Vlastné funkcie

Riešenia čiastkových problémov programu je dobré umiestňovať do vlastných funkcií. Dosiahneme tým sprehľadnenie programu a ak potrebujeme vykonať rovnakú postupnosť príkazov viackrát s rôznymi vstupnými hodnotami, aj skrátenie kódu.

Definícia funkcie má tvar

```
def meno_funkcie (parameter1, parameter2,...):
        blok_príkazov
```

pričom meno_funkcie je identifikátor a parametre sú nepovinné (zátvorky áno!).

Ukážka 1:

```
def vypisNadpis():
   print("*****************************")
   print("*  OBSAH A OBVOD OBDĹŽNIKA  *")
   print("*****************************")
# volanie funkcie vypisNadpis()
vypisNadpis()
```

```
**************************
* OBVOD A OBSAH OBDĹŽNIKA *
**************************
```

Funkciu môžeme volať, uvedením jej mena a prípadných parametrov, až pod definíciou funkcie, inak nastane výnimka, v našom prípade NameError: name 'vypisNadpis' is not defined.

Ukážka 2:

```
def vypisNadpis(text):
   kolko = len(text)+2          # funkcia len(reťazec) vracia počet znakov - dĺžku (length) reťazca
   print(kolko*"*")             # použitý opakovací operátor * pre reťazce, konkrétne pre "*"
   print("*" + text + "*")
   print(kolko*"*")

def vypisObsahObvodObdlznika(stranaA, stranaB):
   print("Obsah = {:.2f}".format(stranaA*stranaB))
   print("Obvod = {:.2f}".format(2*( stranaA + stranaB)))

# ================= HLAVNÝ PROGRAM ==================
vypisNadpis(" OBSAH A OBVOD OBDĹŽNIKA ")
a = float(input("Zadaj stranu a obdĺžnika: "))
b = float(input("Zadaj stranu b obdĺžnika: "))
vypisObsahObvodObdlznika(a,b)
```

```
**************************
* OBVOD A OBSAH OBDĹŽNIKA *
**************************
Zadaj stranu a obdĺžnika: 5
Zadaj stranu b obdĺžnika: 2.5
Obsah = 12.50
Obvod = 15.00
```

V ukážke sú použité parametre. Slúžia na dovezenie hodnoty do funkcie. Pri písaní definície funkcie používame tzv. formálne parametre (v ukážke pomenované text, stranaA, stranaB), pri volaní funkcie do zátvoriek píšeme tzv. skutočné parametre – atribúty, premenné (v ukážke pomenované a, b) alebo rovno hodnoty (v ukážke reťazec " OBSAH A OBVOD OBDĹŽNIKA "), s ktorými sa uskutoční výpočet vo funkcii. Názov formálneho a skutočného parametra môžu ale nemusia byť rovnaké (ukážeme ďalej).

Poznámka „... = HLAVNÝ PROGRAM = ...“ slúži len na optické oddelenie oblasti definície funkcií od oblasti ich použitia.

Ukážka 3:

```
def vypisNadpis(text):
   kolko = len(text)+2
   print(kolko*"*")
   print("*" + text + "*")
   print(kolko*"*")
```

```
def vratRozmeryObdlznika():
   strA = float(input("Zadaj stranu a obdĺžnika: "))
   strB = float(input("Zadaj stranu b obdĺžnika: "))
   return strA, strB

def vypisObsahObvodObdlznika(stranaA, stranaB):
   print("Obsah = {:.2f}".format(stranaA*stranaB))
   print("Obvod = {:.2f}".format(2*(stranaA + stranaB)))
# ================= HLAVNÝ PROGRAM ==================
vypisNadpis(" OBSAH A OBVOD OBDĹŽNIKA ")
a, b = vratRozmeryObdlznika()
vypisObsahObvodObdlznika(a,b)
```

V ukážke 3 je niekoľko nových vecí. Po prvé, úlohou funkcie vratRozmeryObdlznika je vyviesť z funkcie v nej získané rozmery obdĺžnika. Dosahuje sa to použitím príkazu return. Príkazom **return** sa ukončí výpočet funkcie a uvedená hodnota sa **vyvezie** ako výsledok funkcie. Vyviesť možno napríklad číselnú hodnotu/y, reťazec, False alebo True,... Ak sa má z funkcie niečo vyviesť, v jej bloku príkazov sa musí aspoň raz vyskytnúť príkaz return. Poznámka: Ak je príkaz return použitý vo funkcii v cykle, jeho vykonaním sa vystúpi z funkcie, a teda aj z cyklu, ako by bol použitý aj príkaz break.
V ukážke sú použité dva druhy premenných. Premenné, ktoré zavedieme v definícii funkcie, možno použiť len vo funkcii a mimo nej neexistujú, hovoríme im **lokálne** (v ukážke strA, strB, stranaA, stranaB), existujú len počas vykonávania funkcie. Premenné zavedené v nadradenom bloku (mimo definícií funkcií), sa nazývajú **globálne**. Lokálnu aj globálnu premennú možno zrušiť príkazom del premenná. Identifikátory ako int, float a ďalšie vyhradené slová sú štandardné a nemožno ich zrušiť ani ich názov použiť v inom význame.

