program vykreslí mriežku. Kliknutím ľavého tlačidla myši sa na mieste kliknutia nakreslí krúžok alebo krížik, podľa toho, ktorý hráč je na ťahu.




```
# p1-01.py
import tkinter
canvas = tkinter.Canvas(width=400, height=400)
canvas.pack()
pocet_policok_x, pocet_policok_y = 10, 8
v = 30  #veľkost políčka
mriezka_x, mriezka_y = 10, 10       # x a y začiatku mriežky
vypln = 1

def kresli_mriezku(px, py, v, mx, my):
    for stlpec in range(px):
        for riadok in range(py):
            canvas.create_rectangle(stlpec*v + mx, riadok*v + my,
                                    (stlpec+1)*v + mx, (riadok+1)*v + my)

def kresli(stlpec, riadok, vypln):
    if vypln == 1:
        canvas.create_oval(stlpec*v + mriezka_x +1, riadok*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -2, (riadok+1)*v+mriezka_y-2,
                           fill = 'red')
    if vypln == 2:
        canvas.create_line(stlpec*v + mriezka_x +1, riadok*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -1, (riadok+1)*v+mriezka_y+1,
                           fill = 'blue', width = 2)
        canvas.create_line(stlpec*v + mriezka_x +1, (riadok+1)*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -1, riadok*v + mriezka_y+1,
                           fill = 'blue', width = 2)

def pozicia(x, y):
    stlpec = (x - mriezka_x) // v
    riadok = (y - mriezka_y) // v
    return stlpec, riadok

def oznac(sur):
    global vypln
```

```
    vypln = 3 - vypln
    stlpec, riadok = pozicia(sur.x, sur.y)
    kresli(stlpec, riadok, vypln)

kresli_mriezku(pocet_policok_x, pocet_policok_y, v, mriezka_x, mriezka_y)

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

Otázky:

1. Ktorý útvar sa nakreslí po prvom kliknutí?
2. Ktorý príkaz zabezpečí, že sa pri ďalšom klinutí nakreslí iný útvar?
3. Akými inými spôsobmi môžeme zabezpečiť výmenu kresliaceho sa útvaru (`vypln = 3 - vypln`)?
4. Prečo sa nám nepodarí kliknutím na hranici políčka nakresliť krížik alebo krúžok cez dve políčka?
5. Aký index má v programe prvý stĺpec (resp. prvý riadok) hracej plochy?
6. Čo sa stane, ak na niektoré políčko klikneme viackrát?
7. Čo sa stane, ak klikneme mimo hracej mriežky?
8. Akým spôsobom zväčšíme počet políčok hracej plochy, veľkosť jedného políčka, alebo posunieme celú hraciu plochu?
9. Ako zistíme, koľko útvarov naklikal používateľ počas hry? Nechceme upravovať pôvodný (existujúci) program, ale na vyriešenie môžeme použiť len tlačidlo a vytvoriť funkciu, ktorá túto informáciu vypíše.

Program si zatiaľ v žiadnej štruktúre nepamätá naklikané krížiky a krúžky. Informácie o existujúcich útvaroch by sme vedeli zistiť pomocou vlastností vytvorených grafických útvarov v canvase (s týmito vlastnosťami sme pracovali v 9. kapitole 2. dielu učebnice). Bolo by to ale zbytočne komplikované. Pomohlo by nám vhodné vytvorenie značiek k útvarom (**`tags`**).

My si ukážeme iný spôsob. Naklikané krížiky a krúžky si budeme pamätať v zozname. Každému (aj prázdnemu) políčku mriežky bude v zozname zodpovedať jedno číslo. Napríklad: prázdne políčko bude reprezentované číslom 0, krížik číslom 1 a krúžok číslom 2. Pre mriežku s rozmermi 10 stĺpcov a 8 riadkov by nám postačoval 80-prvkový zoznam. Ako zistíme, čo je umiestnené v treťom stĺpci a druhom riadku? Zakaždým by sme si museli počítať index prvku v zozname. Napríklad výpočtom **`(riadok-1) * pocet_policok_x + stlpec - 1`**.

