

# Úvod do jazyka Python a počítačového programování

(Určeno pro vnitřní potřebu SOUE Plzeň, kopírování bez předchozího souhlasu je zakázáno)

## Modul 3

### Logické hodnoty, podmíněné provádění, smyčky, seznamy a zpracování seznamů, logické a bitové operace

V tomto modulu se budete zabývat následujícími tématy:

- datový typ Boolean;
- relační operátory;
- rozhodování v jazyce Python (if, if-else, if-elif, else).
- jak opakovat provádění kódu pomocí cyklů (while, for)
- jak provádět logické a bitové operace v jazyce Python;
- seznamy v jazyce Python (konstrukce, indexování a řezání; manipulace s obsahem)
- jak třídit seznamy pomocí algoritmů bublinového třídění;
- vícerozměrné seznamy a jejich použití.

#### 3.1.1.1 Rozhodování v jazyce Python

##### Otázky a odpovědi

Programátor napíše program a ten mu klade otázky.

Počítač program spustí a poskytne odpovědi. Program musí být schopen reagovat podle obdržených odpovědí.

Počítače naštěstí znají pouze dva druhy odpovědí:

- Ano, to je pravda;
- ne, toto je pravda není.

Nikdy nedostanete odpověď typu Nech mě přemýšlet...., nevím, nebo Pravděpodobně ano, ale nevím to jistě.

K pokládání otázek používá Python sadu velmi speciálních operátorů. Projdeme si je jeden po druhém a ukážeme si jejich účinky na několika jednoduchých příkladech.

## Porovnání: operátor rovnosti

Otázka: Jsou si dvě hodnoty rovny?

K položení této otázky se používá operátor `==` (rovná se).

Nezapomeňte na tento důležitý rozdíl:

- `=` je operátor přiřazení, např. `a = b` přiřadí a hodnotu b;
- `==` je otázka, zda se tyto hodnoty rovnají; `a == b` porovnává a a b.

Je to binární operátor s levostrannou vazbou. Potřebuje dva argumenty a kontroluje, zda se rovnají.

Otázka č. 1: Jaký je výsledek následujícího porovnání?

`2 == 2`

Pravda - samozřejmě, 2 se rovná 2. Python odpoví `True` (pamatujte na tuto dvojici předdefinovaných literálů `True` a `False` - jsou to také klíčová slova Pythonu).

Otázka č. 2: Jaký je výsledek následujícího porovnání?

`2 == 2.`

Tato otázka není tak snadná jako první. Python naštěstí umí převést celočíselnou hodnotu na její reálný ekvivalent, a proto je odpověď `True`.

Otázka č. 3: Jaký je výsledek následujícího porovnání?

`1 == 2`

Tohle by mělo být snadné. Odpověď bude (nebo spíše vždy bude) `False`.

### 3.1.1.2 Rozhodování v jazyce Python

Operátor `==` (rovná se) porovnává hodnoty dvou operandů. Pokud se rovnají, je výsledkem porovnání hodnota `True`. Pokud se nerovnají, výsledkem porovnání je `False`.

Podívejte se na porovnání rovnosti níže - jaký je výsledek této operace?

```
var == 0
```

Všimněte si, že odpověď nemůžeme zjistit, pokud nevíme, jaká hodnota je aktuálně uložena v proměnné `var`.

Pokud byla proměnná během provádění programu mnohokrát změněna nebo je její počáteční hodnota zadána z konzoly, může odpověď na tuto otázku dát pouze Python a pouze za běhu.

Nyní si představte programátora, který trpí nespavostí a musí počítat černé a bílé ovce zvlášť, dokud je černých ovcí přesně dvakrát více než bílých.

Otázka bude znít následovně:

```
black_sheep == 2 * white_sheep
```

Vzhledem k nízké prioritě operátoru `==` se tato otázka považuje za rovnocennou:

```
black_sheep == (2 * white_sheep)
```

Pojďme si nyní procvičit porozumění operátoru `==` - dokážete odhadnout výstup níže uvedeného kódu?

```
var = 0 # Assigning 0 to var
print(var == 0)

var = 1 # Assigning 1 to var
print(var == 0)
```

Spusťte kód a zkontrolujte, zda jste měli pravdu.

### Nerovnost: operátor nerovná se (`!=`)

Operátor `!=` (nerovná se) také porovnává hodnoty dvou operandů. Zde je rozdíl: pokud se rovnají, výsledkem porovnání je `False`. Pokud se nerovnájí, je výsledkem porovnání `True`.

