KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
jan.lastovicka@upol.cz
www.inf.upol.cz/lide/jan-lastovicka
17. listopadu 12, 771 46 Olomouc

Základy programování pro IT 1

4. Funkce

verze z 15. října 2024

Funkce pojmenovávají určitou část programu a umožňují později tuto část vykonat. **Jména funkcí** většinou obsahují sloveso. Například `get_abs` může být jméno funkce. Jména proměnných a funkcí musí být různá. Nejprve se podíváme na to, jak funkce definovat.

Vezměme proměnné *parameter1*, ..., *parametern* jméno funkce *function*, program *program* a proměnnou *variable*. Program *program* musí být spustitelný v prostředí pouze s proměnnými *parameter1*, ..., *parametern*. Proměnná *variable* může být jedna z proměnných *parameter1*, ..., *parametern* nebo proměnná přidaná programem *program*. Příkaz:

```
def function(parameter1, ..., parametern):
    program
    return variable
```

se vykoná tak, že definuje funkci *function*. Proměnné *parameter1*, ..., *parametern* se nazývají **parametry funkce**, program *program* se nazývá **tělo funkce** a proměnná *variable* **návratová proměnná**. Například příkaz:

```
def get_max(x, y):
    if x >= y:
        z = x
    else:
        z = y
    return z
```

definuje funkci `get_max` s parametry `x` a `y`. Dále si ukážeme, jak si vyžádat vykonání těla funkce.

Vezměme funkci *function* a hodnoty *value1*, ..., *valuem*. Pak můžeme funkci *function* **zavolat** s hodnotami *value1*, ..., *valuem*. Hodnoty, které se volání účastní, se nazývají **argumenty**. Volání funkce probíhá následovně.

1. Vezmeme parametry *parameter1*, ..., *parametern* funkce *function*.

2. Zkontrolujeme, že počet parametrů se rovná počtu argumentů. Tedy, že m se rovná n.
3. Pokud se počty liší, volání funkce vyvolá výjimku a tím volání končí.
4. Jinak se vytvoří nové prostředí, kde parametry mají postupně hodnoty argumentů. Tedy *parameter*i má hodnotu *value*i pro každé $1 \leq i \leq n$. Nově vzniklé prostředí bude vypadat takto:

parameter1	*value1*
⋮	⋮
parameter$n$	*value*n

5. V právě vytvořeném prostředí se vykoná tělo funkce.
6. Získá se hodnota návratové proměnné funkce.

Získané hodnotě se říká **návratová hodnota** volání funkce. Například zavolání funkce `get_max` s argumenty `3` a `4` povede na vytvoření nového prostředí:

`x`	`3`
`y`	`4`

Po vykonáním těla funkce:

```
if x >= y:
    z = x
else:
    z = y
```

se prostředí změní na:

`x`	`3`
`y`	`4`
`z`	`4`

Hodnota návratové proměnné `z` je číslo `4`, které bude i návratovou hodnotou volání funkce.

Volání funkce můžeme zapsat výrazem. Pokud *function* je funkce a *expression1*, ..., *expression*m jsou výrazy, pak

```
function(expression1, ..., expressionm)
```

je výraz **volání funkce**, který se vyhodnotí tak, že se postupně vyhodnotí výrazy *expression1*, ..., *expressionm* tím získáme hodnoty *arg1*, ..., *argm* a následně se zavolá funkce *function* s argumenty *arg1*, ..., *argm*. Návratová hodnota volání funkce bude hodnotou výrazu. Například vyhodnocení výrazu

```
get_max(3 + 2, 2 + 1)
```

povede na zavolání funkce `get_max` s argumenty `5` a `3`, které vrátí `5`. Výraz se tedy vyhodnotí na číslo `5`.

Tím, že volání funkce vytváří nové prostředí, ve kterém se vykonává jeho tělo, nedojde ke změně proměnných, ze kterého byla funkce volána. Například:

```
>>> x = 1
>>> get_max(3, 4)
4
>>> x
1
```

Funkci můžeme samozřejmě zavolat i z jiné funkce. Vezměme například funkci:

```
def get_max3(x, y, z):
    m1 = get_max(x, y)
    m2 = get_max(m1, z)
    return m2
```

Funkce vrací maximum ze tří zadaných čísel:

```
>>> get_max(1, 5, 2)
5
```

Tělo funkce můžeme zjednodušit:

```
def get_max3(x, y, z):
    m2 = get_max(get_max(x, y), z)
    return m2
```

Funkce, která vrací logickou hodnotu, se nazývá **predikát**. Říkáme, že predikát **rozhoduje** jistou otázku pokud v případě, že je odpověď ano vrací `True` jinak `False`. Například predikát:

```
def is_even(number):
    val = (number % 2) == 0
    return val
```

rozhoduje, zda je zadané číslo sudé. Ukázky volání funkce:

```
>>> is_even(4)
True
>>> is_even(5)
False
```