```
>>> riadok = 2
>>> stlpec = 3
>>> (riadok-1) * pocet_policok_x + stlpec - 1
12
>>> riadok = 1
>>> stlpec = 1
>>> (riadok-1) * pocet_policok_x + stlpec - 1
0
>>> riadok = 8
>>> stlpec = 10
>>> (riadok-1) * pocet_policok_x + stlpec - 1
79
>>> riadok = 3
>>> stlpec = 1
>>> (riadok-1) * pocet_policok_x + stlpec - 1
20
```

Aby sa nám pracovalo s indexmi políčok jednoduchšie, budeme si informácie pamätať inak. Pre každý riadok mriežky vytvoríme jeden zoznam - v našom prípade 8 zoznamov. Pridaním týchto zoznamov do jedného zoznamu získame zoznam zoznamov. Jedným prvkom tohto zoznamu bude práve zoznam čísel jedného riadku. Jeden riadok je zoznam 10 čísel.

```
# p1-01a.py
plocha = []
for i in range(8):
    plocha.append([0]*10)
print(plocha)
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Chceme zmeniť prázdne políčko v druhom riadku a treťom stĺpci v zozname **plocha** na krížik. Najprv uvedieme index pre daný riadok (čiže 1, lebo index 0 má prvý riadok, index 1 druhý riadok) a v ňom uvedieme index pre stĺpec (čiže 2, lebo prvý stĺpec má v zozname index 0). Zmeníme teda hodnotu pre **plocha[1][2]** a priradíme tam hodnotu **2**.

```
>>> plocha[1][2] = 2
>>> plocha
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 2, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
>>>
```

Ešte to zovšeobecníme podľa rozmerov plochy, ktoré máme nastavené v premenných, a doplníme to do programu pred definíciu funkcie **kresli_mriezku**:

```
#p1-01a-2.py
plocha = []
for i in range(pocet_policok_y):
    plocha.append([0]*pocet_policok_x)
print(plocha)
```

Teraz už môžeme doplniť funkciu **oznac()** o kontrolu, či je políčko prázdne. Iba ak je prázdne, zmeníme políčku hodnotu, vykreslíme obrázok a zmeníme typ výplne.

```
def oznac(sur):                #p1-01b.py
    global vypln
    stlpec, riadok = pozicia(sur.x, sur.y)
    if plocha[riadok][stlpec] == 0:
        vypln = 3 - vypln
        plocha[riadok][stlpec] = vypln
        kresli(stlpec, riadok, vypln)
```

Otázky:

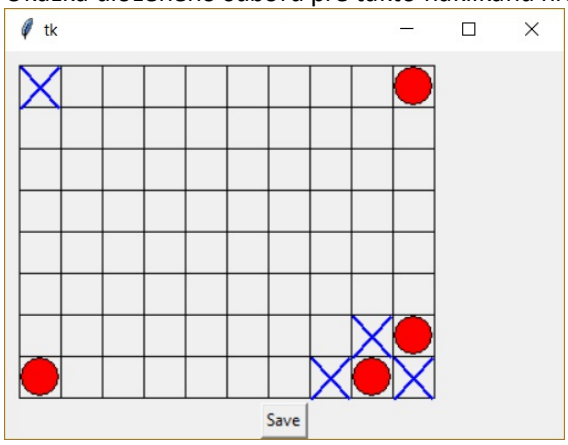
10. Prečo program ohlási chybu, keď klikneme mimo mriežky?
11. Ako zabezpečíme, aby sme tejto chybe predišli?

