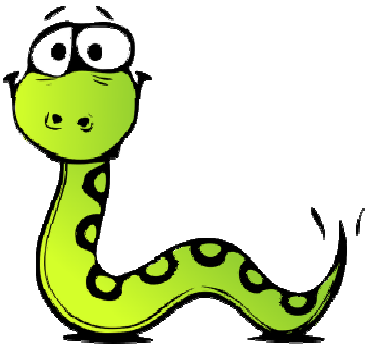
Základy programování v jazyce Python pro střední školy

Metodický list pro učitele

Lekce 9 – Náhoda



Andrej Blaho

Ľubomír Salanci

Václav Šimandl

beta verze - 2019

Cíle lekce

- Seznámit se s ideou generátoru náhodných čísel z určitého rozsahu (intervalu)
- Naučit se používat lokální proměnné v podprogramech (intuitivně bez nutnosti vysvětlování)
- Pochopit, že vygenerovanou hodnotu je nutné v některých případech uložit do proměnné (pro její vícenásobné použití)

Dovednosti

- Zápis `import random` na úplném začátku programu

Osvojená syntaktická pravidla

- Příkaz `import` pro zpřístupnění knihovny `random`
- Zápis volání generátoru náhodných čísel (např. `random.randint(1, 6)`)

Průběh výuky

První úloha slouží k opakování podprogramů:

1. Napiš program `sit.py`, ve kterém definuješ dva podprogramy s příkazy `print`:

   - podprogram `cara` zobrazí v jednom řádku 19 znaků `+--+--+--+--+--+--+`
   - podprogram `hulky` zobrazí střídavě hůlku a dvě mezery tak, aby znaků (včetně mezer) bylo 19.

   Na konci programu zavolej střídavě podprogramy `cara` a `hulky` tak, aby se zobrazilo:

```
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
```

Řešení:

```
def cara():
    print('+--+--+--+--+--+--+')

def hulky():
    print('|  |  |  |  |  |  |')

cara()
hulky()
cara()
hulky()
cara()
```

2. Přidej do předchozího programu ještě jeden podprogram `ctvereckovany_papir`. Ten využije tvé podprogramy `cara` a `hulky` tak, že jejich voláním zobrazí čtvercovou síť jako na obrázku níže. Tento nový podprogram zavolej pro zobrazení sítě.

```
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
|  |  |  |  |  |  |
+--+--+--+--+--+--+
```

Řešení:

```
def cara():
    print('+--+--+--+--+--+--+')

def hulky():
    print('|  |  |  |  |  |  |')

def ctvereckovany_papir():
    cara()
    hulky()
    cara()
    hulky()
    cara()
    hulky()
    cara()
    hulky()
    cara()

ctvereckovany_papir()
```

Cílem dalších úloh je naučit žáky pracovat s náhodnými čísly. V následující úloze žáky necháme experimentovat s generátorem náhodných čísel:

3. V počítači funguje náhoda. Zadej do příkazového řádku tyto příkazy:

```
>>> import random
>>> random.randint(1, 6)
```

Počítač zobrazí nějaké číslo, například:

5

Nech vykonat příkaz `random.randint(1, 6)` několikrát. Diskutuj se svým spolužákem, jaká čísla počítač zobrazil tobě a jemu.

```
>>> random.randint(1, 6)
6
```

```
>>> random.randint(1, 6)
4
>>> random.randint(1, 6)
4
```

Slovo `random` znamená náhodný. Při vykonaní příkazu `random.randint(1, 6)` si počítač vymyslí nějaké číslo od `1` do `6`. Je to podobné, jakoby si počítač hodil hrací kostkou.

Vidíme, že činnost generátoru náhodných čísel se nejlépe projeví, když jej vyvoláme vícekrát.

V metodice programování lze postupovat tak, že:

- buď nejdříve učíme žáky cyklus a potom podprogramy,
- nebo nejdříve učíme žáky podprogramy a potom cyklus.

V jazyce Python je však problém: když nejdříve učíme cyklus, můžeme žákům zadávat jen takové úlohy, jejichž řešení mají v těle cyklu pouze příkazy `print`, přiřazení nebo kreslení grafiky na danou pozici. Tato množina úloh je však malá, vede k náročným, téměř matematickým problémům, které mohou být pro začátečníky velmi obtížné. Kdybychom ještě cyklus zkombinovali s náhodností, úlohy by se těžko vysvětlovaly, případně by se jejich řešení obtížně krokovala.

