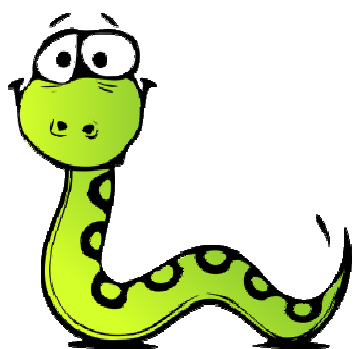




# Programování v jazyce Python pro střední školy

Metodický list pro učitele  
Lekce 13 – Výrazy v cyklu



Andrej Blaho  
Lubomír Salanci  
Václav Šimandl

## Cíle lekce

- Pochopit mechanismus přiřazení: vyhodnocení výrazu vpravo a následné přiřazení do proměnné vlevo
- Seznámit se s akumulací výsledku, tj. postupným zvyšováním hodnoty určité proměnné v cyklu

## Dovednosti

- Znázorňování obsahu proměnných na papír
- Vyplňování krokovací tabulky
- Kreslení náčrtů na papír při uvažování a odvozování vzorců

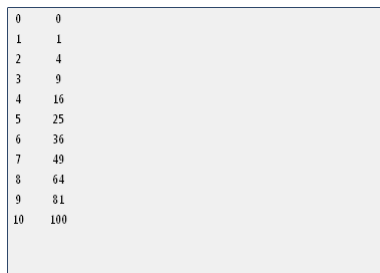
## Poznámky

Cílem této lekce je naučit žáky počítat výsledek postupně pomocí cyklů (tj. akumulovat výsledek). Nejdříve jsou použity úlohy, ve kterých se zvyšuje proměnná o konstantu, později jsou využity úlohy, ve kterých se k výsledku připočítává proměnná cyklu nebo jiná proměnná. Na úlohách je náročné porozumět principu zvyšování hodnoty v proměnné (tj.  $proměnná = proměnná + cosi$ ).

## Průběh výuky

Začínáme úlohou na opakování:

1. Minule jsme vytvářeli program, který v textovém režimu pomocí příkazů `for` a `print` vypisoval čísla a jejich druhé mocniny. Vytvoř podobný program `druhe_mocniny_platno.py`, v němž budou čísla a jejich druhé mocniny zobrazeny v grafické ploše pomocí příkazu `canvas.create_text`.



0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

Řešení:

```
import tkinter

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

for i in range(11):
    canvas.create_text(10, 10 + i * 20, text=i)
    canvas.create_text(50, 10 + i * 20, text=i * i)
```

Následující program dělá skoro to samé, co předchozí, jen nevypisuje mocniny. Ve výpisu se však poprvé vyskytuje koncept **zvyšování hodnoty** proměnné (inkrementace). Necháme žáky program vyzkoušet, aby viděli, co dělá.

2. Je dán následující program:

```
import tkinter

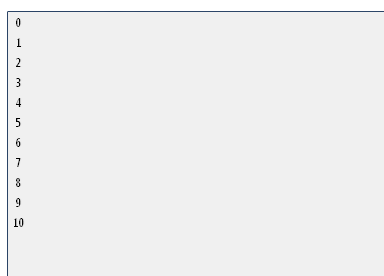
canvas = tkinter.Canvas()
canvas.pack()

y = 10
for i in range(11):
    canvas.create_text(10, y, text=i)
    y = y + 20
```

Program vyzkoušej a doplň do následující tabulky, jak se mění proměnné  $i$  a  $y$  během vykonávání cyklu:

Jakou hodnotu bude mít proměnná  $y$  po skončení cyklu?

