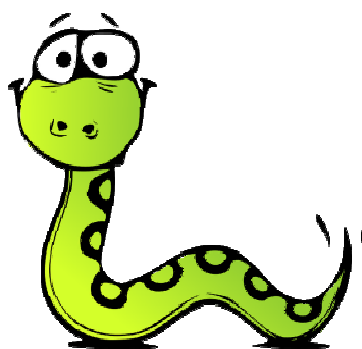




Programování v jazyce Python pro střední školy

Metodický list pro učitele
Lekce 20 – Kreslení myší



Andrej Blaho
Lubomír Salanci
Václav Šimandl

Cíle lekce

- Seznámit se se **souběhem procesů**, tj. s událostmi spojenými s tažením myši a stiskem tlačítka myši
- Naučit se svázat (`bind`) událost s určitým podprogramem
- Pochopit, že parametrem funkce (podprogramu) může být i název jiného podprogramu
- Seznámit se s parametrem, který obsahuje jiné proměnné jako své složky

Dovednosti

- Správné řazení podprogramů a příkazu `bind`, který je používá

Osvojená syntaktická pravidla

- Speciální zápis proměnných, které jsou složkami jiné proměnné, pomocí tečkové notace

Průběh výuky

1. Vytvoř nový program `mys.py` a přepiš do něj následující kód. Program poté spust'.

```
import tkinter

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

def klik(mys):
    print(mys.x, mys.y)

canvas.bind('<B1-Motion>', klik)
```

Přibyl zde nový příkaz `canvas.bind`, díky němuž bude od této chvíle grafická plocha vědět, co má udělat, když nad ní stiskneme levé tlačítko myši a myši potom táhneme. V textovém okně se začnou vypisovat dvojice celých čísel. Víš, jaká jsou to čísla?

V úloze se objevila nová konstrukce `canvas.bind('<B1-Motion>', klik)`. Pomocí takového zápisu **oznamujeme** grafické ploše, aby **sledovala** tažení myši se stisknutým levým tlačítkem (nestačí jen kliknout, je potřeba i táhnout). Od této chvíle se při každém tažení myši automaticky zavolá podprogram `klik`. Přesněji vyjádřeno, tímto zápisem zajistíme, že reakcí na událost tažení myši se stisknutým levým tlačítkem bude zavolání podprogramu `klik`.

Podprogram `klik` musí být zapsaný i s **jedním parametrem** – v této lekci jsme jej nazvali `mys` (mohli bychom zvolit libovolný název parametru, ale slovo `mys` bude asi nejčitelnější). Tento parametr obsahuje komplexní informace o události, k níž došlo (v tomto případě o události tažení myši se stisknutým levým tlačítkem). My však z tohoto parametru využijeme jen informaci o místě v grafické ploše, na němž se kliklo myši (přesněji, přes nějž se táhlo se stisknutým tlačítkem myši).

Zápis `mys.x` a `mys.y` v podprogramu `klik` označuje *x*-ovou a *y*-ovou souřadnici místa v grafické ploše, kde bylo kliknuto myší (přesněji, přes nějž se táhlo se stisknutým tlačítkem myši).

Pro úplnost uvádíme i další možnosti příkazu `canvas.bind` (žákům je však neprozrazujeme):

- '`<B1-Motion>`' tažení se stisknutým levým tlačítkem myši
- '`<B2-Motion>`' tažení se stisknutým středním tlačítkem myši
- '`<B3-Motion>`' tažení se stisknutým pravým tlačítkem myši
- '`<Motion>`' tažení bez ohledu na to, zda je nebo není stisknuté nějaké tlačítko myši

V dalších úlohách využijeme také některé z následujících možností (žákům je však v tuto chvíli neprozrazujeme):

- '`<ButtonPress-1>`' kliknutí levým tlačítkem myši
- '`<ButtonPress-2>`' kliknutí středním tlačítkem myši
- '`<ButtonPress-3>`' kliknutí pravým tlačítkem myši
- '`<ButtonPress>`' kliknutí libovolným tlačítkem myši

Pro každou z těchto situací můžeme vytvořit nějaký podprogram (s jedním parametrem `mys`) a spojit je s grafickou plochou pomocí příkazu `canvas.bind`. Potom například tažení se stisknutým levým tlačítkem myši může dělat něco úplně jiného než tažení se stisknutým pravým tlačítkem myši.