Když začneme s podprogramy jako s prvními, můžeme žáky přirozeně motivovat, že skupinu příkazů zabalíme a pojmenujeme. Potom podprogram stačí jednou nebo několikrát vyvolat i bez použití cyklu. Tento metodický postup se nám zdá výrazně jednodušší a vede i k přehlednějším řešením a zápisům.

Zápis `import random` má podobný význam jako už známý zápis `import tkinter`. Našemu programu zpřístupní externí knihovnu nových příkazů (podprogramů). Knihovna `tkinter` obsahuje kompletní grafiku a my jsme ji využívali v zápise „`tkinter.Canvas()`“. Knihovna `random` obsahuje několik užitečných příkazů, které generují náhodná čísla. My z nich prozatím využijeme jen příkaz `random.randint(od, do)`, který náhodně vybere hodnotu z daného intervalu celých čísel `<od, do>`.

Cílem následující ukázky je naučit žáky používat proměnné pro zapamatování náhodně vygenerované hodnoty:

4. Náhodné číslo si můžeš zapamatovat – napiš program `kostka.py` s následujícím kódem a spusť jej (i vícekrát):

```
import random
n = random.randint(1, 6)
print('Na kostce padla', n)
```

Proměnná `n` je zde globální proměnnou.

Dále chceme zabalit házení kostkou do podprogramu, aby se hody kostkou daly jednoduše vícekrát zopakovat. Z pohledu žáka jsou v následující ukázce dva nové jevy: použití proměnné v podprogramu (tj. lokální proměnná `n`) a použití náhodných čísel v podprogramu.

5. Uprav program `kostka.py` – vytvoř podprogram `hod_kostkou` a doplň kód programu tak, aby se simulovalo deset hodů za sebou:

```
import random

def hod_kostkou():
     n = random.randint(1, 6)
     print('Na kostce padla', n)

hod_kostkou()
```

Měl bys vidět výpis podobný následujícímu:

```
Na kostce padla 5
Na kostce padla 3
Na kostce padla 4
Na kostce padla 1
Na kostce padla 3
Na kostce padla 2
Na kostce padla 1
Na kostce padla 1
Na kostce padla 1
Na kostce padla 3
```

Řešení bez použití cyklu (tj. copy-paste):

```
import random

def hod_kostkou():
    n = random.randint(1, 6)
    print('Na kostce padla', n)

hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
hod_kostkou()
```

Pojmy lokální a globální proměnná zatím není potřeba žáky učit. Plánujeme řešit jen takové úlohy, ve kterých by se neměly vyskytnout konflikty s proměnnými.

Z pohledu Pythonu lokální proměnná existuje pouze při běhu podprogramu, tj. vznikne až po zavolání podprogramu. Po skončení běhu podprogramu tato proměnná dále neexistuje (automaticky se zruší). Globální proměnná existuje od svého vzniku (přiřazením mimo

podprogram), ale na rozdíl od lokální proměnné její hodnotu nemůžeme v podprogramu měnit.

Toto žákům nemusíme vysvětlovat, ale je to dobré vědět, když bude potřeba řešit nějaký problém.

Následují úlohy, ve kterých se žáci lépe seznámí s generátorem náhodných čísel. Ve všech úlohách předpokládáme, že žáci upravené podprogramy zavolají vícekrát, aby je otestovali.

6. Uprav předcházející program tak, aby počítač simuloval jeden hod na dvacetistěnné kostce.

Řešení – upraví se pouze podprogram `hod_kostkou`:

```
def hod_kostkou():
    n = random.randint(1, 20)
    print('Na kostce padla', n)
```

7* Máme binární hrací kostku, která má na třech stěnách `0` a zbylých třech `1`. Uprav předcházející program, aby simuloval hod takovou kostkou.

Řešení – upraví se pouze podprogram `hod_kostkou`:

```
def hod_kostkou():
    n = random.randint(0, 1)
    print('Na kostce padla', n)
```

8* Máme „sudou“ hrací kostku, která má na stěnách čísla 2, 4, 6, 8, 10, 12. Uprav předcházející program, aby simuloval hod takovou kostkou.

Řešení – upraví se pouze podprogram `hod_kostkou`:

```
def hod_kostkou():
    n = random.randint(1, 6) * 2
    print('Na kostce padla', n)
```

9* Máme „lichou“ hrací kostku, která má na stěnách čísla 1, 3, 5, 7, 9, 11. Uprav předcházející program, aby simuloval hod takovou kostkou.

