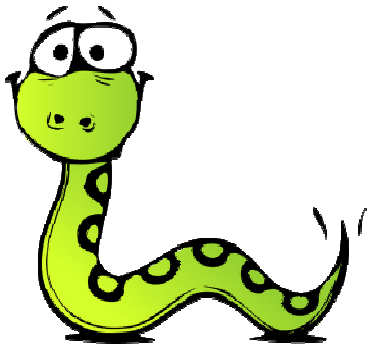
Programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 12 – Proměnná cyklu



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

Cíle lekce

- Pochopit, že hodnota proměnné cyklu se automaticky mění při každém dalším průchodu cyklem
- Naučit se používat proměnnou cyklu v těle cyklu
- Umět odvodit vzorec pro výpočet hodnot určité proměnné v závislosti na proměnné cyklu
- Naučit se ručně krokovat programy s for cykly

Dovednosti

- Vyplnění krokovací tabulky hodnotami proměnných a výrazy v jednotlivých průchodech cyklem

Poznámky

- Stále jde jen o jednoduché cykly s konstantním počtem opakování bez vnořených konstrukcí

Průběh výuky

Cílem této lekce je naučit žáky používat proměnnou cyklu. V úlohách zatím používáme proměnnou `i`. V dalších lekcích prozradíme, že se dají používat i jinak nazvané proměnné. Nyní bude důležité, aby žáci porozuměli, jak se proměnná mění a jak se dá použít v těle cyklu a ve výrazech.

1. Tvůj mladší sourozenec našel následující říkanku:

```
kočka leze dírou
pes oknem
pes oknem
nebude-li pršet
nezmoknem
nebude-li pršet
nezmoknem
```

Vytvoř program `rikanka.py`, který ji vypíše pomocí příkazů `print`. Použij `for` cykly, aby bylo v programu co nejméně příkazů `print`.

Řešení:

```
print('Kočka leze dírou')
for i in range(2):
    print('pes oknem')
for i in range(2):
    print('nebude-li pršet')
    print('nezmoknem')
```

Někteří žáci patrně přijdou jen na použití jednoho `for` cyklu, ale možného použití druhého cyklu si nevšimnou. Takové žáky přivedeme na myšlenku, že se v říkance opakují ještě jiné verše, a tak je možné použít ještě jeden `for` cyklus.

Úlohu lze řešit i bez for cyklů, například takto:

```
print('Kočka leze dírou' + 2 * '\npes oknem' +
      2 * '\nnebude-li pršet\nnezmoknem')
```

nebo i takto:

```
print('''Kočka leze dírou
pes oknem
pes oknem
nebude-li pršet
nezmoknem
nebude-li pršet
nezmoknem''')
```

Taková řešení však od žáků neočekáváme – žáky jsme tyto triky neučili a ani nepožadujeme, aby je znali. Pokud však nějaký žák takto úlohu vyřeší, pochválíme jej a požádáme, aby úlohu vyřešil s použitím `for` cyklů.

Další úloha ilustruje použití proměnné `i` a (pro žáky doposud) neznámou, ale důležitou činnost `for` cyklu.

2. Vytvoř program `rada_cisel.py` a pomocí následujícího kódu vypiš celá čísla od 0 do 9:

```
for i in range(10):
    print('číslo', i)
```

Jestli jsi kód zapsal správně, program po spuštění vypíše:

```
číslo 0
číslo 1
číslo 2
číslo 3
číslo 4
číslo 5
číslo 6
číslo 7
číslo 8
číslo 9
```

Jak to funguje?

