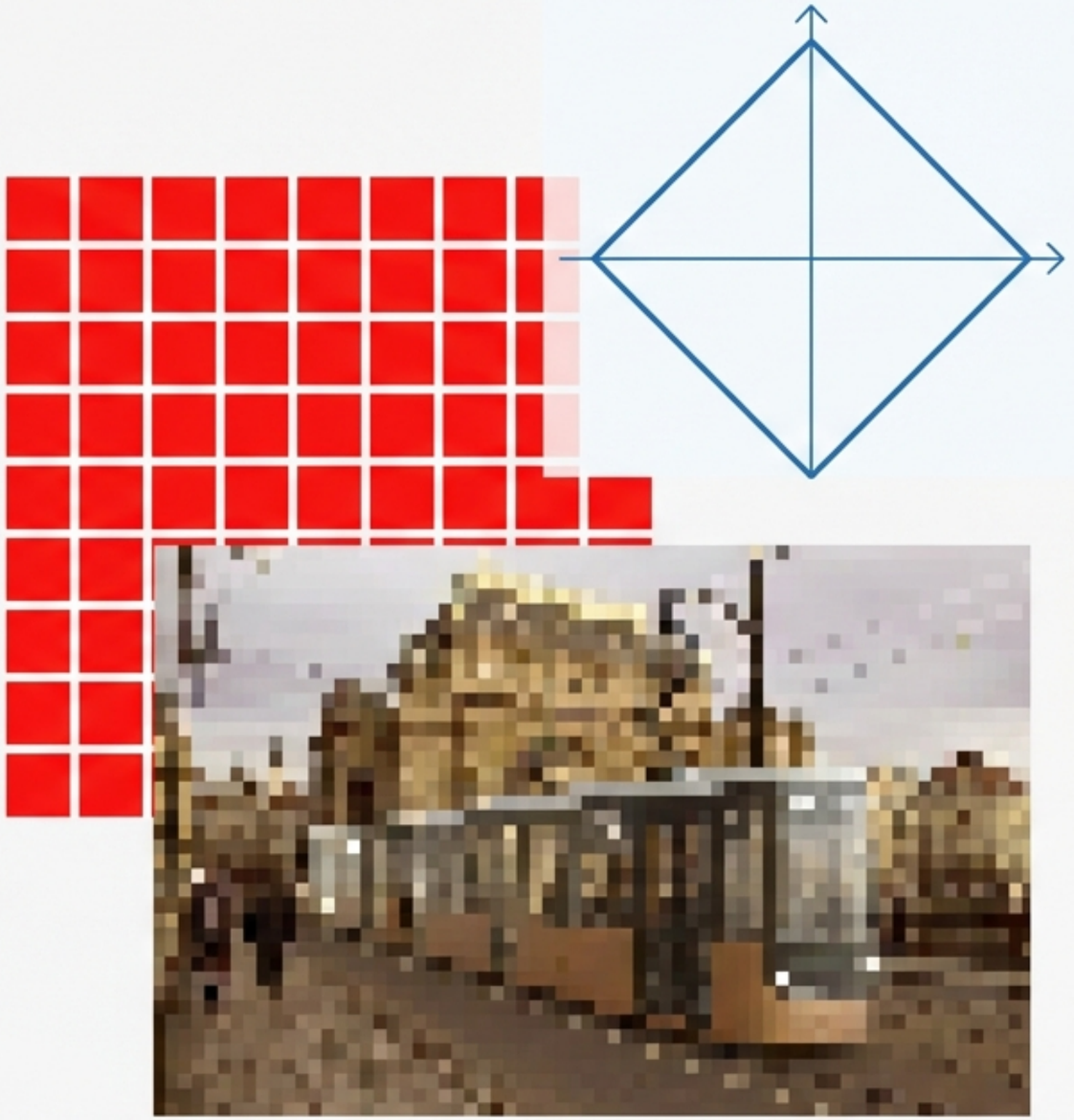
Bitmapová grafika a Pillow (PIL)

Průvodce kreslením pixel po pixelu a zpracováním obrazu v Pythonu.



Praktický workbook pro programování grafiky.

