

Operátory

Poznámky k téme

Operátory sú znaky, ktorými zapisujeme výpočty a porovnania: `+`, `*`, `//`, `%`, `==`, `and` ... Táto téma prechádza všetky operátory, ktoré budeš bežne používať, a vysvetľuje, v akom poradí sa vyhodnocujú. Najviac pozornosti venuje operátorom `//` a `%`, ktoré sa v úlohách objavujú neustále – pri párnosti, cifrách čísla aj prevodoch času.

Čo sa naučíš

- počítat s operátormi `+`, `-`, `*`, `/`, `//`, `%`, `**` a poznať ich prednosť,
- používať celočíselné delenie a zvyšok (cifry, párnosť, prevody jednotiek),
- porovnávať hodnoty a skladať podmienky pomocou `and`, `or`, `not`,
- skracovať zápis pomocou `+=`, `-=` a podobných.

1 Aritmetické operátory

Operátor	Význam	Príklad	Výsledok
<code>+</code>	sčítanie	<code>7 + 2</code>	9
<code>-</code>	odčítanie	<code>7 - 2</code>	5
<code>*</code>	násobenie	<code>7 * 2</code>	14
<code>/</code>	delenie (vždy desatinné číslo)	<code>7 / 2</code>	3.5
<code>//</code>	celočíselné delenie	<code>7 // 2</code>	3
<code>%</code>	zvyšok po delení	<code>7 % 2</code>	1
<code>**</code>	mocnina	<code>7 ** 2</code>	49

```
>>> 7 / 2
3.5
>>> 8 / 2
4.0
>>> 7 // 2
3
>>> 7 % 2
1
>>> 2 ** 10
1024
>>> 9 ** 0.5
3.0
```

`9 ** 0.5` je druhá odmocnina (mocnina na 1/2).

1.1 Prednosť operátorov

Ak je vo výraze viac operátorov, Python ich vyhodnocuje v tomto poradí (od najvyššej priority):

1. zátvorky `( )`,
2. mocnina `**`,

3. znamienko mínus pred číslom (`-x`),
4. `*`, `/`, `//`, `%` – zľava doprava,
5. `+`, `-` – zľava doprava,
6. porovnania (`==`, `<`, ...),
7. `not`, potom `and`, nakoniec `or`.

```
>>> 2 + 3 * 4
14
>>> (2 + 3) * 4
20
>>> 10 - 4 - 3
3
>>> 2 * 3 ** 2
18
>>> -2 ** 2
-4
>>> (-2) ** 2
4
```

Pozor na `-2 ** 2`: mocnina má prednosť pred mínusom, preto sa počíta ako `-(2 ** 2)`. Ak poradie nevieš naisto, použi zátvorky – nikdy neuškodí a výraz sa lepšie číta.

2 Celočíselné delenie a zvyšok

Predstav si, že 17 jablák rozdeľuješ rovnakým dielom medzi 5 detí. Každé dostane `17 // 5 = 3` jablká a `17 % 5 = 2` jablká ostanú. Platí:

`17 = 5 · (17 // 5) + 17 % 5`, teda `17 = 5 · 3 + 2`.

```
>>> 17 // 5
3
>>> 17 % 5
2
>>> divmod(17, 5)
(3, 2)
```

Funkcia `divmod` vráti oba výsledky naraz.

2.1 Na čo sa `%` a `//` používajú

Párnosť a deliteľnosť. Číslo `n` je deliteľné číslom `d`, keď je zvyšok nula: `n % d == 0`. Párne čísla majú zvyšok 0 po delení dvoma.

```
>>> 14 % 2 == 0
True
>>> 15 % 2 == 0
False
>>> 36 % 9 == 0
True
```

Cifry čísla. Posledná cifra je zvyšok po delení 10; číslo bez poslednej cifry je celočíselný podiel 10:

```
>>> n = 4728
>>> n % 10
8
>>> n // 10
472
>>> n // 10 % 10
2
>>> n // 100 % 10
7
>>> n // 1000
4
```

Prevody jednotiek. Koľko hodín, minút a sekúnd je 7384 sekúnd?

```
sekundy = 7384
hodiny = sekundy // 3600
zvysok = sekundy % 3600
minuty = zvysok // 60
sek = zvysok % 60
print(hodiny, "h", minuty, "min", sek, "s")
```

Výstup:

```
2 h 3 min 4 s
```

Cyklické opakovanie. Zvyšok sa „točí dookola“: dni v týždni, hodiny na ciferníku. Ak je dnes streda (deň č. 2, pri pondelku č. 0), o 100 dní bude deň $(2 + 100) \% 7$:

```
>>> (2 + 100) % 7
4
>>> (22 + 5) % 24
3
```

O 100 dní bude piatok (deň č. 4) a 5 hodín po 22:00 budú 3:00.