Nyní se podívejte na porovnání nerovností níže - dokážete odhadnout výsledek této operace?

```
var = 0 # Assigning 0 to var
print(var != 0)

var = 1 # Assigning 1 to var
print(var != 0)
```

Spusťte kód a zkontrolujte, zda jste měli pravdu.

### Operátory porovnávání: větší než

Můžete také položit srovnávací otázku pomocí operátoru `>` (větší než). Pokud chcete zjistit, zda je více černých ovcí než bílých, můžete ji zapsat takto:

```
black_sheep > white_sheep # Greater than
```

`True` to potvrzuje, `False` to popírá.

### Operátory porovnávání: větší než nebo rovno

Operátor větší než má ještě jednu speciální, nestriktní variantu, která se však značí jinak než v klasickém aritmetickém zápisu: `>=` (větší než nebo rovno).

Následná znaménka jsou dvě, nikoliv jedno.

Oba tyto operátory (striktní i nestriktní), stejně jako dva další, o nichž bude řeč v následující části, jsou **binární operátory s levostrannou vazbou** a jejich **priorita je větší než priorita**, kterou vykazují operátory `==` a `!=`.

Chceme-li zjistit, zda máme či nemáme nosit teplou čepici, položíme si následující otázku:

```
centigrade_outside >= 0.0 # Greater than or equal to
```

## Operátory porovnávání: menší než nebo rovno

Jak jste již pravděpodobně uhodli, operátory použité v tomto případě jsou: operátor `<` (méně než) a jeho nepřísrný sourozenec: `<=` (méně než nebo rovno).

Podívejte se na tento jednoduchý příklad:

```
current_velocity_mph < 85 # Less than
current_velocity_mph <= 85 # Less than or equal to
```

Zjistíme, zda hrozí pokuta od dálniční policie (první otázka je přísná, druhá ne).

## Využití odpovědí

Co můžete udělat s odpovědí (tj. s výsledkem porovnávací operace), kterou získáte z počítače?

Existují přinejmenším dvě možnosti: zaprvé si ji můžete zapamatovat (uložit do proměnné) a využít ji později. Jak to uděláte? No, použijete libovolnou proměnnou, například takto:

```
answer = number_of_lions >= number_of_lionesses
```

Obsah proměnné vám prozradí odpověď na položenou otázku.

Druhá možnost je pohodlnější a mnohem častější: na základě získané odpovědi můžete rozhodnout o budoucnosti programu. K tomuto účelu potřebujete speciální instrukci, kterou si brzy probereme.

Nyní musíme aktualizovat naši tabulku priorit a zařadit do ní všechny nové operátory. Ta nyní vypadá následovně:

Priorita	Operátor	
1	<code>+</code> , <code>-</code>	unární
2	<code>**</code>	
3	<code>*</code> , <code>/</code> , <code>//</code> , <code>%</code>	
4	<code>+</code> , <code>-</code>	binární
5	<code>&lt;</code> , <code>&lt;=</code> , <code>&gt;</code> , <code>&gt;=</code>	
6	<code>==</code> , <code>!=</code>	

Pozn:

`**` - Exponentiation

`//` - floor division - zaokrouhlí výsledek dělení na nejbližší celé číslo směrem dolů

`%` - modulus (zbytek po celočíselném dělení)

## LAB

### Odhadovaný čas

5-10 minut

## Úroveň obtížnosti

Velmi snadné

## Cíle

seznámení se s funkcí `input()`; seznámení se s operátory porovnávání v jazyce Python.

## Scénář

Pomocí některého z operátorů porovnávání v jazyce Python napište jednoduchý dvouřádkový program, který jako vstup přijme parametr `n`, což je celé číslo, a vypíše `False`, pokud je `n` menší než 100, a `True`, pokud je `n` větší nebo rovno 100.

Nevytvářejte žádné bloky `if` (o nich budeme hovořit velmi brzy). Otestujte svůj kód pomocí dat, která jsme vám poskytli.

## Testovací data

Sample input: `55`

Expected output: `False`

Sample input: `99`

Expected output: `False`

Sample input: `100`

Expected output: `True`

Sample input: `101`

Expected output: `True`

Sample input: `-5`

Expected output: `False`

Sample input: `+123`

Expected output: `True`

## Odkazy:

Cisco Programming Essentials in Python

Root.cz

ITNetwork.cz

Internet