V nasledujúcej ukážke 4 sme použili lokálnu funkciu vratStranu a v nej priamo príkaz return. Výstup je rovnaký ako v predchádzajúcich ukážkach. Zároveň sme použili rovnaké označenie pre lokálne aj globálne premenné, to však neznamená, že v programe boli použité len dve pamäťové miesta označené a a b!

Ukážka 4:

```
def vypisNadpis(text):
   kolko = len(text)+2
   print(kolko*"*")
   print("*" + text + "*")
   print(kolko*"*")

def vratRozmeryObdlznika():

   def vratStranu(ktoru):
      return float(input("Zadaj stranu " + ktoru + " obdĺžnika: "))

   a = vratStranu("a")                    # volanie funkcie vratStranu so skutočným parametrom "a"
   b = vratStranu("b")                    # volanie funkcie vratStranu so skutočným parametrom "b"
   return a, b

def vypisObsahObvodObdlznika(a, b):
   print("Obsah = {:.2f}".format(a*b))
   print("Obvod = {:.2f}".format(2*(a + b)))
# ================= HLAVNÝ PROGRAM ==================
vypisNadpis(" OBSAH A OBVOD OBDĹŽNIKA ")
a, b = vratRozmeryObdlznika()          # jediné globálne premenné
vypisObsahObvodObdlznika(a,b)
```

Poznámka

Pri niektorých údajových typoch dochádza k takému prepojeniu formálneho a skutočného parametra, že pri niektorých operáciách s formálnym parametrom sa tieto zmeny automaticky realizujú aj so skutočným parametrom, t.j. „vyvážajú“ sa bez použitia príkazu return (pozri záver kapitoly Lineárne vyhľadávanie príklad L.10).

Dokumentačný reťazec

Ak **pod hlavičkou funkcie**, t.j. pod riadkom def meno_funkcie(...):, je uvedená viacriadková poznámka (pozri študijný text Údajové typy), táto sa stáva tzv. **dokumentačným reťazcom** (**docstring**). Docstring, aj s úplnou hlavičkou funkcie, môže užívateľ zobraziť volaním funkcie **help** - zápisom: help(meno_funkcie). Docstring môže byť jedno alebo viacriadkový, ale musí začínať a končiť troma apostrofmi alebo troma úvodzovkami. Okrem toho, čo funkcia robí, sa v dokumentačnom reťazci často uvádza aj úloha a rozsah dovolených hodnôt jednotlivých parametrov funkcie.

Napríklad:

```
def mlen(s):
   '''vráti dĺžku reťazca s, t.j. počet jeho znakov '''
   dlzka = 0
   for znak in s:
      dlzka += 1
   return dlzka
```

Použitie:

```
>>> help(mlen)
```

vypíše:

```
Help on function mlen in module __main__:

mlen(s)
   vráti dĺžku reťazca s, t.j. počet jeho znakov
```

alebo

```
def mcopyEndStr(s,od):
   """
   vráti podreťazec z reťazca s od zadaného znaku do konca reťazca s
   s - reťazec
   od - nezáporné celé číslo
   """
   novy = ""
   for i in range(od, len(s)):
      novy += s[i]
   return novy
```

vypíše:

```
Help on function mcopyEndStr in module __main__:

mcopyEndStr(s, od)
   vráti podreťazec z reťazca s od zadaného znaku do konca reťazca s
   s - reťazec
   od - nezáporné celé číslo
```

Moduly

Často používané funkcie alebo konštanty si môžeme ľahko umiestniť do „vlastného“ modulu. Modul musí byť do programu, v ktorom chceme použiť funkcie modulu, importovaný[1]. Všetky funkcie a premenné definované v importovanom module sú prístupné cez bodkovú notáciu modul.funkcia alebo modul.premenná.
Ukážeme si to na module ktorý nazveme vratcislo. Modul bude obsahovať funkcie na vrátenie ošetreného zadaného celého alebo reálneho čísla, prípadne nezáporného celého čísla.

Modul vratcislo je uložený v súbore vratcislo.py:

```
# modul vratcislo

def cele(text):
    """ umožňuje zadať ošetrené celé číslo
        parametrom je text zobrazený pri volaní input
    """
    while True:
        vstup = input(text)
        try:
            cele_cislo = int(vstup)
            return cele_cislo
        except ValueError:
            print("Zadaný prázdny reťazec alebo nedovolený znak!")

def realne(text):
    """ umožňuje zadať ošetrené reálne číslo
        parametrom je text zobrazený pri volaní input
    """
    while True:
        vstup = input(text)
        if "," in vstup: print("Opravil som desatinnú čiarku na bodku!")
        vstup = vstup.replace(',','.')
        try:
            realne_cislo = float(vstup)
            break
        except ValueError:
            if len(vstup) == 0:
                print("Zadal si prázdny reťazec!")
            else:
                print("Zadal si nedovolený znak!")
    return realne_cislo

def nezaporneCele(text):
    """ umožňuje zadať ošetrené nezáporné celé číslo
        parametrom je text zobrazený pri volaní input
    """
```

