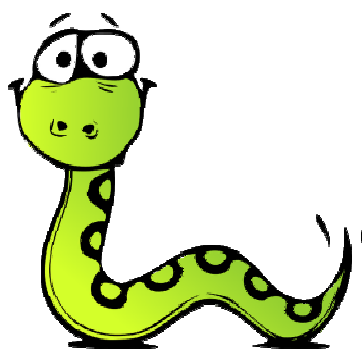




Programování v jazyce Python pro střední školy

Metodický list pro učitele
Lekce 8 – Podprogramy



Andrej Blaho
Lubomír Salanci
Václav Šimandl

Cíle lekce

- Seznámit se s pojmem podprogram, s jeho definováním a voláním (zatím bez parametrů)
- Uvažovat o řešení problému pomocí podprogramů (například rozložením na podproblémy)
- Naučit se, že podprogramy je možné zavolat teprve poté, co byly definované

Dovednosti

- Používání kláves <Tab> a <Enter> při formátování zápisu podprogramu

Osvojená syntaktická pravidla

- Způsob zápisu definice podprogramu: hlavička a tělo
- Odsazení všech příkazů v těle podprogramu od levého okraje
- Způsob zápisu volání podprogramu (prázdné závorky za názvem podprogramu)

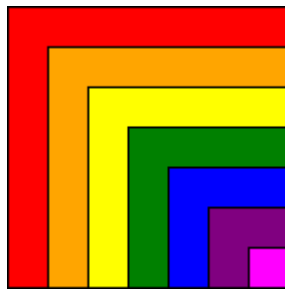
Poznámka

- Podprogramy mohou používat globální proměnnou `canvas`

Průběh výuky

První úloha slouží k opakování použití proměnných při kreslení:

1. Vytvoř program `duha.py`, který nakreslí kostičkovou duhu. Do proměnných `x`, `y` přiřaď souřadnice pravého dolního rohu kostičkové duhy a použij je při kreslení barevných čtverců. Nejmenší čtverec má rozměry 20 x 20 a každý další je o 20 větší:



Řešení:

```
import tkinter

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

x = 200
y = 200
canvas.create_rectangle(x - 140, y - 140, x, y, fill = 'red')
canvas.create_rectangle(x - 120, y - 120, x, y, fill = 'orange')
canvas.create_rectangle(x - 100, y - 100, x, y, fill = 'yellow')
canvas.create_rectangle(x - 80, y - 80, x, y, fill = 'green')
canvas.create_rectangle(x - 60, y - 60, x, y, fill = 'blue')
canvas.create_rectangle(x - 40, y - 40, x, y, fill = 'purple')
canvas.create_rectangle(x - 20, y - 20, x, y, fill = 'magenta')
```

V dalších úlohách chceme žáky naučit vytvářet a používat podprogramy, zatím bez parametrů a návratové hodnoty. Vyskytuje se zde několik nových principů: definování podprogramu, tělo podprogramu, volání podprogramu. Žáci si musí dát pozor na syntaktická pravidla: slovo `def`, symboly `()` : a odsazení od kraje. Kromě toho je nezbytné mít na paměti, že nejdříve je nutno podprogram definovat a až potom je možné jej volat. Vzhledem ke komplexnosti problematiky předkládáme žákům ukázkou práce s podprogramy, ale v prvním kroku nevysvětlujeme detaily (ty zmiňujeme až později).

2. Doposud jsi mohl psát jen takové příkazy, které počítač znal. Teď ho naučíš nové, své vlastní příkazy – tzv. **podprogramy**. Postupuj následovně:

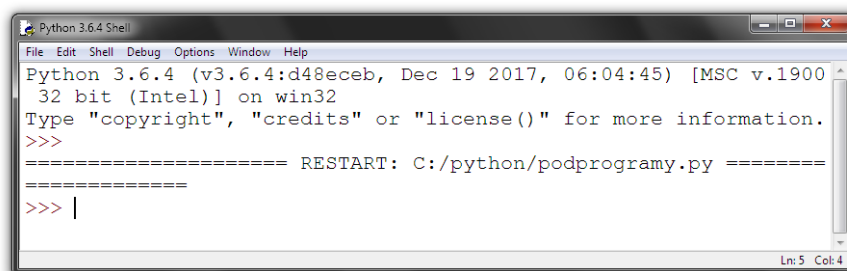
A) Vytvoř nový program `vypis.py`, ve kterém bude napsaný jen následující kód:

prázdné závorky i dvojtečka jsou velmi důležité

```
def vypis_text():
    print('*****')
    print('** Python **')
    print('*****')
```

