

Premenné a dátové typy

Poznámky k téme

Programy pracujú s údajmi – s vekom, cenou, menom, výsledkom výpočtu. Aby sme si údaj mohli zapamätať a neskôr ho použiť, uložíme ho do **premennej**. Každý údaj má zároveň **typ**: celé číslo, desatinné číslo, text alebo logická hodnota. Typ určuje, čo sa s údajom dá robiť – čísla sa dajú sčítať, text sa dá spájať.

Čo sa naučíš

- vytvoriť premennú, zmeniť jej hodnotu a správne ju pomenovať,
- rozlíšiť typy `int`, `float`, `str` a `bool`,
- zistiť typ hodnoty funkciou `type`,
- prevádzať hodnoty medzi typmi (`int`, `float`, `str`) a vyhnúť sa typickým chybám.

1 Premenné a priradenie

Premenná je meno, pod ktorým si program pamätá hodnotu. Hodnotu do premennej uložíme **priradením**:

```
vek = 16
meno = "Ema"
print(meno, "má", vek, "rokov.")
```

Výstup:

```
Ema má 16 rokov.
```

Znak `=` v programovaní **neznamená rovnosť** ako v matematike. Je to príkaz: *vypočítaj, čo je vpravo, a ulož to pod meno, ktoré je vľavo*. Preto je úplne správny aj zápis, ktorý by v matematike nedával zmysel:

```
body = 10
body = body + 5      # vezmi starú hodnotu, pripočítaj 5, ulož späť
print(body)
```

Výstup:

```
15
```

Pri novom priradení sa stará hodnota **prepíše** – premenná si pamätá vždy iba poslednú hodnotu.

1.1 Trasovanie: ako sa menia hodnoty

Keď chceš pochopiť, čo program robí, prejdí ho riadok po riadku a zapisuj si hodnoty premenných do tabuľky. Tomuto postupu sa hovorí **trasovanie**.

```
a = 3
b = a * 2
a = a + b
b = b - 1
print(a, b)
```

Výstup:

```
9 5
```

po riadku	a	b	čo sa stalo
1	3	–	a dostane hodnotu 3
2	3	6	b = 3 · 2
3	9	6	a = 3 + 6 – stará hodnota a sa prepíše
4	9	5	b = 6 – 1

1.2 Pravidlá pre mená premenných

- Meno sa skladá z písmen, číslíc a podčiarkovníka `_`.
- **Nesmie začínať číslicom** (2pokus je chyba, pokus2 je v poriadku) a nesmie obsahovať medzeru.
- Python rozlišuje **veľké a malé písmená**: `vek`, `Vek` a `VEK` sú tri rôzne premenné.
- Nesmie to byť **klúčové slovo** jazyka, napríklad `if`, `for`, `while`, `def`, `True`, `class`, `in`.

Odporúčania, vďaka ktorým sa program dobre číta:

- voľ **výstižné mená**: `cena_spolu` povie viac než `x`;
- píš malými písmenami a slová oddeľuj podčiarkovníkom: `pocet_ziakov` (tzv. *snake_case*);
- vyhni sa diakritike v menách (`dĺzka`, nie `dĺžka`) – Python ju síce povolí, ale na inej klávesnici sa ťažko píše.

1.3 Viac priradení naraz

```
x = y = 0          # obe premenné dostanú 0
a, b = 5, 8        # a dostane 5, b dostane 8
print(x, y, a, b)
```

Výstup:

```
0 0 5 8
```

Druhý zápis (*rozbaľovanie*) sa podrobnejšie preberá v téme N-tice.

1.4 Premenná musí existovať skôr, než ju použiješ

```
print(skore)
skore = 100
```

Výstup (program skončí chybou):

```
Traceback (most recent call last):
  File "program.py", line 1, in <module>
    print(skore)
    ^^^^^
NameError: name 'skore' is not defined
```

Python vykonáva príkazy zhora nadol, takže na prvom riadku premenná `skore` ešte neexistuje.

