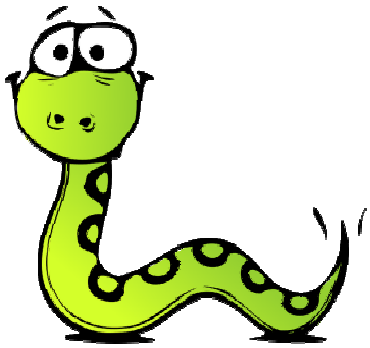
Programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 16 – Větvení



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

Cíle lekce

- Seznámit se s příkazem větvení
- Pochopit mechanismus vykonání/nevykonání příkazů
- Seznámit se s pravdivostními hodnotami a porovnáváním číselných hodnot
- Naučit se přepsat situace z přirozeného jazyka do formální podoby a jazyka srozumitelného pro počítač

Dovednosti

- Odsazování skupiny příkazů v příkazu větvení

Osvojená syntaktická pravidla

- Zápis podmíněného příkazu `if … else …`
- Zápis podmínek jako výrazů s porovnáním hodnot

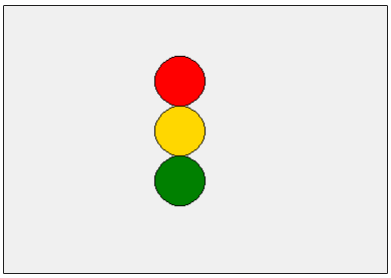
Poznámky

- V této úvodní lekci jsou řešeny úlohy s jednoduchými podmínkami a s jedním příkazem v každé větvi příkazu větvení
- V této úvodní lekci jsou řešeny jen úlohy, ve kterých stačí použít jedno větvení

Průběh výuky

Začínáme úlohou na opakování:

1. Napiš program `semafor_velky.py`, který pomocí barevných kruhů nakreslí semafor se třemi světly jako na obrázku níže:



Řešení (průměry kruhů jsou `50`):

```
import tkinter

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

canvas.create_oval(150, 50, 200, 100, fill='red')
canvas.create_oval(150, 100, 200, 150, fill='gold')
canvas.create_oval(150, 150, 200, 200, fill='green')
```

Také druhá úloha slouží k opakování:

2* Napiš program `nahodny_pozdrav.py`, který pomocí příkazu `random.choice([...])` vypíše jeden z následujících pozdravů: `Dobré ráno`, `Dobrý den`, `Dobrý večer`, `Dobrou noc`. Použij cyklus, aby se vypsalo 10 pozdravů. Například:

```
Dobrou noc
Dobrý den
Dobrou noc
Dobrý večer
Dobrou noc
Dobrou noc
Dobré ráno
Dobrou noc
Dobrý večer
Dobrou noc
```

Řešení:

```
import random

for i in range(10):
    pozdrav = random.choice(['Dobré ráno', 'Dobrý den',
      'Dobrý večer', 'Dobrou noc'])
    print(pozdrav)
```

V následující úloze by žáci měli objevit, že výsledky porovnávání mají hodnoty buď `True` (pravda) nebo `False` (nepravda):

3. Počítač dokáže porovnávat čísla. Napiš do příkazového řádku:

`>>> 1 < 2` a potvrď klávesou *Enter*

Počítač čísla porovná a vypíše:

`True` ... počítač nám odpověděl, že je pravda, že `1` je menší než `2`

Víme, že `3` není větší než `4`. Vyzkoušej, co počítač vypíše, když do příkazového řádku zadáš:

```
>>> 3 > 4
```

Řešení:

```
>>> 3 > 4
False
```

4. Můžeš porovnávat i složitější výrazy. Jaké výsledky počítač vypíše, když zadáš:

```
a) >>> 1 + 2 > 3
b) >>> -1 > -2
c) >>> (1 + 2) * 3 < 1 + 2 * 3
d) >>> a = 100
   >>> a < 101
```

Řešení:

```
a) >>> 1 + 2 > 3
   False
b) >>> -1 > -2
   True
c) >>> (1 + 2) * 3 < 1 + 2 * 3
   False
d) >>> a = 100
   >>> a < 101
   True
```

