



Jak psát krásný kód v Pythonu: Průvodce PEP 8

Od logiky k čistému a čitelnému stylu

```
sends = [  
def func_sends(oo):  
    with no  
        sor2s = 0  
        for i in rangee):  
            $ar1tv = $or1tv  
            return parlex(ob)
```

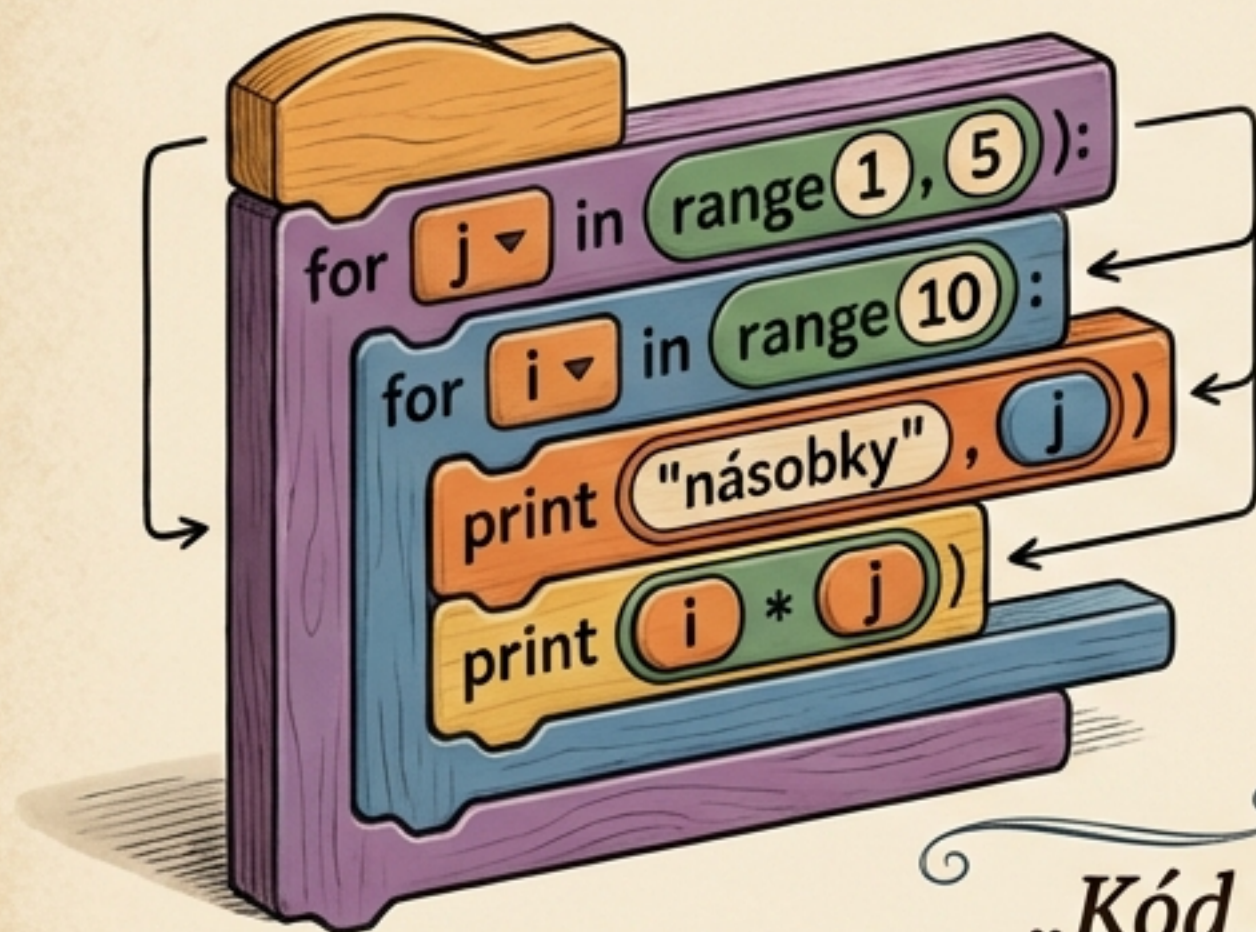
```
return 0
```

Based on Python Enhancement Proposal 8

Proč na stylu záleží?