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

B) Program spusť – jestli je všechno v pořádku, uvidíš:



C) Do příkazového řádku napiš:

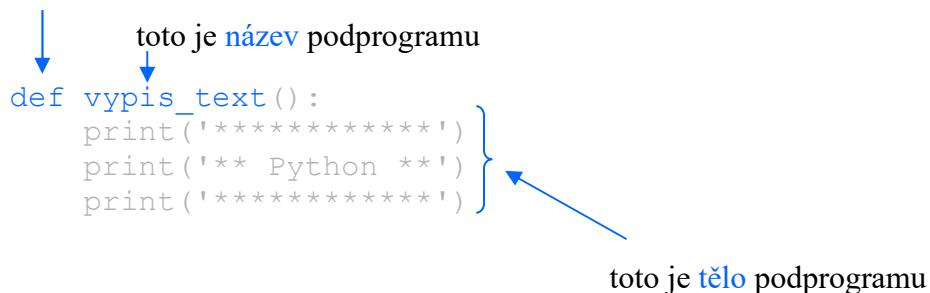
```
>>> vypis_text()
```

D) Jestli jsi postupoval správně, Python zobrazí text:

```
*****
** Python **
*****
```

Co se stalo?

slovem `def` začíná **definice** tvého nového příkazu – podprogramu



```
def vypis_text():
    print('*****')
    print('** Python **')
    print('*****')
```

toto je **tělo** podprogramu

Po spuštění programu se počítač naučil nový příkaz `vypis_text`. Počítač ho zatím nevykonával, jen se ho naučil. Skupinu příkazů `print` – tedy tělo podprogramu `vypis_text` – počítač vykoná až tehdy, když do příkazového řádku napíšeš:

```
>>> vypis_text()
```

prázdné závorky jsou velmi důležité

Takovýto zápis se nazývá **volání** podprogramu.

V části 2.B může žáky zmást, že se po spuštění programu zdánlivě nic nestane, jen se vypíše informace o restartu programu. Pokud by se někteří žáci dotazovali na správnost svého postupu, vysvětlíme jim, že postupovali správně, ale že se jimi zadané příkazy vykonají až po zavolání podprogramu v části 2.C.

Technická poznámka: Pokud si žáci navykli kopírovat ukázkové kódy z pracovních listů do svých programů pomocí schránky (např. pomocí klávesových zkratk `Ctrl+C` a `Ctrl+V`), měli by se od této chvíle naučit takto zkopírované kódy dodatečně kontrolovat. Je totiž pravděpodobné, že se při kopírování nezachová případné odsazení řádků od kraje, kvůli čemuž programy nebudou fungovat nebo budou fungovat chybně. Je tedy nutné zkopírovaný kód porovnat se zdrojovým kódem v pracovním listu a odsazení řádků od kraje doplnit.

3. Přidej do programu `vypis.py` další příkazy (jsou zvýrazněny žlutě) – pozor, tyto příkazy nesmí mít odsazení, protože už nepatří do podprogramu:

```
def vypis_text():
    print('*****')
    print('** Python **')
    print('*****')

print('Vítej!')
vypis_text()
print()
vypis_text()
print('to je konec')
```

Když program spustíš, uvidíš takovýto výsledek:

```
Vítej!
*****
** Python **
*****

*****
** Python **
*****
to je konec
```

V tomto programu se nejdříve definoval podprogram `vypis_text`. Za ním následují příkazy `print` a příkazy pro volání podprogramu `vypis_text`. Python zobrazil svoji vizitku dvakrát, protože v programu jsou dvě volání podprogramu `vypis_text`.

Na základě této úlohy by žáci měli pochopit, že podprogram můžeme zavolat i vícekrát.

4. Změň předchozí program tak, aby počítač vypsal:

```
Hello!
*****
** I am Python **
*****

How are you?
*****
** I am Python **
*****

I am fine.
*****
** I am Python **
*****

The end
```

Řešení:

```
def vypis_text():
    print('*****')
    print('** I am Python **')
    print('*****')

print('Hello')
vypis_text()
print('How are you?')
vypis_text()
print('I am fine.')
vypis_text()
print('The end')
```

Pokud to uznáme za vhodné, můžeme žákům prozradit, že alternativně lze odsazení příkazů od kraje zajistit pomocí jednoho stisku klávesy <Tab>. Python na dané místo vloží čtyři mezery.

