Datové typy

Co to je?

Datový typ

Datový typ je doslova typ (forma, podoba, ...) informace uložené v paměti počítače.

- Narozdíl od pseudokódu, v programovacích jazycích musíte kromě názvu proměnné uvádět i její typ.
- Základní typy v Pythonu jsou `int`, `float`, `str`, `set`, `list`, `tuple`, `dict`

Měnné vs. neměnné

- Python rozlišuje mezi měnnými (mutable) a neměnnými (immutable) datovými typy.
- Do struktury měnných typů (seznamy, slovníky, ...) můžete zasahovat během programu, ale do struktury neměnných (čísla, slova, ...) nikoliv.

Datové typy

Číselné typy

Celá čísla

Datový typ `int`

Zkratkou `int` (z angl. integer) Python označuje typ celých čísel, tj. čísel bez desetinné části.

Celá čísla

Python umí následující operace na celých číslech.

- součet (+);
- rozdíl (−);
- součin (∗);
- celočíselný podíl (//), např. `11 // 3 == 3`;
- zbytek po dělení (%), např. `11 % 3 == 2`;
- mocninu (∗∗), např. `4 ** 3 == 64`.

Desetinná čísla

Datový typ `float`

Zkratka `float` (z angl. floating point) označuje v Pythonu typ desetinných čísel.

Poznámka. Celá čísla jsou samozřejmě zároveň desetinná. Aby je Python v tomto případě rozlišil, píše `2.0` pro „desetinné číslo“ dva a `2` pro „celé číslo“ dva.

Desetinná čísla

Python umí následující operace na desetinných číslech.

- součet (+);
- rozdíl (−);
- součin (*);
- podíl (/);
- mocninu (**).

celá ↔ desetinná

- Slova `int` a `float` jsou zároveň názvy funkcí/procedur v Pythonu pro převod mezi číselnými typy.
- `int(x: float)` vrátí tzv. „celou část“ z `x`; např. `int(3.9) == 3`.
- `float(x: int)` převede celé číslo `x` na desetinné prostě tak, že k němu přidá „`.0`“. Takže třeba `float(3) == 3.0`.

Datové typy

Řetězce

Řetězce (stringy)

Datový typ `str`

Zkratkou `str` (z angl. string) Python označuje typ „řetězců znaků“, tj. posloupností v zásadě libovolných symbolů.

- Stringy se píší do uvozovek, buď jednoduchých (`'`) nebo dvojitých (`"`). Na výběru nezáleží, ale string musí začínat končit stejnou uvozovkou.
- Python používá pro kódování textu UTF-8 (**U**nicode **T**ransformation **F**ormat – 8-bit). Tedy umí rozpoznat každý znak v tomto kódování.

Řetězce (stringy)

Python umí následující operace na řetězcích.

- součet/spojení (+ nebo mezera)
  - např. `"auto" + "bus" == "autobus"`
  - např. `"mrt" "vola" == "mrtvola"`
- součin/opakování (*): např. `"hehe" * 3 == "hehehehehehe"`
- výběr prvku (`str[pořadí prvku]`): např. `"python"[2] == "t"`.
  Pozor! Python čísluje od 0.

stringy ↔ čísla

- Zkratka `str` je zároveň procedura na převod dané proměnné na string. V případě čísel máme
  - `str(x: int)` převede celé číslo na string. Třeba `str(3) == "3"`.
  - `str(x: float)` převede desetinné číslo na string. Např. `str(3.14159) == "3.14159"`.

  Pozor! Python neřeší, jestli je ve stringu číslo. Takže třeba `"1" + "1" == "11"`, ale `1 + 1 == 2`.

stringy ↔ čísla

Procedury `int` a `float` taky převádějí stringy na číslo, pokud to lze. Např.

- `int("69") == 69`,
- `float("3.14159") == 3.14159`, ale
- `float("hehe")` i `int("9.11")` hodí chybu.

Datové typy

Seznamy

Seznamy

Datový tip `list`

Slovem `list` označuje Python seznam; vlastně množinu, kde každý prvek má jednoznačné pořadí. **Prvky v seznamu mohu nahrazovat, přidávat a odebírat**.

