Jazyk Python

Python je interpretr. Interpretr nevytváří spustitelný kód (například soubor.exe v systému Windows), proto Python musí být nainstalován do vašeho počítače, aby spustil program. Interpretr také umožňuje interaktivní práci s prostředím.

Python pracuje s několika různými typy dat:

1) **Celá čísla** - Mají stejný význam, jaký známe z matematiky, jsou psána v desetinném systému a mohou začít plus nebo mínus znamení. Jejich velikost (počet cifer) je omezena pouze kapacitou pracovní pamětí.

2) **Desetinná čísla** - Obsahují desetinnou tečku nebo exponenciální část (například 1E+15). Nastávají i v důsledku některých operací, například dělením dvou celých čísel. Mají omezenou přesnost (přibližně 16-17 platný cifer).

3) **Znakové řetězce** - Jejich délka (počet znaků) je omezena pouze kapacitou pracovní paměti.

Zapisují se mezi apostrofy '*text*', nebo uvozovky "*text*". Oba zápisy jsou rovnocenné, pouze musí začínat a končit stejně. Mohou obsahovat diakritiku. Mohou být i prázdné (' ').

V Pythonu má každý typ své jméno:

Int - Pro celá čísla, například 0, 1, 15, -123456789,…

float - Pro desetinná čísla, například 0,0, 3,14159, 2,0000000001, 33E50,…

srt - Pro znakové řetězce jako 'a', "ABC", '', "Jsem šťastný"

Zjistit o jaký se jedná typ můžeme v Pythonu pomocí standardní funkce: **type ()**

Různé typy hodnot mají různé operandy:

Celočíselné operace (Oba operandy musí mít celočíselný typ, není možné dělit 0)

+ (sčítání) - např. 1 + 2 má hodnotu 3

- (odečítání) - např. 2 - 5 má hodnotu -3

***** (násobení) - např. 3 * 37 má hodnotu 111

// (celočíselné dělení - např. 22 // 7 má hodnotu 3

% (zbytek po celočíselném dělení) - např. 22 % 7 má hodnotu 1

****** (umocňování) - např. 2 ** 8 má hodnotu 256

Operace s desetinnými čísly (Alespoň jeden z operandů musí být typ desetinného pole, výjimkou je dělení, není možné dělit 0)

+ (sčítání) - např. 1 + 0,2 má hodnotu 1,2

- (odečítání) - např. 6 - 2.86 má hodnotu 3,14

***** (násobení) - např. 1.5 * 2.5 má hodnotu 3,75

/ (dělení) - např. 23/3 má hodnotu 7,6666666666667

// (celočíselné dělení) - např. 23.0 // 3 má hodnotu 7,0

% (zbytek po celočíselném dělení) - např. 23,0 % 3 má hodnotu 2,0

****** (umocňování) - např. 3 ** 3. má hodnotu 27.0

** (1/2) (odmocňování) - např. 3 ** (1/2) má hodnotu 1.7320508075688772

Operace se znakovými řetězci

+ (spojení dvou řetězců) - např. 'a' + 'b' má hodnotu 'ab'

***** (vícenásobné zřetězení) - např. 3 * 'x' má 'xxx'

Proměnné a přiřazení

Proměnná je pojmenována nějaká existující hodnota v paměti. Proměnná vzniká vykonáním přiřazujícího příkazu (určité existující hodnotě je přiřazena název).

Jméno proměnné může obsahovat písmena, číslice a znak podtržítka a musí se lišit od pythonovských příkazů (tzv. rezervovaných slov), např. *if*, *de*f, *return*, atd. Python rozlišuje malá a velká písmena.

Přiřazení proměnné zapisujeme znakem rovná se:

Proměnná = hodnota

Tímto se v Pythonu vytvoří vazba na danou hodnotu. Při přiřazení jiného odkazu k proměnné se tato vazba mění. Více proměnných může odkazovat na stejnou hodnotu. Na jeden název může být v jednu chvíli přiřazená pouze jedna hodnota. Pokud proměnná odkazuje na jinou proměnnou (např. x = a), neodkazuje tak na její název, ale hodnotu. Python udržuje všechny proměnné v paměti názvů proměnných a všechny aktuálně vytvořené hodnoty v paměti hodnot.

