C2184 Úvod do programování v Pythonu

2. Syntaxe, 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

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

Out[4]:
```
3
```

In [70]:
```
2 * 5 / 6
```

Out[70]:
```
1.6666666666666667
```

Mocniny a odmocniny

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

Out[6]:
```
25
```

In [7]:
```
25**(1/2)
```

Out[7]:
```
5.0
```

Priorita aritmetických operátorů

- Jako v matematice: najdřív **, pak * /, nakonec + -

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

Out[8]:
```
11
```

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

In [9]:
```
(10 * ((5-4) * 2 + 1)) ** 2
```

Out[9]:
```
900
```

Celočíselné dělení

Alice a Bob si chtějí rozdělit 7 jablíček...

7 ÷ 2 = 3 (zbytek 1)

In [10]:
```
7 // 2  # Výsledek celočíselného dělení
```

Out[10]: 3

In [11]:
```
7 % 2  # Zbytek po dělení
```

Out[11]: 1

Zbytku po dělení se říká také *modulo* (říkáme "7 modulo 2").

Funkce

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

In [71]:
```
abs
```

Out[71]:
```
<function abs(x, /)>
```

In [13]:
```
abs(-5)
```

Out[13]:
```
5
```

In [14]:
```
print('hello')
```

```
hello
```

- Funkce může mít i více než jeden parametr

In [15]:
```
max(1, 5, 2)
```

Out[15]: 5

- Nebo taky žádný

In [16]:
```
print()
```

- Funkce lze vnořovat

In [17]:
```
print(max(1, abs(-5), 2+2))
```

```
5
```

- Později si ukážeme, jak vytvářet vlastní funkce

Typy

Každá hodnota v Pythonu má svůj typ.

Základní typy:

- `int` = celá čísla (*integers*)
- `float` = reálná čísla (*floating-point numbers*)
- `complex` = komplexní čísla
  - 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`

Funkce `type`

- Zjišťuje, jakého typu je hodnota

In [18]:
```
type(1)
```

Out[18]:
```
int
```

In [19]:
```
type(3.14)
```

Out[19]:
```
float
```

In [20]:
```
type(1.0)
```

Out[20]:
```
float
```

In [21]:
```
type(True)
```

Out[21]:
```
bool
```

In [67]:
```
type('10')
```

Out[67]:
```
str
```

Všechno má svůj typ

In [37]:
```
type(print)
```

Out[37]:
```
builtin_function_or_method
```

In [38]:
```
type(print())
```

Out[38]:
```
NoneType
```

In [39]:
```
type(int)
```

Out[39]:
```
type
```

In [40]:
```
type(type)
```

Out[40]:
```
type
```

Přetypování (*type conversion*)

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

In [42]:
```
type(10)
```

Out[42]:
```
int
```

In [43]:
```
float(10)
```

Out[43]:
```
10.0
```

In [44]:
```
type(float(10))
```

Out[44]:
```
float
```

In [45]:
```
str(10)
```

Out[45]:
```
'10'
```

In [46]:
```
type(str(10))
```

Out[46]:
```
str
```

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 číslem a nesmí být shodný s klíčovým slovem (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

```
time, average_water_temperature, x1, x2, V
```

- Propojení proměnné s hodnotou pomocí přiřazení = (*assignment*)

In [47]:
```
a = 10
b = 5
```

In [48]:
```
a
```

Out[48]:
```
10
```

In [49]:
```
b
```

Out[49]:
```
5
```

In [50]:
```
a = b
a
```

Out[50]:
```
5
```

In [51]:
```
c
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-51-2b66fd261ee5> in <module>
----> 1 c

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

Proměnné nemají typ

- Do jedné proměnné lze ukládat hodnoty různých typů

In [52]:
```
x = 10
type(x)
```

Out[52]:
```
int
```

In [53]:
```
x = 'hello'
type(x)
```

Out[53]:
```
str
```

In [54]:
```
x = abs
type(x)
```

Out[54]:
```
builtin_function_or_method
```

- Tomuto principu se říká *dynamické typování*

Speciální přiřazení

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

In [55]:
```
a = 1
a += 1
a
```

Out[55]: 2

In [56]:
```
a = 2
a *= 8
a
```

Out[56]: 16

Konstanty

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

