

# C2184 Úvod do programování v Pythonu (2021)

## 2. Syntax, čísla a matematické operace

### Čísla

- Celá (*integer*): 1, 5, -25, 0
- Reálná (*float*): 3.14, -5.5, 6.022e23, 1.6e-19

### Aritmetické operátory

- Klasické sčítání, odčítání, násobení a dělení: + - \* /

```
[1]: 5 + 2 - 4
```

```
[1]: 3
```

```
[2]: 2 * 5 / 6
```

```
[2]: 1.6666666666666667
```

- Mocniny: \*\*

```
[3]: 5**2
```

```
[3]: 25
```

### Priorita aritmetických operátorů

- Jako v matematice: nejdřív \*\*, pak \* /, nakonec + -

```
[4]: 2**8 - (5 + 5) * 5 * 5 + 5
```

```
[4]: 11
```

- Závorky jsou vždy jen (), nepoužíváme [] a {} jako v matematice

```
[5]: (10 * ((5-4) * 2 + 1)) ** 2
```

```
[5]: 900
```

## Celočíselné dělení (*integer division*)

Alice a Bob si chtějí rozdělit 7 jablíček...

$$7 \div 2 = 3 \text{ (zbytek 1)}$$

- `//` počítá celočíselný podíl
- `%` počítá zbytek po dělení (také *modulo*, řekneme např. "7 modulo 2 rovná se 1").

[6]: `7 // 2`

[6]: 3

[7]: `7 % 2`

[7]: 1

## Otázky:

Které z těchto čísel je největší?

- A) 5/3
- B) 5//3
- C) 5%3
- D) 5,3

Které z těchto čísel je nejmenší?

- A) 6.12
- B) 6.1e2
- C) 6.1 e 2
- D) 6.1 \*\* 2

## Funkce (*function*)

- Objekt, kterému dáme nějaké parametry a on nám něco vrátí a případně něco udělá
- Funkci voláme pomocí závorek

[8]: `abs`

[8]: `<function abs(x, /)>`

[9]: `abs(-5)`

[9]: 5

```
[10]: print('hello')
```

hello

- Funkce může mít i více než jeden parametr

```
[11]: max(1, 5, 2)
```

```
[11]: 5
```

- Nebo taky žádný

```
[12]: print()
```

- Funkce lze vnořovat

```
[13]: print(max(1, abs(-5), 2+2))
```

5

- Později si ukážeme, jak vytvářet vlastní funkce

## Typy (*types*)

Každá hodnota v Pythonu má svůj typ.

Základní typy:

- int = celá čísla (*integers*)
- float = reálná čísla (*floating-point numbers*)
- complex = komplexní čísla (*complex numbers*)
  - komplexní složka se označuje j (např. 1+2j, 3-1j)
- bool = logické hodnoty (*Boolean*): True, False
- str = řetězce (*strings*), např. 'Hello World'
- NoneType = typ, který má pouze jednu hodnotu: None ("nic")
  - hodnotu None vrací např. funkce print

Složitější typy:

- funkce
- kolekce (list, tuple, dict...)
- třídy
- ...

## Funkce type

- Zjišťuje, jakého typu je hodnota

```
[14]: type(1)
```

```
[14]: int
```

```
[15]: type(3.14)
```

```
[15]: float
```

```
[16]: type(1.0)
```

```
[16]: float
```

```
[17]: type(True)
```

```
[17]: bool
```

```
[18]: type('10')
```

```
[18]: str
```

## Všechno má svůj typ

```
[19]: type(print)
```

```
[19]: builtin_function_or_method
```

```
[20]: type(print())
```

```
[20]: NoneType
```

```
[21]: type(int)
```

```
[21]: type
```

```
[22]: type(type)
```

```
[22]: type
```

## Přetypování (*type conversion*)

- Název funkce pro konkrétní typ se jmenuje stejně jako daný typ (např. desetinná čísla `float()`)

```
[23]: type(10)
```

```
[23]: int
```

```
[24]: float(10)
```

```
[24]: 10.0
```

```
[25]: type(float(10))
```

```
[25]: float
```

```
[26]: str(10)
```

```
[26]: '10'
```

```
[27]: type(str(10))
```

```
[27]: str
```

## Otázky:

Který z těchto příkazů vypíše na výstup 200?

- A) `print 200`
- B) `print(100+100)`
- C) `print('100+100')`
- D) `print(float(200))`

Která z těchto hodnot je typu `int`?

