C2184 Úvod do programování v Pythonu

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: najdřív `**`, 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 dělění (*integer division*)

Alice a Bob si chtějí rozdělit 7 jablíček…

7 ÷ 2 = 3 (zbytek 1)

- `//` počítá celočíselný podíl
- `%` počíta 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`

Proměnné (*variables*)

- “Krabičky” pro uložení hodnot
- Každá proměnná má svůj název (identifikátor, *identifier*)
- Odkazuje na místo v paměti počítače, kde je uložená hodnota (*value*)
- Název proměnné
  - Popisuje její význam
  - Může obsahovat písmena bez diakritiky, číslice, podtržítko _
  - Nesmí začínat číslicí a nesmí být shodný s klíčovým slovem (seznam: https://docs.python.org/3/reference/lexical_analysis.html#keywords)
  - Doporučuje se anglicky

  - 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!

[8]:
```
a = 10
b = 5
```

[9]:
```
a
```

[9]:
```
10
```

[10]:
```
b
```

[10]:
```
5
```

[11]:
```
a = b
a
```

[11]:
```
5
```

[12]:
```
c
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
  ↪last)
/home/adam/School/Praca/Python/2022/Python/cviko_02/02_Syntax.ipynb␣
  ↪Cell 26 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/home/adam/School/Praca/Python/
  ↪2022/Python/cviko_02/02_Syntax.ipynb#X34sZmlsZQ%3D%3D?line=0'>1</a>␣
  ↪c

NameError: name 'c' is not defined
```

Můžeme naplnit víc proměnných současně

[13]:
```
weight, volume = 3.5, 2.0
```

[14]:
```
x = y = z = 1
```

[15]:
```
a, b = b, a
```

Proměnnou lze i smazat

[16]:
```
del a
```

Speciální přiřazení

- Operátory `+=` `-=` `*=` `/=` `//=` `%=` `**=`
- Obecně `p ?= v` je zkratka pro `p = p ? v` (kde `p` je proměnná, `?` je operátor, `v` může být proměnná nebo hodnota)

[17]:
```
a = 1
a += 1
a
```

[17]: 2

[18]:
```
a *= 8
a
```

[18]: 16

Konstanty (*constants*)

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

[19]:
```
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 #
- Doplňují kód, aby bylo jasné, co dělá a proč
- Python je ignoruje

[20]:
```
U = 1.5  # voltage [V]
R = 500  # resistance [Ω]
I = U / R  # compute electric current [A] by Ohm's law
```

[21]:
```
# print(I)
```

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

[22]:
```
weight *= 1000  # proměnnou m vynásobíme 1000 (zbytečný komentář)
weight *= 1000  # přepočet z kilogramů na gramy (užitečný komentář)
```

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

[23]:
```
#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*):

[24]:
```
n = 25  # number of students
```

Hodnota `...`

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

[25]:
```
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
```

Funkce (*function*)

- Objekt, kterému dáme nějaké parametry a on nám něco vrátí a případně něco udělá

- Funkci voláme pomocí závorek

[26]:
```
abs
```

[26]:
```
<function abs(x, /)>
```

[27]:
```
abs(-5)  # Absolutní hodnota
```

[27]:
```
5
```

[28]:
```
round(2.85)  # Zakrouhlení
```

[28]:
```
3
```

[29]:
```
print('hello')  # Vypiš na výstup
```

```
hello
```

- Funkce může mít i více než jeden parametr

[30]:
```
max(1, 5, 2)
```

[30]:
```
5
```

- Nebo taky žádný

[31]:
```
print()
```

- Funkce lze vnořovat

[32]:
```
print(max(1, abs(-5), 2+2))
```

```
5
```

- Později si ukážeme, jak vytvářet vlastní funkce

Hodnota `None`

- "Nic"
- Vyjádřuje, že něco neexistuje / chybí / nelze najít

[33]:
```
my_favorite_number = 2
my_favorite_color = None
```

- Některé funkce mají vedlejší efekt, ale nevrací žádný výsledek
  - Např. `print` – vypíše text, ale nic nevrací
- Výsledek takové funkce je `None`

[34]:
```
x = print(5+5)
```

```
10
```

- Jupyter notebook automaticky vypisuje výsledek posledního řádku buňky, pouze `None` se nevypisuje

[35]:
```
x
```

[36]:
```
print(x)
```

```
None
```

Moduly (*modules*)

- Modul je soubor proměnných, konstant, funkcí a dalších objektů
- Modul načítáme pomocí klíčového slova `import`
  - Zpravidla všechny moduly načítáme hned na začátku programu
- Objekty z modulu vybíráme pomocí tečky `.`