Co je Pillow a jak začít

- **Pillow (PIL)**: Standardní knihovna v Pythonu pro manipulaci s obrázky.
- Instalace:

```
pip install Pillow
```

- Import:

```
from PIL import Image
```

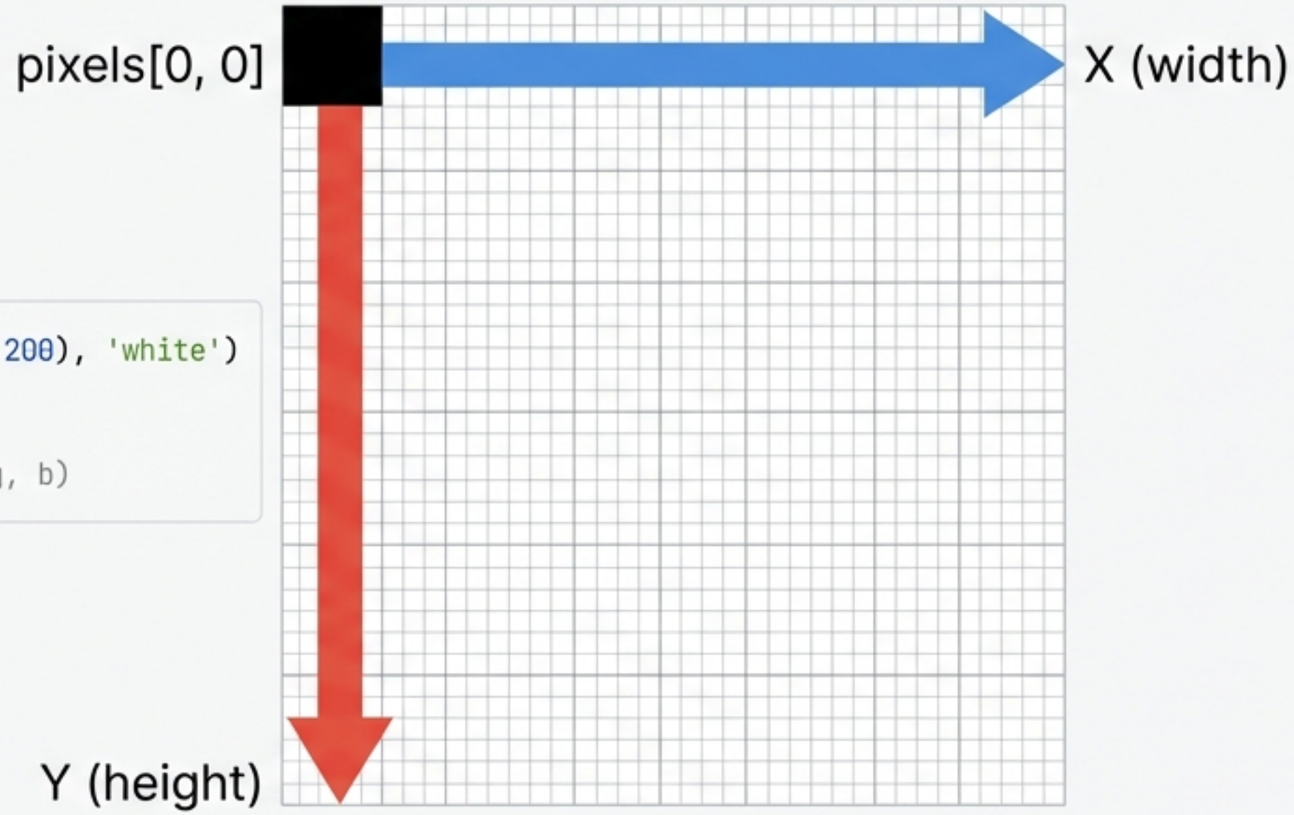
Režimy (Modes):

- **RGB**: Barevný (Red, Green, Blue). Nejčastější.
- **L**: Stupně šedi (0–255).
- **RGBA**: Barevný s průhledností (Alpha).

Plátno a souřadnice

```
img = Image.new('RGB', (200, 200), 'white')
pixels = img.load()

# Zápis: pixels[x, y] = (r, g, b)
```



⚠️ **Pozor:** Souřadnice jsou [x, y]. Počátek [0, 0] je vlevo nahoře.

Kreslíme čtverce: Vnořené cykly

Pro plošné kreslení používáme dva cykly: vnější pro sloupce (x), vnitřní pro řádky (y).

