 Python akademie - lekce 1 - 17.10.2024

▼

# 01\_01 Čísla

jako datové typy

---

## Zajímavé odkazy z této lekce:

- [Problém s plovoucí destinnou čárkou](#)
  - [Operator precedence determines the order](#)
  - [Doporučení pep8, formátování matematických operátorů s různou prioritou](#)
- 

## ▼ Úvod do datových typů

---

Pro naučení jakéhokoliv **programovacího jazyka**, si potřebuješ osvojit znalosti z tzv. tří teoretických pilířů. Na těchto pilířích stojí téměř všechny programovací jazyky:

1. **Datové typy** (čísla, sekvence, aj.)
2. **Syntaxe** (funkce, podmínky, smyčky, aj.)
3. **Knihovny** (decimal, aj.)

**Datové typy**, nebo také **datové struktury**. Začneme u něčeho, co je nám blízké a to jsou číselné hodnoty. Python je standardně vybavený dvěma číselnými datovými typy:

1. `int`, celá čísla (z angl. integer)
2. `float`, desetinná čísla (z angl. float)

## ✓ Celá čísla (integer)

---

```
print(100 + 200) # v interaktivním interpretu začneš psát za ">>>"
```

⇒ 300

V zápise je vhodné **psát mezery**, aby byl přehlednější. Je to pouze obecně platné doporučení. Výstup bude stejný, pokud je nepoužijeme:

```
print(100+200) # stejné hodnoty, jiný zápis bez mezer
```

⇒ 300

Všimni si symbolu `#`, který jsme použili v obou zápisech výše. Mřížka naznačuje, že jde o **jednořádkový komentář**. Python bude zápis za ní ignorovat, takže ji můžeš využít pro tvoje vlastní vysvětlivky.

Znaménko mezi čísly se často označuje jako tzv. **operátor** (čísla na jeho stranách potom jako **operandy**). Pokud na toto označení narazíš, tak ať ti není cizí.

## ✓ Aritmetické operace

---

Standartní operace, které známe z matematiky:

```
print(10 + 5)
```

 15

```
print(10 - 5)
```

 5

```
print(10 * 5)
```


 50

```
print(10 / 5)
```


 2.0

Práce s jednotlivými datovými typy může být záludná. Opatrně na výsledné typy:

```
print(10 / 3)
```

 3.3333333333333335

```
type(10 / 5)
```

 float

Méně známé operace dostupné v Pythonu:

| Operátor | Význam            |
|----------|-------------------|
| //       | celočíslné dělení |
| %        | modulo            |
| **       | umocňování        |

```
print(10 // 3)
```

 3

```
print(10 // 4)
```

 2


```
print(10 % 3)
```

 1

```
print(11 % 3)
```

 2

```
print(10 ** 3)
```

 1000

```
print(2 ** 3)
```

 8

Pokud budete časem potřebovat specializovanější matematické funkce, doporučuji vyzkoušet knihovnu `math`.

## ✓ Ověření datového typu


---

Za pomoci zabudované funkce `type` ověříme, jestli je číselná hodnota skutečně *integer*.

```
print(type(222))
```

 <class 'int'>

```
type(222)
```

 int

V obou případech jsme dostali zpátky výstup interpreta, jehož formát nám zatím nemusí být jasný. Zásadní je část toho výstupu, kde stojí `int` (tedy integer). Takže jde skutečně o celé číslo a Python to ví.

## ✓ Desetinná čísla (floats)

---

**Pozor!** Desetinným oddělovačem je **tečka**. Čárka slouží k jiným účelům.

```
print(0.1)
```

⇒ 0.1

```
print(0.1 + 0.3)
```

⇒ 0.4

```
type(0.3)
```

⇒ float

Občas se při práci s **desetinnými čísly** setkáš s fenoménem známým jako **plovoucí řádová čárka**.

Ten je způsobený tím, že některá desetinná čísla nemají odpovídající **binární tvar**. Proto jsou použity přibližné hodnoty.

```
print(0.1 + 0.2)
```

⇒ 0.30000000000000004

Nemusíš se ničeho obávat, není to chyba na tvé straně, ale obecný fakt. Pokud budeš do budoucna potřebovat práci s **přesnými desetinnými čísly**, doporučujeme pracovat s knihovnou `decimal` (o práci s knihovnami se budeme bavit v pozdějších lekcích)

► ⚡ Pokročilé, zde pouze na ukázkou

## ✓ Komplikace s čísly

---

```
print(2 + 3 * 2)
```

⇒ 8

```
print((2 + 3) * 2)
```

⇒ 10

Různé operátory mají různé priority. Pokud budete používat různé operátory, doporučuji odlišovat mezi jejich důležitostí:

```
print(2 + 3*2)
```

 8

## Hierarchie matematických operátorů

| Pořadí | Operátor | Proces     |
|--------|----------|------------|
| 1.     | ()       | závorky    |
| 2.     | **       | umocňování |
| 3.     | *        | násobení   |
| 4.     | /        | dělení     |
| 5.     | +        | sčítání    |
| 6.     | -        | odčítání   |