```
AVOGADRO_NUMBER = 6.022e23
GOLDEN_RATIO = (1 + 5**(1/2)) / 2
```

Komentáře (*comments*)

- Doplňují kód, aby bylo jasné, co dělá
- Python je ignoruje
- Komentáře na jeden řádek začínají #
- Komentář na více řádku je uzavřen v """ ... """

```
def pocitej_odpor(napeti, proud):
    """Vypocet odporu dle Ohmova zakona.
    napeti -- vstupni napeti [V]
    """
    odpor = napeti / proud
    return odpor

napeti = 5  # Vstupni napeti [V]
```

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

```
m *= 1000  # Proměnnou m vynásobíme 1000 (zbytečný komentář)
m *= 1000  # Přepočet z kilogramů na gramy (užitečný komentář)
```

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

```
#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*)
  - Smradlavý kód:

```
n = 25  # Number of students
```

  - Opravený kód

```
number_of_students = 25
n_students = 25
```

Moduly (*modules*)

- Modul je soubor proměnných, konstant, funkcí a dalších objektů
- Modul načítáme pomocí klíčového slova `import`
- Objekty z modulu vybíráme pomocí tečky `.`

In [57]:
```
import math
math.pi
```

Out[57]:
```
3.141592653589793
```

In [58]:
```
math.sqrt(2)  # Odmocnina
```

Out[58]:
```
1.4142135623730951
```

In [59]:
```
math.log(100)  # Přirozený logaritmus
```

Out[59]:
```
4.605170185988092
```

In [60]:
```
math.log10(100)  # Desítkový logaritmus
```

Out[60]:
```
2.0
```

In [61]:
```
math.sin(90)  # Sínus úhlu v radiánech
```

Out[61]:
```
0.8939966636005579
```

In [62]:
```
math.radians(90)
```

Out[62]:
```
1.5707963267948966
```

In [63]:
```
math.degrees(2 * math.pi)
```

Out[63]:
```
360.0
```

Porovnávací operátory (*comparison operators*)

- Větší, menší:

```
a > b, a < b
```

- Rovno, není rovno:

```
a == b, a != b
```

- Větší rovno, menší rovno:

```
a >= b, a <= b
```

- Pozor, nelze zaměňovat = a ==

```
x = 5   # Do proměnné x přiřaď hodnotu 5!
x == 5  # Je x rovno pěti?
```

- Výsledkem těchto operátorů je vždy **logická hodnota**.

Logické hodnoty

- Existují pouze dvě:

  `True, False`

- Tyto hodnoty jsou typu `bool` (zkratka od angl. *Boolean*, zavedl je matematik George Boole)

Logické operátory

- Platí a a zároveň b

```
a and b
```

- Platí a nebo b

```
a or b
```

- Neplatí a (platí opak a, negace a)

```
not a
```

In [ ]:
```
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

In [65]:
```
100 <= 200 and 5 > 10 or 2 + 2 == 4
```

Out[65]:
```
True
```

In [66]:
```
not True or 9 + 3 > 11 and 5 != 6
```

Out[66]:
```
True
```

Zkratky:

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

je to stejné jako

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

Cvičení

Spočítejte výsledek z hlavy:

In [ ]:
```
a = True
b = False

a and b or not a and not b
```

In [ ]:
```
x = 5
y = 10

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

In [ ]:
```
13 // 5 > 13 % 5 and not False and (True or True == False)
```

Řešení:

In [3]:
```
a = True
b = False

a and b or not a and not b
```

Out[3]:
```
False
```

In [4]:
```
x = 5
y = 10

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

Out[4]:
```
True
```

In [5]:
```
13 // 5 > 13 % 5 and not False and (True or True == False)
```

Out[5]:
```
False
```