Vyplněný čtverec

```
for x in range(25, 75):
    for y in range(25, 75):
        pixels[x, y] = (0, 0, 255)
```

Prázdný rám

```
if x == 25 or x == 74 or y == 25…
    pixels[x, y] = (0, 0, 255)
```

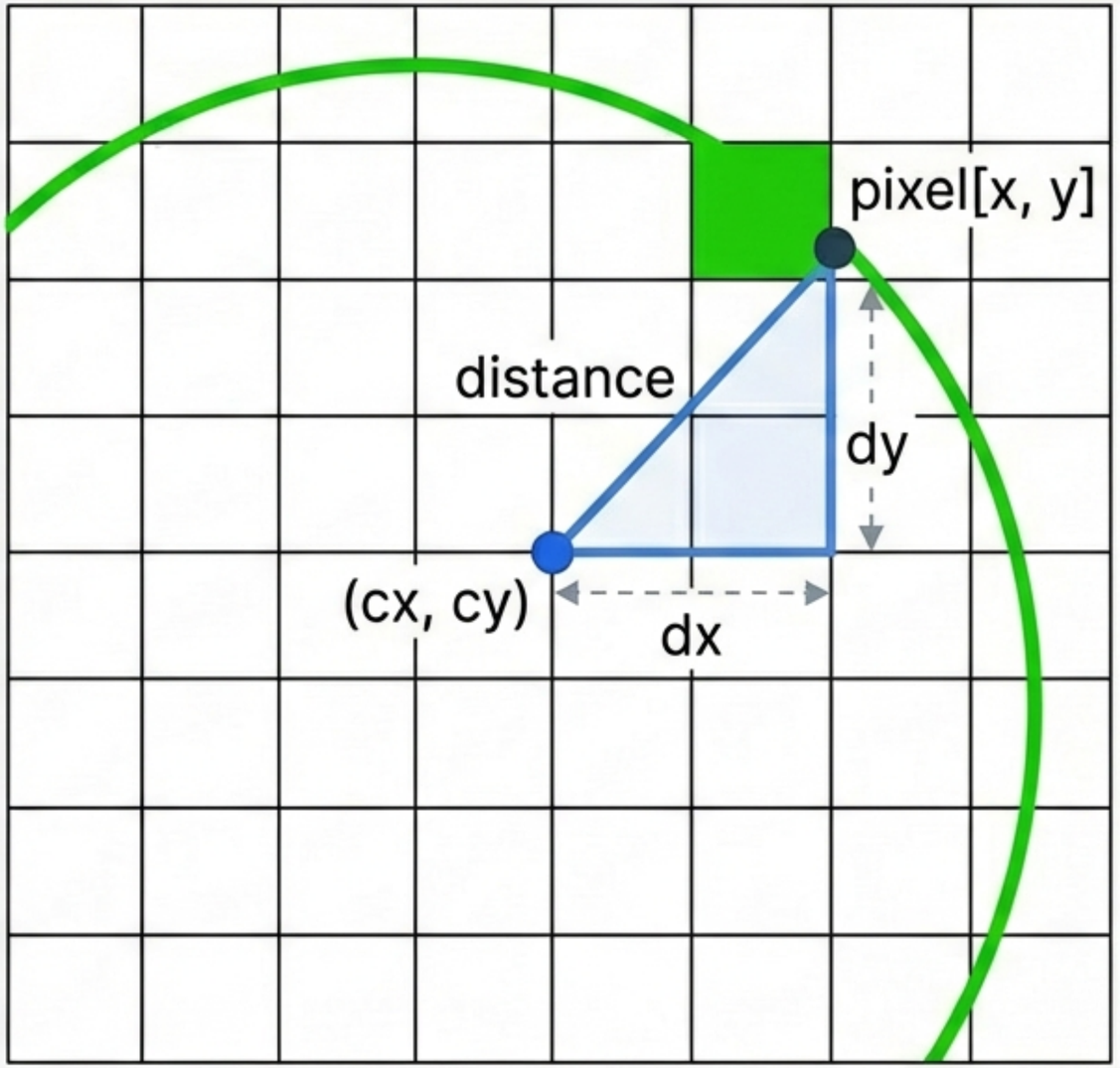
`if` podmínka uvnitř cyklu určuje tvar.