Možnosti zkráceného zápisu různých přiřazovacích příkazů:

jméno_proměnné += hodnota (jméno_proměnné = jméno_proměnné + hodnota)

jméno_proměnné -= hodnota (jméno_proměnné = jméno_proměnné - hodnota)

jméno_proměnné *= hodnota (jméno_proměnné = jméno_proměnné * hodnota)

jméno_proměnné /= hodnota (jméno_proměnné = jméno_proměnné / hodnota)

jméno_proměnné //= hodnota (jméno_proměnné = jméno_proměnné // hodnota)

jméno_proměnné %= hodnota (jméno_proměnné = jméno_proměnné % hodnota)

jméno_proměnné **= hodnota (jméno_proměnné = jméno_proměnné ** hodnota)

Příklady použití:

```
x = 45
x -= x        # to stejné jako x = 0
x += x        # to stejné jako x *= 2

z = 'abc'
z += z
print(z) >>> abcabc
```

Je také možné přiřadit tu samou hodnotu do více proměnných:

x = 0
sucet = 0
pocet = 0
ab = 0

A to je možné také zapsat jediným **vícenásobným přiřazením**:

x = sucet = pocet = ab = 0

Další užitečnou variantou je **paralelní přiřazení**, kdy můžeme přiřadit více hodnot více proměnným, či je prohodit mezi sebou. V takovémto případě musí být dodržen stejný počet proměnných a jejich hodnot.

x, y, meno, pi = 120, 255, 'bod A', 3.14

Dobrým využitím této možnosti je **výměna obsahů** u proměnných.

a, b = b, a

Paralelní přiřazení funguje i v případě, kdy na straně příkazu je jediný znakový řetězec, pokud má přesně tolik znaků, kolik je proměnných.

a, b, c, d, e, f = 'Python'

print(a, b, c, d, e, f) >>> P y t h o n

Režim programování

V **IDLE** vytváříme nové okno (File > New File, nebo Ctrl+N), otevře se nové textové okno bez 3 šipek (>>>). Do tohoto okna nebudeme zadávat výrazy, ale příkazy. Po vytvoření programu neboli skriptu, je potřeba ho uložit (nejčastěji s příponou **.py**)a spustit (Run, nebo F5). Po spuštění se v původním okně **SHELL** nejprve celý Python restartuje a vyčistí paměť a po té vykoná příkazy programu.

Programy mohou být spuštěny nejen z prostředí **IDLE**, ale také dvojitým kliknutím na soubor v operačním systému. Pokud je Python správně nainstalován v operačním systému, dvojité kliknutí na příponu souboru **.py** otevře se nové okno, kde jsou provedeny příkazy a poté je okno okamžitě uzavřeno. Abychom takové okno okamžitě nezavřeli, ale počkalo na to, až stiskneme například Enter, je potřeba přidat do programu příkaz (funkci) **Print ()**, nebo **Input ().**

print ()

Jedná o volání speciální funkce. Tato funkce zapisuje hodnoty výrazů, které jsou uvedeny mezi závorkami. Hodnoty jsou ve výpisu odděleny mezerami.

Print () bez parametrů vloží pouze do aktuálního umístění prázdný řádek.

Text musí být uzavřen v apostrofech, nebo uvozovkách.

Jednotlivé hodnoty jsou odděleny čárkou.

input ()

Jedná o volání speciální funkce. Tato funkce nejprve zapíše zadaný řetězec znaků (pokud je zadán) a poté čeká na vstupní řetězec ukončený **Enter**, funkce po té vrací programu tento řetězec znaků, který jsme zadali.

Do input() se píše pouze text uzavřen v apostrofech, nebo uvozovkách.

Input() vždy vrací řetězec, pokud chceme číslo, musíme ho přetypovat.

help ()

Jedná se o funkci, která nám vypíše informace o jiných funkcích a hodnot.

