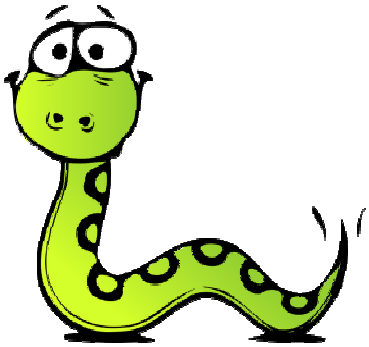
Základy programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 11 – Program s opakováním



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

beta verze - 2019

Cíle lekce

- Naučit se zajistit opakování skupiny příkazů pomocí for cyklu
- Umět rozpoznat části, které se mají opakovat, od těch, které se opakovat nemají
- Zvládnout zapisovat pomocí for cyklu programy, ve kterých se opakují výpisy, přiřazení náhodných hodnot a vykreslování

Dovednosti

- Odsazování skupiny příkazů (tělo cyklu)
- Kreslení pomocného obrázku na papír v případě řešení náročnější úlohy

Osvojená syntaktická pravidla

- Zápis příkazu cyklu `for i in range(n)`, kde `n` je počet opakování
- Tělo cyklu nesmí být prázdné a všechny řádky musí být odsazené od kraje o stejný počet mezer (nejlépe o 4 mezery)
- Nastavení velikosti písma v příkazu `create_text` pomocí pojmenovaného parametru `font=`

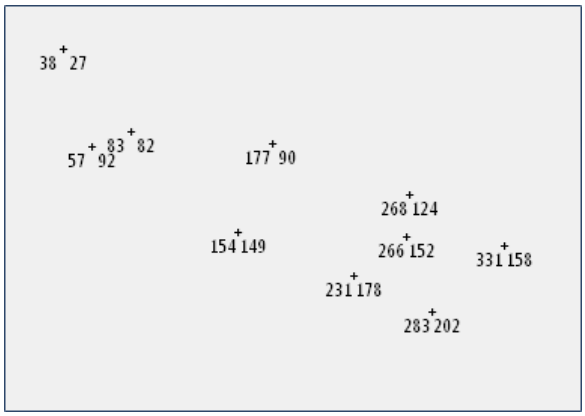
Poznámky

- V této úvodní lekci jsou řešeny úlohy jen s jedním for cyklem, v jehož těle jsou pouze jednoduché příkazy (přiřazení, výpisy pomocí příkazu `print` a grafické příkazy `create_rectangle` a `create_text`)
- Počet opakování je konstantní a v těle cyklu není používána proměnná cyklu
- Zatím se neřeší akumulování hodnot z předchozích průchodů cyklem (např. součet hodnot)

Průběh výuky

Začínáme opakovacími úlohami a sledujeme jimi i další cíl – přípravu na výuku cyklů:

1. Běháme po louce a zaznamenáváme si naši GPS pozici. Vytvoř nový program `gps.py` a v něm podprogram `gps`, který vygeneruje náhodné souřadnice `x`, `y` představující GPS pozici. Na tomto místě nakreslí značku `'+'` a pod ni vypíše danou pozici – čísla `x`, `y`. Po deseti zavoláních podprogramu `gps` můžeš dostat například takovýto výsledek:



Řešení:

```
import tkinter
import random

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

def gps():
    x = random.randint(20, 360)
    y = random.randint(20, 240)
    canvas.create_text(x, y, text='+')
    canvas.create_text(x - 10, y + 10, text=x)
    canvas.create_text(x + 10, y + 10, text=y)

gps()
gps()
gps()
gps()
gps()
gps()
gps()
gps()
gps()
gps()
```

Program vypisuje souřadnice `x` a `y` pomocí dvou volání příkazu `create_text`. Aby tato dvě čísla byla vypsána vedle sebe, je v příkazu `create_text` posunuta jejich `x`-ová souřadnice vlevo resp. vpravo. Příkaz `create_text` by zvládl vypsat obě čísla najednou, avšak tato čísla by musela být zapsána v kulatých závorkách. Podprogram `gps` by pak bylo možné zapsat takto:

```
def gps():
    x = random.randint(20, 360)
    y = random.randint(20, 240)
    canvas.create_text(x, y, text='+')
    canvas.create_text(x, y + 10, text=(x, y))
```

Touto „fintou“ žáky zbytečně nezatěžujeme, ale když se někdo z nich zeptá, můžeme mu tento postup prozradit.

Výpis obou čísel pomocí jediného volání příkazu `create_text` by bylo možné řešit též na základě převodu čísel na řetězce a jejich následným zřetězením, ale to je ještě náročnější zápis.

2. Vytvoř program `tesim_se.py` bez grafické plochy, který pomocí příkazu `print` vypíše text `'Těším se na prázdniny'` pětkrát pod sebe.