Matematika kruhu: Euklidovská vzdálenost

Pro každý pixel spočítáme vzdálenost od středu (cx, cy). Pokud je menší než poloměr, pixel obarvíme.

$$distance = \sqrt{(x - cx)^2 + (y - cy)^2}$$

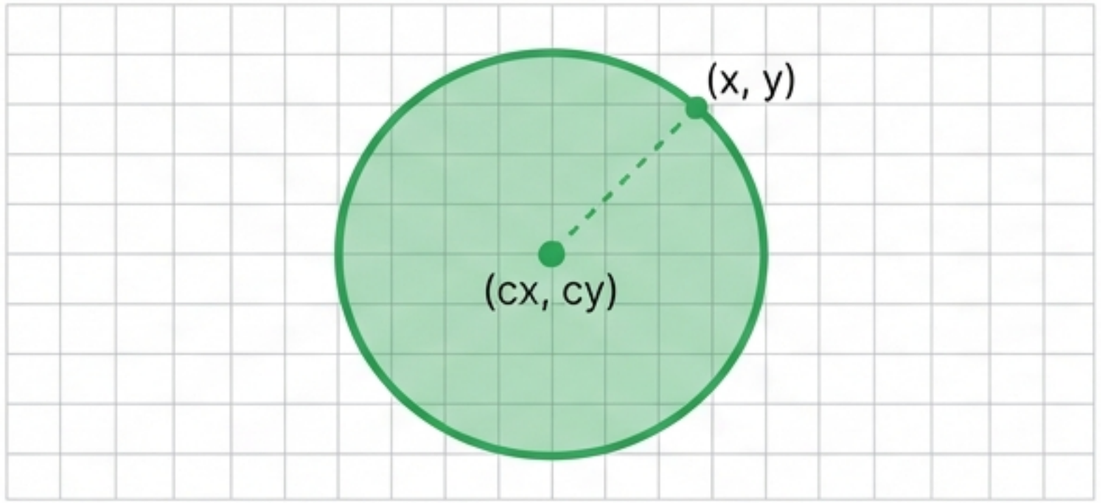
```
dx = x - cx
dy = y - cy
dy = y - cy
dist = (dx*dx + dy*dy)**0.5
if dist <= radius:
    pixels[x, y] = (0, 255, 0)
```



Alternativní tvary: Diamant

Manhattan Distance (Taxicab geometry)

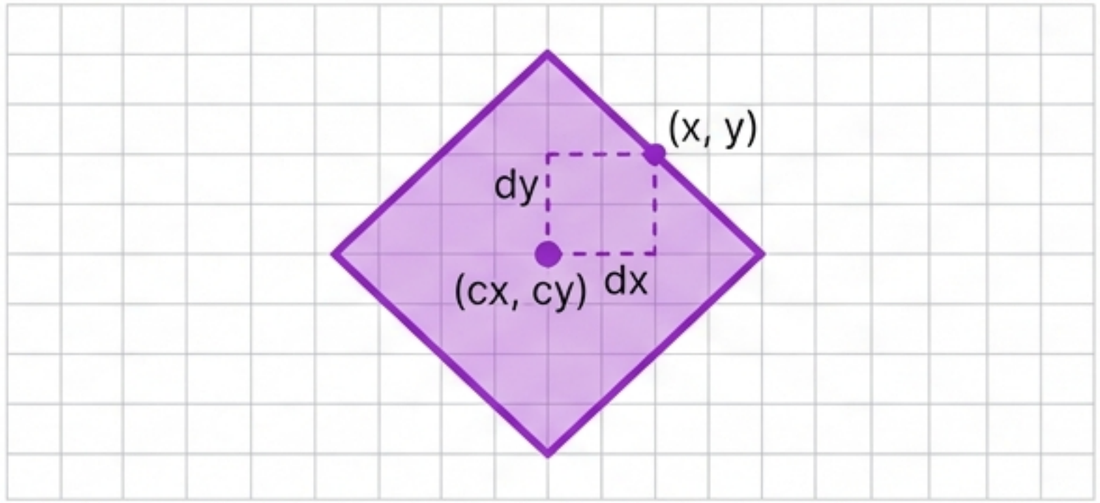
Euklidovská (Kruh)



