Zpracování obrázků

Pillow

Pro práci s obrázky budeme používat knihovnu Pillow, která vychází z knihovny PIL (Python Imaging Library).

```
pip install pillow
```

Možnosti nabízí mnohé, například konverze mezi formáty (PNG, JPEG, PPM, GIF, TIFF, BMP), kreslení bodů (zejména pro výstup matematických funkcí) a geometrických tvarů, ořezávání, otáčení, změny velikosti, změny barevnosti, rozostření, přidávání textů a další.

Kreslení do obrázku

Následující skript přidá do obrázku rastr z teček. Lze tím vytvořit zajímavé efekty, které běžný software pro práci s rastry nenabízí.

```
from PIL import Image, ImageDraw
import sys

try:
    obrázek = Image.open("C:\\temp\\test.jpg")
    š, v = obrázek.size
    kontext = ImageDraw.Draw(obrázek,'RGBA')
    for x in range(0,š,4):
        for y in range(0,v,4):
            kontext.point([(x, y)], fill=(200,200,200,128))
    del kontext

except IOError:
    print("Nepodařilo se nahrát obrázek")
    sys.exit(1)

obrázek.show()
obrázek.save("C:\\temp\\out.jpg", "JPEG", quality=80,\
optimize=True, progressive=True)
```

Rastr vykreslujeme bod po bodu metodou Draw. Režim je zvolený RGB s alfa kanálem pro práci s průhledností.
Ve dvou cyklech for, procházíme souřadnice obrázku v rozsahu od nuly do šířky, resp. výšky v krocích po čtyřech. Vložení bodu do kontextu provádí metoda point(), u které je prvním parametrem souřadnice bodu a druhým barva. V tomto případě zadáváme barvu jako R, G, B, A, kde A je alfa kanál, čili průhlednost.
Pokud bychom obrázek otevřeli v režimu RGB a nikoliv RGBA, mohli bychom barvu zadat i ve formátu používaném na webu, tzn. fill='# c8c8c8'.

Co kdyby zdrojový obrázek byl indexovaný png? Museli bychom jej zkonvertovat na RGBA funkcí convert('RGBA'), kterou bychom navázali na metodu open(), tzn.:

```
obrázek = Image.open("C:\\temp\\test.png").convert('RGBA')
```

Odstranění metadat z fotografií (smazání EXIF informací)

Fotoaparáty generují spousty zbytečných dat a ukládají je spolu s obrazovými daty. EXIF navíc prozradí řadu informací, které třeba nechcete zveřejňovat (např. že vlastníte velmi drahý přístroj, jaká byla při focení nastavená clona a čas a řadu dalších věcí).

EXIF dat se však lze snadno zbavit přeuložením pomocí Pillow.

Soubor bude potom menší a nejen díky likvidaci metadat, ale také díky kompresi (neplatí to obecně; záleží jak je nastavená komprese ve foťáku). Doporučuji v Pillow nastavit kvalitu JPEG na 90. Při nižších hodnotách bývá vidět ztrátová komprese. Záleží samozřejmě na typu fotografie. Jsou-li na fotce texty nebo detaily obličejů, potom to bývá vidět mnohem víc, než třeba na fotografii krajiny.

```
from PIL import Image
import os

os.chdir('C:\\temp\\obrázky\\test')
```

```
soubory = os.listdir()
for f in soubory:
    název_souboru, přípona = os.path.splitext(f)
    if(přípona == '.jpg'):
        obrázek = Image.open(f)
        data = list(obrázek.getdata())
        obrázek_nový = Image.new(obrázek.mode, obrázek.size)
        obrázek_nový.putdata(data)
        soub = 'C:\\temp\\obrázky\\out\\'+obrázek.filename
        obrázek_nový.save(soub, 'JPEG', quality=90)
        print(soub)
```