2 Čísla: int a float

Python rozlišuje dva základné číselné typy:

Typ	Význam	Príklady
<code>int</code> (<i>integer</i>)	celé číslo	0, 42, -7, 1000000
<code>float</code> (<i>floating point</i>)	desatinné číslo	3.14, -0.5, 2.0, 1e6 (= 1 000 000.0)

Celé čísla môžu byť v Pythone ľubovoľne veľké:

```
>>> 2 ** 100
1267650600228229401496703205376
>>> 7 / 2
3.5
>>> 6 / 3
2.0
>>> 7 // 2
3
>>> 2 + 1.5
3.5
```

Dôležité pravidlá:

- Operátor `/` (delenie) dáva **vždy** `float`, aj keď vyjde celé číslo: `6 / 3` je `2.0`.
- Celočíselné delenie `//` dáva pri celých číslach `int` (`7 // 2` je `3`).
- Ak sa vo výpočte stretne `int` s `float`, výsledok je `float`.

Podrobný prehľad operátorov nájdeš v téme Operátory.

2.1 Desatinné čísla nie sú vždy presné

Počítač ukladá desatinné čísla v dvojkovej sústave, a tak niektoré čísla (napríklad 0,1) nevie uložiť úplne presne. Prejaví sa to drobnými odchýlkami:

```
>>> 0.1 + 0.2
0.30000000000000004
>>> round(0.1 + 0.2, 2)
0.3
>>> 0.1 + 0.2 == 0.3
False
```

Pri výpise preto výsledok **zaokrúhli** (`round(x, 2)`) a desatinné čísla nikdy neporovnávajú presne cez `==` – porovnávajú zaokrúhlené hodnoty alebo či je rozdiel veľmi malý.

2.2 Užitočné funkcie pre čísla

```
>>> abs(-8)
8
>>> round(3.14159, 2)
3.14
>>> round(2.7)
3
>>> max(4, 9, 2)
9
>>> min(4, 9, 2)
2
```

3 Text: typ str

Text sa v Pythone volá **reťazec** (typ `str`, z anglického *string*). Píše sa do úvodzoviek. Aj prázdny text `""` je reťazec – iba nemá žiadne znaky.

```
>>> "Ahoj" + " " + "Ema"
'Ahoj Ema'
>>> "ha" * 3
'hahaha'
>>> len("Python")
6
>>> len("")
0
```

- `+` dva reťazce **spojí**,
- `*` reťazec **zopakuje** (`"ha" * 3` je `"hahaha"`),
- `len` vráti **počet znakov** (aj medzery a interpunkcia sú znaky).

3.1 Číslo verzus text

5 a `"5"` sú dve úplne odlišné hodnoty – prvá je číslo, druhá je text s jedným znakom:

```
print(5 + 5)
print("5" + "5")
print("5" * 3)
print(5 * 3)
```

Výstup:

```
10
55
555
15
```

S reťazcami sa toho dá robiť oveľa viac (vyberať znaky, meniť veľkosť písmen, deliť text na slová) – tomu sa venuje téma Reťazce.

4 Logické hodnoty: bool

Typ `bool` má iba dve hodnoty: `True` (pravda) a `False` (nepravda). Píšu sa s **veľkým začiatočným písmenom** a bez úvodzoviek. Najčastejšie vznikajú ako výsledok porovnania:

```
>>> 5 > 3
True
>>> 10 == 9
False
>>> vek = 16
>>> vek >= 18
False
```

Logické hodnoty sa používajú v podmienkach (`if`) a cykloch (`while`) – rozhodujú, či sa niečo vykoná.

5 Zistenie typu: type()

Funkcia `type` prezradí, akého typu je hodnota:

```
>>> type(42)
<class 'int'>
>>> type(3.0)
<class 'float'>
>>> type("42")
<class 'str'>
>>> type(True)
<class 'bool'>
>>> type(2 + 0.5)
<class 'float'>
```

`3.0` je `float`, hoci je jeho hodnota celá – rozhoduje desatinná bodka. `"42"` je `str`, lebo je v úvodzovkách.