2.2 Záporné čísla

Pri záporných číslach Python zaokrúhľuje celočíselný podiel **nadol** (smerom k mínus nekonečnu) a zvyšok má rovnaké znamienko ako deliteľ:

```
>>> -7 // 2
-4
>>> -7 % 2
1
>>> 7 // -2
-4
```

Vzťah $a = b \cdot (a // b) + a \% b$ platí stále: $-7 = 2 \cdot (-4) + 1$. V bežných úlohách s kladnými číslami sa s tým nestretnáš, ale je dobré to vedieť.

3 Porovnávanie

Porovnanie vráti logickú hodnotu `True` alebo `False`.

Operátor	Význam
<code>==</code>	rovná sa
<code>!=</code>	nerovná sa
<code><</code> , <code>></code>	menšie, väčšie
<code><=</code> , <code>>=</code>	menšie alebo rovné, väčšie alebo rovné

```
>>> 5 == 5
True
>>> 5 != 5
False
>>> 3 < 8
True
>>> "a" == "A"
False
>>> x = 7
>>> 1 <= x <= 10
True
```

Python dovoľuje porovnania **reťazit'**: `1 <= x <= 10` znamená „x je od 1 do 10“, presne ako v matematike.

3.1 Porovnávanie textov

Reťazce sa porovnávajú znak po znaku podľa ich kódov (podobne ako v slovníku). Veľké písmená majú menšie kódy ako malé, takže `"Zebra" < "auto"`:

```
>>> "auto" < "bicykel"
True
>>> "Zebra" < "auto"
True
>>> "10" < "9"
True
```

Posledný príklad: ako **texty** sa porovnáva najprv `"1"` s `"9"`, preto je `"10"` „menšie“. Čísla zo vstupu preto vždy preved' na `int` skôr, než ich porovnáš.

3.2 Porovnávanie desatinných čísel

Pre nepresnosť desatinných čísel (téma Premenné a dátové typy) nepoužívaj `==` na výsledky výpočtov s desatinnými číslami:

```
>>> 0.1 + 0.2 == 0.3
False
>>> abs((0.1 + 0.2) - 0.3) < 1e-9
True
>>> round(0.1 + 0.2, 10) == 0.3
True
```

4 Logické operátory

Logické operátory spájajú podmienky:

- `a and b` – pravda, iba ak platí **obe** časti,
- `a or b` – pravda, ak platí **aspoň jedna** časť,
- `not a` – opak.

a	b	a and b	a or b	not a
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

```
>>> vek = 15
>>> vek >= 6 and vek <= 18
True
>>> vek < 6 or vek > 65
False
>>> not vek > 18
True
```

Poradie: najprv `not`, potom `and`, nakoniec `or`. Výraz `a or b and c` znamená `a or (b and c)`. Pri zložitejších podmienkach používaj zátvorky.

4.1 Skrátené vyhodnocovanie

Python vyhodnocuje `and` a `or` zľava a skončí, keď je výsledok jasný. Pri `and` sa druhá časť nevyhodnotí, ak je prvá nepravdivá. To sa dá využiť na ochranu pred chybou:

```
>>> x = 0
>>> x != 0 and 10 / x > 2
False
```

Keďže `x != 0` je `False`, delenie `10 / x` sa vôbec nevykoná a nenastane delenie nulou.

4.2 Častá chyba: `x == 1 or 2`

Podmienku „x je 1 alebo 2“ nemožno zapísať ako `x == 1 or 2`. Python ju vyhodnotí ako `(x == 1) or 2` a číslo 2 sa v logickom výraze správa ako pravda:

```
>>> x = 5
>>> x == 1 or 2
2
>>> x == 1 or x == 2
False
>>> x in (1, 2)
False
```

Správne je porovnať `x` s každou hodnotou zvlášť, alebo použiť `in` (téma N-tice).

5 Skrátené priradenie

Zápis `x = x + 5` sa dá skrátiť na `x += 5`. Podobne fungujú aj ostatné operátory:

Skrátene	Znamená
<code>x += 3</code>	<code>x = x + 3</code>
<code>x -= 3</code>	<code>x = x - 3</code>
<code>x *= 3</code>	<code>x = x * 3</code>
<code>x /= 3</code>	<code>x = x / 3</code>
<code>x //= 3</code>	<code>x = x // 3</code>
<code>x %= 3</code>	<code>x = x % 3</code>
<code>x **= 3</code>	<code>x = x ** 3</code>

```
skore = 100
skore += 25      # bonus
skore -= 40      # trest
skore *= 2       # dvojnásobok
print(skore)
```

Výstup:

```
170
```

Skrátené priradenie sa najčastejšie používa v cykloch na počítadlá (`pocet += 1`) a súčty (`sucet += x`).