Keď si pamätáme naklikané útvary, môžeme rozohratú hru uložiť do textového súboru. Doplníme do programu tlačidlo **Save**, ktoré hru uloží. Do prvého riadku uložíme veľkosť hracej plochy (v počte políčok).

```
def save():                    #p1-01c.py
    subor = open('piskvorky.txt', 'w')
    subor.write(str(pocet_policok_x)+' '+str(pocet_policok_y)+'\n')
    for riadok in plocha:
        zapisat = ''
        for prvok in riadok:
            zapisat = zapisat + str(prvok) + ' '
        zapisat = zapisat[:-1]+'\n'
```

```
        subor.write(zapisat)
    subor.close()
button1 = tkinter.Button(text='Save', command=save)
button1.pack()
```

Ukážka uloženého súboru pre takto naklikanú hraciu plochu:




```
10 8
2 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 2 1
1 0 0 0 0 0 0 2 1 2
```

Ešte doplníme do programu tlačidlo **Load**, ktoré načíta do zoznamu textový súbor, vykreslí mriežku a útvary a zmení rozmery plochy podľa informácií v súbore. Keďže po prečítaní riadku a použití metódy **split()** dostaneme z čísel v riadku zoznam reťazcov, a tie potrebujeme po jednotlivých prvkoch premeniť funkciou **int()** na čísla, vyrobíme si funkciu **int_zoznam()**. Funkcia **int_zoznam()** nám všetky prvky zoznamu zmení na čísla a na výstupe vráti zoznam čísel.

```
def int_zoznam(zoznam):      #p1-01d.py
    vysledok = []
    for prvok in zoznam:
        vysledok.append(int(prvok))
    return vysledok
```

```
def load():                  #p1-01e.py
    global pocet_policok_x, pocet_policok_y, plocha
    subor = open('piskvorky.txt', 'r')
    info = subor.readline()
    info = info.strip()
    rozmery = info.split()
    pocet_policok_x, pocet_policok_y = int_zoznam(rozmery)
    plocha = []
    for riadok in subor:
        riadok = riadok.strip()
        riadok_zoznam = riadok.split()
        riadok_zoznam = int_zoznam(riadok_zoznam)
        plocha.append(riadok_zoznam)
    canvas.delete('all')
    kresli_mriezku(pocet_policok_x, pocet_policok_y, v, mriezka_x, mriezka_y)
    kresli_plochu()

button2 = tkinter.Button(text='Load', command=load)
button2.pack()
```

Funkcia **kresli_plochu()**, ktorú voláme po prečítaní súboru, prejde celým zoznamom **plocha** a postupne vykreslí jednotlivé útvary volaním funkcie **kresli()**.

```
def kresli_plochu():        #p1-01f.py
    for r in range(pocet_policok_y):
        for s in range(pocet_policok_x):
            kresli(s, r, plocha[r][s])
```