`i` je proměnná, do které příkaz `for` postupně přiřazuje celá čísla od `0` do `9`

rozsah čísel je z intervalu `<0, 10)`

```
for i in range(10):
    print('číslo', i)
```

pro každé číslo se vykoná tělo cyklu, a tak se vypíše hodnota proměnné `i`

Očekáváme:

- Diskuzi učitele se žáky
- Případné názorné odkrokování činnosti cyklu na tabuli s doprovodným komentářem:
  - „Když cyklus začíná, do proměnné `i` přiřadí `0`“ a na tabuli kreslíme obsah proměnné `i` jako krabičku,
  - „Příkaz `print` vypíše ...“ a kreslíme na tabuli výsledek výpisu
  - „Potom se do `i` přiřadí `1`“ a změníme obsah proměnné v krabičce
  - atd.
  - „Nakonec se do `i` přiřadí `9`“ a změníme obsah proměnné v krabičce
  - „Příkaz `print` vypíše ...“ a kreslíme na tabuli výsledek výpisu
  - „Cyklus skončí. Číslo 10 se už nepřiřadí.“
- Žákům nevysvětlujeme různé varianty příkazu `range`. Zatím stačí, když mu budou intuitivně rozumět například takto: „`range` určuje rozsah hodnot od 0 až po číslo o jedno menší, než je uvedeno v závorce“. K tomuto poznání by měly přispět i následující úlohy

Také v následujících úlohách předpokládáme diskuzi se žáky.

3. Urči, co je potřeba v předchozím programu změnit, aby se vypsala čísla:

a) `0, 1, ... 10` – tedy i číslo 10
b) `1, 2, ... 10`
c) `2, 4, ... 20`
d) `10, 20, ... 100`

Program pokaždé vyzkoušej, abys ověřil, zda byla tvá domněnka správná.

Příkaz `for` čteme: „*`pro i v rozsahu(...) vykonej tělo cyklu`*“

Řešení pro `0, 1, ... 10`:

```
for i in range(11):
    print('číslo', i)
```

Řešení pro `1, 2, ... 10`:

```
for i in range(10):
    print('číslo', i + 1)
```

Řešení pro `2, 4, ... 20`:

```
for i in range(10):
    print('číslo', (i + 1) * 2)
```

nebo:

```
for i in range(10):
    print('číslo', i * 2 + 2)
```

Řešení pro `10, 20, ... 100`:

```
for i in range(10):
    print('číslo', (i + 1) * 10)
```

nebo:

```
for i in range(10):
    print('číslo', i * 10 + 10)
```

V této úloze se žáci poprvé setkávají s výpočty založenými na proměnné cyklu, což může některým žákům činit potíže. Pokud to bude potřeba, je vhodné se žáky individuálně diskutovat o tom, jak by bylo možno řadu požadovaných čísel (např. `2, 4, … 20`) vytvořit na základě řady čísel `0` až `9`. Lze použít například následující postup:

- V prvním kroku necháme žáka napsat na papír do sloupce čísla, která potřebujeme vypsat (tj. `2, 4, … 20`) a vedle nich do druhého sloupce čísla, kterých nabývá proměnná `i` (tj. `0` až `9`)
- Ve druhém kroku se žáka zeptáme, zda čísla v jednotlivých řádcích nemají „něco společného“. Žák by měl objevit souvislost mezi čísly (v tomto případě, že číslo ve druhém sloupci se rovná dvojnásobku čísla v prvním sloupci zvětšenému o 2).
- Následně by měl žák úvahu zobecnit a odvodit potřebný vzorec (v tomto případě `i * 2 + 2`), který použije ve výpisu

Tyto úlohy lze řešit alternativně jen pomocí vhodných parametrů příkazu `range`: `range(1, 11)`, `range(2, 21, 2)`, `range(10, 101, 10)`. Nechceme však, aby úlohy žáci takto řešili, ani jim různé varianty příkazu `range` neprozrazujeme.

Ačkoliv se varianta příkazu `range(m, n)` zdá být analogická k příkazu `random.randint(m, n)`, který si žáci osvojili v 9. lekci, je mezi nimi významný rozdíl. Zatímco v příkazu `random.randint(m, n)` jsou generována čísla od `m` (včetně) do `n` (včetně), v příkazu `range(m, n)` jsou generována čísla od `m` (včetně) do `n` (mimo), tedy posledním generovaným číslem je `n-1`. Tento rozdíl činí žákům obtíže, a proto doporučujeme zejména začátečníky variantu příkazu `range(m, n)` neučit a spokojit se pouze s variantou `range(n)`.