5. Chceme vytvořit program `teplo_zima.py`, který by nám řekl, zda je venku teplo nebo zima. Mohlo by to fungovat tak, že do proměnné `teplota` přiřadíme číslo. Potom by počítač pro teplotu větší než 20 stupňů vypsal, že je teplo. Jinak by nám řekl, že je zima. Takto to naprogramuješ:

```
teplota = 25
print('Je', teplota, 'stupňů.')
if teplota > 20:
    print('Dnes je teplo.')
else:
    print('Dnes je zima.')
print('Správně se obleč.')
```

Dvojtečky jsou velmi důležité

Příkazy `print` nech odsazené od kraje (Python tam automaticky vložil 4 mezery)

Jestli jsi program napsal správně, po jeho spuštění uvidíš:

```
Je 25 stupňů.
Dnes je teplo.
Správně se obleč.
```

V této úloze se žáci poprvé seznamují s příkazem větvení – v obou větvích je vždy jeden příkaz.

Konstrukce větvení `if … else …` funguje zcela stejně jako ve všech jiných programovacích jazycích. Liší se jen ve způsobu zápisu:

- První řádek obsahuje slovo `if`, za kterým následuje nějaká podmínka (nejčastěji porovnání číselných hodnot), a řádek je ukončený povinnou dvojtečkou `:`
- Následuje alespoň jeden od kraje odsazený řádek s příkazy, které se vykonají pouze v případě, že podmínka je splněná (podmínka za `if` má hodnotu `True`)
- Další řádek obsahuje jen slovo `else` s povinnou dvojtečkou `:`. Toto slovo nesmí být odsazené od kraje, ale musí být na stejné úrovni jako řádek `if`
- Následuje alespoň jeden od kraje odsazený řádek s příkazy, které se vykonají pouze v případě, že podmínka není splněná (podmínka za `if` má hodnotu `False`)

Je zřejmé, že právě jedna z větví (první odsazená skupina příkazů za `if`, nebo druhá za `else`) se vykoná a ta druhá ne. Která z nich to bude, to závisí jen na podmínce.

Pokud to bude potřeba, fungování příkazu větvení `if … else …` můžeme v tuto chvíli demonstrovat na konkrétním jednoduchém příkladu pomocí kresby na tabuli.

6. Změň v programu `teplo_zima.py` hodnotu proměnné `teplota` z `25` na `10`. Program spusť. Co se vypíše?

   Jak program funguje?

   `if ... else ...` je nový příkaz pro větvení programu

   podmínka, podle které se počítač rozhodne, kterou větev dále vykoná

```
teplota = 10
print('Je', teplota, 'stupňů.')
if teplota > 20:
    print('Dnes je teplo.')      } větev if
else:
    print('Dnes je zima.')       } větev else
print('Správně se obleč.')
```

   Když počítač uvidí příkaz `if ... else ...`, nejdříve vyhodnotí podmínku. Když je podmínka splněná, vykoná se příkaz ve větvi `if`, jinak se vykoná příkaz ve větvi `else`.

Řešení:

```
teplota = 10
print('Je', teplota, 'stupňů.')
if teplota > 20:
    print('Dnes je teplo.')
else:
    print('Dnes je zima.')
print('Správně se obleč.')
```

Po spuštění se vypíše:

```
Je 10 stupňů.
Dnes je zima.
Správně se obleč.
```

Když to uznáme za vhodné, můžeme zavést konstrukci `input`. Ta si vyžádá od uživatele určitou hodnotu, se kterou se bude dále v programu pracovat. Nejvhodnější je následující varianta zápisu:

```
teplota = int(input('Zadej teplotu: '))
print('Je', teplota, 'stupňů.')
...
```

Zápis `int(` před samotným příkazem `input` označuje, že načítaná hodnota bude celé číslo. Pokud bychom předpokládali, že se mohou zadávat i desetinná čísla (s desetinnou tečkou), je potřeba použít slovo `float(`. Příkladem budiž následující ukázka:

```
teplota = float(input('Zadej teplotu: '))
print('Je', teplota, 'stupňů.')
...
```

Parametrem příkazu `input` je libovolný text, který se vypíše do příkazového řádku a za který bude uživatel zapisovat vstupní hodnotu.