V další úloze by si žáci měli uvědomit, že podprogram musí být definován ještě před prvním příkazem volání podprogramu:

5. Vytvoř nový program `pisen.py`, který bude obsahovat následující kód:

```
refren()
refren()
print()
print('když já jím dám ovsa')
print('oni skáčou hopsa')
print()
refren()
refren()

def refren():
    print('já mám koně vraný koně')
    print('to jsou koně mí')
```

Když program spustíš, Python vypíše chybové hlášení:

```
Traceback (most recent call last):
  File "D:\projekty-python\pisen.py", line 1, in <module>
    refren()
NameError: name 'refren' is not defined
```

Python ti tímto hlášením oznamuje, že na 1. řádku programu není možné volat podprogram `refren`, protože tento podprogram ještě nebyl definován.

Uprav program `pisen.py` tak, aby se úryvek písně vypsál správně.

Řešení:

```
def refren():
    print('já mám koně vraný koně')
    print('to jsou koně mí')

refren()
refren()
print()
print('když já jím dám ovsa')
print('oni skáčou hopsa')
print()
refren()
refren()
```

Žákům můžeme zdůraznit, že v Pythonu je zvykem všechny podprogramy zapisovat na začátku programu a až za nimi následuje samotná posloupnost příkazů, které mohou tyto podprogramy volat.

6. Před několika týdny jsi vytvářel program, který zobrazil tvoji vizitku, které byla podobná následující:

```
+-----+
|           www           |
| Petr      ( o o )      |
| LEV       ( ~ )        |
|           "             |
| Počítačový král       |
+-----+
```

Vytvoř nový program vizitka.py a v něm definuj podprogram vizitka, který takovou vizitku zobrazí. Nakonec tento podprogram zavolej, abys ověřil, že funguje správně.

Řešení:

```
def vizitka():
    print('+-----+')
    print('|           www           |')
    print('| Petr      ( o o )      |')
    print('| LEV       ( ~ )        |')
    print('|           "             |')
    print('| Počítačový král       |')
    print('+-----+')
vizitka()
```

Část žáků může mít podobné problémy se zápisem některých symbolů jako v 11. úloze 3. lekce. Můžeme jim připomenout symbol zpětného lomítka (\), pomocí kterého lze do textu vložit apostrof (jako \') nebo zpětné lomítko (jako \\).

7. Dopln do programu vizitka.py volání podprogramu vizitka tak, aby se pod sebe zobrazilo 10 tvých vizitek.

Řešení:

```
def vizitka():
    print('+-+-----+')
    print('|                www      |')
    print('| Petr      ( o o )  |')
    print('| LEV        ( ~ )    |')
    print('|                "      |')
    print('| Počítačový král    |')
    print('+-+-----+')
```

vizitka()
vizitka()
vizitka()
vizitka()
vizitka()

vizitka()
vizitka()
vizitka()
vizitka()
vizitka()

Tato úloha je zřejmou přípravou na cykly. Když se na ně budou žáci ptát, můžeme je ubezpečit, že za několik vyučovacích hodin se budou cykly učit i v tomto předmětu.

8. Ve svém programu můžeš definovat i více podprogramů. Vytvoř nový program obrazce.py a definuj v něm tři podprogramy. Každý z nich zobrazí jeden z následujících obrázků:

- podprogram noha nakreslí takovouto nohu (dole jsou dvě podtržítka vlevo i vpravo):

```
  |
__|__
```

- podprogram obdelnik nakreslí takovýto obdélník:

```
####
#   #
####
```

- podprogram trojuhelnik nakreslí takovýto trojúhelník:

```
  *
 ***
*****
```

Na konec programu vlož volání podprogramů, abys každý z nich otestoval. Potom zkus pomocí vytvořených podprogramů zobrazit následující obrázky:

<pre> * *** ***** * *** ***** ___ ___ </pre>	<pre> * *** ***** ##### # # ##### ___ ___ </pre>	<pre> ##### # # ##### ___ ___ ##### # # ##### </pre>	<pre> toto je noha: ___ ___ toto je obdélník: ##### # # ##### toto je trojúhelník: * *** ***** </pre>
---	--	--	--