Následuje série úloh, ve kterých se používá proměnná `i` ve výrazech. Pokud to uznáme za vhodné, můžeme jednotlivé kroky kreslit na tabuli:

4. Vytvoř program `druhe_mocniny.py`, který pomocí `for` cyklu vypíše čísla a jejich druhé mocniny:

```
0 na druhou je 0
1 na druhou je 1
2 na druhou je 4
3 na druhou je 9
4 na druhou je 16
5 na druhou je 25
6 na druhou je 36
```

Řešení:

```
for i in range(7):
    print(i, 'na druhou je', i * i)
```

5. Máme takovouto povídku:

```
Na stromě bylo 0 vrabců, jeden přiletěl a už je tam 1 vrabců
Na stromě bylo 1 vrabců, jeden přiletěl a už je tam 2 vrabců
Na stromě bylo 2 vrabců, jeden přiletěl a už je tam 3 vrabců
Na stromě bylo 3 vrabců, jeden přiletěl a už je tam 4 vrabců
Na stromě bylo 4 vrabců, jeden přiletěl a už je tam 5 vrabců
Na stromě bylo 5 vrabců, jeden přiletěl a už je tam 6 vrabců
Na stromě bylo 6 vrabců, jeden přiletěl a už je tam 7 vrabců
Na stromě bylo 7 vrabců, jeden přiletěl a už je tam 8 vrabců
Na stromě bylo 8 vrabců, jeden přiletěl a už je tam 9 vrabců
Na stromě bylo 9 vrabců, jeden přiletěl a už je tam 10 vrabců
```

Zapiš ji pomocí `for` cyklu do nového programu `povidka.py`.

Řešení:

```
for i in range(10):
    print('Na stromě bylo', i, 'vrabců, jeden přiletěl a už je
        tam', i + 1, 'vrabců')
```

Tato i následující úloha poskytuje prostor k diskuzi o české gramatice. Lze diskutovat, jak by bylo nutné programy upravit, aby generovaly gramaticky správné věty.

6. Vrabci z předchozí povídky odlétají – vymysli v programu `povidka.py` kód, který to bude pomocí `for` cyklu vyprávět:

```
Na stromě bylo 10 vrabců, jeden odletěl a zůstalo tam 9 vrabců
Na stromě bylo 9 vrabců, jeden odletěl a zůstalo tam 8 vrabců
Na stromě bylo 8 vrabců, jeden odletěl a zůstalo tam 7 vrabců
Na stromě bylo 7 vrabců, jeden odletěl a zůstalo tam 6 vrabců
Na stromě bylo 6 vrabců, jeden odletěl a zůstalo tam 5 vrabců
Na stromě bylo 5 vrabců, jeden odletěl a zůstalo tam 4 vrabců
Na stromě bylo 4 vrabců, jeden odletěl a zůstalo tam 3 vrabců
Na stromě bylo 3 vrabců, jeden odletěl a zůstalo tam 2 vrabců
Na stromě bylo 2 vrabců, jeden odletěl a zůstalo tam 1 vrabců
Na stromě bylo 1 vrabců, jeden odletěl a zůstalo tam 0 vrabců
```

Řešení:

```
for i in range(10):
    print('Na stromě bylo', 10 - i, 'vrabců, jeden odletěl
        a zůstalo tam', 9 - i, 'vrabců')
```

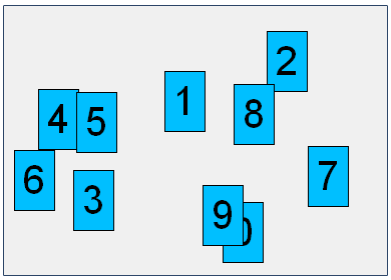
V této úloze se žáci poprvé setkávají s tím, že se v cyklu vypisované číslo snižuje. Někteří žáci se proto mohou snažit upravovat parametry příkazu `range` tak, aby se do proměnné `i` nepřiřazovala čísla od `0` do `9`, ale například od `10` do `1`. Tyto snahy však obvykle nevedou k požadovanému řešení.

