Lekce 1
Syntaxe a čísla

C2184 Úvod do programování v Pythonu
podzim 2016

Stanislav Geidl
Národní centrum pro výzkum biomolekul
Masarykova univerzita

Ukázka kódu

```
# komentar
"""
Toto je
komentar
"""
a = 5
b = [1,2,3]
c = sum(b)
if a > c:
        c = a
        b = 0
def pozdrav():
        print("Hello World!")
        return True
import math
print(math.pi)
d = math.sqrt(2)
```


Komentář

- komentář na jeden řádek začíná #
- komentář na více řádku je uzavřen v bloku """... """
- komentář slouží pro doplňující informace ke kódu pro autora nebo dalšího vývojaře - zvyšují míru čitelnosti a přehlednosti kódu
- obecné zásady psaní komentářů:
  - komentáře nesmí duplikovat kód
  - komentář musí jednoznačně osvětlit popisovanou část kódu
  - komentuje se hlavička každého souboru, se kterým se pracuje
  - komentuje se hlavička všech objetků, funkcí a konstrukcí procedur
  - komentují se všechny netriviální konstrukce

Proměnná (variable)

- má svůj název (identifikátor, identifier) `a`
- odkazuje na místo v paměti, kde je uložena tzv. hodnota (value) proměnné
- propojení proměnné a její hodnoty provádíme pomocí přiřazení (assign) (=)

```
a = 1
```

- Python je jazyk s dynamickou typovou kontrolou (do proměnné můžete uložit cokoliv, ale hodnoty si pamatují svůj typ)

```
10 + "a"
```

- název proměnné se může skládat z malých (a velkých) písmen (bez diakritiky), čísel a podtržítka (_), nesmí začínat číslem, podtžítkem začínají speciální proměnné, název proměnné se nesmí shodovat s žádným z 34 klíčových slov jazyka Python: `and, as, assert, break, class, continue, def, del, elif, else, expect, finally, for, from, global, if, import, in, is, lambda, nonlocal, not, or, pass, raise, return, try, while, with, yield, True, False, None`

Funkce a jejich použití

- **funkce** je část programu (je podprogram, angl. subroutine), která lze volat opakovaně z různých části programu
- každá funkce má svůj název doplněný o závorky ;)

```
print("Hello World!")
```

- **knihovna** je soubor proměnných, konstant, funkcí a dalších objektů, které můžeme načíst pro daný skript pomocí klíčového slova import
- normálně pak používáme název knihovny před každou proměnnou, funkcí či jiným objektem (math.pi nebo math.sqrt())

```
import math
D = math.pi + math.sqrt(2)
```

Základní datové typy

1. **bool**ean (booleovský typ) nabývá buď hodnoty `True` nebo `False`.
2. Čísla mohou být celá (**int**eger; 1 a 2), reálná (**float**; 1.1 a 1.2) nebo komplexní (**complex**; 3+1j).
3. Řetězce (**str**ing) jsou posloupnosti Unicode znaků. Tuto podobu může mít například html dokument.
4. Bajty (**byte**) a pole bajtů, například soubor s obrázkem ve formátu jpeg.
5. Seznamy (**list**) jsou uspořádané posloupnosti hodnot.
6. N-tice (**tuple**) jsou uspořádané, neměnné posloupnosti hodnot.
7. Množiny (**set**) jsou neuspořádané kolekce hodnot.
8. Slovníky (**dict**ionary) jsou neuspořádané kolekce dvojic klíč-hodnota.

Zjištění a změna typu

- funkce type()

```
>>> type(10)
<type 'int'>
>>> type(1.5)
<type 'float'>
>>> type(3+1j)
<type 'complex'>
>>> type(True)
<type 'bool'>
>>> type("test")
<type 'str'>
>>> type(None)
<type 'NoneType'>
```

- změna pomocí funkce, která nese název podle typu, který bude výsledkem, např. int()

Zjištění a změna typu

- funkce type()
- změna pomocí funkce, která nese název podle typu, který bude výsledkem, např. int()

```
>>> int("10")
10
>>> int("10a")
ValueError: invalid literal for int()
with base 10: '10a'
>>> int(10.6)
10
>>> float(1)
1.0
>>> bool(0)
False
>>> bool(1)
True
>>> bool(None)
False
>>> str(1)
'1'
```


Bloky

- **Bloky slouží k seskupení příkazů**, například uvnitř cyklů, funkcí, objektů, struktur atd.
- Například v Pascalu jsou bloky označeny slovy BEGIN a END a v jazyce C slouží pro vytváření bloků špičaté závorky . V Pythonu se pro vytváření bloků používá odsazování.
- Nový blok vytvoříte tak, že napíšete na začátku řádků před příkazy, které spolu tvoří blok, libovolný počet mezer nebo tabulátorů. Ale **před každým dalším příkazem v bloku musí být stejný počet mezer a tabulátorů**! Odsadíte-li o mezeru více, nebo pokud místo tabulátoru použijete mezery, začnete tím jiný blok. Pokud vytvoříte nový blok na místě, kde nemá být, skončí program s chybou.
- bílé znaky = mezera, tabulator, \n , \r

```
prikaz1:
  blok1
  blok1
prikaz2:
  blok2
  blok2
  prikaz3:
    blok3
```

Matematické operátory

$+$	sčítání (např. $10 + 3$)
$-$	odčítání (např. $10 - 3$)
$*$	násobení (např. $10 * 3$)
$/$	dělení (např. $10/3$)
$//$	celočíselné dělění (např. $10 // 3 = ??$)
%	modulo (zbytek po celočíselném dělení; např. $10 \% 3 = ??$)
$**$	umocňování a odmocňování (např. $10 * *3$ nebo $10 * *0.5$)
()	závorky
`abs()`	funkce, která vrací absolutní hodnotu

- pořadí operací jako v matematice
- Co bude výsledkem $2 * *8 - (5 + 5) * 5 * 5 + 5$?

Speciální typy přiřazení: `+=` `-=` `*=` `/=` `%=` `//=`

```
a += 1 # a = a + 1
```


Příklad - řešení kvadratické rovnice

Máme kvadratickou rovnici:

$$ax^2 + bx + c = 0 \tag{1}$$

známe hodnoty proměných a, b a c a hledáme hodnoty proměnných $x1$ a $x2$

```
a = 1
b = -6
c = 5

x1 = ?
x2 = ?

print(x1)
print(x2)
```


Příklad - řešení kvadratické rovnice

$$ax^2 + bx + c = 0 \tag{2}$$

Řešení pomoci diskrimintatu:

$$D = b^2 - 4ac \tag{3}$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \tag{4}$$

Příklad - řešení kvadratické rovnice

```
a = 1
b = -6
c = 5

D = b**2 - 4*a*c
x1 = (-b+math.sqrt(D)) / (2*a)
x2 = (-b-math.sqrt(D)) / (2*a)

print(x1)
print(x2)
```