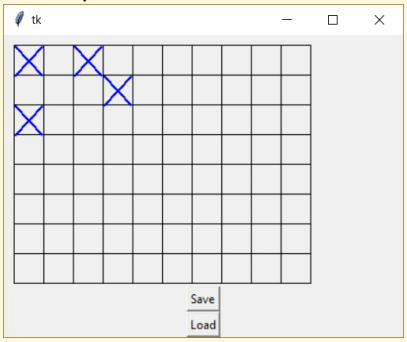
Zatiaľ máme spravený tento program:

```
# p1-01g.py
import tkinter
canvas = tkinter.Canvas(width=400, height=400)
canvas.pack()
pocet_policok_x, pocet_policok_y = 10, 8
v = 30  #veľkost políčka
mriezka_x, mriezka_y = 10, 10       # x a y začiatku mriežky
vypln = 1
plocha = []
for i in range(pocet_policok_y):
    plocha.append([0]*pocet_policok_x)

def kresli_mriezku(px, py, v, mx, my):
    for stlpec in range(px):
        for riadok in range(py):
            canvas.create_rectangle(stlpec*v + mx, riadok*v + my,
                                    (stlpec+1)*v + mx, (riadok+1)*v + my)

def kresli(stlpec, riadok, vypln):
    if vypln == 1:
        canvas.create_oval(stlpec*v + mriezka_x +1, riadok*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -2, (riadok+1)*v+mriezka_y-2,
                           fill = 'red')
    if vypln == 2:
        canvas.create_line(stlpec*v + mriezka_x +1, riadok*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -1, (riadok+1)*v+mriezka_y+1,
                           fill = 'blue', width = 2)
        canvas.create_line(stlpec*v + mriezka_x +1, (riadok+1)*v+mriezka_y+1,
                           (stlpec+1)*v+mriezka_x -1, riadok*v + mriezka_y+1,
                           fill = 'blue', width = 2)

def pozicia(x, y):
    stlpec = (x - mriezka_x) // v
    riadok = (y - mriezka_y) // v
    return stlpec, riadok

def oznac(sur):
    global vypln
    stlpec, riadok = pozicia(sur.x, sur.y)
    if plocha[riadok][stlpec] == 0:
        vypln = 3 - vypln
        plocha[riadok][stlpec] = vypln
        kresli(stlpec, riadok, vypln)

def save():
    subor = open('piskvorky.txt', 'w')
    subor.write(str(pocet_policok_x)+' '+str(pocet_policok_y)+'\n')
    for riadok in plocha:
        zapisat = ''
        for prvok in riadok:
            zapisat = zapisat + str(prvok) + ' '
        zapisat = zapisat[:-1]+'\n'
        subor.write(zapisat)
```

```
    subor.close()

def int_zoznam(zoznam):
    vysledok = []
    for prvok in zoznam:
        vysledok.append(int(prvok))
    return vysledok

def kresli_plochu():
    for r in range(pocet_policok_y):
        for s in range(pocet_policok_x):
            kresli(s, r, plocha[r][s])

def load():
    global pocet_policok_x, pocet_policok_y, plocha
    subor = open('piskvorky.txt', 'r')
    info = subor.readline()
    info = info.strip()