$$dist = \sqrt{((x-cx)^2 + (y-cy)^2}$$

Vzdušná čára

Manhattan (Diamant)




```
dist = abs(x - cx) + abs(y - cy)
```

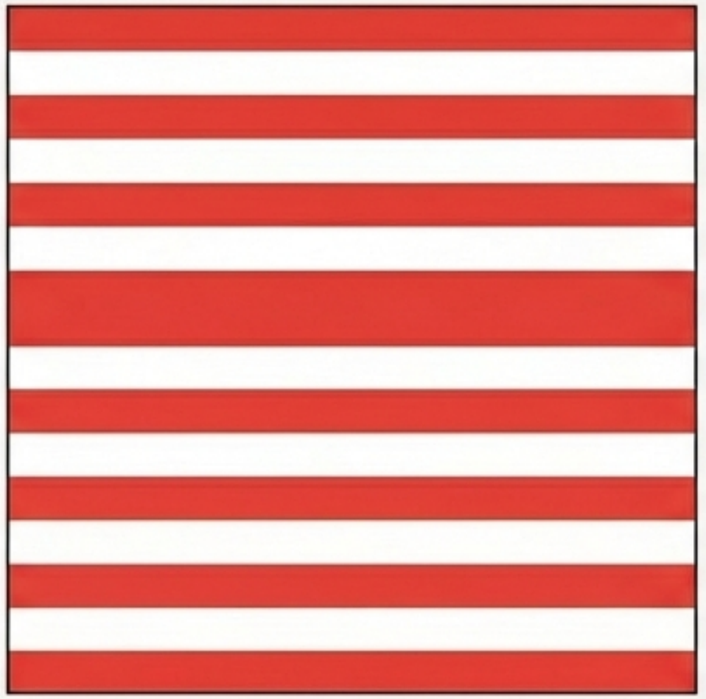
Součet absolutních rozdílů

```
if (abs(x - cx) + abs(y - cy)) <= radius:
    pixels[x, y] = (255, 0, 255)
```

Algoritmické vzory: Pruhy a šachovnice

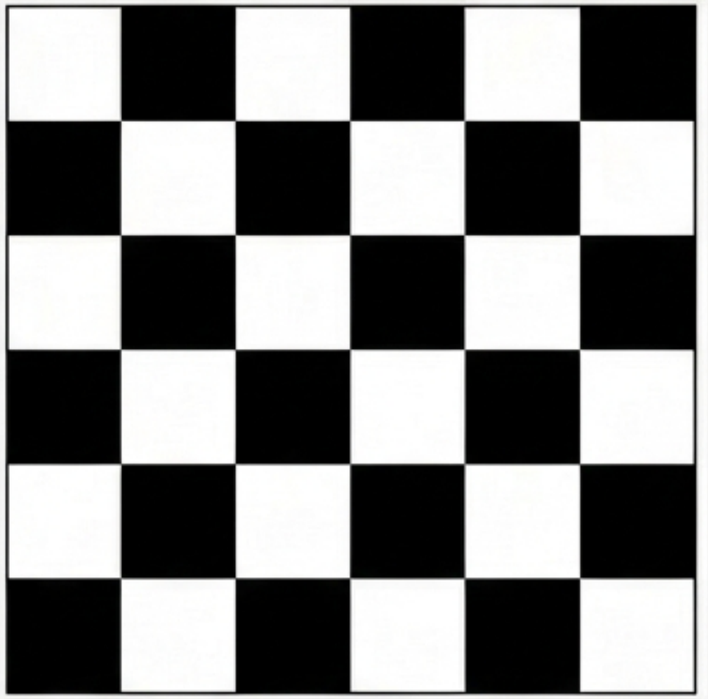
Operátor % (zbytek po dělení) umožňuje střídat barvy.

Pruhy (Stripes)



```
if (y // 20) % 2 == 0:
    # Sudý pruh -> Barva A
```

Šachovnice (Checkerboard)



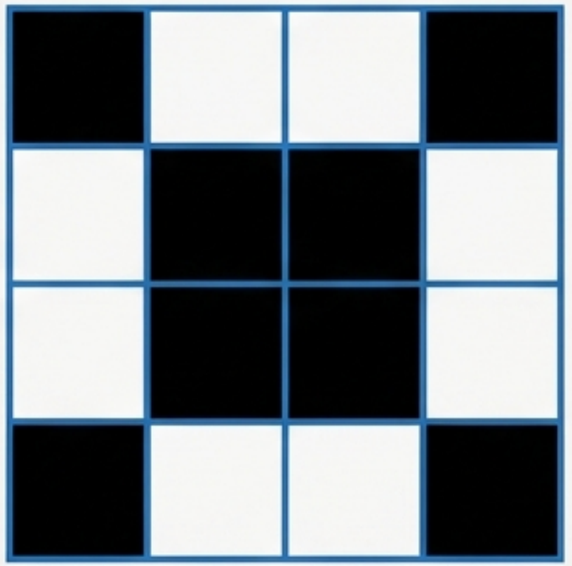
```
if (x // 20 + y // 20) % 2 == 0:
    # Sudé políčko -> Barva A
```

Od dat k obrazu (ASCII Art)

Obrázek můžeme definovat jako seznam řetězců.