Program nakreslí:



Zápis  $y = y + 20$  bývá náročný na porozumění. Vypadá jako matematická rovnice, ale rovnice to není. Žákům pomůže následující vysvětlení:

- „ $y = y + 20$  má dvě části – proměnnou **vlevo** od znaménka rovná se a výraz na **pravé** straně od znaménka rovná se“ ... současně ukazujeme na tabuli
- „Vykonání přiřazení proběhne ve dvou krocích – nejdříve se **vyhodnotí** výraz na pravé straně od znaménka rovná se a ve druhém kroku se výsledná hodnota **přiřadí** do proměnné  $y$ “

Když to bude potřeba, krokovací tabulku (nebo aspoň její začátek) doporučujeme vyplnit společně se žáky. Taktéž můžeme na tabuli znázorňovat, jak se mění obsah proměnné  $y$ .

Na začátku

$y = 10$

$y$

10

Znázorňujeme a program krokujeme:



Po 1. průchodu

$$y = \underbrace{10}_{\text{y}} + 20$$

10 + 30

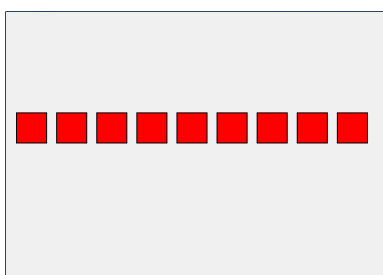
Řešení:

i	y v příkazu create_text	y po vykonání $y = y + 20$
0	10	30
1	30	50
2	50	70
3	70	90
4	90	110
5	110	130
6	130	150
7	150	170
8	170	190
9	190	210
10	210	230

Po skončení cyklu bude v proměnné  $y$  hodnota 230.

V následující úloze nechceme, aby žáci souřadnice odvozovali od hodnoty proměnné cyklu  $i$ , ale aby použili koncept zvyšování hodnoty proměnné podobně jako v předchozí úloze.

3. Vytvoř nový program `rada_ctvercu.py` a v něm pomocí cyklu nakresli devět čtverců s délkou strany 30. Mezi čtverci bude mezera o velikosti 10. Použij proměnnou  $x$ , ve které bude uložena  $x$ -ová souřadnice levého horního rohu kresleného čtverce. Hodnota této proměnné bude v cyklu zvýšena pokaždé o 40.



Řešení:

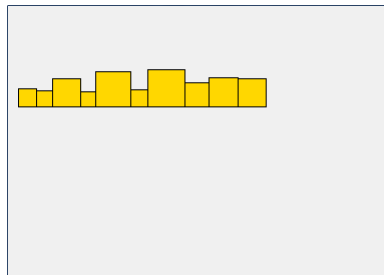
```
import tkinter

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

x = 10
for i in range(9):
    canvas.create_rectangle(x, 100, x + 30, 130, fill='red')
    x = x + 40
```

V následující úloze je potřeba uložit náhodně vygenerovanou hodnotu do další proměnné:

4. Zlatokop našel poklad – 10 zlatých krychliček různých velikostí. Ty postupně ukládal na stůl těsně vedle sebe. Vytvoř program `zlaty_poklad.py`, který takový poklad nakreslí. Každá zlatá krychlička má náhodně zvolenou velikost z rozsahu od 10 do 40. Použij proměnnou, do které budeš ukládat náhodné číslo pro velikost krychličky. Kromě ní použij další proměnnou, pomocí níž budeš evidovat x-ovou pozici krychličky.



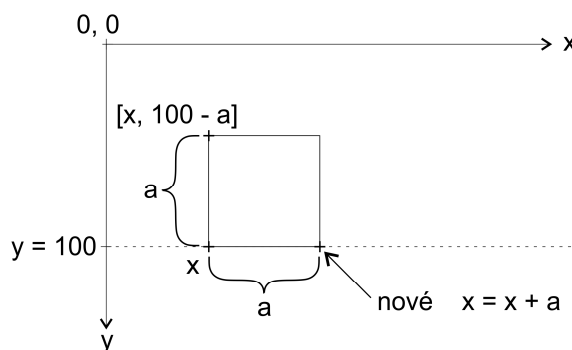
Řešení:

```
import tkinter
import random

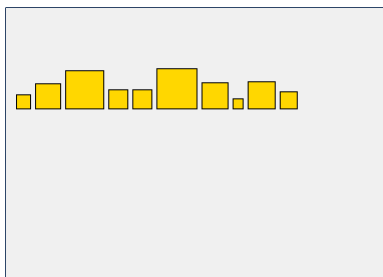
canvas = tkinter.Canvas()
canvas.pack()

x = 10
for i in range(10):
    a = random.randint(10, 40)
    canvas.create_rectangle(x, 100 - a, x + a, 100,
                           fill='gold')
    x = x + a
```