2. Zápis `mys.x` a `mys.y` v programu `mys.py` označuje *x*-ovou a *y*-ovou souřadnici místa v grafické ploše, kde jsi klikl(a). Namísto příkazu `print` v podprogramu `klik` použij příkaz `canvas.create_text`, pomocí něhož vykreslí znak '*'. Znak by se měl vykreslit na pozici, kde jsi klikl(a) myší:

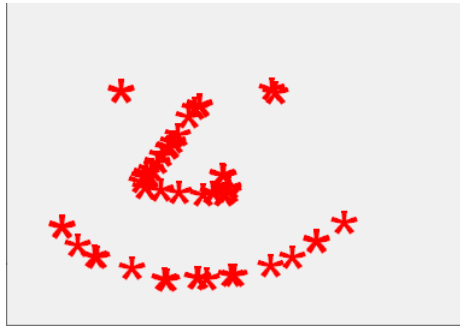
```
import tkinter

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

def klik(mys):
    .....

canvas.bind('<B1-Motion>', klik)
```

Program nyní při tažení myši kreslí malé hvězdičky. Pomocí parametrů `font='...'` a `fill='...'` můžeš velikost těchto znaků zvětšit na 50 a změnit jejich barvu na červenou. Jestli jsi postupoval(a) správně, mělo by být možné vytvořit například takovýto obrázek:



Zkus takto nakreslit i něco zajímavějšího a výsledným obrázkem se pochlub spolužákovi.

Řešení:

```
import tkinter

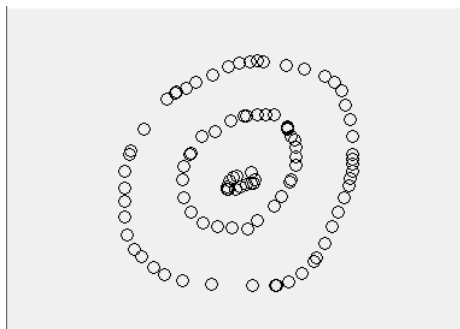
canvas = tkinter.Canvas()
canvas.pack()

def klik(mys):
    canvas.create_text(mys.x, mys.y, text='*',
                      font='arial 50', fill='red')

canvas.bind('<B1-Motion>', klik)
```

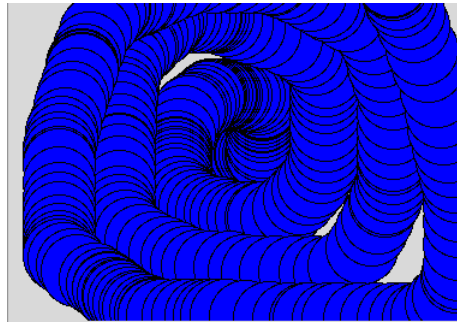
Místo vypisování hvězdičky '*' mohou žáci zkusit vypisovat i jiné texty, například 'O', '/' nebo slovo 'PYTHON'.

3. Pomocí příkazu `canvas.create_oval(x-5, y-5, x+5, y+5)` umíš nakreslit malý kruh se středem `[x, y]` a s poloměrem 5. V programu `mys.py` místo příkazu `canvas.create_text` vhodně použij příkaz pro kreslení malého kruhu. Nyní by se místo znaků '*' měly na místa, kudy jsi táhl(a) myši, kreslit kruhy podobně jako na následujícím obrázku:



Ted' změň poloměr kreslených kruhů například na hodnotu 30. Jak se změní kreslené kruhy?

Co je ještě nutné změnit, aby bylo možné vytvořit kresbu jako na následujícím obrázku? Svou domněnku ověř.



Řešení:

```
import tkinter

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

def klik(mys):
    canvas.create_oval(mys.x-5, mys.y-5, mys.x+5, mys.y+5)

canvas.bind('<B1-Motion>', klik)
```

Řešení by bylo možné zapsat i takovýmto způsobem:

```
import tkinter

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

def klik(mys):
    x = mys.x
    y = mys.y
    canvas.create_oval(x-5, y-5, x+5, y+5)

canvas.bind('<B1-Motion>', klik)
```

Velké modré kruhy nakreslíme pomocí následujícího kódu:

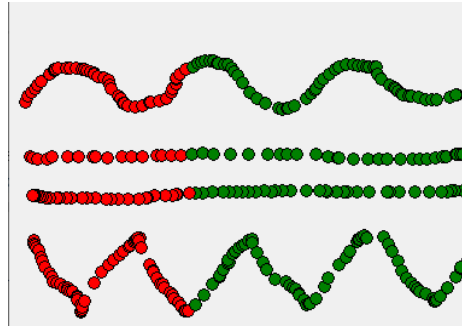
```
import tkinter

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

def klik(mys):
    canvas.create_oval(mys.x-30, mys.y-30, mys.x+30,
                      mys.y+30, fill='blue')

canvas.bind('<B1-Motion>', klik)
```

