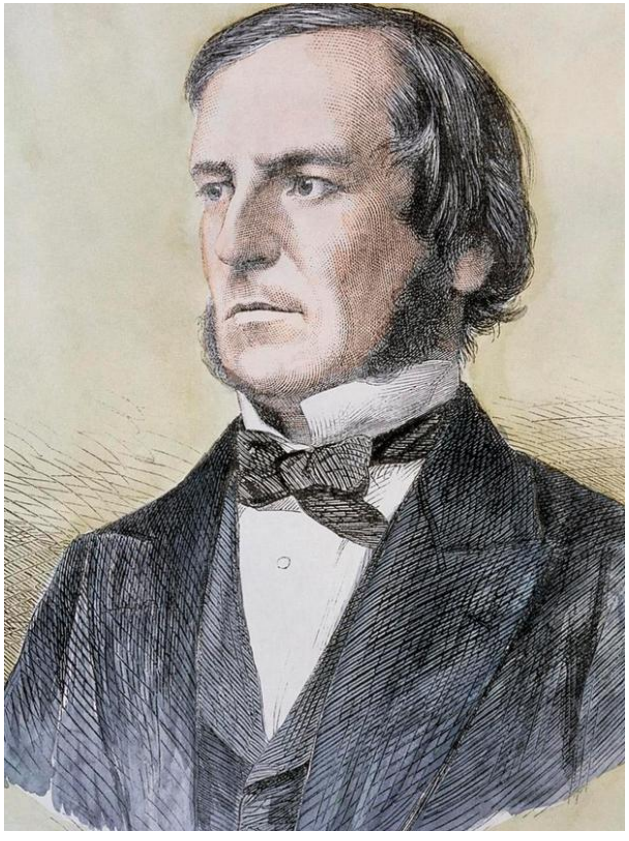
Podmienky

bool

Boolean je datový typ, ktorý nadobúda 2 hodnôt: `True` a `False`

```
print(type(True))
```



Obr. 1: George Boole

Väčšie, menšie, rovná sa

```
a = 1
b = 3
print(a < b)
```

Operátory, ktoré budeme potrebovať:

`<` `>` `<=` `>=` `==` `!=` `not` `and` `or`

`!=` je *nerovná sa*.

`not` je unárny operátor (1 argument, ostatné hore sú binárne):

```
print(not True)
```

Rozhodovanie

```
if 8 > 4:
    print('Podmienka plati')
```

```
x = 6
if x > 4:
    print('X je vacsie nez 4')
else:
    print('X nie je vacsie nez 4')
```

Rozhodovanie

Napíšte funkciu, ktorá vypíše či je argument kladný, záporný alebo nula.

```
def print_sign(x):
    if x < 0:
        print("negative")
    else:
        print("non-negative")
```

Nezabudnúť na `:` a indentovaný blok inak dostanete `SyntaxError`.

Rozhodovanie

```
def print_sign(x):
    if x < 0:
        print("negative")
    else:
        if x == 0:
            print("zero")
        else:
            print("positive")
```

Rozhodovanie

```
def print_sign(x):
    if x < 0:
        print("negative")
    elif x == 0:
        print("zero")
    else:
        print("positive")
```

Logické operátory v if

```
if 0 < x:
    if x < 10:
        print('x is a positive single-digit number.')
```

```
if 0 < x and x < 10:
    print('x is a positive single-digit number.')
```

Koncept rovnosti vo väčšom detaile

```
a = 1
b = 1.0

print(type(a))
print(type(b))

a == b     # ma to byt True?
```

“Silnejšia” rovnosť

Objekty majú svoju identitu. V implementácií Pythonu *CPython* je identita adresa v pamäti.

```
a = 3
b = "B"

print(id(a))
print(id(b))
```

Porovnanie identity je možné pomocou operátora `is` alebo `is not`:

```
a = 1
b = 1.0

print(a is b) # ekvivalentne: print(id(a) == id(b))
print(a is not b) # print(id(a) != id(b))
```

Poznámka: `is not` je jeden operátor (nie `is (not ..)`).

Kedy použiť `is` namiesto ==?

- Keď vás zaujíma či dva objekty sú naozaj jeden objekt - sú na rovnakom mieste v pamäti.
- Porovnanie s `None` (`None` je len jeden)

Poznámka: Existuje celá rada podivností, keď pracujete s typom "ako keby bol" bool a pri použití `is` a `not`:

```
print(not None)
print(not not None)
print(not [])
print(not 0)
print(not 1)
```

alebo toto, alebo toto, alebo toto.

Poučenie: nechovajte sa k iným datovým typom ako k `bool`-om.

Cvičenie 1

Python má zabudovanú funkciu `abs`, ktorá vráti absolútnu hodnotu čísla.

```
abs(1.21)     # -> 1.21
abs(-1.21)    # -> 1.21
```

Vytvorte si vlastnú verziu tejto funkcie a porovnajte s `abs`.

Cvičenie 2

Python má zabudovanú funkciu `round`, ktorá zaokrúhli číslo na najbližšie celé číslo (toto nie je celá pravda, ale budeme sa tváriť, že je to tak, inak by to bolo na dlho 1 2 3).

```
round(1.50001)    # -> 2
```

Ako by ste túto funkciu napísali pomocou funkcie `int`? Vyskúšajte a porovnajte s funkciou `round` pre nasledujúce hodnoty: `1`, `1.1`, `1.499`, `1.51`, `-1.1`, `-1.51`.

(`int` vyberá celé číslo bližšie k nule.)

Cvičenie 3

Ak je 1 strana trojuholníka väčšia ako súčet ostatných dvoch strán, tak tento trojuholník nenakreslíte. Inak áno. Napíšte funkciu, ktorá bere 3 dĺžky strán, a overí, či je možné zostaviť trojuholník z týchto strán.

Cvičenie 4

Veľká Fermatova veta hovorí:

Nejestvujú celé čísla x, y a z väčšie ako nula, pre ktoré by platilo $x^n + y^n = z^n$, kde n je prirodzené číslo väčšie ako 2.

Napíšte funkciu `check_fermat`, ktorá vezme 4 parametre a vyskúša či Fermatova veta pre ne platí.

Podľa seriálu The Simpsons by veta nemala platiť pre čísla $x = 3987$, $y = 4365$, $z = 4472$, $n = 12$.