Řešení – definování podprogramů + je potřeba k nim přidat další příkazy podle obrázku:

```

def trojuhelnik():
    print('  *')
    print(' ***')
    print('*****')

def obdelnik():
    print('#####')
    print('#      #')
    print('#####')

def noha():
    print('  |')
    print('___|___')

```

Řešení – zobrazení stroměčku:

```

trojuhelnik()
trojuhelnik()
noha()

```

Řešení – zobrazení domečku na kuří nožce:

```

trojuhelnik()
obdelnik()
noha()

```

Řešení – zobrazení činky:

```

obdelnik()
noha()
obdelnik()

```

Řešení – volání podprogramů s jejich popisky:

```
print('toto je noha:')  
noha()  
print()  
print('toto je obdélník:')  
obdelnik()  
print()  
print('toto je trojúhelník:')  
trojuhelnik()
```

Doposud podprogramy pracovaly pouze v textovém režimu. Proto následuje ukázka, ve které se kreslí pomocí podprogramů do grafické plochy, a úlohy, které toto procvičují:

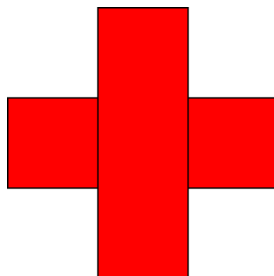
9. Teď budeš vytvářet podprogramy, které kreslí do grafického okna. Vytvoř nový program `kresba_podprogram.py` a vyzkoušej:

```
import tkinter  
  
canvas = tkinter.Canvas()  
canvas.pack()  
  
def kresli():  
    canvas.create_rectangle(10, 20, 30, 40, fill='red')  
  
kresli()
```

V této úloze tělo podprogramu obsahuje jediný příkaz – nakreslení obdélníku.

Proměnná `canvas` je zde tzv. globální proměnná. Globální proměnná existuje od svého vzniku (přiřazením mimo podprogram), v podprogramu se sice používá, avšak její hodnotu nelze v podprogramu měnit. Tyto informace však žákům vzhledem k jejich současným programátorským zkušenostem nebudeme prozrazovat.

10. Vytvoř nový program `kriz.py` a v něm definuj podprogram `kriz`, který po zavolání nakreslí červený kříž:



Řešení:

```
import tkinter

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

def kriz():
    canvas.create_rectangle(150 - 90, 100 - 30, 150 + 90,
                           100 + 30, fill='red')
    canvas.create_rectangle(150 - 30, 100 - 90, 150 + 30,
                           100 + 90, fill='red')

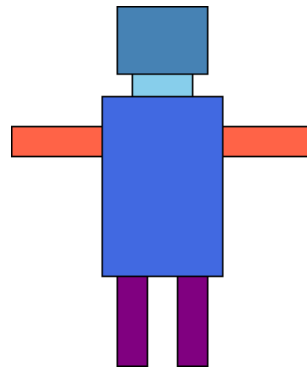
kriz()
```

V této úloze tělo podprogramu obsahuje posloupnost dvou příkazů. Při každém volání tohoto podprogramu se nakreslí nejprve první obdélník a přes něj druhý.

11* Vytvoř nový program `robot.py`, který bude schopen nakreslit robota. V programu budou čtyři podprogramy – hlava, ruce, nohy, telo – a každý z nich bude schopen nakreslit část robota. Když je zavoláš v následujícím pořadí:

```
hlava()
ruce()
nohy()
telo()
```

nakreslí se celý robot jako na obrázku vpravo:



Řešení:

```
import tkinter

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

def hlava():
    canvas.create_rectangle(160, 40, 200, 100,
                           fill='sky blue')
    canvas.create_rectangle(150, 10, 210, 55,
                           fill='steel blue')

def telo():
    canvas.create_rectangle(140, 70, 220, 190,
                           fill='royal blue')
```

```
def ruce():  
    canvas.create_rectangle(80, 90, 280, 110,  
        fill='tomato')  
  
def nohy():  
    canvas.create_rectangle(150, 160, 170, 250,  
        fill='purple')  
    canvas.create_rectangle(190, 160, 210, 250,  
        fill='purple')  
  
hlava()  
ruce()  
nohy()  
telo()
```

Toto je ukázka jednoho z možných řešení. Žáci zřejmě navrhnou nakreslení jiných obdélníků, jehož výsledkem bude akceptovatelný robot.