Řešení – upraví se pouze podprogram `hod_kostkou`:

```
def hod_kostkou():
    n = random.randint(1, 6) * 2 - 1
    print('Na kostce padla', n)
```

10* Máme exotickou hrací kostku, která má na stěnách čísla 1, 4, 9, 16, 25, 36. Uprav předcházející program, aby simuloval hod takovou kostkou.

Řešení – upraví se pouze podprogram `hod_kostkou`:

```
def hod_kostkou():
    n = random.randint(1, 6) ** 2
    print('Na kostce padla', n)
```

Pokud budou mít žáci s řešením problémy, je potřeba s nimi řešení diskutovat a učit je výrazy sestavovat.

11. Napiš program `predpoved.py` a v něm podprogram `predpoved`, který vypíše zprávu:

```
Dnes bude 15 stupňů.
```

Jako číselný údaj program zvolí náhodné celé číslo od `-15` do `35`.

Možné řešení:

```
import random

def predpoved():
    teplota = random.randint(-15, 35)
    print('Dnes bude', teplota, 'stupňů.')

predpoved()
```

12. Vytvoř program `pin.py`, který vygeneruje náhodný PIN pro tvůj mobil. Do čtyř proměnných `a`, `b`, `c`, `d` přiřaď náhodná čísla od 0 po 9 a potom je jediným příkazem `print` vypiš. Výpis může vypadat například takto:

```
Tvůj nový PIN je 1 3 7 3
```

Řešení:

```
import random

a = random.randint(0, 9)
b = random.randint(0, 9)
c = random.randint(0, 9)
d = random.randint(0, 9)
print('Tvůj nový PIN je', a, b, c, d)
```

13. Vytvoř nový program `datumy.py` – generátor náhodných datumů (pro jednoduchost nechť má každý měsíc 30 dní). Po spuštění program vypíše informaci s vygenerovaným náhodným datem, například:

```
Pokoj si uklidím 30 . 2 . 2019
```

Možné řešení s použitím proměnných:

```
import random

den = random.randint(1, 30)
mesic = random.randint(1, 12)
rok = random.randint(2018, 2099)
print('Pokoj si uklidím', den, '.', mesic, '.', rok)
```

Možné řešení bez použití proměnných:

```
import random

print('Pokoj si uklidím',
      random.randint(1, 30), '.',
      random.randint(1, 12), '.',
      random.randint(2018, 2099))
```

Uvedené výpisy se některým žákům nemusí líbit, protože vypisované tečky ve vygenerovaném datu jsou od čísel oddělené mezerami. Pokročilejší formátování výstupů je náročnější téma a přesahuje možnosti tohoto kurzu. Jen pro zajímavost uvádíme různé způsoby zápisu řádku s příkazem `print`:

Zápis pro základní výpis, v němž jsou mezi číslicemi a tečkami mezery (tento zápis byl využit ve výše uvedeném řešení):

```
print('Pokoj si uklidím', den, '.', mesic, '.', rok)
```

Alternativní zápisy pro výpis, v němž nejsou mezi číslicemi a tečkami mezery (všechny uvedené zápisy vypíší stejný výsledek):

- `print(f'Pokoj si uklidím {den}.{mesic}.{rok}')`
- `print('Pokoj si uklidím {}.{}.{}'.format(den, mesic, rok))`
- `print('Pokoj si uklidím ', str(den) + '.' + str(mesic) + '.' + str(rok))`
- `print('Pokoj si uklidím ', den, '.', mesic, '.', rok, sep='')`

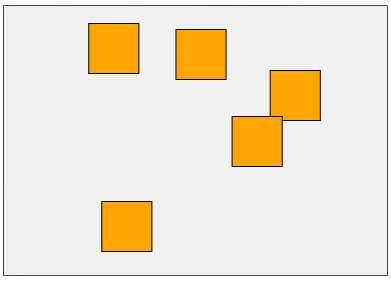
Žáky, vzhledem k jejich současným programátorským zkušenostem, však tyto alternativní zápisy nedoporučujeme učit.