Otázky a úkoly k semináři

1. Napište funkci `get_min`, která vrací minimum ze dvou zadaných čísel.
2. Napište funkci `get_min3` vracející minimum ze tří zadaných čísel.
3. Napište funkci `get_abs` vracející absolutní hodnotu zadaného čísla.
4. Napište predikát `is_odd`, který rozhoduje, zda je zadané číslo liché.
5. Napište funkci `get_last_digit`, která vrátí poslední číslici zadaného čísla.
6. Napište funkci `remove_last_digit`, která odebere poslední číslici ze zadaného čísla.
7. Napište funkci `append_digit`, která očekává číslo a číslici. Funkce přidá zadanou číslici za poslední číslici zadaného čísla. Například:

   ```
   >>>  append_digit(12, 3)
   123
   ```

8. Definujte funkci `get_butlast_digit`, která vrátí předposlední číslici zadaného čísla. Funkce musí v těle volat funkce `get_last_digit` a `remove_last_digit`.
9. Napište predikát `is_nonzero_divisor`, který rozhodne, zda nenulové číslo dělí jisté číslo. Například:

   ```
   >>> is_nonzero_divisor(2, 6)
   True
   >>> is_nonzero_divisor(5, 6)
   False
   ```

10. Napište predikát, který rozhodne, zda pro tři zadaná čísla *`number1`*, *`number2`* a *`number3`* platí, že součet čtverců čísel *`number1`* a *`number2`* je roven čtverci *`number3`*. Funkce například musí vrátit pravdu pro čísla `3`, `4` a `5`, protože $3^2 + 4^2 = 5^2$.

11. Napište predikát rozhodující trojúhelníkovou nerovnost. Tedy zda pro tři zadaná čísla platí $x + y > z$.

12. Napište predikát rozhodující, zda lze sestrojit trojúhelník, kde máme zadány délky jeho stran.

13. Napište predikát `is_divisor`, která rozhodne, zda jedno číslo je dělitelem druhého. Přitom x je dělitel y, jestliže existuje číslo z takové, že $x \cdot z = y$. Predikát musí fungovat pro libovolná celá čísla.

14. Napište takzvanou znaménkovou funkci, která pro kladná čísla vrátí jedničku, pro nulu nulu a pro záporná minus jedničku.

15. V následujících úkolech budete často žádáni, že máte něco spočítat. Myslí se tím, že máte napsat funkci, která výpočet provede. Sečtěte všechna čísla v uzavřeném intervalu zadaném jeho mezemi.

16. Spočítejte faktoriál zadaného čísla. Faktoriál nuly je jedna a faktoriál nenulového čísla n je roven součinu n a faktoriálu čísla $n - 1$.

17. Napište funkci, `get_fibonacci`, která pro nezáporné číslo i vrátí i plus první prvek Fibonacciho posloupnosti. Například:

    ```
    >>> get_fibonacci(0)
    0
    >>> get_fibonacci(1)
    1
    >>> get_fibonacci(6)
    8
    ```

    První dva prvky Fibonacciho posloupnosti jsou nula a jedna. Každý další prvek je součtem dvou předchozích prvků.

18. Vraťte největšího společného dělitele dvou zadaných čísel.

19. Vraťte nejmenší společný násobek dvou zadaných čísel.

20. Rozhodněte, zda je číslo prvočíslem.

21. Rozhodněte, zda je číslo čtvercové (je druhou mocninou nějakého čísla).

22. Rozhodněte, zda je možné číslo vyjádřit součtem dvou čtvercových čísel.

23. Spočítejte ciferný součet zadaného čísla.

24. Vraťte celou část druhé odmocniny zadaného čísla.

25. V následujících úkolech můžete ve výrazech použít pouze přičítání a odečítání jedničky a porovnání čísla s nulou. Dále je povoleno volat funkce, které si v této části úkolů naprogramujete. Předpokládejte, že všechna čísla, o kterých se dále mluví jsou nezáporná.

(a) Spočítejte součet dvou zadaných čísel.

(b) Spočítejte rozdíl dvou čísel a, b, kde b je menší nebo rovno než a.

(c) Jsou zadána dvě čísla a a b. Rozhodněte, zda je a menší nebo rovno než b.

(d) Rozhodněte, zda se dvě zadaná čísla rovnají.

(e) Spočítejte součin dvou zadaných čísel.

(f) Spočítejte celočíselný podíl dvou zadaných čísel.

(g) Spočítejte zbytek po celočíselného podílu dvou zadaných čísel.

(h) Umocněte zadané číslo na zadaný exponent.