6 Riešené úlohy

6.1 Priestupný rok

Zadanie: Rok je priestupný, ak je deliteľný 4, ale nie 100 – s výnimkou rokov deliteľných 400, ktoré priestupné sú. Načítaj rok a vypíš `True` alebo `False`.

Riešenie: Pravidlo prepíšeme priamo do logického výrazu. Rok 2024 je priestupný (deliteľný 4, nie 100), rok 1900 nie (deliteľný 100, nie 400), rok 2000 áno (deliteľný 400).

```
rok = int(input("Rok: "))
priestupny = (rok % 4 == 0 and rok % 100 != 0) or rok % 400 == 0
print(priestupny)
```

Výstup:

```
Rok: 2024
True
```

Zátvorky tu nie sú nutné (`and` má prednosť pred `or`), ale výraz s nimi lepšie čítame.

6.2 Ciferný súčet trojciferného čísla

Zadanie: Načítaj trojciferné číslo a vypíš súčet jeho cifier.

Riešenie: Stovky sú `n // 100`, desiatky `n // 10 % 10`, jednotky `n % 10`. Pre ľubovoľne dlhé čísla sa ciferný súčet počíta cyklom (téma Cykly).

```
n = int(input("Trojciferné číslo: "))
stovky = n // 100
desiatky = n // 10 % 10
jednotky = n % 10
print("Ciferný súčet:", stovky + desiatky + jednotky)
```

Výstup:

```
Trojciferné číslo: 472
Ciferný súčet: 13
```

6.3 Vydávanie mincí

Zadanie: Automat vydáva drobné mincami 2 €, 1 €, 50 centov, 20 centov, 10 centov, 5 centov, 2 centy a 1 cent. Pre sumu v centoch vypíš, koľko kusov ktorej mince vydá, ak vydáva čo najväčšie mince.

Riešenie: Celočíselné delenie povie, koľko mincí danej hodnoty sa vojde; zvyšok sa rozmení menšími mincami. Pre prehľadnosť ukážeme prvé štyri mince – ostatné by sa riešili rovnako (s cyklom a zoznamom by bol program kratší).

```
suma = int(input("Suma v centoch: "))
print("2 €:", suma // 200)
suma = suma % 200
print("1 €:", suma // 100)
suma = suma % 100
print("50 c:", suma // 50)
suma = suma % 50
print("20 c:", suma // 20)
suma = suma % 20
print("zvyšok v centoch:", suma)
```

Výstup:

```
Suma v centoch: 387
2 €: 1
1 €: 1
50 c: 1
20 c: 1
zvyšok v centoch: 17
```

6.4 Je bod v obdĺžniku?

Zadanie: Obdĺžnik má ľavý dolný roh v bode [0, 0] a pravý horný v bode [8, 5]. Načítaj súradnice bodu a vypíš, či leží vnútri obdĺžnika alebo na jeho hranici.

```
x = float(input("x: "))
y = float(input("y: "))
vnutri = 0 <= x <= 8 and 0 <= y <= 5
print(vnutri)
```

Výstup:

```
x: 3
y: 6
False
```

Zhrnutie

- `/` dáva desatinné číslo, `//` celočíselný podiel, `%` zvyšok, `**` mocninu.
- Prednosť: zátvorky, `**`, mínus pred číslom, `*` / `//` `%`, `+` `-`, porovnania, `not`, `and`, `or`.
- `n % d == 0` – deliteľnosť; `n % 10` – posledná cifra; `n // 10` – číslo bez poslednej cifry.
- Porovnania vracajú `True/False`, dajú sa reťaziť (`1 <= x <= 10`).
- `and` – obe, `or` – aspoň jedna, `not` – opak; „x je 1 alebo 2“ je `x == 1 or x == 2`.
- `x += 1` je skratka pre `x = x + 1`.

Úlohy na precvičenie

1. Načítaj počet sekúnd a vypíš ho v tvare `H:MM:SS` (minúty a sekundy vždy dvojmiestne – pomôže formát `:02`).
2. Načítaj celé číslo a vypíš, či je deliteľné 3 aj 5 súčasne.
3. Načítaj štvorciferné číslo a vypíš ho odzadu (napr. 4728 → 8274) iba pomocou `//` a `%`.
4. Načítaj vek a vypíš `True`, ak človek platí detské vstupné (do 15 rokov) alebo seniorské (od 62 rokov).
5. Vypočítaj bez spustenia: `3 + 4 * 2 ** 2 // 3`, `not 3 > 2 or 1 > 2`, `-9 // 4`. Potom over výsledky v Pythone.

Úlohy k tejto téme nájdeš na learnwithmente.com v časti Informatika → Python.