4. Podprogram `klik` v programu `mys.py` se ještě předtím, než nakreslí barevný kroužek, může pomocí příkazu větvení rozhodnout, jestli bude kreslit červený nebo zelený kroužek. Uprav podprogram `klik` tak, aby se kroužky kreslily červeně, pokud je jejich `x`-ová souřadnice menší než 150; jinak se kreslily zeleně. Poloměr všech kroužků bude 5. Příklad fungování programu můžeš vidět na následujícím obrázku:



Řešení:

```
import tkinter

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

def klik(mys):
    if mys.x < 150:
        canvas.create_oval(mys.x-5, mys.y-5, mys.x+5,
                           mys.y+5, fill='red')
    else:
        canvas.create_oval(mys.x-5, mys.y-5, mys.x+5,
                           mys.y+5, fill='green')

canvas.bind('<B1-Motion>', klik)
```

Správným řešením je i takovýto kód:

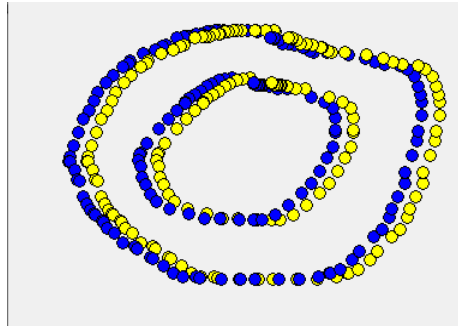
```
import tkinter

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

def klik(mys):
    if mys.x < 150:
        barva = 'red'
    else:
        barva = 'green'
    canvas.create_oval(mys.x-5, mys.y-5, mys.x+5, mys.y+5,
                       fill=barva)

canvas.bind('<B1-Motion>', klik)
```

5. Vytvoř nový program `dvojite.py` a zkopíruj do něj kód z programu `mys.py`. Uprav v programu `dvojite.py` podprogram `klik` tak, aby kreslil všechny kroužky modře s poloměrem 5. Zajisti, aby se kromě modrého kroužku nakreslil i stejně velký žlutý kroužek. Jeho střed však bude o 15 posunutý vpravo (k x-ové souřadnici přičteš 15). Při tažení myši by měl vzniknout efekt jako na následujícím obrázku:



Řešení:

```
import tkinter

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

def klik(mys):
    canvas.create_oval(mys.x-5, mys.y-5, mys.x+5, mys.y+5,
                      fill='blue')
    canvas.create_oval(mys.x-5+15, mys.y-5, mys.x+5+15,
                      mys.y+5, fill='yellow')

canvas.bind('<B1-Motion>', klik)
```

Podprogram `klik` je možné zapsat například též tímto způsobem:

```
def klik(mys):
    x = mys.x
    y = mys.y
    canvas.create_oval(x-5, y-5, x+5, y+5, fill='blue')
    x = x + 15
    canvas.create_oval(x-5, y-5, x+5, y+5, fill='yellow')
```

6. Vytvoř si nový program odstranit.py a zkopíruj do něj kód z programu dvojite.py. Přepiš do programu odstranit.py následující žlutě označený kód:

```
import tkinter

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

def klik(mys):
    ...
    ...

def smaz(mys):
    canvas.delete('all')

canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

Nyní by mělo vše fungovat stejně, ale program bude také reagovat na situaci, kdy do grafické plochy klikneš pravým tlačítkem myši. Zkus něco do plochy nakreslit a potom klikni do plochy pravým tlačítkem. Můžeš to opakovat i vícekrát. Diskutuj se spolužákem, co se po kliknutí pravým tlačítkem myši stalo.

Použili jsme tu nový příkaz `canvas.delete('all')`, pomocí kterého se z grafické plochy vymaže doposud vytvořená kresba.

Kompletní program nyní vypadá takto:

```
import tkinter

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

def klik(mys):
    canvas.create_oval(mys.x-5, mys.y-5, mys.x+5, mys.y+5,
                       fill='blue')
    canvas.create_oval(mys.x-5+15, mys.y-5, mys.x+5+15,
                       mys.y+5, fill='yellow')

def smaz(mys):
    canvas.delete('all')

canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

Pomocí příkazu `canvas.bind('<ButtonPress-3>', smaz)` bude grafická plocha reagovat i na kliknutí pravým tlačítkem myši (opravdu na kliknutí, tedy nikoliv na tažení se stisknutým tlačítkem). Při každém takovém kliknutí se v tomto případě zavolá podprogram `smaz`.