    rozmery = info.split()
    pocet_policok_x, pocet_policok_y = int_zoznam(rozmery)
    plocha = []
    for riadok in subor:
        riadok = riadok.strip()
        riadok_zoznam = riadok.split()
        riadok_zoznam = int_zoznam(riadok_zoznam)
        plocha.append(riadok_zoznam)
    subor.close()
    canvas.delete('all')
    kresli_mriezku(pocet_policok_x, pocet_policok_y, v, mriezka_x, mriezka_y)
    kresli_plochu()

button1 = tkinter.Button(text='Save', command=save)
button1.pack()
button2 = tkinter.Button(text='Load', command=load)
button2.pack()

kresli_mriezku(pocet_policok_x, pocet_policok_y, v, mriezka_x, mriezka_y)

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

Otázky:

12. Keď načítame akúkoľvek uloženú hru (vytvorenú týmto programom), ktorý útvar sa nakreslí v hre pri nasledujúcom kliknutí? Ako môžeme zabezpečiť, aby to bol vždy ten správny útvar?
13. Ako by sme museli upraviť náš program, ak by sme v súbore vo všetkých riadkoch (okrem prvého) vynechali medzery medzi číslami?
14. Istý hráč používaním nášho programu (nemenil program ani údaje v textovom súbore) vytvoril takéto hracie pole:



Odhadnite, akým postupom sa mu to mohlo podariť.

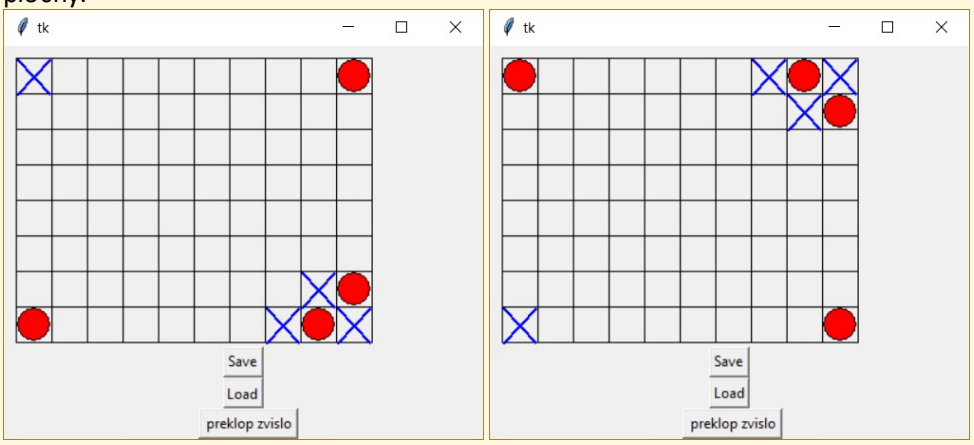
Úlohy:

1 Doplňte do programu tlačidlo `vymen`, ktoré vymení navzájom krížiky a krúžky (nielen na obrazovke, ale aj v zozname `plocha`).

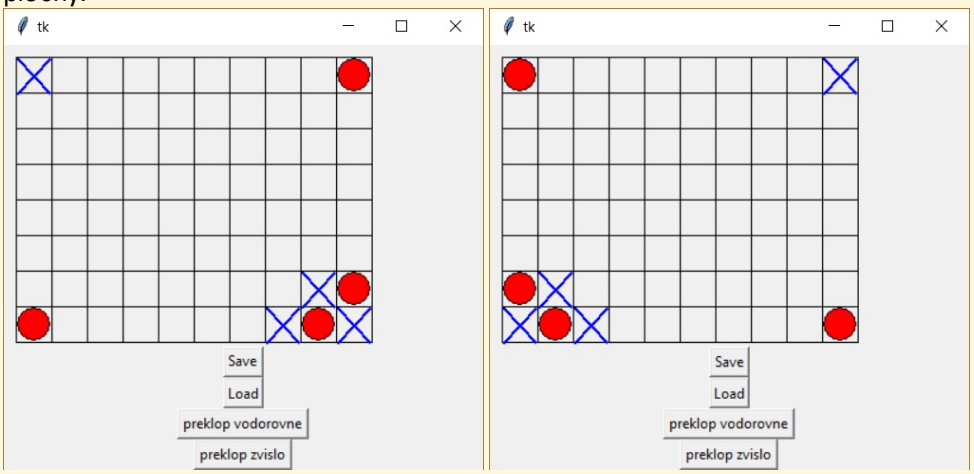
2 Doplňte do programu widget `scale` (posúvač), ktorým budeme meniť veľkosť jedného políčka. Hneď po zmene sa nám prekreslí obsah canvasu.

3 Vytvorte funkciu `pocty()`, ktorá spočíta v hracej ploche počet krížikov a počet krúžkov.

4 Doplňte do programu tlačidlo `preklop zvislo`, ktoré zrkadlovo otočí (preklopí) obsah hracej plochy.



5 Doplňte do programu tlačidlo `preklop vodorovne`, ktoré zrkadlovo otočí (preklopí) obsah hracej plochy.



6 Vytvorte program, ktorý overí, či v textovom súbore je zapísaný plán reálnej hry, teda či v ňom len niekto neprepísal v poznámkovom bloku niektoré hodnoty. Napríklad overte, či rozmery hracej plochy korešpondujú s údajmi v riadkoch, či počet krúžkov je rovný alebo o jedno väčší ako počet krížikov.

Otázky:

15. Ako by ste naprogramovali, aby program v hre Piškvorky zistil súvislý rad aspoň piatich symbolov v rovnakom smere?
16. Čo robia nasledovné funkcie `A()` a `B()`? Prečo vo funkcii `A()` je príkaz `global`, a vo funkcii `B()` nie je?