Když je to potřeba, žákům pomáháme, případně kreslíme některé kroky řešení na tabuli. Je vhodné, aby při tom žáci viděli, odkud se berou a proč se v programu vyskytují jednotlivé proměnné (v našem případě  $a$ ,  $x$ ):



5. Vylepši předchozí program tak, aby byly mezi zlatými krychličkami mezery o velikosti 5.



Řešení:

```
import tkinter
import random

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

x = 10
for i in range(10):
    a = random.randint(10, 40)
    canvas.create_rectangle(x, 100 - a, x + a, 100,
                           fill='gold')
    x = x + a + 5
```

V následující úloze se poprvé hodnota z proměnné cyklu připočítává k jiné proměnné:

6. Hrajeme počítačovou hru, která má 10 úrovní. Po úspěšném průchodu  $i$ -tou úrovní získáme  $i$  bodů. Po průchodu první úrovní tedy získáme 1 bod. Po průchodu druhou úrovní se nám ke skóre připočtou 2 body, takže celkem už máme 3 body. Po průchodu třetí úrovní se nám připočtou 3 body, takže naše skóre bude 6 bodů atd. Vytvoř nový program `skore_hry.py`, který pomocí příkazu `print` a cyklu vypíše, jak se zvyšuje skóre po průchodu každou úrovní. Začátek výpisu je naznačený níže:

```
Po levelu 1 bude tvé skóre 1 bodů.  
Po levelu 2 bude tvé skóre 3 bodů.  
Po levelu 3 bude tvé skóre 6 bodů.  
Po levelu 4 bude tvé skóre 10 bodů.  
Po levelu 5 bude tvé skóre 15 bodů.  
...
```

Jaké bude skóre po průchodu desátou úrovní?

Po průchodu 10. úrovní bude skóre 55 bodů.

Řešení:

```
skore = 0  
for i in range(10):  
    uroven = i + 1  
    skore = skore + uroven  
    print('Po levelu', uroven, 'bude tvoje skóre', skore,  
          'bodů.')
```

Alternativní řešení bez použití proměnné `uroven`:

```
skore = 0  
for i in range(10):  
    skore = skore + i + 1  
    print('Po levelu', i + 1, 'bude tvoje skóre', skore,  
          'bodů.')
```

V této úloze se proměnná `skore` zvyšuje o proměnnou odvozenou od proměnné cyklu. Pokud to je potřeba, žákům můžeme individuálně napovědět, že princip výpočtu hodnoty proměnné `skore` bude podobný jako v předchozí úloze, jen hodnota přičítané proměnné závisí na hodnotě proměnné cyklu.

Poznámka pro učitele – výpis v cyklu lze zapsat i následujícím způsobem:

```
print(f'Po {i + 1}. levelu bude tvoje skóre {skore} bodů.')
```

Takové řešení však od žáků neočekáváme – žáky jsme tento trik neučili a ani nepožadujeme, aby jej znali.

Matematicky zdatnější žáci možná úlohu zvládnou vyřešit i bez proměnné `skore`:

```
for i in range(10):  
    print('Po levelu', i+1, 'bude tvoje skóre',  
          (i+1) * (i+2) // 2, 'bodů.')
```

V tomto řešení jsme použili dělení `//`, kdy je výsledkem vždy celé číslo. Přesněji, výsledkem takového dělení je největší celé číslo menší než nebo rovné podílu vzniklého běžným dělením.