Číslice 3 v zápisu `'<ButtonPress-3>'` označuje pravé tlačítko myši. Kdyby se nahradila hodnotou 1, grafická plocha by se smazala při každém kliknutí levým tlačítkem myši a tedy před každým tažením při kreslení. Žákům tuto informaci můžeme, avšak nemusíme prozradit.

7. Nyní se naučíme používat nový grafický příkaz `canvas.create_line(x1, y1, x2, y2)`. Pomocí něho lze nakreslit jednoduchou čáru (úsečku) z bodu `[x1, y1]` do bodu `[x2, y2]`. Vytvoř nový program `paprsky.py` a zkopíruj do něj kód z programu `odstranit.py`. V programu `paprsky.py` uprav podprogram `klik` podle následujícího kódu (změny jsou vyznačeny žlutě). Program poté spust'.

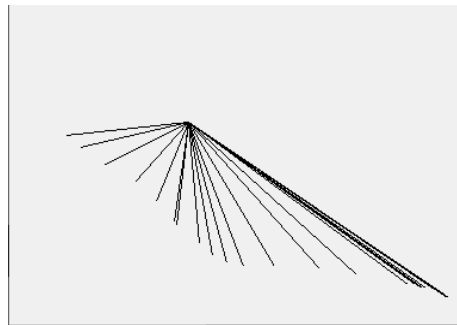
```
import tkinter

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

def klik(mys):
    canvas.create_line(150, 100, mys.x, mys.y)
def smaz(mys):
    canvas.delete('all')

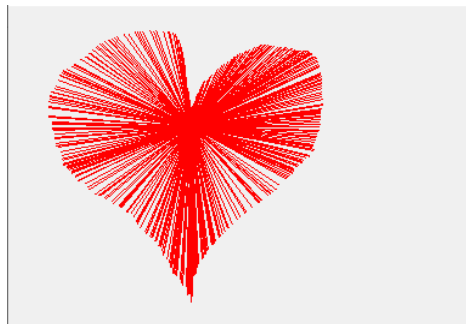
canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

Když program spustíš, můžeš s jeho pomocí vytvořit například takovýto obrázek:



Při tažení myši se kreslí úsečky z bodu `[150, 100]` do aktuální pozice myši. Proto mají všechny tyto úsečky společný jeden vrchol.

Dokážeš nakreslit červené srdce jako je na následujícím obrázku? Barvu úsečky nastavíš stejně jako například barvu textu v příkazu `canvas.create_text` nebo barvu výplně v příkazu `canvas.create_oval`.



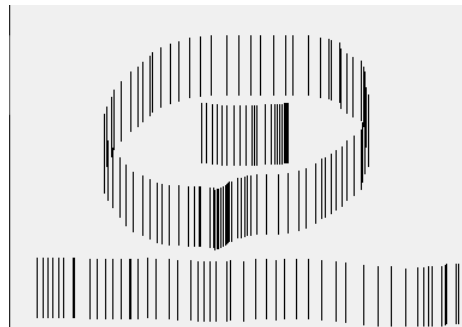
V příkazu `canvas.create_line` můžeme měnit barvu úseček pomocí pojmenovaného parametru `fill='red'`. Pro kreslení červených úseček je potřeba podprogram `klik` upravit následujícím způsobem:

```
def klik(mys):  
    canvas.create_line(150, 100, mys.x, mys.y, fill='red')
```

Kdybychom chtěli příkazem `canvas.create_line` kreslit různě silné úsečky, je možné sílu úseček měnit pomocí pojmenovaného parametru `width=5`.

8. Vytvoř nový program `spendliky.py` a zkopíruj do něj kód z programu `paprsky.py`. Uprav kód programu `spendliky.py` tak, aby každá úsečka začínala na pozici myši (`mys.x, mys.y`) a končila v bodě posunutém o 50 směrem vzhůru (`y`-ová souřadnice konce úsečky bude o 50 zmenšená).

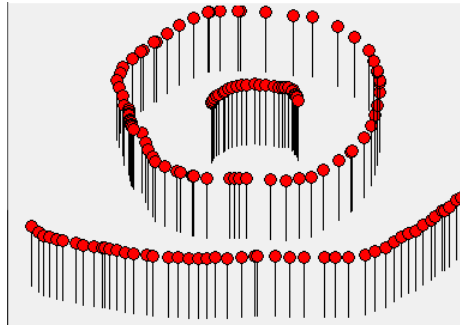
Pomocí programu můžeš nakreslit například takovýto obrázek:



Řešení:

```
import tkinter  
  
canvas = tkinter.Canvas()  
canvas.pack()  
  
def klik(mys):  
    canvas.create_line(mys.x, mys.y, mys.x, mys.y-50)  
  
def smaz(mys):  
    canvas.delete('all')  
  
canvas.bind('<B1-Motion>', klik)  
canvas.bind('<ButtonPress-3>', smaz)
```

9. Uprav program `spendliky.py` tak, aby byl na konci každé úsečky nakreslen červený kroužek. Jestli jsi postupoval správně, mělo by být možné pomocí programu nakreslit například takovýto obrázek:



Řešení:

```
import tkinter

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

def klik(mys):
    canvas.create_line(mys.x, mys.y, mys.x, mys.y-50)
    canvas.create_oval(mys.x-5, mys.y-5-50, mys.x+5,
                      mys.y+5-50, fill='red')

def smaz(mys):
    canvas.delete('all')

canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

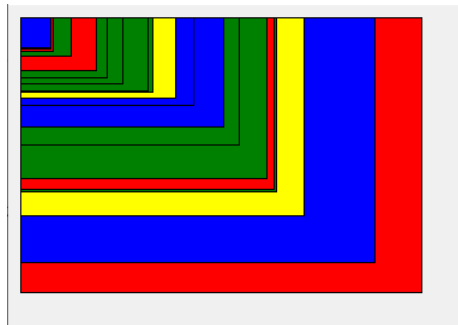
Podprogram `klik` je možné zapsat i tímto způsobem:

```
def klik(mys):
    x1 = mys.x
    y1 = mys.y
    x2 = x1
    y2 = y1 - 50
    canvas.create_line(x1, y1, x2, y2)
    canvas.create_oval(x2-5, y2-5, x2+5, y2+5, fill='red')
```

10. Vytvoř nový program `mys_obdelniky.py` a zkopíruj do něj kód z programu `spendliky.py`. V programu `mys_obdelniky.py` uprav podprogram `klik` tak, aby byl schopen nakreslit obdélník, jehož levý horní roh bude mít vždy souřadnice `[10, 10]` a pravý dolní roh bude na aktuální pozici myši `[mys.x, mys.y]`. Tento obdélník bude vybarvený náhodně zvolenou barvou. Pro náhodný výběr jedné ze čtyř barev můžeš využít následující kód:

```
barva = random.choice(['red', 'yellow', 'blue', 'green'])
```

Při tažení myši by se ti mělo podařit vytvořit podobnou kresbu jako na následujícím obrázku:



Řešení:

```
import tkinter
import random

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

def klik(mys):
    barva = random.choice(['red', 'yellow', 'blue',
                           'green'])
    canvas.create_rectangle(10, 10, mys.x, mys.y,
                           fill=barva)

def smaz(mys):
    canvas.delete('all')

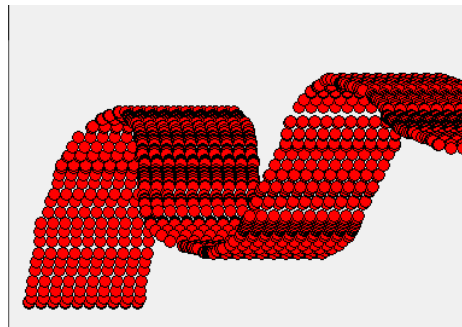
canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

11. Vrať se k programu `dvojite.py` a uprav v něm kód podprogramu `klik` tak, aby byl schopen kreslit 10 červených kroužků. Tyto kroužky budou nakreslené těsně vedle sebe:

- První kroužek bude na pozici myši
- Každý další bude mít svůj střed posunutý o 10 vpravo oproti předchozímu kroužku (tedy x-ovou souřadnici zvětší o 10)

Vykreslení jednotlivých kroužků v podprogramu `klik` zajisti pomocí cyklu.

Při tažení myši by se ti mohlo podařit vytvořit podobnou kresbu jako na následujícím obrázku:



Řešení:

```
import tkinter

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

def klik(mys):
    x = mys.x
    y = mys.y
    for i in range(10):
        canvas.create_oval(x-5, y-5, x+5, y+5, fill='red')
        x = x + 10

def smaz(mys):
    canvas.delete('all')

canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

Podprogram `klik` můžeme méně čitelně zapsat též tímto způsobem:

```
def klik(mys):
    for i in range(10):
        canvas.create_oval(mys.x-5+i*10, mys.y-5,
                           mys.x+5+i*10, mys.y+5, fill='red')
```

12* Vytvoř nový program `sprej.py` a zkopíruj do něj kód z programu `dvojite.py`. Nyní budeš upravovat program `sprej.py` tak, aby vznikl efekt spreje.