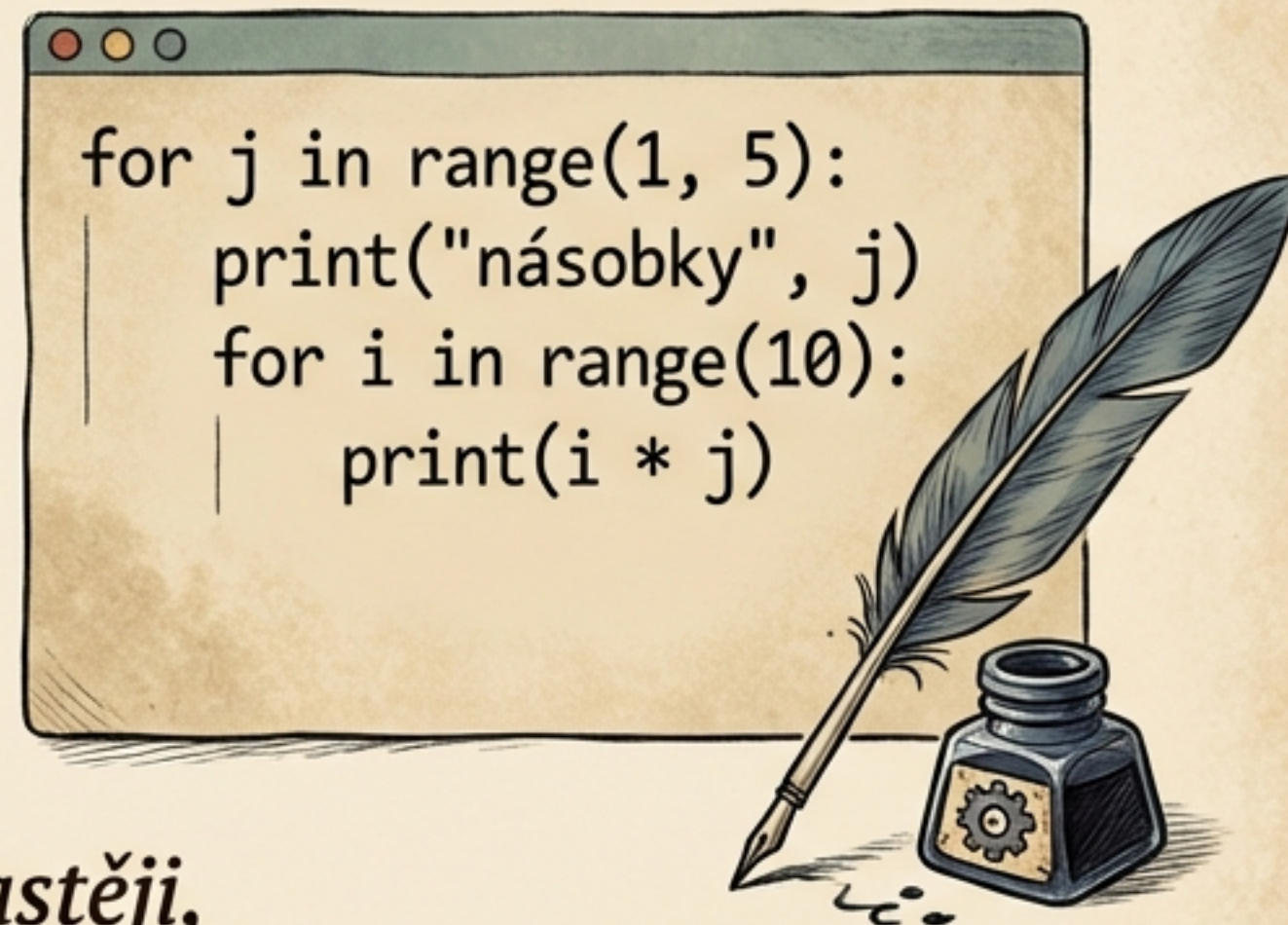
Logika (Blockly)

Editor hlídá tvary a barvy.



Python (Text)

Struktura je na vás.



*„Kód se čte mnohem častěji,
než se píše.“ — Guido van Rossum*

- Blockly/Edu Blocks: Editor hlídá tvary a barvy.
- 🔑 Python: Struktura je na vás.
- 🔧 Čitelný kód = Snadnější ladění a méně chyb.

Pojmenování: Proměnné a funkce

Používejte snake_case (malá písmena s podtržítky)

SPRÁVNĚ ✓

```
pocet_bodu = 100  
jmeno_studenta = 'Jan'  
  
def vypocti_prumer():
```

ŠPATNĚ ✗

```
pocetBodu = 100 # CamelCase je pro třídy  
PocetBodu = 100  
JmenoStudenta = 'Jan'
```


Volba jmen: Buďte konkrétní

Explicitní je lepší než implicitní

 `s = 'Jan'`

Student?
Sobota?
Škola?

NEJASNÉ / AMBIGUOUS

 `jmeno_studenta = 'Jan'` 

JASNÉ / CLEAR



Nepoužívejte v názvech diakritiku (č, ř, ž)!

Vyhněte se znakům:
l (malé L), O (velké O),
I (velké i)

Pojmenování: Třídy a Konstanty

Třídy (Classes)

Používejte CapWords
(CamelCase).

```
class ModelAuta:  
    pass  
class TridaZaku:  
    pass
```

Konstanty (Constants)

Používejte VELKÁ_PÍSMENA
(UPPER_CASE).

```
PI = 3.14159  
MAX_POKUS = 3  
NAZEV_SKOLY = 'Gymnázium'
```


Odsazení: Páteř vašeho kódu

```
if x > 5:
```

```
    ....print('Větší než 5')    # ← Tady jsou 4 mezery
```

4 mezery

- ✓ Vždy používejte 4 mezery.
- ✗ Nikdy nepoužívejte tabulátory.
- 💡 Odsazení určuje logické bloky.

