Funkce

Funkce je předdefinovaný sled příkazů, který se vyvolává zadáním názvu funkce.

Funkce něco vykoná (napíše, přečte, vypočítá, ...) a případně něco udělá (vytiskne), nebo vrátí nějakou hodnotu, kterou můžeme dále zpracovat.

Funkce můžou (ale nemusí) mít i předefinované parametry, které se pak zapisují do kulatých závorek uvedených za názvem funkce.

Definování funkce

Funkce se definuje příkazem „***def***", za kterým následuje námi zvolený název funkce, kulaté závorky a dvojtečka. Následuje pole příkazů odskočených o 4 mezery:

```
def meno_funkcie():     # zapamatuj si blok příkazů jako nový příkaz
    příkaz
    příkaz
    ...
```

Parametry funkce

Parametr funkce je ***dočasná proměnná*****,** která se vytvoří při volání funkce a skrze kterou můžeme do funkce poslat nějakou hodnotu.

Parametr funkce definujeme během definice funkce v hlavičce funkce. Pokud je jich více, oddělujeme je čárkou.

Při definování funkce v hlavičce funkce uvádíme tzv. ***formální parametry*** – to jsou nové proměnné, které vznikají až při volání funkce.

Při volání funkce musíme do závorek zapsat hodnoty, které se stanou tzv. ***skutečnými parametry*** – tyto hodnoty se při volání funkce přiřadí do ***formálních parametrů***.

Proměnné, které vznikají po čas běhu funkce, se stávají ***lokálními proměnnými*** – budou existovat jen po čas běhu funkce a po skončení funkce se automaticky zruší. Stejně tak i proměnné definované v těle funkce vznikají při startu funkce a ruší se při jejím skončení.

Vyvolání funkce

K vyvolání funkce stačí zadat název funkce se závorkami (kde mohou, ale nemusí být uvedeny další údaje – záleží na požadavcích pro funkci) a funkce se spustí.

Funkci samozřejmě můžeme volat pouze tehdy, když Python již zná její definici.

Seznam kroků, které se vykonají po spuštění funkce:

1. *přeruší se vykonávání běžícího programu (Python si přesně zapamatuje místo, kde se to stalo)*
2. *skočí se na začátek volané funkce*
3. pokud jsou uvedeny - vytvoří se nové proměnné (***formální parametr***) a přiřadí se do něj hodnota ***skutečného parametru***, který jsme uvedli v závorce při volání funkce.
4. *postupně se vykonají všechny příkazy funkce*
5. zruší se všechny proměnné, které vznikly během vykonávání funkce.
6. *když se dojde na konec funkce, zrealizuje se* ***návrat*** *na zapamatované místo, kde se přerušilo vykonávání programu a pokračuje se ve vykonávání dalších příkazů za voláním funkce.*

Jmenný prostor

K tomu, abychom pochopili, jak fungují lokální proměnné, je potřeba pochopit, jak funguje jmenný prostor (namespace). Všechny identifikátory v Pythoně jsou jedním z následujících 3 typů (3 různých tabulek jmen):

1) **standardní** (builtins)
   Tabulka standardních jmen je v Pythonu předefinovaná a pro celý program jen jedna.
2) **globální** (main)
   Tabulku globálních jmen vytváříme na nejvyšší úrovni programu přidělováním proměnných (včetně názvů funkcí) a je pro celý program jen jedna.
3) **lokální** - vznikají po čas běhu funkce
   Tabulka lokálních jmen se „soukromě“ vytváří pro každou funkci při jejím startu a ruší se při jejím konci.

Když na nějakém místě použijeme identifikátor, Python ho nejprve hledá v ***lokální tabulce jmen***, když zde identifikátor nenajde, hledá ho v ***globální tabulce jmen***, a když ho nenajde ani tady, hledá ho v ***standardní tabulce jmen***. A když ho nenajde v žádné z těchto tabulek, nahlásí chybu:

NameError: name 'identifikátor' is not defined

Příkaz dir()

vypíše tabulku ***globálních jmen***. Tato tabulka, i když je „prázdná“ obsahuje několik speciálních jmen, které začínají a končí dlouhými podtržítky. Za nimi se pak objevují námi vytvořené globální proměnné.