```
def A():
    global plocha
    plocha = plocha[::-1]
```

```
def B():
    for r in range(len(plocha)):
        plocha[r] = plocha[r][::-1]
```

17. Ako by sme vytvorili funkcie `A()` a `B()` (v predchádzajúcej otázke), ak by sme nepoznali alebo nemohli použiť rezy?

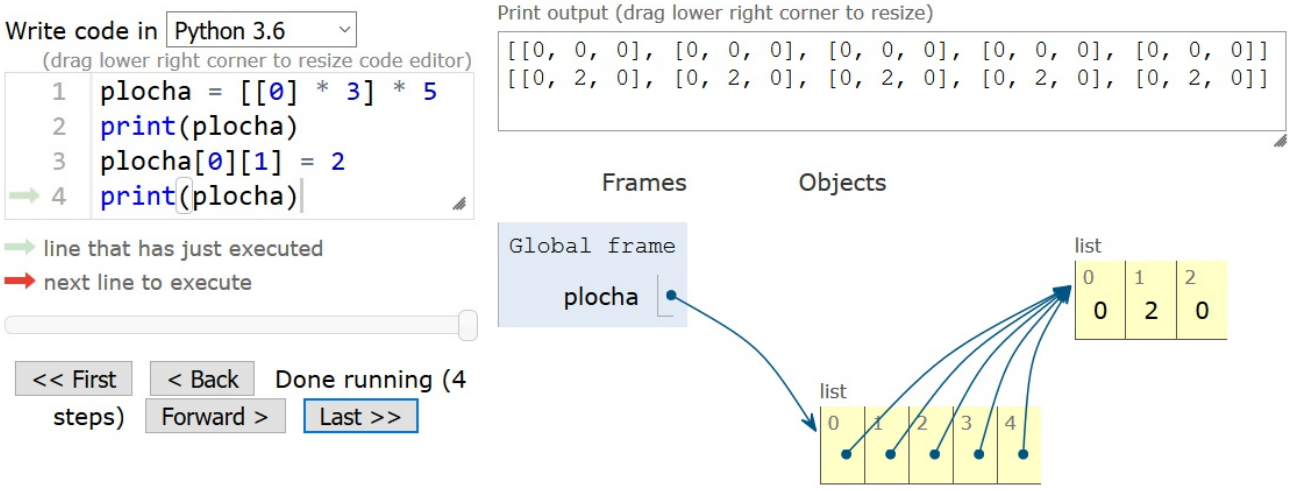
Vráťme sa k vytvoreniu zoznamu zoznamov. V programe **`p1-01a.py`** sme ho vytvárali pomocou **`for`** cyklu.

```
# p1-01a.py
plocha = []
for i in range(8):
    plocha.append([0]*10)
print(plocha)
```

Môžeme zoznam zoznamov vytvoriť aj jednoduchšie? Pozrime sa na takéto riešenie:

```
>>> plocha = [[0] * 3] * 5
>>> plocha
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> plocha[0][1] = 2
>>> plocha
[[0, 2, 0], [0, 2, 0], [0, 2, 0], [0, 2, 0], [0, 2, 0]]
>>>
```

Z pokusu v príkazovom riadku vidíme, že zoznam zoznamov sa na prvý pohľad vytvoril aj týmto spôsobom správne. Keď však zmeníme hodnotu jedného prvku niektorého zoznamu, tento prvok sa naraz zmení vo všetkých zoznamoch v zozname. Prečo to tak je? Zápis **`[0] * 3`** vytvoril jeden trojprvkový zoznam s nulami. Následne zápisom **`[[0] * 3] * 5`** sa vytvoril päťprvkový zoznam. Problémom však je, že ten obsahuje päťkrát referenciu (odkaz na miesto v pamäti), kde sa nachádza pôvodne vytvorený trojprvkový zoznam. Všetkých päť prvkov teda referuje (odkazuje) na **jeden a ten istý zoznam troch núl**. Keď v ktoromkoľvek z piatich zoznamov zmeníme jeden prvok, stále meníme ten istý trojprvkový zoznam. Najlepšie to uvidíme vo vizualizácii pomocou **`www.pythontutor.com`**



Rovnako nevhodné je aj toto riešenie:

Príkazovník

Zoznam zoznamov (dvojrozmerné pole)