6 Prevod typov

Často potrebujeme hodnotu jedného typu zmeniť na iný. Slúžia na to funkcie s menami typov:

Funkcia	Čo urobí	Príklad	Výsledok
<code>int(x)</code>	prevedie na celé číslo	<code>int("42")</code> , <code>int(3.99)</code>	<code>42</code> , <code>3</code>
<code>float(x)</code>	prevedie na desatinné číslo	<code>float("2.5")</code> , <code>float(4)</code>	<code>2.5</code> , <code>4.0</code>
<code>str(x)</code>	prevedie na text	<code>str(16)</code>	<code>"16"</code>
<code>bool(x)</code>	prevedie na logickú hodnotu	<code>bool(0)</code> , <code>bool(7)</code>	<code>False</code> , <code>True</code>

```
>>> int("42") + 8
50
>>> int(3.99)
3
>>> round(3.99)
4
>>> float("2.5") * 2
5.0
>>> str(16) + " rokov"
'16 rokov'
>>> int(" 7 ")
7
```

Všimni si:

- `int(3.99)` je 3 – `int` desatinnú časť **odreže**, nezaokrúhľuje. Na zaokrúhlenie slúži `round`.
- `int` si poradí s medzerami na okrajoch textu (" 7 "), ale nie s textom, ktorý nie je celé číslo:

```
>>> int("3.5")
ValueError: invalid literal for int() with base 10: '3.5'
>>> int("dva")
ValueError: invalid literal for int() with base 10: 'dva'
>>> int(float("3.5"))
3
```

Prevod je dôležitý najmä pri **vstupe**: funkcia `input` vždy vráti text, aj keď niekto napíše číslo. Preto sa číselný vstup číta ako `int(input())` – podrobne v téme Vstup a výstup.

6.1 Text a číslo v jednom výpise

Text s číslom sa cez `+` spojiť nedá (`TypeError`). Máš tri možnosti:

```
vek = 16
print("Mám " + str(vek) + " rokov.") # číslo prevedieme na text
print("Mám", vek, "rokov.")          # čiarky v print
print(f"Mám {vek} rokov.")           # f-reťazec
```

Výstup:

```
Mám 16 rokov.
Mám 16 rokov.
Mám 16 rokov.
```

Tretí spôsob, **f-reťazec**, je najpohodlnejší: pred úvodzovky napíšeš `f` a do zložených zátvoriek `{ }` vložíš premennú alebo výraz.

7 Časté chyby

Desatinná čiarka namiesto bodky. Python čiarku chápe ako oddelenie dvoch hodnôt – nevznikne chyba, ale niečo úplne iné, než sme chceli:

```
cena = 3,5
print(cena)
print(type(cena))
```

Výstup:

```
(3, 5)
<class 'tuple'>
```

Vznikla takzvaná n-tica dvoch čísel (téma N-tice). Správne je `cena = 3.5`.

Spájanie textu s číslom:

```
vek = 16
print("Mám " + vek + " rokov.")
```

Výstup (program skončí chybou):

```
Traceback (most recent call last):
  File "program.py", line 2, in <module>
    print("Mám " + vek + " rokov.")
    ~~~~~^~~~~
TypeError: can only concatenate str (not "int") to str
```

Meno so zlou veľkosťou písmen:

```
Meno = "Ema"
print(meno)
```

Výstup (program skončí chybou):

```
Traceback (most recent call last):
  File "program.py", line 2, in <module>
    print(meno)
    ^^^
NameError: name 'meno' is not defined
```

`=` **namiesto** `==`. Jedno `=` je priradenie, dve `==` sú porovnanie. Vo výrazoch, ktoré majú dať `True` alebo `False`, musí byť `==`.

8 Riešené úlohy

8.1 Výmena hodnôt dvoch premenných

Zadanie: V premennej `a` je 5, v `b` je 8. Vymeň ich hodnoty.