Pokud to bude potřeba, je vhodné se žáky individuálně diskutovat o tom, jak by bylo možno řadu postupně se snižujících čísel (např. `10` až `1`) vytvořit na základě řady postupně se zvyšujících čísel `0` až `9`. Lze použít například následující postup:

- V prvním kroku necháme žáka napsat na papír do sloupce čísla, která potřebujeme vypsat (tj. `10` až `1`) a vedle nich do druhého sloupce čísla, kterých nabývá proměnná `i` (tj. `0` až `9`)
- Ve druhém kroku se žáka zeptáme, zda čísla v jednotlivých řádcích nemají „něco společného“. Žák by měl objevit, že jejich součet je vždy `10`. Pokud to neobjeví, můžeme se jej přímo zeptat, jaký je součet těchto čísel
- Ve třetím kroku (pokud si to žák již neuvědomil sám) se žáka zeptáme, jak lze čísla v prvním sloupci odvodit z čísla `10` a čísel ve druhém sloupci. Žák by měl odpovědět, že je nutné od čísla `10` odečíst číslo ve druhém sloupci
- Následně by měl žák úvahu zobecnit a odvodit vzorec `10 - i`, který použije ve výpisu

V následující úloze se proměnná cyklu používá v kombinaci s grafikou. Doporučujeme nevytvářet kreslící podprogramy, ale grafické příkazy zapisovat do těla cyklu.

7. Máme kartičky s čísly od 0 do 9, které chceme náhodně rozložit po ploše. Vytvoř program `deset_karticek.py`, který pomocí cyklu postupně nakreslí deset takových kartiček na náhodných pozicích:

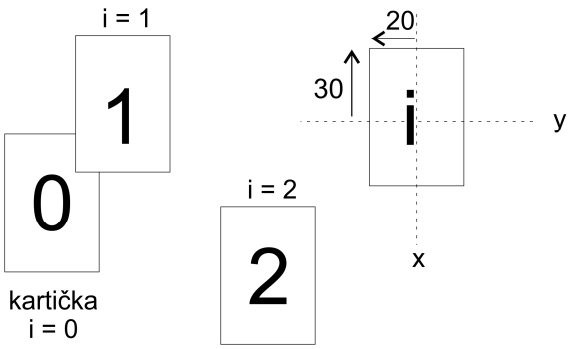


Kdybys chtěl na kartičkách nakreslit velká čísla jako na obrázku výše, přidej do příkazu `create_text` žlutě zvýrazněný kód: `canvas.create_text(x, y, text=i, font='arial 30')`

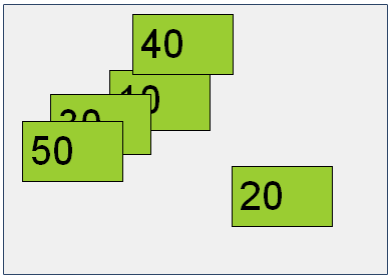
Řešení (souřadnice `x`, `y` jsou vygenerované tak, aby odpovídaly středu kartičky):

```
import tkinter
import random

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

for i in range(10):
    x = random.randint(20, 340)
    y = random.randint(30, 230)
    canvas.create_rectangle(x - 20, y - 30, x + 20, y + 30,
        fill='deepskyblue')
    canvas.create_text(x, y, text=i, font='arial 30')
```

Je možné, že budeme muset kreslení jedné kartičky vymyslet společně se žáky a toto promýšlení postupu předvést i na tabuli:



8. Na chodníku je rozhozených pět cizokrajných bankovek s hodnotami `10`, `20`, `30`, `40` a `50`. Napiš program `bankovky.py`, který takové bankovky nakreslí pomocí `for` cyklu:



V úloze není číslo úmyslně umístěné ve středu kartičky. Záleží však na kreativitě žáků, jak budou kreslené bankovky nakonec vypadat.