```
zoznam = []
for i in range(5):
    zoznam.append([0] * 10)
zoznam[4][9] = hodnota
print(zoznam[3][5])
print(zoznam)
```

```
for riadok in range(5):
    for stlpec in range(10):
        print(zoznam[riadok][stlpec])
for riadok in zoznam:
    for prvok in riadok:
        print(prvok)
```

Rastrový obrázok

```
obr = tkinter.PhotoImage(file='súbor.png')
canvas['width'] = obr.width()
canvas['height'] = obr.height()
canvas.create_image(0, 0, image=obr)
```

```
farba = obr.get(súradnica_x, súradnica_y)
print(farba[0], farba[1], farba[2]) #RGB
obr.put('farba', (súradnica_x, súradnica_y))
canvas.update()
```

Korytnačia grafika

```
import turtle
k = turtle.Turtle()
k.forward(vzdialenosť)
k.right(uhol)
k.left(uhol)
k.penup()
k.pendown()
k.pensize(hrúbka_pera)
k.pencolor('farba')
k.backward(vzdialenosť)
k.dot(priemer_bodky)
k.setposition(súradnica_x, súradnica_y)
k.setheading(natočenie)
k.towards(súradnica_x, súradnica_y)
print(k.position())
print(k.heading())
```

```
ts = turtle.TurtleScreen(canvas)
k = turtle.RawTurtle(ts)
canvas.config(scrollregion=(0, 0, 800, 400))
ts.tracer(0)
ts.delay(0)
ts.update()
k.speed(0)
k.reset()
k.home()
k.clear()
k.fillcolor('farba')
k.begin_fill()
k.end_fill()
turtle.mainloop()
```

Rekurzia a frakftály

```
def meno_funkcie(parametre):
    if podmienka:
       ...príkazy...
       meno_funkcie(parametre)

def vypocet(n):
    if n < 1:
        return 0
    return n + vypocet(n - 1)
```

```
def meno_funkcie(parametre):
    if podmienka:
       triviálny prípad
    else:
       ...príkazy...
       meno_funkcie(parametre)
       ...príkazy...
       meno_funkcie(parametre)
       ...príkazy...
```

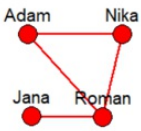
Triedy a objekty

```
class Názov_triedy:
    def __init__(self, parametre):
        self.atribút = parameter1
        ...príkazy...
    def metóda(self, parametre)
        ...príkazy...

inštancia = Názov_triedy(parametre_konštruktoru)
inštancia.metóda(parametre)
```

```
class Odvodená_trieda(Rodič):
    def __init__(self, parametre):
       Rodič.__init__(self, parametre)
       self.atribút = parameter1
       ...príkazy...
    def metóda(self, parametre)
       ...príkazy...
```

Grafy

```
vrcholy = ['Adam', 'Nika', 'Jana', 'Roman']
graf = [[0, 1, 0, 1],
        [1, 0, 0, 1],
        [0, 0, 0, 1],
        [1, 1, 1, 0]]
```




```
graf1 = {'Adam':{'Roman':1, 'Nika':1},
         'Nika':{'Adam':1, 'Roman':1},
         'Jana':{'Roman':1},
         'Roman':{'Adam':1, 'Jana':1, 'Nika':1}}
```

Ďalšie možnosti jazyka Python

```
premenná = hodnota1 if podmienka else hodnota2
zoznam = [výraz for prvok in zoznam]
zoznam = [výraz for prvok in zoznam if filtrujúca_podmienka]

def funkcia(parameter1, parameter2=inicializačná_hodnota)
    ...príkazy...

funkcia(hodnota1)
funkcia(hodnota1, hodnota2)
funkcia(hodnota1, parameter2=hodnota2)
```

```
try:
    ...príkazy...
except:
    príkazy spracovania výnimky
```