- A) `int`
- B) `type(int)`
- C) `int(24/7)`
- D) `'9'`

## Proměnné (*variables*)

- “Krabičky” pro uložení hodnot
- Každá proměnná má svůj název (identifikátor, *identifier*)

- Odkazuje na místo v paměti počítače, kde je uložena hodnota (*value*)
- Název proměnné
  - Popisuje její význam
  - Doporučuje se anglicky
  - Může obsahovat písmena bez diakritiky, číslice, podtržítka \_
  - Nesmí začínat číslem a nesmí být shodný s klíčovým slovem (seznam: [https://docs.python.org/3/reference/lexical\\_analysis.html#keywords](https://docs.python.org/3/reference/lexical_analysis.html#keywords))
  - Používají se malá písmena, slova se oddělují podtržítkem
  - Příklady: `time`, `average_water_temperature`, `x1`, `x2`, `V`
- Do proměnné vkládáme hodnotu pomocí operátoru přiřazení = (*assignment*)
  - `kam = co`, ne `naopak!`

```
[28]: a = 10
      b = 5
```

```
[29]: a
```

```
[29]: 10
```

```
[30]: b
```

```
[30]: 5
```

```
[31]: a = b
      a
```

```
[31]: 5
```

```
[32]: c
```

```

↳
-----
NameError                                Traceback (most recent call last)
<ipython-input-32-2b66fd261ee5> in <module>
----> 1 c

NameError: name 'c' is not defined

```

## Můžeme naplnit víc proměnných současně

```
[33]: weight, volume = 3.5, 2.0
```

```
[34]: x = y = z = 1
```

```
[35]: a, b = b, a
```

## Proměnnou lze i smazat

```
[36]: del a
```

## Proměnné nemají typ

- Do jedné proměnné lze ukládat hodnoty různých typů

```
[37]: x = 10  
type(x)
```

```
[37]: int
```

```
[38]: x = 'hello'  
type(x)
```

```
[38]: str
```

```
[39]: x = abs  
type(x)
```

```
[39]: builtin_function_or_method
```

- Tomuto principu se říká *dynamické typování*

## Speciální proměnná `_`

- Pouze v interaktivním módu
- `_` obsahuje výsledek posledního spuštěného příkazu (v notebooku poslední spuštěné buňky), pokud nebyl uložen do jiné proměnné a nebyl `None`
- Obsah této proměnné se nám automaticky vypisuje
  - Pozor, jen v interaktivním módu (v normálním módu musíme použít `print`)

```
[40]: abs(2 - 7)
```

```
[40]: 5
```

[41]: \_

[41]: 5

### Speciální přiřazení

- Operátory +=, -=, \*=, /=, //=, %=, \*\*=
- Máme obecně  $p \text{ ?} = v$ , Python interpretuje tento zápis jako  $p = p \text{ ?} v$ , kde  $p$  je proměnná,  $?$  je operátor,  $v$  může být proměnná nebo hodnota

```
[42]: a = 1
a += 1
a
```

[42]: 2

```
[43]: a = 2
a *= 8
a
```

[43]: 16

### Konstanty (*constants*)

- Proměnné, kterých hodnota by sa neměla měnit
- Nazýváme je velkými písmeny

```
[44]: AVOGADRO_NUMBER = 6.022e23
GOLDEN_RATIO = (1 + 5**(1/2)) / 2
```

### Otázky:

```
a, b = 5, 2
a += b
b = a
a -= 1
```

Co bude v proměnných  $a$ ,  $b$  po vykonání uvedeného kódu?

- A) 5, 2
- B) 1, 2
- C) 6, 7
- D) 6, 6



## Komentáře (*comments*)

- Komentář je všechno za znakem #
- Doplnují kód, aby bylo jasné, co dělá a proč
- Python je ignoruje

```
[45]: U = 1.5 # voltage [V]
      R = 500 # resistance [Ω]
      I = U / R # compute electric current [A] by Ohm's law
```

```
[46]: # print(I)
```

- VSCode - zakomentování/odkomentování celého řádku nebo více řádků pomocí Ctrl + / (na české klávesnici -)
- Komentáře doplňují informace, neduplikují kód

```
[47]: weight *= 1000 # proměnnou m vynásobíme 1000 (zbytečný komentář)
      weight *= 1000 # přepočítání z kilogramů na gramy (užitečný komentář)
```

- Přehlednost - mezera za #, aspoň dvě mezery před #

```
[48]: #Ugly comment
      a=5#another ugly comment

      # Nice comment
      a = 5 # another nice comment
```

- Příliš mnoho komentářů značí, že možná něco děláme špatně (*code smell*):

```
[49]: n = 25 # number of students
```

## Hodnota ...

- Hodnota ... (nebo Ellipsis) nemá pro výpočty praktické využití
- Můžeme ji využít např. jestli chceme něco doplnit později

```
[50]: circle_radius = 2.5
      circle_area = ... # TODO find the formula
      print('A circle with radius', circle_radius, 'has area', circle_area)
```