[1] Modul je tiež možné importovať pod novým menom (napríklad kratším) import staré_meno_modulu as nové_meno. Použitie nové_meno.funkcia(...).

```
    while True:
        vstup = input(text)
        try:
            nezaporne_cele_cislo = int(vstup)
            if nezaporne_cele_cislo < 0:
                print("Zadal si záporné číslo!")
                continue
            else:
                break
        except ValueError:
            if len(vstup) == 0:
                print("Zadal si prázdny reťazec!")
            else:
                print("Zadal si nedovolený znak!")
    return nezaporne_cele_cislo
```

Použitie:

```
import vratcislo

cele = vratcislo.cele("Zadaj celé číslo: ")
print(cele)

nezaporne_cele = vratcislo.nezaporneCele("Zadaj nezáporné celé číslo: ")
print(nezaporne_cele)

realne = vratcislo.realne("Zadaj reálne číslo: ")
print(realne)
```

Výpis:

```
Zadaj celé číslo: 25
25
Zadaj nezáporné celé číslo: -0,5
Zadal si nedovolený znak!
Zadaj nezáporné celé číslo: 0
0
Zadaj reálne číslo: 6,78
Opravil som desatinnú čiarku na bodku!
6.78
```

Modul pole uložený v súbore pole.py obsahuje funkcie na vytvorenie poľa celých čísel a poľa reálnych čísel zadaním hodnôt alebo náhodným generovaním, po zadaní počtu prvkov poľa a funkcie na vypísanie prvkov poľa v riadku a pod seba. Funkcie modulu využívajú modul vratcislo.

```
import vratcislo, random

def vypis (pole, text="Prvky poľa:"):
    """ Vypíše zadaný text a prvky poľa """
    print(text)
    print(pole)
```

```
def vytvorIntZadanim():
    """ Vytvorí pole zadaním hodnôt prvkov poľa """
    pocetPrvkov = vratcislo.nezaporneCele("Počet prvkov poľa: ")
    pole = []
    for i in range(pocetPrvkov):
        zadal = vratcislo.cele(str(i+1)+". prvok poľa: ")
        pole.append(zadal)
    vypis(pole)
    return pole

def vytvorIntNahodne(od=1,po=9):
    """ Vytvorí pole vygenerovaním celých náhodných čísel od 'od' po 'po' """
    pocetPrvkov = vratcislo.nezaporneCele("Počet prvkov poľa: ")
    pole = [random.randint(od,po) for _ in range(pocetPrvkov)]
    vypis(pole)
    return pole

def vypisPodSeba(pole):
    """ Vypíše nadpis a prvky poľa """
    print("Prvky poľa:")
    for prvok in pole:
        print(prvok)

def vytvorRealZadanim():
    """ Vytvorí pole zadaním hodnôt prvkov poľa """
    pocetPrvkov = vratcislo.nezaporneCele("Počet prvkov poľa: ")
    pole = []
    for i in range(pocetPrvkov):
        pole.append(vratcislo.realne(str(i+1)+". prvok: "))
    vypisPodSeba(pole)
    return pole

def vytvorRealNahodne(nasobok=1):
    """ Vytvorí pole vygenerovaním reálnych náhodných čísel z <0;nasobok) """
    pocetPrvkov = vratcislo.nezaporneCele("Počet prvkov poľa: ")
    pole = []
    for _ in range(pocetPrvkov):
        pole.append(random.random()*nasobok)
    vypisPodSeba(pole)
    return pole
```

Použitie:

```
import pole

pole.vytvorIntNahodne(0,100)
pole.vytvorRealZadanim()
pole.vytvorRealNahodne(10)
```

Výpis:

```
Počet prvkov poľa: 5
Prvky poľa:
[49, 23, 66, 93, 44]
Počet prvkov poľa: 7
1. prvok: 3,9
Opravil som desatinnú čiarku na bodku!
2. prvok: 2.4
3. prvok: -5.8
4. prvok: 0
5. prvok: 2
6. prvok: 1.98
7. prvok: -120.0
Prvky poľa:
3.9
2.4
-5.8
0.0
2.0
1.98
-120.0
Počet prvkov poľa: 6
Prvky poľa:
6.480693542887606
5.738841779925635
8.474320957629214
6.642225950279521
4.230932903860523
2.324155597495823
```

Ak chcete mať možnosť využívať funkcie modulu vo svojom ľubovoľnom programe, umiestnite súbor modulu (pole.py) do svojho priečinka k zvyšným pythonovským programom alebo využite postup uvedený v študijnom texte Úvod, inštalácia na strane 3. Nezabudnite na rovnaké miesto (priečinok) umiestniť aj súbor vratcislo.py.