V následujících úlohách se kombinuje kreslení a náhodná čísla:

14. Vytvoř nový program `nahodny_ctverec.py`, ve kterém pomocí následujícího kódu nakreslíš náhodně umístěný čtverec:

```
import tkinter
import random

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

def nahodny_ctverec():
    x = random.randint(10, 300)
    y = random.randint(10, 200)
    canvas.create_rectangle(x, y, x + 50, y + 50,
       fill='orange')

nahodny_ctverec()
```

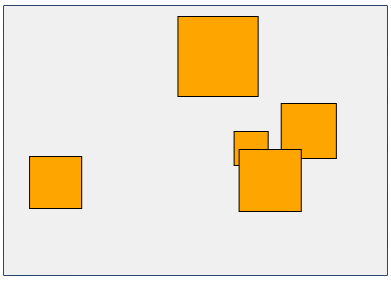
15. Doplň do programu `nahodny_ctverec.py` příkazy tak, aby program nakreslil pět náhodných čtverců:



Řešení – upraví se pouze volání podprogramu:

```
nahodny_ctverec()
nahodny_ctverec()
nahodny_ctverec()
nahodny_ctverec()
nahodny_ctverec()
```

16. Uprav program `nahodny_ctverec.py` tak, aby se čtverce kreslily nejen na náhodných pozicích, ale také aby měl každý čtverec náhodnou velikost z intervalu od `10` do `100`:

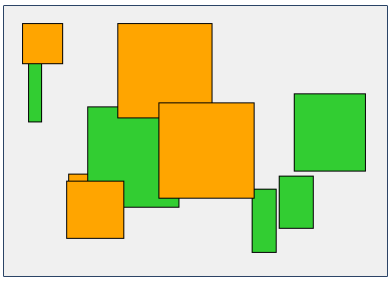


Řešení – upraví se pouze podprogram `nahodny_ctverec`:

```
def nahodny_ctverec():
    x = random.randint(10, 300)
    y = random.randint(10, 200)
    a = random.randint(10, 100)
    canvas.create_rectangle(x, y, x + a, y + a, fill='orange')
```

Poznámka: Proměnné `x`, `y`, `a` jsou lokální proměnné. Proměnná `canvas` je globální proměnná.

17. Doplň do programu `nahodny_ctverec.py` nový podprogram `nahodny_obdelnik`. Ten vygeneruje náhodné souřadnice i rozměry obdélníku a nakreslí jej. Vlož příkazy, které střídavě nakreslí pět náhodných čtverců a pět obdélníků.

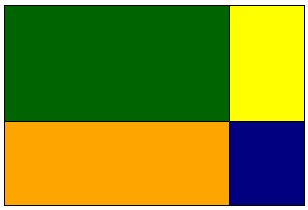


Řešení:

```
import tkinter
import random

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

def nahodny_ctverec():
    x = random.randint(10, 300)
    y = random.randint(10, 200)
    a = random.randint(10, 100)
    canvas.create_rectangle(x, y, x + a, y + a, fill='orange')

def nahodny_obdelnik():
    x = random.randint(10, 300)
    y = random.randint(10, 200)
    a = random.randint(10, 100)
    b = random.randint(10, 100)
    canvas.create_rectangle(x, y, x+a, y+b, fill='lime green')

nahodny_ctverec()
nahodny_obdelnik()
nahodny_ctverec()
nahodny_obdelnik()
nahodny_ctverec()
nahodny_obdelnik()
nahodny_ctverec()
nahodny_obdelnik()
nahodny_ctverec()
nahodny_obdelnik()
```

Podprogram `nahodny_obdelnik` zřejmě vznikne jako kopie podprogramu `nahodny_ctverec`, kterou žáci přejmenují a přidají do ní lokální proměnnou `b` pro výšku obdélníku. Barvu si mohou zvolit dle vlastního uvážení.

18. Vytvoř nový program `sportovni_vlajka.py`, který bude kreslit sportovní vlajku vaší třídy. Vlajka bude tvořena čtyřmi barevnými obdélníky, které se vzájemně dotýkají v jediném bodě jako na obrázku níže:



Program si náhodně zvolí souřadnice `x`, `y`, které představují místo dotyku všech čtyř obdélníků. Vnější rozměry vlajky nechť jsou `300x200`; barvy na vlajce si urči podle svého uvážení.

Řešení:

```
import tkinter
import random

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

x = random.randint(50, 250)
y = random.randint(50, 150)
canvas.create_rectangle(10, 10, x, y, fill='darkgreen')
canvas.create_rectangle(x, 10, 310, y, fill='yellow')
canvas.create_rectangle(10, y, x, 210, fill='orange')
canvas.create_rectangle(x, y, 310, 210, fill='navy')
```

Vnější vrcholy obdélníku je možné zvolit libovolně. V našem řešení má levý horní vrchol souřadnice `[10, 10]` a protože rozměry obdélníku mají být `300x200`, protilehlý vrchol (pravý dolní) má souřadnice `[310, 210]`. Vnitřní náhodně zvolený bod jsme vygenerovali tak, aby byl od okrajů obdélníku vzdálen alespoň `50`, tedy každá ze čtyř oblastí bude mít rozměry minimálně `50x50`.