Riešenie: Zápis `a = b` a potom `b = a` nefunguje – po prvom priradení je pôvodná hodnota `a` stratená. Potrebujeme pomocnú premennú, do ktorej si ju odložíme:

```
a = 5
b = 8
pomocna = a    # odložíme si 5
a = b          # a = 8
b = pomocna    # b = 5
print(a, b)
```

Výstup:

```
8 5
```

V Pythone sa to dá zapísať aj jedným riadkom `a, b = b, a` (téma N-tice).

8.2 Nákup s DPH

Zadanie: Tovar stojí bez DPH 24,90 €. Vypočítaj DPH 23 % a cenu s DPH, zaokrúhlenú na centy.

Riešenie: DPH je 23 % z ceny, teda `cena * 0.23`. Cenu s DPH dostaneme sčítaním. Na centy zaokrúhlime funkciou `round(..., 2)`.

```
cena_bez_dph = 24.90
dph = cena_bez_dph * 0.23
cena_s_dph = cena_bez_dph + dph
print("DPH:", round(dph, 2), "€")
print("Cena s DPH:", round(cena_s_dph, 2), "€")
```

Výstup:

```
DPH: 5.73 €
Cena s DPH: 30.63 €
```

8.3 Index telesnej hmotnosti (BMI)

Zadanie: BMI sa počíta ako hmotnosť v kilogramoch delená druhou mocninou výšky v metroch. Vypočítaj BMI pre hmotnosť 62 kg a výšku 172 cm a vypíš ho na jedno desatinné miesto.

Riešenie: Výšku musíme previesť na metre (`172 / 100`). Druhú mocninu zapíšeme operátorom `**`.

```
hmotnost = 62
vyska_cm = 172
vyska_m = vyska_cm / 100
bmi = hmotnost / vyska_m ** 2
print("BMI:", round(bmi, 1))
```

Výstup:

```
BMI: 21.0
```

Mocnina má prednosť pred delením, takže `hmotnost / vyska_m ** 2` sa vypočíta ako `hmotnost / (vyska_m ** 2)`.

8.4 Kruh

Zadanie: Pre polomer 2,5 cm vypočítaj obvod a obsah kruhu. Použi $\pi \approx 3,14159$.

```
PI = 3.14159
r = 2.5
obvod = 2 * PI * r
obsah = PI * r ** 2
print("Obvod:", round(obvod, 2), "cm")
print("Obsah:", round(obsah, 2), "cm²")
```

Výstup:

```
Obvod: 15.71 cm
Obsah: 19.63 cm²
```

Meno `PI` je napísané veľkými písmenami – tak sa zvyknú označovať hodnoty, ktoré sa v programe nemenia (*konštanty*). Presnejšie π poskytuje modul `math` (téma Modulárny program).

Zhrnutie

- `meno = hodnota` uloží hodnotu do premennej; nové priradenie starú hodnotu prepíše.
- `x = x + 1` znamená „zvýš `x` o 1“, nie rovnicu.
- Mená: písmená, číslice, `_`, nezačínajú číslicou, rozlišujú veľkosť písmen.
- Typy: `int` (celé), `float` (desatinné, s bodkou), `str` (text v úvodzovkách), `bool` (`True/False`).
- `/` dáva vždy `float`; desatinné čísla pri výpise zaokrúhľuj (`round`).
- `type(x)` zistí typ; `int()`, `float()`, `str()` prevádzajú medzi typmi.
- `int(3.99)` je 3 (odreže); `int("3.5")` je chyba `ValueError`.
- Text s číslom spájaj cez `str()`, čiarky v `print` alebo f-reťazec.

Úlohy na precvičenie

1. Ulož do premenných dĺžku a šírku obdĺžnika a vypíš jeho obsah a obvod.
2. Vytrasuj (na papieri) program `x = 2`, `y = x + 3`, `x = x * y`, `y = y - x` a potom over výsledok spustením.
3. Prepočítaj 75 minút na hodiny ako desatinné číslo a výsledok zaokrúhli na 2 miesta.
4. Zisti typ výsledku výrazov `7 / 7`, `7 // 7`, `"7" * 2` a `3 > 2`.
5. Prepíš výpis `print("Výsledok je " + vysledok)` tak, aby fungoval, aj keď je `vysledok` číslo.

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