Takové řešení však od žáků neočekáváme. Když úlohu někdo z nich takto vyřeší, pochválíme jej a požádáme, aby úlohu vyřešil s použitím proměnné `skore`, do níž se připočítává proměnná cyklu.

Následují úlohy na trénování:

7. Znáš pověst o králi, který slíbil mudrcovi za odměnu tolik zrněk pšenice, kolik jich bude na všech políčkách šachovnice? Král mudrcovi dovolil, aby na první políčko dal 10 zrněk, na druhé 20, na třetí 30 atd. Pomoz králi v rozhodování, zda je taková odměna přiměřená a vytvoř pro něj program `zrnka_sachovnice.py`, který vypíše **celkový počet** zrněk na šachovnici. Políček na šachovnici je 64.

Řešení:

```
zrnek = 0  
for i in range(64):  
    zrnek = zrnek + (i + 1) * 10  
print('Počet zrněk na šachovnici bude:', zrnek)
```

Někteří žáci mohou při řešení využít pomocné proměnné (my ji nazýváme `policko`):

```
zrnek = 0  
for i in range(64):  
    policko = (i + 1) * 10  
    zrnek = zrnek + policko  
print('Počet zrněk na šachovnici bude:', zrnek)
```

Úlohu lze řešit též pomocí matematického vzorce; takové řešení však od žáků neočekáváme:

```
print('Počet zrněk na šachovnici bude', (10 + 640) * 64 / 2)
```

Celkový počet zrn bude 20800, což je přibližně 832 gramů.



8. Jiná verze pověsti praví, že král měl mudrcovi dovolit dát na první políčko jen 1 zrnko, ale na každé další políčko mu dovolil dát dvakrát více zrněk než na předchozí (tj. 2, 4, 8, 16, ...). Uprav svůj program tak, aby zjistil celkový počet zrněk na šachovnici podle této verze pověsti.

Očekávané řešení:

```
zrnek = 0
policko = 1
for i in range(64):
    zrnek = zrnek + policko
    policko = policko * 2
print('Počet zrněk na šachovnici bude:', zrnek)
```

Úlohu lze řešit také následujícím způsobem:

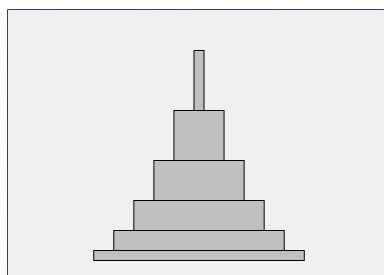
```
zrnek = 0
for i in range(64):
    zrnek = zrnek + 2 ** i
print('Počet zrněk na šachovnici bude:', zrnek)
```

Úlohu lze řešit i pomocí matematického vzorce; takové řešení však od žáků neočekáváme:

```
print('Počet zrněk na šachovnici bude:', 2 ** 64 - 1)
```

Celkový počet zrn bude 18 446 744 073 709 551 615 (18.4 trilionů), což je přibližně 737 869 762 948 tun (737.9 miliard tun).

9\* Vytvoř program `jested.py`, který pomocí cyklu nakreslí vysílač na Ještědu:



Kreslení můžeš začít od spodního obdélníku. Ten má šířku 210 a výšku 10. Každý další obdélník leží na předchozím, je užší o 40 a vyšší o 10.

Řešení:

```
import tkinter

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

y = 250
a = 210
b = 10
for i in range(6):
    canvas.create_rectangle(190 - a / 2, y - b, 190 + a / 2,
                            y, fill='silver')
    y = y - b
    a = a - 40
    b = b + 10
```

Když to bude potřeba, žákům pomáháme, případně je navedeme na ideu použití proměnných `a`, `b` pro šířku a výšku obdélníků.