Řešení:

```
print('Těším se na prázdniny')
print('Těším se na prázdniny')
print('Těším se na prázdniny')
print('Těším se na prázdniny')
print('Těším se na prázdniny')
```

Žáky necháme upravit a vyzkoušet následující řešení – minimalizujeme výklad, nepočítáme s tím, že budeme něco vysvětlovat:

3. V obou předchozích programech jsi měl vícekrát nakopírované příkazy `gps()` nebo `print(...)`. Abys je nemusel opakovaně kopírovat, můžeš to zapsat jednodušeji. Kód programu `tesim_se.py` uprav stejně, jako je uvedeno níže:

Dvojtečka je velmi důležitá

```
for i in range(5):
    print('Teším se na prázdniny')
```

Příkaz nech odsazený od kraje (Python tam automaticky vložil 4 mezery)

Tento program spusť a urči, co program vykonal.

Program vypíše:

```
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
```

4. Zkus místo čísla `5` dát číslo `10` a program znovu spusť. Experimentuj i s jinými čísly, například `1`, `100` a podobně. Urči, co je tímto číslem ovlivňováno.

Žáky necháme experimentovat. Předpokládáme, že žáci velmi rychle odhalí, že číslo v závorkách u příkazu `range` označuje počet opakování od okraje odsazeného příkazu.

V Pythonu se doporučuje odsazovat vnořené příkazy přesně o 4 mezery, ačkoliv by program fungoval i s odsazením o libovolný počet mezer větší než 0.

V další úloze je demonstrováno, že tělo cyklu může obsahovat více příkazů:

5. Uprav program stejně, jako je uvedeno níže a spusť jej:

```
for i in range(5):
    print('Těším se na prázdniny')
    print('====================')
```

Jestli jsi postupoval správně, po spuštění uvidíš:

```
Těším se na prázdniny
====================
Těším se na prázdniny
====================
Těším se na prázdniny
====================
Těším se na prázdniny
====================
Těším se na prázdniny
====================
>>>
```

Jak program funguje?

slovem `for` začíná příkaz cyklu

toto číslo znamená počet opakovaní

```
for i in range(5):
    print('Těším se na prázdniny')
    print('====================')
```

tělo cyklu – tyto příkazy se vykonají 5-krát

Je žádoucí, aby si žáci na základě experimentování uvědomili, jak se program vykoná / bude chovat, když některý příkaz nebude odsazený od kraje:

6. Je důležité odsadit od kraje příkazy, které tvoří tělo cyklu. Vyzkoušej, co vypíše takto upravený program:

```
for i in range(5):
    print('Těším se na prázdniny')
print('====================')
```

Diskutuj se svým spolužákem, jaký je rozdíl v zápisu kódu programu z úlohy 6 oproti úloze 5. Potom určete, jak se tento rozdíl projevil po spuštění programu.

Program vypíše:

```
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
Těším se na prázdniny
====================
```

Tělo cyklu (posloupnost od kraje odsazených řádků) končí na prvním neodsazeném řádku. Všechny řádky těla cyklu musí být odsazené o stejný počet mezer. Do těla cyklu můžeme vložit i prázdné řádky - ty se budou ignorovat.

Tělo cyklu musí obsahovat aspoň jeden neprázdný řádek, nesmí tedy být prázdné. Následující kód ukazuje příklad chybného zápisu:

```
for i in range(5):

print('konec')
```

Doposud jsme cyklus používali jen při vypisování textu, nyní začneme používat cyklus v kombinaci s grafikou:

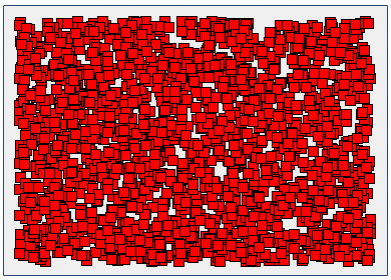
7. Otevři program `gps.py`, který jsi vyřešil v 1. úloze, a opakované volání podprogramu `gps()` zapiš pomocí `for` cyklu. Jestli jsi postupoval správně, měl bys po spuštění programu na obrazovce vidět opět deset GPS pozic.