Po spuštění to může vypadat takto:

```
Zadej teplotu: 15
Je 15 stupňů.
Dnes je zima.
Správně se obleč.
```

Příkaz `input` pro získání vstupních hodnot lze využít nejen na zadávání čísel, ale i textů. Příkladem budiž následující ukázka:

```
piti = input('Co budeš chtít k pití? ')
print(piti, 'Ti hned přineseme')
```

Po spuštění to může vypadat takto:

```
Co budeš chtít k pití? Limonádu
Limonádu Ti hned přineseme
```

V tomto případě jsme před příkaz `input` nezapsali ani `int(` ani `float(`, protože načítaná hodnota není celým ani desetinným číslem, ale je to text.

Žáky ze znalosti výše uvedených příkazů nebudeme v žádném případě zkoušet.

7. Uprav program `teplo_zima.py` tak, aby pro:

   - záporné teploty vypsal `Vezmi si rukavice`,
   - jinak vypsal `Rukavice nejsou potřeba`.

   Ověř, že program funguje správně pro kladné i záporné hodnoty. Co tvůj program vypíše pro `0`?

Řešení:

```
teplota = 0
print('Je', teplota, 'stupňů.')
if teplota < 0:
    print('Vezmi si rukavice.')
else:
    print('Rukavice nejsou potřeba.')
```

Po spuštění se pro různé teploty vypíše:

```
Je 10 stupňů.
Rukavice nejsou potřeba.
======== RESTART ========
Je -5 stupňů.
Vezmi si rukavice.
======== RESTART ========
Je 0 stupňů.
Rukavice nejsou potřeba.
```

8. Chceme poslat doporučený dopis. Vytvoř program `cena_dopisu.py`, který ti poradí s cenou dopisu. Na začátku programu přiřaď do proměnné `hmotnost` číslo s hmotností tvého dopisu. Použij příkaz pro větvení programu, aby pro dopis s hmotností:

   - větší než `50` vypsal `Zaplatíš 55 korun`,
   - jinak vypsal `Zaplatíš 47 korun`.

   Ověř, že program počítá správně cenu dopisu pro hmotnosti: `30`, `50` a `100`.

Řešení:

```
hmotnost = 100
print('Hmotnost dopisu je', hmotnost, 'gramů')
if hmotnost > 50:
    print('Zaplatíš 55 korun')
else:
    print('Zaplatíš 47 korun')
```

Po spuštění se pro různé hmotnosti vypíše:

```
Hmotnost dopisu je 30 gramů
Zaplatíš 47 korun
======== RESTART ========
Hmotnost dopisu je 50 gramů
Zaplatíš 47 korun
======== RESTART ========
Hmotnost dopisu je 100 gramů
Zaplatíš 55 korun
```

9. Jdeme na zmrzlinu. Cena za jeden kopeček zmrzliny je 25 korun. Zmrzlinář však nabízí slevu: když vezmeme víc než 4 kopečky, cena za každý kopeček bude 20 korun. Vytvoř program `zmrzlina.py`, ve kterém na začátku přiřadíš do proměnné `pocet` počet kopečků a on vypíše výslednou cenu. Například:

   ```
   Za 5 kopečků zmrzliny zaplatíš:
   100 korun.
   ```

   Ověř, že program správně počítá cenu zmrzliny pro `4`, `5` a `6` kopečků.

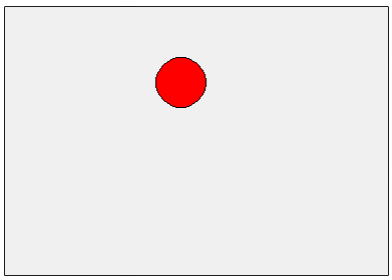
Řešení:

```
pocet = 4
print('Za', pocet, 'kopečků zmrzliny zaplatíš:')
if pocet > 4:
    print(pocet * 20, 'korun.')
else:
    print(pocet * 25, 'korun.')
```

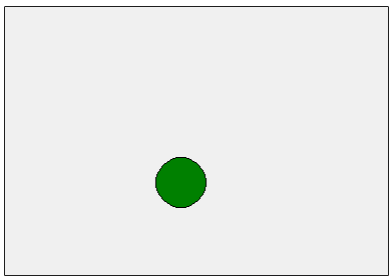
Po spuštění se pro různé počty kopečků vypíše:

```
Za 4 kopečků zmrzliny zaplatíš:
100 korun.
======== RESTART ========
Za 5 kopečků zmrzliny zaplatíš:
100 korun.
======== RESTART ========
Za 6 kopečků zmrzliny zaplatíš:
120 korun.
```