Řešení:

```
import tkinter
import random

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

for i in range(5):
    x = random.randint(30, 300)
    y = random.randint(30, 230)
    canvas.create_rectangle(x - 30, y - 30, x + 70, y + 30,
        fill='yellowgreen')
    canvas.create_text(x, y, text=10 + i * 10,
        font='arial 30')
```

Cílem následující úlohy je předvést, jak se proměnná cyklu používá při výpočtu souřadnic. Proto chceme, aby žáci program odkrokovali a viděli souvislost mezi proměnnou `i` a výpočtem souřadnic. Když to bude potřeba, znázorňujeme činnost programu na tabuli.

9. Vytvoř nový program `kresleni_cisel.py` a přepiš do něj následující kód, který kreslí čísla na grafickou plochu:

```
import tkinter

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

for i in range(8):
    x = i * 50
    canvas.create_text(x, 100, text=i, font='arial 30')
```

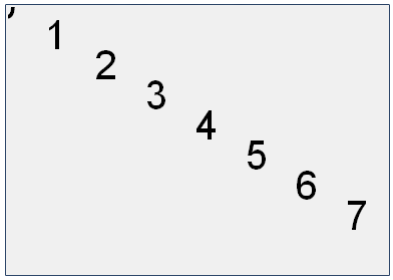
Vytvořený program spusť, abys viděl, co udělá, a vyplň následující tabulku:

Řešení:

	hodnota v proměnné `i`	hodnota v proměnné `x`
když se zobrazí 0	*0*	*0*
když se zobrazí 1	*1*	*50*
když se zobrazí 2	*2*	*100*
když se zobrazí 3	*3*	*150*
když se zobrazí 4	*4*	*200*
když se zobrazí 5	*5*	*250*
když se zobrazí 6	*6*	*300*
když se zobrazí 7	*7*	*350*

Následují gradované varianty předchozí úlohy:

10. Uprav předchozí program tak, aby se čísla kreslila přibližně na úhlopříčce grafické plochy podobně jako na následujícím obrázku:

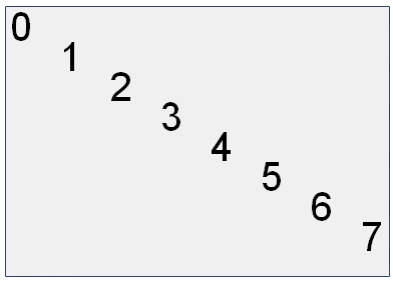


Jaký jsi vymyslel vzorec pro výpočet y-ové souřadnice?

Řešení:

```
import tkinter

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

for i in range(8):
    x = i * 50
    y = i * 30
    canvas.create_text(x, y, text=i, font='arial 30')
```

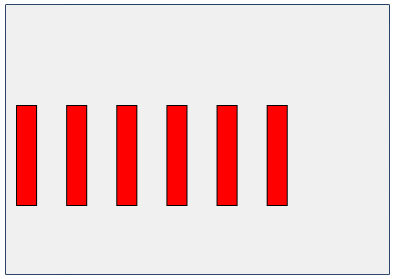
11. V předchozím programu se číslo 0 kreslilo za roh grafické plochy, takže nebylo skoro vidět. Uprav výpočet souřadnic tak, aby byla vidět všechna čísla. Výsledek může vypadat jako na obrázku níže:



Řešení:

```
import tkinter

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

for i in range(8):
    x = i * 50 + 15
    y = i * 30 + 20
    canvas.create_text(x, y, text=i, font='arial 30')
```

12. Víš, jak vypadá padající had z domina? Vytvoř program `domino.py`, který pomocí cyklu a obdélníku nakreslí zatím ještě stojící kostky domina:

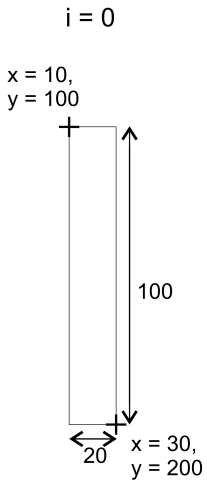


Řešení:

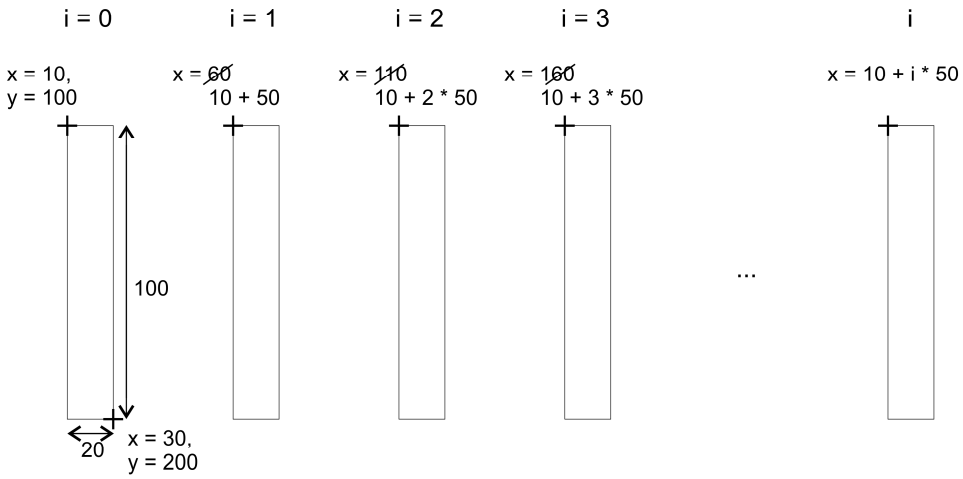
```
import tkinter

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

for i in range(6):
    x = i * 50 + 10
    canvas.create_rectangle(x, 100, x + 20, 200, fill='red')
```

V takovýchto úlohách zaměřených na kreslení bývá pro žáky náročné odvodit vzorec pro výpočet souřadnic pomocí proměnné cyklu. Je vhodné naučit žáky načrtnout si průběh kreslení (například na čtverečkovaný papír) a souřadnice si odvodit:

- začínáme prvním obdélníkem s konkrétními čísly



- postupně přidáváme další pozice s konkrétními čísly, aby žáci viděli vztah mezi pořadovým číslem a souřadnicí



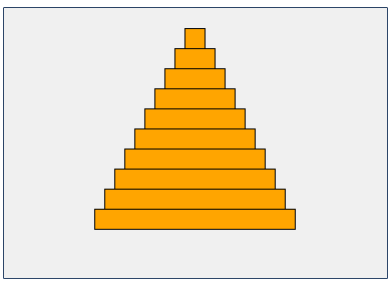
Někteří žáci mohou přijít na řešení, v němž není vzorec pro výpočet souřadnic odvozován na základě proměnné cyklu, ale postupným zvyšováním hodnoty určité proměnné v cyklu. Příkladem takového řešení budiž následující kód:

```
import tkinter

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

x = 10
for i in range(6):
    canvas.create_rectangle(x, 100, x + 20, 200, fill='red')
    x = x + 50
```

Takové postupy však prozatím od žáků neočekáváme, neboť se jimi budeme zabývat až v další lekci. Pokud však nějaký žák takto úlohu vyřeší, pochválíme jej a požádáme, aby úlohu zkusil vyřešit na základě výpočtu souřadnic pomocí proměnné cyklu.

Poslední úloha může být pro žáky obtížná, neboť se v cyklu počítá nejen pozice, ale také velikost obdélníku.

13* Napiš program `velka_pyramida.py`, který pomocí cyklu a obdélníku nakreslí takovouto pyramidu:



Řešení:

```
import tkinter

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

for i in range(10):
    y = 200 - i * 20
    d = 100 - i * 10
    canvas.create_rectangle(190 - d, y, 190 + d, y + 20,
        fill='orange')
```

V našem řešení do proměnné `d` vypočítáváme polovinu délky `i`-tého obdélníku. Pokud se tímto cyklem kreslí obdélníky odzdola (od největšího po nejmenší), polovina délky nejdelšího z nich je `100`, dalšího nad ním `90`, dalšího `80`, atd.