Řešení:

```
import tkinter
import random

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

def gps():
    x = random.randint(20, 360)
    y = random.randint(20, 240)
    canvas.create_text(x, y, text='+')
    canvas.create_text(x - 10, y + 10, text=x)
    canvas.create_text(x + 10, y + 10, text=y)

for i in range(10):
    gps()
```

Následují úlohy na trénování:

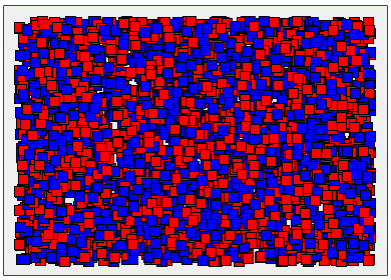
8. Vytvoř nový program `opakovany_ctverec.py` a v něm podprogram `cerveny_ctverec()`. Ten nakreslí na grafickou plochu na náhodné souřadnice červený čtverec se stranou délky `10`. Použij `for` cyklus na to, abys nakreslil `2000` červených čtverců. Výsledek může vypadat například jako na následujícím obrázku:



Řešení:

```
import tkinter
import random

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

def cerveny_ctverec():
    x = random.randint(10, 360)
    y = random.randint(10, 250)
    canvas.create_rectangle(x, y, x + 10, y + 10, fill='red')

for i in range(2000):
    cerveny_ctverec()
```

9. Doplň do programu `opakovany_ctverec.py` podprogram `modry_ctverec()`. Tento podprogram bude kreslit na náhodné souřadnice modrý čtverec se stranou délky `10`. Zajisti, aby tělo cyklu obsahovalo volání podprogramu `cerveny_ctverec()` i podprogramu `modry_ctverec()`. Výsledek může vypadat například jako na následujícím obrázku:



Řešení:

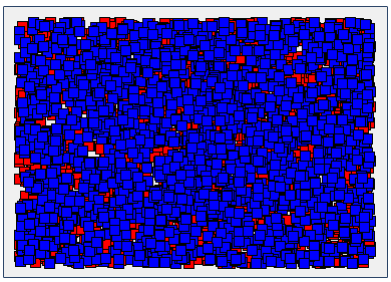
```
import tkinter
import random

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

def cerveny_ctverec():
    x = random.randint(10, 360)
    y = random.randint(10, 250)
    canvas.create_rectangle(x, y, x + 10, y + 10, fill='red')

def modry_ctverec():
    x = random.randint(10, 360)
    y = random.randint(10, 250)
    canvas.create_rectangle(x, y, x + 10, y + 10, fill='blue')

for i in range(2000):
    cerveny_ctverec()
    modry_ctverec()
```

10. Uprav kód programu podle následujícího vzoru tak, aby v něm byly dva cykly za sebou.

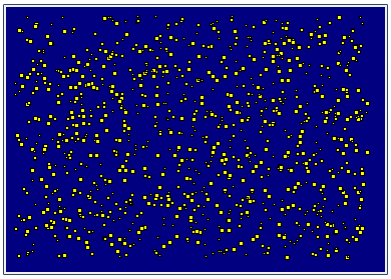
```
for i in range(2000):
    cerveny_ctverec()
for i in range(2000):
    modry_ctverec()
```

Získal jsi stejný obrázek jako předtím? Pokud ne, diskutuj se svým spolužákem, proč je obrázek jiný.

Program nejdříve nakreslí červené čtverce. Potom nakreslí modré. Proto uvidíme jen málo červených ploch, a mnoho modrých:



11. Vytvoř nový program `obloha.py`, který pomocí grafických příkazů nakreslí hvězdnou oblohu:



Návod:

- Napiš podprogram `hvezdicka`, který nakreslí na náhodnou pozici malý žlutý čtvereček. Velikost jeho strany bude náhodné číslo z rozsahu od 2 do 4.
- Tmavomodrou oblohu nakresli jako velký obdélník s barvou `'navy'`.
- Potom zavolej tisíckrát podprogram `hvezdicka`.

Řešení:

```
import tkinter
import random

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

def hvezdicka():
    x = random.randint(10, 360)
    y = random.randint(10, 250)
    a = random.randint(2, 4)
    canvas.create_rectangle(x, y, x + a, y + a, fill='yellow')

canvas.create_rectangle(0, 0, 380, 270, fill='navy')
for i in range(1000):
    hvezdicka()
```

O vykonávání příkazů a jejich pořadí je potřeba se žáky diskutovat. Lze je například navést, aby vyměnili pořadí cyklu a kreslení modrého obdélníku (jako v následujícím kódu), a ptát se, proč uvidí jen modrou plochu:

```
for i in range(1000):
    hvezdicka()
canvas.create_rectangle(0, 0, 380, 270, fill='navy')
```

Modrou oblohu lze nakreslit i upravením příkazu, pomocí kterého se inicializuje grafická plocha:

```
canvas = tkinter.Canvas(bg='navy')
```

Potom není potřebný příkaz:

```
canvas.create_rectangle(0, 0, 380, 270, fill='navy')
```

Je na nás, zda takovéto řešení žáků prozradíme vzhledem k tomu, že se jedná o technický detail. Někteří žáci si však toto řešení mohou oblíbit.

12. Je dán následující program:

```
import random

for i in range(5):
    n = random.randint(1, 100)
    print('bylo vylosováno číslo', n)
```

Diskutuj se svým spolužákem, co program vykoná. Potom na počítači za použití Pythonu zkontroluj, zda byla tvá domněnka správná.