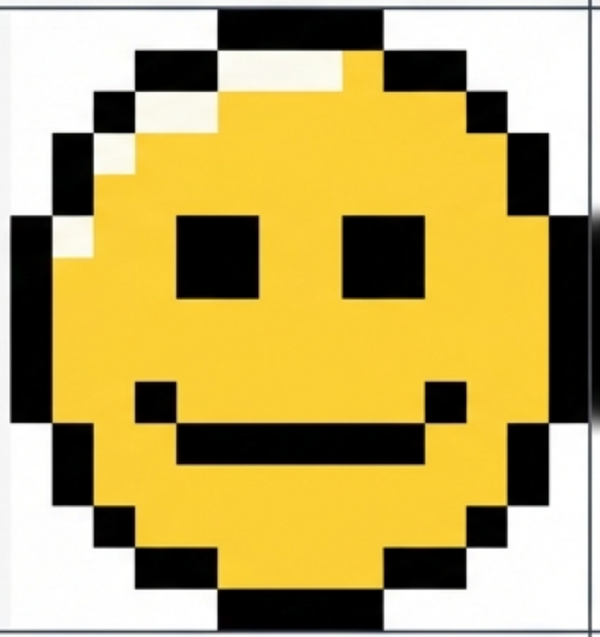
```
data = [
    'X..X',
    '.XX.',
    'X..X'
]
```

```
for y in rows:
    for x in chars:
        if char == 'X':
            Draw Black
```



Zvětšování a Pixel Art

Malé obrázky (8×8) musíme zvětšit. Záleží na metodě!