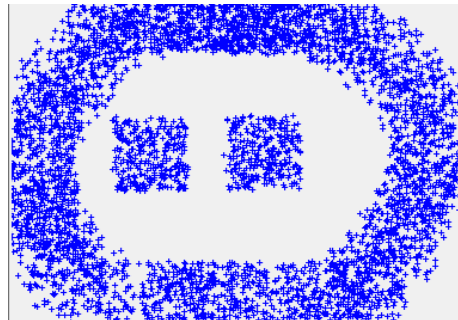
Nejprve před příkaz `canvas.bind` (mimo podprogram `klik`) zapiš kód

```
barva = 'blue'
```

Dále budeš upravovat podprogram `klik`:

- Nejprve se náhodně zvolí dvě čísla `dx` a `dy` z intervalu `<-30, 30>` (pomocí příkazu `random.randint(-30, 30)`)
- Tato dvojice čísel vyjadřuje posunutí nakreslené barevné tečky oproti pozici myši. Tečka tedy bude kreslena na pozici `[mys.x+dx, mys.y+dy]`
- Na tuto posunutou pozici nakresli tečku pomocí příkazu `canvas.create_text` jako znak `'+'`. Jako barvu kresleného znaku `'+'` použij proměnnou `barva`
- Postup z předchozích tří bodů při každém kliknutí zopakuj pomocí cyklu 50krát, čímž se nakreslí 50 malých znaků `'+'`, které nebudou příliš daleko od místa, kde jsi klikl

Program vyzkoušej. Jestli jsi postupoval správně, může vzniknout například takovýto obrázek:



Když budeš chtít změnit barvu spreje, stačí do příkazového řádku zapsat kód:

```
barva = 'yellow'
```

Od tohoto okamžiku bude sprej kreslit žlutou barvou.

Řešení:

```
import tkinter
import random

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

def klik(mys):
    for i in range(50):
        dx = random.randint(-30, 30)
        dy = random.randint(-30, 30)
        canvas.create_text(mys.x + dx, mys.y + dy, text='+',
                           fill=barva)

def smaz(mys):
    canvas.delete('all')

barva = 'blue'
canvas.bind('<B1-Motion>', klik)
canvas.bind('<ButtonPress-3>', smaz)
```

Pokud není barva kreslených teček nastavována v podprogramu `klik`, ale je to globální proměnná `barva`, můžeme její hodnotu měnit i při běhu programu. Jestliže do příkazovém řádku zapíšeme kód

```
>>> barva = 'yellow'
```

bude od daného okamžiku sprej kreslit žlutou barvou.

Touto lekcí končí celá učebnice. Žáci se při řešení úloh jednotlivých lekcí naučili v Pythonu pracovat s číselnými výrazy, používat proměnné a vytvářet textové výpisy v interaktivním režimu i v rámci jimi vytvářených programů. Naučili se zapisovat příkazy ve správném pořadí, využívat náhodně generovaná čísla, kreslit obrazce (čtverce, obdélníky, kruhy, elipsy a texty) do grafického okna. Osvojili si vytváření podprogramů bez parametrů i s parametry, používání cyklů pro opakované provádění příkazů a využívání proměnné cyklu pro výpočet hodnot jiných proměnných v těle cyklu. Naučili se používat příkazy větvení, a to i v kombinaci s cykly a kreslením na plátno. V této poslední lekci se seznámili s kreslením pomocí myši.

Seznam témat, kterým se věnovala tato učebnice, pochopitelně není vyčerpávající. Budete-li mít zájem, můžete se se žáky věnovat dalším tématům, mezi něž patří například:

- vytváření a volání podprogramů s více parametry a podprogramů s návratovou hodnotou
- vzájemné převody mezi datovými typy vyjadřujícími text, celá a desetinná čísla
- načítání vstupních dat zadaných do příkazové řádky uživatelem
- práce se seznamy a procházení prvků seznamu
- ošetřování výjimek