Program vypíše (pravděpodobně s jinými čísly) následující:

```
bylo vylosováno číslo 52
bylo vylosováno číslo 26
bylo vylosováno číslo 72
bylo vylosováno číslo 80
bylo vylosováno číslo 48
```

Žáků se následně můžeme zeptat, co program vykonává. Možné slovní popisy mohou být například:

- „Program v cyklu vymyslí náhodné číslo, které vypíše i s nějakým textem, a toto opakuje 5 krát.“
- „Program vypíše 5 vylosovaných čísel z rozsahu od 1 do 100.“

Nechceme však, aby žáci program jen přečetli: „for i in range(5), do n přiřaď ...“.

Následují úlohy na trénování cyklů:

13. Napiš program `dve_kostky.py`, který simuluje hody dvěma kostkami. Zapiš pomocí `for` cyklu pět hodů, kdy se v těle cyklu do dvou proměnných přiřadí dvě náhodná čísla, ta se vypíšou a vypíše se i jejich součet. Výpis může vypadat například takto:

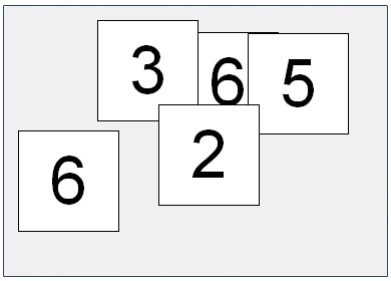
```
Na první kostce padlo číslo 4
Na druhé kostce padlo číslo 3
Součet obou čísel je 7

Na první kostce padlo číslo 2
Na druhé kostce padlo číslo 4
Součet obou čísel je 6

Na první kostce padlo číslo 5
Na druhé kostce padlo číslo 2
Součet obou čísel je 7
```

```
Na první kostce padlo číslo 3
Na druhé kostce padlo číslo 1
Součet obou čísel je 4

Na první kostce padlo číslo 1
Na druhé kostce padlo číslo 4
Součet obou čísel je 5
```

Řešení:

```
import random

for i in range(5):
    a = random.randint(1, 6)
    b = random.randint(1, 6)
    print('Na první kostce padlo číslo', a)
    print('Na druhé kostce padlo číslo', b)
    print('Součet obou čísel je', a + b)
    print()
```

14. Napiš program `kostky_s_cisly.py`, který pomocí grafických příkazů nakreslí na náhodných místech pět hracích kostek. Kostku nakresli jako čtverec, do kterého je vepsané náhodně vygenerované číslo. Použij `for` cyklus, ve kterém budou všechna přiřazení i kreslení.



Kdybys chtěl nakreslit kostku s velkými čísly jako na obrázku výše, přidej do příkazu `create_text` žlutě zvýrazněný kód:

```
canvas.create_text(x, y, text=random.randint(1, 6),
  font='arial 50')
```

Řešení:

```
import tkinter
import random

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

for i in range(5):
    x = random.randint(60, 330)
    y = random.randint(60, 210)
    canvas.create_rectangle(x - 50, y - 50, x + 50, y + 50,
       fill='white')
    canvas.create_text(x, y, text=random.randint(1, 6),
       font='arial 50')
```

Parametr `font` v příkazu `canvas.create_text` nemusíme komentovat – je to pro žáky bonus.

Informace pro učitele:

- Do parametru lze napsat i jiný název písma, například:

  ```
  font='Courier 22'
  font='Times 22'
  font='Impact 22'
  ```

- název písma musí být jednoslovný – ne `'times new roman 50'`, ale `('times new roman', 50)` . Tento zápis žákům však určitě neprozrazujeme.

V poslední úloze může být náročné vymyslet vzorec pro generování souřadnic, aby čtverečky ležely v mřížce.

15* Vytvoř nový program `qr_kod.py`, který bude představovat generátor náhodného QR kódu a který bude schopen generovat podobný QR kód jako na obrázku níže:

Obrázek se skládá z velkého počtu černých čtverečků. Každý má délku strany 10 a je nakreslený v jednom náhodně vybraném políčku mřížky, která obsahuje 21 x 21 políček.

Když to bude potřeba, žákům pomůžeme s odvozením vzorců. Lze použít například následující nápovědy:

- „Čtverečky budou ležet v mřížce.“ – nakreslíme si ji
- „Jak generovat pozice? Zkusme vymyslet generování x-ové souřadnice.“
- „Náhodná čísla umíme generovat pomocí `randint` ... Ale jak zabezpečit rozestupy?“
- „Jak generovat čísla `10, 20, 30, 40 ... 210`?“
- Kreslíme

Řešení:

```
import tkinter
import random

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

for i in range(250):
    x = random.randint(1, 21) * 10
    y = random.randint(1, 21) * 10
    canvas.create_rectangle(x, y, x + 10, y + 10, fill='black')
```