Přetypování hodnot

Názvy **int**, **float** a **srt** zároveň slouží pro přetypování, a dokážou tak vytvořit z jednoho typu jiný.

• **int (hodnota) -** z dané hodnoty vytvoří celé číslo. Např*. int (3.14) => 3*, nebo *int ('37 ') => 37.*

• **float (hodnota)** z dané hodnoty vytvoří desetinné číslo. Např. *float (333) => 333,0*, nebo *float ('3.14') => 3,1*.

• **str (hodnota)** z dané hodnoty vytvoří znakoví řetězec. Např. *srt (356) => '356'*, nebo *srt (3.14) => '3,14'*.

Přetypování ze znakového řetězce na číslo bude fungovat, jen pokud je řetězec zadán v správném číselném formátu.

Některá z dohodnutých pravidel, jak správně zapisovat Pythonovský kód:

• Názvy proměnných obsahují pouze malá písmena.

• Pro znak **=** v přiřazovacím příkaze dáváme mezeru před i za.

• Operace v aritmetických výrazech jsou obvykle také odděleny od operandu mezerou.

• Řádky programu by neměly být delší než 79 znaků.

• Za čárky, které oddělují parametry v příkazu - např. *tisk ()*, vždy dáváme mezeru.

V Pythonu je definováno několik standardních funkcí, které pracují s čísly. Zde jsou dvě z nich:

abs (*číslo***)** - Funkce vrátí absolutní hodnotu zadaného čísla. Zadané číslo může být celé, nebo desetinné. Např. *abs (13) => 13*, nebo *abs (-3,14) => 3,14*.

round(*číslo***)** - Funkce vrátí zaokrouhlenou hodnotu zadaného čísla na celé číslo. Např. *round(3.14) => 3,* nebo *round(-0.74) => -1.*

round(*číslo*, *počet***) -** Funkce vrátí zaokrouhlenou hodnotu zadaného čísla na příslušný počet desetinných míst. Např. *round(3.14, 1) => 3.1,* nebo *round(2563, -2) => 2600.*

Číslo může být celé nebo desetinné, počet musí být celé číslo a vyjadřuje na kolik desetinných míst se bude zaokrouhlovat.

Speciální znaky formátování řetězce:

'\n' tento znak označuje při výpise funkcí *Print ()* kde se má přejít na nový řádek. To znamená, že kdykoliv použije ve v znakovém řetězci tento znak, při výpise se zde v tomto bodě odskočí na nový řádek.

```
a = 'prvý riadok\nstredný\ntretí riadok'
print(a) >>>        prvý riadok
                    stredný
                    tretí riadok
```

Python umožňuje zápis na více řádků i tím, že znakový řetězec začíná a končí třemi apostrofami (**'''**), nebo uvozovkami (**"""**). Při takovémto zápisu můžeme zapisovat na více řádku, kde Python následně automaticky nahradí všechny přechody na nový řádek znakem '**\n**'. Takovýto zápis může obsahovat i apostrofy a uvozovky.

```
a = '''prvý riadok
   stredný
   tretí riadok'''
```

Když chceme v řetězci spojit více různých hodnot a proměnných vložíme před začátek řetězce (apostrofy, nebo uvozovky) písmeno **'f'** a na proměnné a výpočty použijeme kudrnaté závorky **'{}'**. Python nejprve vyhodnotí obsah kudrnatých závorek a po té vypíše celý řetězec.

```
meno, x, y = 'A', 180, 225
r = f'bod {meno} na súradniciach ({x},{y})'
print(r)>>> bod A na súradniciach (180,225)
```

Další možností jak zapsat zápis obsahující číselné proměnné je pomocí formátovací šablony, kdy v znakovém řetězci na místo, kde má být číselná proměnná vložíme kudrnaté závorky a po ukončení znakového řetězce (apostrofy, nebo uvozovkami) napíšeme funkci **'.format'** a za to do závorek **'()'** napíšeme ve shodném pořadí jména proměnných.

```
meno, x, y = 'A', 180, 225
r = 'bod {} na súradniciach ({},{})'.format(meno, x, y)
print(r) >>> bod A na súradniciach (180,225)
```