Délka řádku a zalamování

Limit 79 znaků

```
seznam_jmen = [  
    'Anna', 'Barbora', 'Cecilie',  
    'Dana', 'Eva', 'Františka'  
]
```

- ☒ Dlouhé řádky se špatně čtou.
- ☒ Zalamujte logicky uvnitř závorek ().

Bílý znak: Vzduch pro váš kód

Oddělujte operátory mezerami

✗ Stísněné

vysledek=(a+b)*c

✓ Čitelné

vysledek = (a + b) * c

Priorita operací

y = x*2 - 1

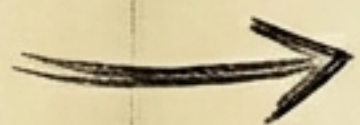
Žádné mezery u
vyšší priority (*)

Mezery u nižší priority (-)

Kde mezery naopak nedělat

1. Uvnitř závorek:

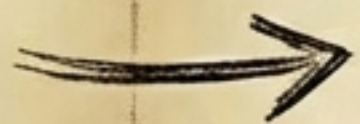
~~×~~ [~~×~~1, ~~×~~2]



✓ [1, 2]

2. Před čárkou:

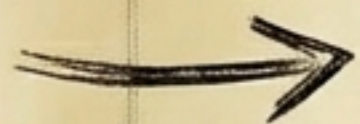
~~×~~ print(x~~×~~, y)



✓ print(x, y)

3. Před závorkou funkce:

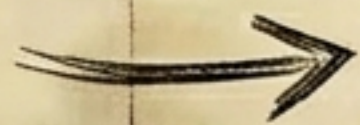
~~×~~ funkce~~×~~(arg)



✓ funkce(arg)

4. Kolem = u parametrů:

~~×~~ def f(a~~×~~=5)



✓ def f(a=5)

Organizace importů

✓ SPRÁVNĚ

Standardní knihovny

```
import os  
import sys
```

Knihovny třetích stran

```
import numpy  
import pandas
```

Lokální moduly

```
import moje_funkce
```

✗ ŠPATNĚ

```
✗ import os, sys, numpy
```


Komentáře: Vysvětlujte "Proč", ne "Co"

✗ ŠPATNĚ (BAD)

```
x = x + 1 # Zvětši x o 1
```

✗ Zbytečné (to vidíme)

✓ SPRÁVNĚ (GOOD)

```
x = x + 1 # Kompenzace pro okraje
```

✓ Užitečné (vysvětluje důvod)

- Blokové komentáře: Pro složitou logiku.
- Docstringy: Dokumentace funkcí na prvním řádku.

Programátorská doporučení

Pište "Pythonic" kód

✗ Neelegantní

Porovnání s None

```
if x == None:
```

Prázdný seznam

```
if len(seznam) > 0:
```

Boolean

```
if active == True:
```

✓ Pythonic

```
if x is None:
```

```
if seznam:
```

```
if active:
```


Časté chyby začátečníků

1. ☒ ~~pocetBodu~~ → ☒ pocet_bodu (CamelCase vs snake_case)
2. ☒ ~~x=5+3~~ → ☒ x = 5 + 3 (Chybějící mezery)
3. ☒ ~~p,tmp~~ → ☒ pozice, temp_file (Kryptické názvy)
4. ☒ ~~počet~~ → ☒ pocet (Diakritika)
5. ☒ ~~Importy uprostřed kódu~~ → ☒ Importy na začátku

Nástroje: Nechte počítač pracovat

Pravidla si nemusíte pamatovat nazpaměť



Lintery (Kontrola)

Flake8, Pylint

Podtrhnou chyby v editoru.



Autoformátory (Oprava)

Black, Ruff

Upraví kód při uložení.



VS Code Rozšíření

Nainstalujte si rozšíření a automatizujte to!

Shrnutí: 5 pravidel pro krásný kód

1.

Odsazení: 4 mezery.

```
1 code:
2     sne?
3     cap:
4     ed:
5     ine
```

2.

Jména: snake_case (proměnné), CapWords (třídy).

Příklad: snake_case (proměnné), CapWords (třídy).

3.

Vzduch: Mezery kolem operátorů.

Příklad: `x = ' ' + ' ' + 0`

4.

Struktura: Importy nahoře, řádky do 79 znaků.

Příklad: ...

5.

Význam: Popisné názvy a "Proč" komentáře.

Popisné názvy rak komentáře vreněim, plamute "Proč" sounázvoy.

Simple is better than complex. — The Zen of Python