10. V 1. úloze jsi vytvářel program `semafor_velky.py`, který kreslil semafor. Nyní bychom chtěli vylepšit tento program tak, že na semaforu bude svítit buď jen červené, nebo jen zelené světlo. Do proměnné `cas` přiřaď počet sekund. Když je toto číslo menší než `30`, nakreslí se červené světlo, jinak se nakreslí zelené světlo.



pro `cas = 5`



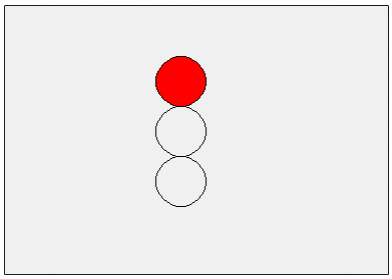
pro `cas = 50`

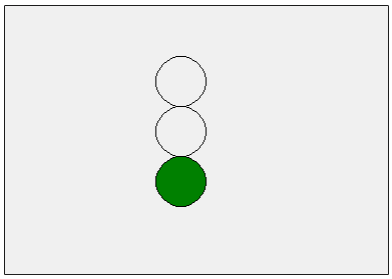
Ověř, že tvůj program kreslí správné světlo pro časy `5`, `30` i `50` sekund.

Řešení:

```
import tkinter

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

cas = 5
if cas < 30:
    canvas.create_oval(150, 50, 200, 100, fill='red')
else:
    canvas.create_oval(150, 150, 200, 200, fill='green')
```

11. Vylepši program `semafor_velky.py` tak, aby byla vidět i nesvítící světla podobně jako na obrázcích níže:





Řešení:

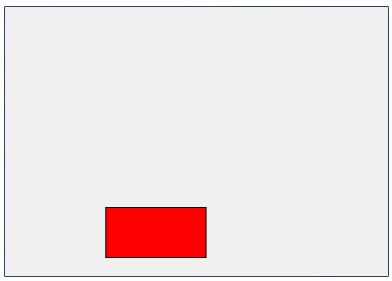
```
import tkinter

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

```
cas = 5
if cas < 30:
    canvas.create_oval(150, 50, 200, 100, fill='red')
else:
    canvas.create_oval(150, 150, 200, 200, fill='green')
canvas.create_oval(150, 50, 200, 100)
canvas.create_oval(150, 100, 200, 150)
canvas.create_oval(150, 150, 200, 200)
```

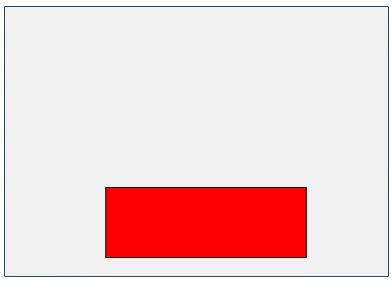
V tomto řešení si všimněme, že ve větvích příkazu větvení se nakreslí buď červený vrchní kruh, nebo zelený spodní kruh. Po skončení tohoto příkazu následují tři příkazy na kreslení všech tří kruhů, tedy i toho, který se nakreslil v příkazu větvení. Jestliže tyto tři kruhy nemají výplň, nevadí, že překreslí už nakreslený kruh. Pokud bychom požadovali, aby zhasnutá světla byla nakreslena šedou barvou, museli bychom použít jiný postup. Řešení by mohlo vypadat například takto:

```
import tkinter

canvas = tkinter.Canvas(bg='white')
canvas.pack()