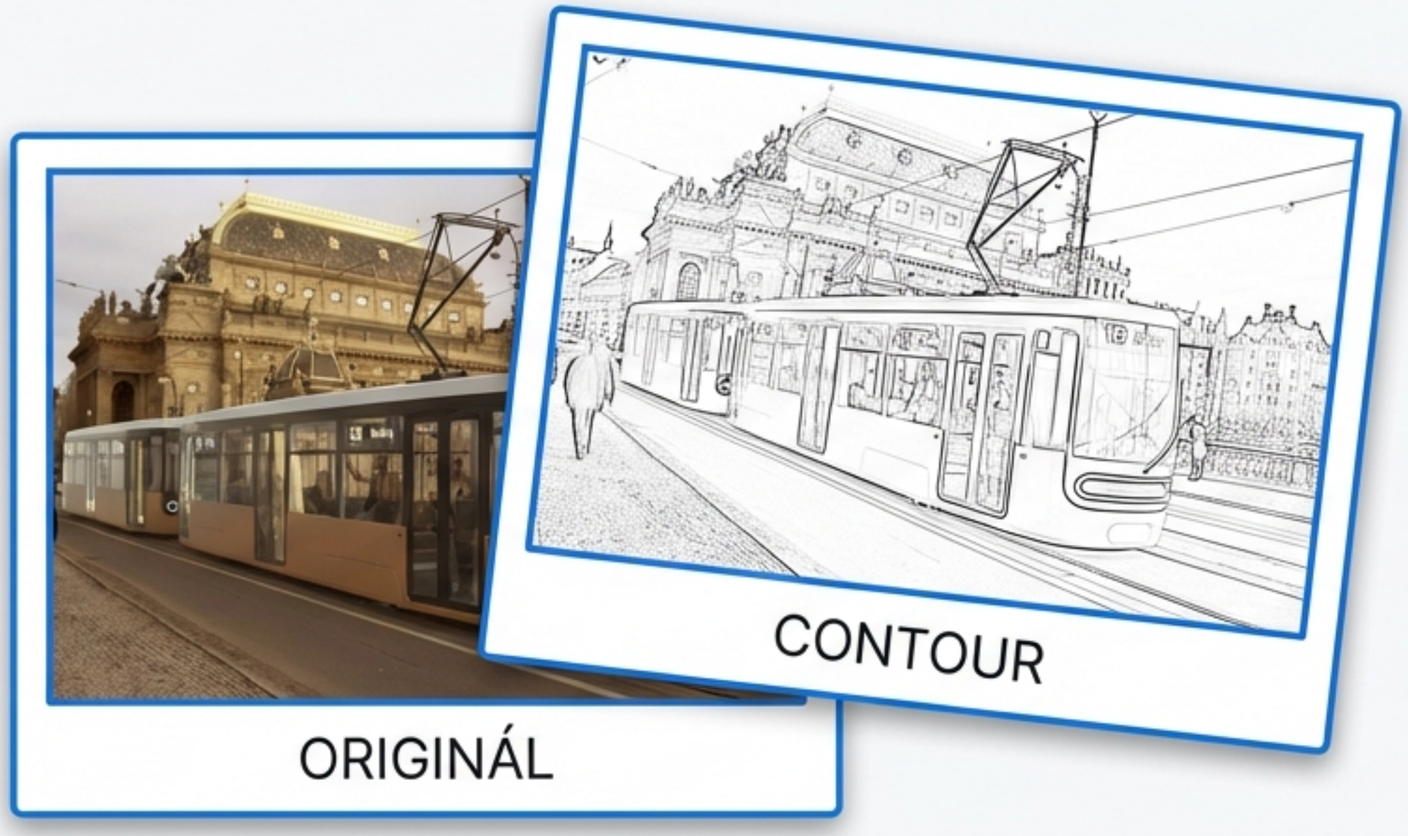
`Image.NEAREST`	`Image.BILINEAR`
	
Zachová ostré hrany (Pixel Art)	Rozmaže hrany (Fotky)

```
img.resize((big_w, big_h), resample=Image.NEAREST)
```

Filtry a úprava fotografií

Aplikace filtrů pomocí ImageFilter.

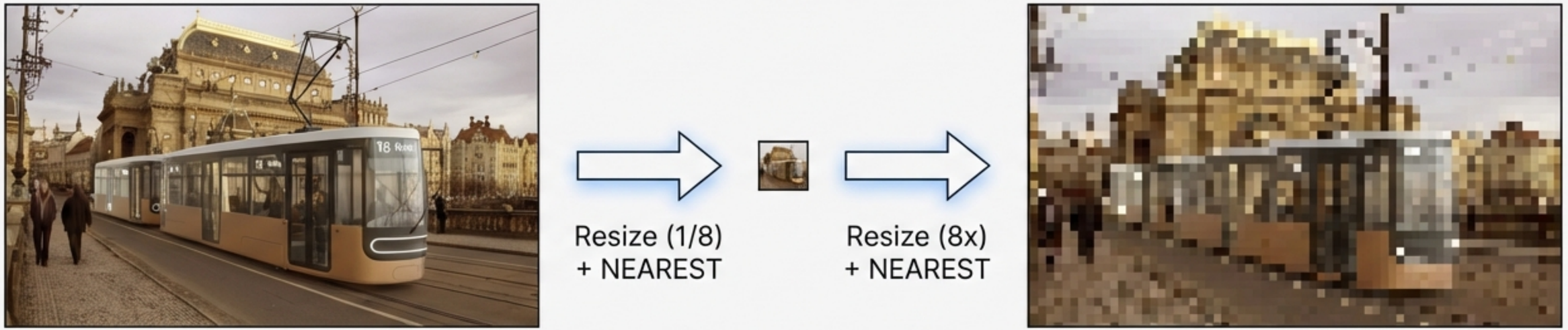
```
img = Image.open('tramvaj.jpg')
img.filter(ImageFilter.CONTOUR)
```



- BLUR
- SHARPEN
- CONTOUR
- EMBOSS

Efekt 'Minecraft' (Pixelizace)

Algoritmus: Zmenšit (zahodit detaily) -> Zvětšit (zachovat hrubost).




```
img.resize((small, small), resample=Image.NEAREST)
img.resize((original, original), resample=Image.NEAREST)
```

Práce s barvami

Základní paleta

`(0, 0, 0)`	:	Černá
`(255, 255, 255)`	:	Bílá
`(255, 0, 0)`	:	Červená
`(255, 255, 0)`	:	Žlutá

Programovaný gradient

Barva vypočítaná ze souřadnice.