[37]:
```
import math
math.pi
```

[37]:
```
3.141592653589793
```

[38]:
```
math.sqrt(2)  # Odmocnina
```

[38]:
```
1.4142135623730951
```

[39]:
```
math.log(100)  # Přirozený logaritmus
```

[39]:
```
4.605170185988092
```

[40]:
```
math.log10(100)  # Desítkový logaritmus
```

[40]:
```
2.0
```

[41]:
```
math.exp(1.5)  # e**1.5
```

[41]:
```
4.4816890703380645
```

[42]:
```
math.sin(90)  # Sínus úhlu v radiánech
```

[42]:
```
0.8939966636005579
```

[43]:
```
math.radians(90)  # Stupně -> radiány
```

[43]:
```
1.5707963267948966
```

[44]:
```
math.degrees(2 * math.pi)  # Radiány -> stupně
```

[44]:
```
360.0
```

Typy (*types*)

Každá hodnota v Pythonu má svůj typ.

Základní typy:

- `int` = celá čísla (*integers*): `1`, `2` ...
- `float` = reálná čísla (*floating-point numbers*): `1.5`, `2.0` ...
- `complex` = komplexní čísla (*complex numbers*)
  - komplexní složka se označuje `j` (`1+2j`, `3-1j` ...)
- `bool` = logické hodnoty (*Boolean*): `True`, `False`
- `str` = řetězce (*strings*): `'Hello World'`, `'a'` ...
- `NoneType` = typ, který má pouze jednu hodnotu: `None`
- `type` = typ samotných typů: `int`, `float`, `str`, `type` ...

Složitější typy:

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

Funkce `type`

- Zjišťuje, jakého typu je hodnota

[45]:
```
type(1)
```

[45]:
```
int
```

[46]:
```
type(3.14)
```

[46]:
```
float
```

[47]:
```
type(1.0)
```

[47]:
```
float
```

[48]:
```
type(True)
```

[48]:
```
bool
```

[49]:
```
type('10')
```

[49]:
```
str
```

Všechno má svůj typ

[50]:
```
type(print)
```

[50]:
```
builtin_function_or_method
```

[51]:
```
type(print())
```

[51]:
```
NoneType
```

[52]:
```
type(int)
```

[52]:
```
type
```

[53]:
```
type(type)
```

[53]:
```
type
```

Přetypování (*type conversion*)

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

[54]:
```
float(10)
```

[54]:
```
10.0
```

[55]:
```
str(10)
```

[55]:
```
'10'
```

[56]:
```
int(3.5)
```

[56]:
```
3
```

[57]:
```
int('1000')
```

[57]:
```
1000
```

Proměnné nemají pevný typ

- Do jedné proměnné lze ukládat hodnoty různých typů
- Tomuto principu se říká *dynamické typování*

[58]:
```
x = 10
type(x)
```

[58]:
```
int
```

[59]:
```
x = 'hello'
type(x)
```

[59]:
```
str
```

[60]:
```
x = abs
type(x)
```

[60]:
```
builtin_function_or_method
```

- VSCode – když podržíme kurzor nad proměnnou, ukáže nám její aktuální typ (nebo i hodnotu)

[61]:
```
a = 5 + 2
b = a / 2
c = str(b)
```

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'`

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*)

- Je rovno: `a == b`
- Není rovno: `a != b`
- Větší, menší: `a > b, a < b`
- Větší rovno, menší rovno: `a >= b, a <= b`
- Výsledkem těchto operátorů je vždy **logická hodnota** (`True`/`False`).

[62]:
```
10 < 20
```

[62]:
```
True
```

[63]:
```
a == b
```

[63]:
```
False
```

[64]:
```
a != b
```

[64]:
```
True
```

- Pozor, nelze zaměňovat = a ==

[65]:
```
x == 5  # Je x rovno pěti?
```

[65]:
```
False
```

[66]:
```
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, $\neg$)

Příklady:

```
je_duha = prsi and sviti_slunce

mam_volno = je_sobota or je_nedele or je_svatek

musim_do_prace = not mam_volno
```

[67]:
```
p = 2 < 5
p
```

[67]: True

[68]:
```
r = 2 + 2 == 5
r
```

[68]: False

[69]:
```
p and r  # Platí p a zároveň r (obě současně)?
```

[69]: False

[70]:
```
p or r  # Platí p nebo r (aspoň jedno z nich)?
```

[70]: True

[71]:
```
not p  # Je pravda, že neplatí p?
```

[71]: False

Priorita operátorů

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

Pokud to chceme jinak, použijeme závorky

[72]:
```
100 <= 200 and 5 > 10 or 2 + 2 == 4
```

[72]: True

[73]:
```
not True or 9 + 3 > 11 and 5 != 6
```

[73]: True

Zkratky:

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

je to stejné jako

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

... ale pozor:

[74]:
```
2 and 8 > 5
```

[74]: True

Priorita: `2 and (8 > 5)`

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

(`0`, `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)
```