A circle with radius 2.5 has area Ellipsis

## Moduly (*modules*)

- Modul je soubor proměnných, konstant, funkcí a dalších objektů
- Modul načítáme pomocí klíčového slova import

- Objekty z modulu vybíráme pomocí tečky .

```
[51]: import math  
      math.pi
```

```
[51]: 3.141592653589793
```

```
[52]: math.sqrt(2) # Odmocnina
```

```
[52]: 1.4142135623730951
```

```
[53]: math.log(100) # Přirozený logaritmus
```

```
[53]: 4.605170185988092
```

```
[54]: math.log10(100) # Desítkový logaritmus
```

```
[54]: 2.0
```

```
[55]: math.exp(1.5) # e**1.5
```

```
[55]: 4.4816890703380645
```

```
[56]: math.sin(90) # Sínus úhlu v radiánech
```

```
[56]: 0.8939966636005579
```

```
[57]: math.radians(90) # Stupně -> radiány
```

```
[57]: 1.5707963267948966
```

```
[58]: math.degrees(2 * math.pi) # Radiány -> stupně
```

```
[58]: 360.0
```

## Logické hodnoty

- Existují pouze dvě: True (pravda), False (nepravda)
- Tyto hodnoty jsou typu bool (zkratka od angl. *Boolean*, zavedl je matematik George Boole)

## Porovnávací operátory (*comparison operators*)

- Větší, menší:  $a > b$ ,  $a < b$
- Rovno, není rovno:  $a == b$ ,  $a != b$

- Větší rovno, menší rovno:  $a \geq b$ ,  $a \leq b$
- Výsledkem těchto operátorů je vždy **logická hodnota** (True/False).
- Pozor, nelze zaměňovat  $=$  a  $==$

```
[59]: x == 5 # Je x rovno pěti?
```

[59]: False

```
[60]: x = 5 # Do proměnné x přiřaď hodnotu 5!
```

## Logické operátory

- Pracují s logickými hodnotami
- and - “a zároveň” (konjunkce,  $\wedge$ )
- or - “nebo” (disjunkce,  $\vee$ )
- not - “neplatí, že” (negace,  $\neg$ )

```
[61]: p = 2 < 5
p
```

[61]: True

```
[62]: r = 2 + 2 == 5
r
```

[62]: False

```
[63]: p and r # Platí p a zároveň r (obě současně)?
```

[63]: False

```
[64]: p or r # Platí p nebo r (aspoň jedno z nich)?
```

[64]: True

```
[65]: not p # Je pravda, že neplatí p?
```

[65]: False

```
[ ]: je_duha = prsi and sviti_slunce
mam_volno = je_sobota or je_nedele or je_svatek
musim_do_prace = not mam_volno
```

## Priorita operátorů

1. Aritmetické operátory `**` `*` `/` `+` `-` ...
2. Porovnávací operátory `==` `!=` `>` `<` `>=` `<=` ...
3. `not`
4. `and`
5. `or`
6. Přiřazení `=`

Pokud to chceme jinak, použijeme závorky

```
[67]: 100 <= 200 and 5 > 10 or 2 + 2 == 4
```

[67]: True

```
[68]: not True or 9 + 3 > 11 and 5 != 6
```

[68]: True

## Zkratky:

```
0 <= x < 10
```

je to stejné jako

```
0 <= x and x < 10
```

## ... ale pozor:

```
[69]: 2 and 8 > 5
```

[69]: True

Priorita: `2 and (8 > 5)`

Číslo 2 jakožto nenulové číslo se považuje za “pravdivé” (*truthy*)

(Nula, None, prázdný řetězec `' '` se považují za “nepravdivé” (*falsy*))

## Procvičení

Spočítejte výsledky následujících buněk z hlavy.

Výsledek je vždy True nebo False nebo může být v buňce chyba.

```
[ ]: # 1
```

```
2**2 == 2*2
```

```
[ ]: # 2
```

```
5e-6 >= abs(-5e6)
```

```
[ ]: # 3
```

```
import math
```

```
type(10/2) == type(math.pi)
```

```
[ ]: # 4
```

```
x = 5  
x -= 1  
x *= x  
y = 10 - x  
  
y > 0
```

```
[ ]: # 5
```

```
n = 20  
n_lost = 7  
n_final = n - n_lost  
  
n_final != 13
```

```
[ ]: # 6
```

```
height = width = 10  
area = height * width  
  
height > 5 and area = 100
```

```
[ ]: # 7
```

```
a = True  
b = False  
  
a and b or not a and not b
```

```
[ ]: # 8
```

```
x = 5
```

```
y = 10
```

```
y > x * x or y >= 2 * x and x < y
```

```
[ ]: # 9
```

```
14 // 5 > 14 % 5
```

```
[ ]: # 10
```

```
14 // 5 > 14 % 5 and not False and (True or True == False)
```