K tabulce standardních jmen se můžeme dostat přes příkaz ***dir(__builtins__)***. Zde se nám pak vypíší všechny předdefinované jména Pythonu.

Příkaz del

Tímto příkazem rušíme identifikátor z globální a lokální tabulky jmen, (v standardní tabulce jmen nelze nic měnit, a zapisuje se: ***del premenná***

Příkaz nejprve zjistí, v které tabulce se identifikátor nachází a potom ho z té tabulky vymaže. Příkaz nejprve kouká do lokální a po té do globální tabulky jmen.

Pokud si k lokální, nebo globální proměnné omylem přiřadíme název ze standardních proměnných a ten pak nefunguje, můžeme příkazem del toto přiřazení zrušit a znovu tak bude fungovat přiřazení ze standardní tabulky jmen.

Vnořené volání

Vzniká, když se v těle funkce nachází volání jiné funkce. V takovém případě se nejprve vytvoří jmenný prostor v lokální tabulce jmen a při jejich volání se vnoří nový jmenný prostor, vykává se příkaz a po té se tento jmenný prostor zruší.

Stránka pro kontrolu krokování a vytváření jmenných prostorů:
http://www.pythontutor.com/visualize.html#mode=edit

Funkce s návratovou hodnotou

Většina standardních (ne všechny) funkcí má nějakou návratovou hodnotu. Návratová hodnota se vytváří pomocí příkazu ***return***.

Příkazem return se ukončí výpočet funkce a zruší se její jmenný prostor. Výsledná hodnota se stává výsledkem funkce.

Rozlišujeme tak dva typy funkcí:

1) ty, které něco dělají (např. vypisují, nebo kreslí), ale nevracejí žádnou návratovou hodnotu.
2) ty, které něco vypočítají a vrátí nějakou návratovou hodnotu (obsahují v sobě příkaz return). Návratová hodnota může obsahovat číslo, nebo logický argument (True, False), nebo řetězec.

V případě logické návratové hodnoty není při jejím zápisu nutné používat parametr „if“:

```
def parne(n):
    if n % 2 == 0:
        return True
    else:
        return False
```

ale zapisuje se ve zkrácené podobě:

```
def parne(n):
    return n % 2 == 0
```

Typy parametrů a typ výsledku

Python nekontroluje typy parametrů, ale kontroluje, co se s nimi dělá ve funkci. To znamená, že funkce, která bude fungovat pro čísla, nebude fungovat pro řetězce a spadne. Je ale možné v těle funkce kontrolovat typ parametru, a tak funkce může fungovat i pro čísla a i pro řetězce.

např. takto:

```
def pocitaj(x):
    if type(x) == str:
        return 2*x + '1'
    else:
        return 2*x + 1
```

Náhradní hodnoty parametrů

Pokud chceme u funkce předvolit některý parametr, tak aby v případě jeho neuvedení došlo k použití k námi předvyplněného parametru, zadáváme v řádku definice funkce do závorky za parametr rovnítko (bez mezer) a za něj uvedeme námi vybranou hodnotu. Tato hodnota pak bude použita vždy, pokud nebude uvedená jiná a stává se tzv. **náhradní hodnotou** (default)

Při tom platí, že pokud nadefinujeme náhradní hodnotu u jedné položky, musíme pak nadefinovat náhradní hodnoty i u všech další položek, které se nacházejí za ní.

```
def kresli_bodku(x, y, farba='red', r=5):
```

Pokud parametry zadáváme pouze hodnotou, musíme je vždy uvádět v pořadí, v kterém jsou uvedeny v definici funkce, pokud ale parametry zadáváme i s jejich popisným jménem, můžeme je zadat v libovolném pořadí.