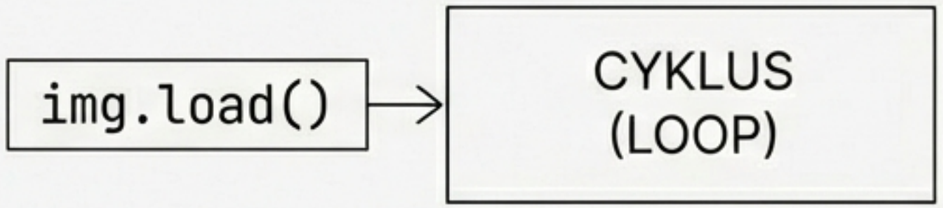
```
pixels[x, y] = (x, 0, 0)
```

Optimalizace a tipy

Rychlost (Performance)

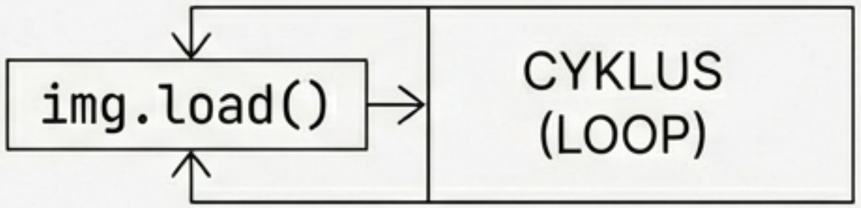
✅ Správně:

`pixels = img.load()` před cyklem.

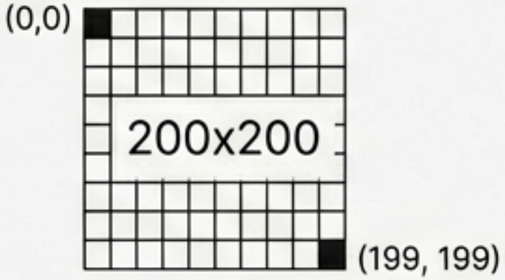


❌ Špatně:

Volání `img.load()` uvnitř cyklu.



Bezpečnost (Safety)



Indexy jdou 0 až 199.

```
if 0 <= x < width and 0 <= y < height:
    # Kresli bezpečně
```

Galerie vzorců (Cheat Sheet)

Kruh

```
sqrt((x-cx)**2 + (y-cy)**2) <= r
```

Diamant

```
abs(x-cx) + abs(y-cy) <= r
```

Pruhy

```
(y // vyska) % 2 == 0
```

Pryhy

```
(y // vyska) % 2 == 0
```

Šachovnice

```
(x // vel + y // vel) % 2 == 0
```

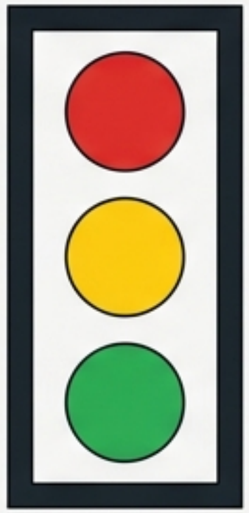
Čtverec

```
x1 <= x <= x2 and y1 <= y <= y2
```

Výzvy pro vás

Experimentujte s matematikou a objevujte nové vzory.

Semafor

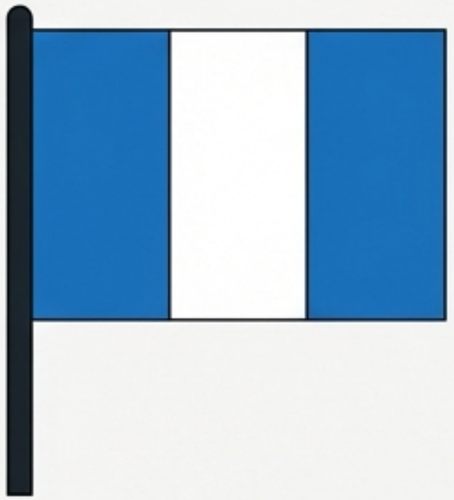


Tři kruhy pod sebou.

Terč

Kružnice v kružnici.

Vlajka



Svislé pruhy.

Hvězda

Kombinace tvarů.