- Seznamy se píší do hranatých závorek `[]` a prvky oddělují čárkami. Třeba `[2, "hora", 4, 7]` je seznam se čtyřmi prvky.
- Prvkem seznamu může být další seznam. Třeba `[1, [2, "tři"], 4]` je seznam, jehož druhým prvkem je seznam `[2, "tři"]`.

Seznamy

Python umí následující operace na seznamech.

- součet/spojení (+)
  - např. `[69, 420] + [911, 1337] == [69, 420, 911, 1337]`.
- součin/opakování (*)
  - např. `[1, 2] * 4 == [1, 2, 1, 2, 1, 2, 1, 2]`.
- výběr prvku (`list[pořadí prvku]`)
  - např. `[1, 4, 7, "hroch"][2] == 7`.
  - např. `[1, 4, 7, "hroch"][-1] == "hroch"`.

  Pozor! Python čísluje buď od 0 nahoru (začátek → konec) nebo od −1 dolu (konec → začátek)

Datové typy

N-tice

N-tice

Datový typ `tuple`

Slovem `tuple` Python označuje n-tici, neboli posloupnost n prvků. **Prvky n-tice nemohu nahrazovat, přidávat ani odebírat**.

- N-tice se píší buď kulatou závorkou s prvky oddělenými čárkou, třeba `(1, 2)`, nebo v mnoha případech i bez závorky, třeba `1, 2`.
- Python umí stejné operace na n-ticích jako na seznamech.

Datové typy

Slovníky

Slovníky

Datový typ `dict`

Zkratkou `dict` (z angl. dictionary) označuje Python typ slovníku, tj. množiny hodnot, které jsou zařazeny pod klíči. **Slovník umožňuje nahrazování, přidávání i odebírání klíčů i hodnot**.

- Slovník se píše do složených závorek `{}` a prvky jsou v podobě `klíč: hodnota` odděleny čárkami. Např. `{(1, 2): "kočka", 3: [4, 5], "pes": 6}`.
- Hodnotou může být cokoli, ale klíč musí být neměnný datový typ (číslo, slovo, n-tice apod.).

Slovníky

Slovníky nelze sčítat/spojovat ani násobit/opakovat. Jedinou základní operací je výběr prvku příkazem `dict[klíč]`. Pár příkladů:

- `{"kočka": 2, "pes": 3}["pes"] == 3`.
- `{(1, 2, 3): "ted'", 4: 5}[(1, 2, 3)] == "ted'"`.
- `{0: "nula", 1: "jedna", 2: "dva"}[1] == "jedna"`.

Datové typy

`list`, `tuple` a `dict` jako procedury

`list` jako procedura

Procedure/funkce `list` umožňuje převod jiného datového typu na seznam, pokud to (podle Pythonu) dává smysl. Obecné pravidlo je, že Python umí převést na seznam jen ty datové typy, **které jsou číslované**.

- `list(x: int|float)` hodí chybu,
- `list(x: str)` převede řetězec na seznam jeho znaků,
  - např. `list("kočka") == ["k", "o", "č", "k", "a"]`.
- `list(x: tuple)` převede n-tici na seznam se stejnými prvky,
  - např. `list((1, 2, 3)) == [1, 2, 3]`.
- `list(x: dict)` převede slovník na seznam klíčů.
  - např. `list({"pes": "haf", "kočka": "mňau"}) == ["pes", "kočka"]`.

tuple jako procedura

Procedura/funkce `tuple` funguje v zásadě stejně jako `list`. Tzn.

- `tuple(x: int|float)` hodí chybu,
- `tuple(x: str)` udělá z řetězce n-tici jeho symbolů,
- `tuple(x: list)` převede seznam na n-tici se stejnými prvky.
- `tuple(x: dict)` převede slovník na n-tici jeho klíčů.

dict jako procedura

Procedura/funkce dict lze použít pouze na převod seznamu nebo n-tice, jejichž **každý prvek má délku 2** (tj. dvojice nebo seznam o dvou prvcích). Příklady:

- `dict([("pes", 2), ("kočka", 3)])` == `{"pes": 2, "kočka": 3}`.
- `dict((["pes", 2], ["kočka", 3]))` == `{"pes": 2, "kočka": 3}`.