cas = 5
canvas.create_oval(150, 50, 200, 100, fill='gray')
canvas.create_oval(150, 100, 200, 150, fill='gray')
canvas.create_oval(150, 150, 200, 200, fill='gray')
if cas < 30:
    canvas.create_oval(150, 50, 200, 100, fill='red')
else:
    canvas.create_oval(150, 150, 200, 200, fill='green')
```

12. Máme krabici ve tvaru obdélníku s délkami stran, které jsou uložené v proměnných `a`, `b`. Krabici chceme položit vždy naležato. Vytvoř program `krabice.py`, ve kterém na začátku přiřadíš do proměnných `a, b` rozměry krabice. Program nakreslí krabici tak, aby **vždy** ležela delší stranou na zemi (příklady jsou na obrázcích níže).



`a = 50, b = 100`



`a = 200, b = 70`

Ověř, že program funguje správně pro různé kombinace hodnot proměnných `a, b`.

V této úloze si musí žáci uvědomit, jaká proměnná je použita pro vyjádření šířky resp. výšky kresleného obdélníku:

- Pokud je hodnota proměnné `a` větší než hodnota proměnné `b`, pak je jako šířka obdélníku použita hodnota proměnné `a` a jako jeho výška je použita hodnota proměnné `b`
- Jinak je jako šířka obdélníku použita hodnota proměnné `b` a jako jeho výška je použita hodnota proměnné `a`

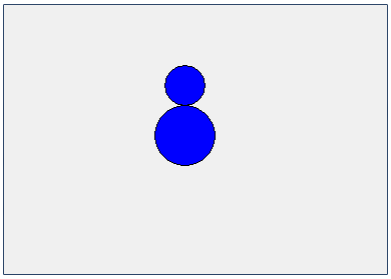
Řešení:

```
import tkinter

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

a = 50
b = 100
if a > b:
    canvas.create_rectangle(100, 250 - b, 100 + a, 250,
        fill='red')
else:
    canvas.create_rectangle(100, 250 - a, 100 + b, 250,
        fill='red')
```

Místo přiřazení konkrétních hodnot do proměnných `a`, `b` zde opět můžeme použít konstrukci `input`, pomocí níž si od uživatele vyžádáme rozměry krabice. Program s příkazem `input` by mohl vypadat například takto:

```
import tkinter

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

a = int(input('Zadej první rozměr krabice: '))
b = int(input('Zadej druhý rozměr krabice: '))

if a > b:
    canvas.create_rectangle(100, 250 - b, 100 + a, 250,
        fill='red')
else:
    canvas.create_rectangle(100, 250 - a, 100 + b, 250,
        fill='red')
```

13* Vytvoř nový program `kruhy_nad_sebou.py`, který bude kreslit nad sebe dva kruhy. Do proměnných `r1`, `r2` přiřaď poloměry kruhů. Napiš kód programu, který nakreslí oba dva kruhy tak, aby menší kruh ležel **vždy** na větším podobně jako na obrázku níže.



`r1 = 30, r2 = 20`

Ověř, že program funguje správně pro různé kombinace poloměrů.

1. varianta řešení (nejprve se nakreslí první kruh s poloměrem `r1` a středem se souřadnicemi `[180, 130]`, druhý kruh s poloměrem `r2` se nakreslí nad první, nebo pod první kruh):

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

r1 = 30
r2 = 50
canvas.create_oval(180 - r1, 130 - r1, 180 + r1, 130 + r1,
    fill='blue')
if r1 < r2:
    canvas.create_oval(180 - r2, 130 + r1, 180 + r2,
        130 + r1 + 2 * r2, fill='blue')
else:
    canvas.create_oval(180 - r2, 130 - r1 - 2 * r2, 180 + r2,
        130 - r1, fill='blue')
```

2. varianta řešení:

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

r1 = 30
r2 = 50
if r1 < r2:
    canvas.create_oval(180 - r1, 10, 180 + r1, 10 + 2 * r1,
        fill='blue')
    canvas.create_oval(180 - r2, 10 + 2 * r1, 180 + r2,
        10 + 2 * r1 + 2 * r2, fill='blue')
else:
    canvas.create_oval(180 - r2, 10, 180 + r2, 10 + 2 * r2,
        fill='blue')
    canvas.create_oval(180 - r1, 10 + 2 * r2, 180 + r1,
        10 + 2 * r2 + 2 * r1, fill='blue')
```

Toto druhé řešení využívá toho, že v každé větvi příkazu větvení může být i více příkazů, které se vykonají v případě, že je podmínka splněná, resp. nesplněná.

Při řešení této úlohy je opět možné využít konstrukci `input`. Pro přiřazení hodnoty do proměnné `r1` bychom použili příkaz:

```
r1 = int(input('Zadej poloměr prvního kruhu: '))
```

a pro přiřazení hodnoty do proměnné `r2` příkaz:

```
r2 = int(input('Zadej poloměr druhého kruhu: '))
```