Linedraw – vektory z bitmapy

Některé moduly se nedají nainstalovat pomocí pip utility. Musíte stáhnout soubory ručně. Většinou je stačí umístit do pracovního adresáře. Zde například máme nadějný modul Linedraw pro zpracování obrazu, který generuje
z obrázku vektorové šrafy.

https://github.com/LingDong-/linedraw

Rozběhat Linedraw může být trochu komplikovanější. Pokud však ještě nemáte v počítači nainstalované moduly, které tento modul používá (pillow, filters, numpy, opencv-python), budou se při pokusech o spuštění objevovat chybové zprávy. Z hlášení překladače a ze zdrojových souborů však u podobných modulů vydedukujete, co je zapotřebí doinstalovat a program nakonec většinou spustíte.

Linedraw není nijak závratný, ale má potenciál. Hodně se naučíte studováním kódu jiných programátorů a testováním. Nebojte se přemýšlet nad cizím kódem, i když působí na první pohled velmi složitě.

Linedraw otestujte z příkazového řádku vašeho operačního systému:

linedraw.py -i statue.png -o test.svg

-i … vstupní soubor
-o … výstupní soubor
-nh … no hatching = bez šrafování
-nc … no contour = bez kontur
--contour_simplify … zjednodušení kontur, např. 1, 2, 3
--hatch_size … délka šrafů … 8, 16, 32

Na obrázku níže je vlevo zdrojový obrázek (byl to png soubor) a vpravo výstup z Linedraw se zapnutým šrafováním a konturami.
Výsledkem je vektorový SVG soubor, který můžete zobrazit třeba v prohlížeči (stačí jej přetáhnout do prohlížeče).

Vektorový obrázek má tu skvělou vlastnost, že je definován matematicky křivkami, takže jej lze libovolně zvětšit a bude stále stejně kvalitní.

Vizualizace dat

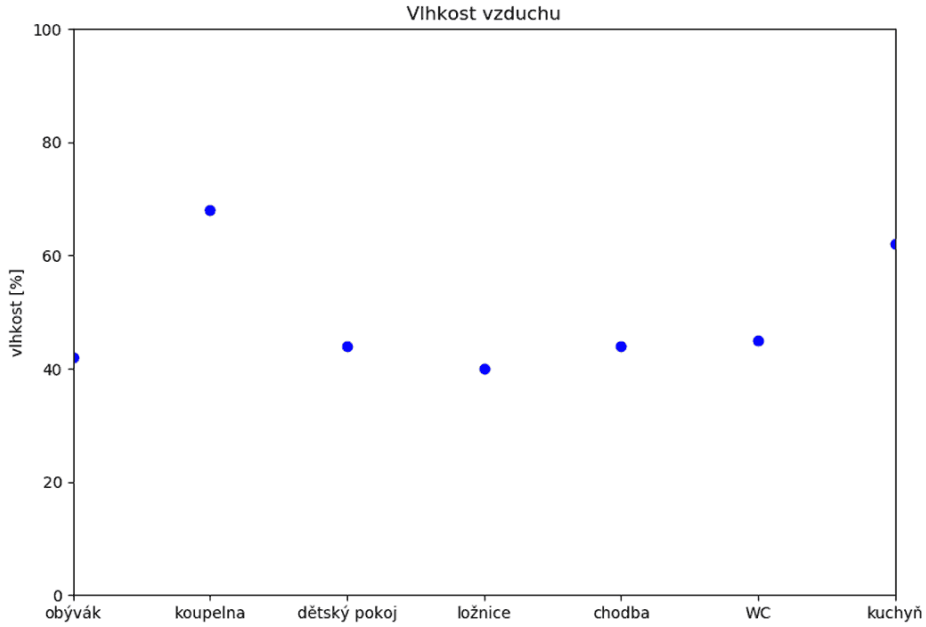
MatPlotLib

Každému je jasné, že vyznat se v datech jen pomocí pročítání čísel je pro člověka krajně vyčerpávající a vlastně nemožné. Musíme data transformovat do grafů. Používat k tomu budeme populární MatPlotLib:

```
pip install matplotlib
```

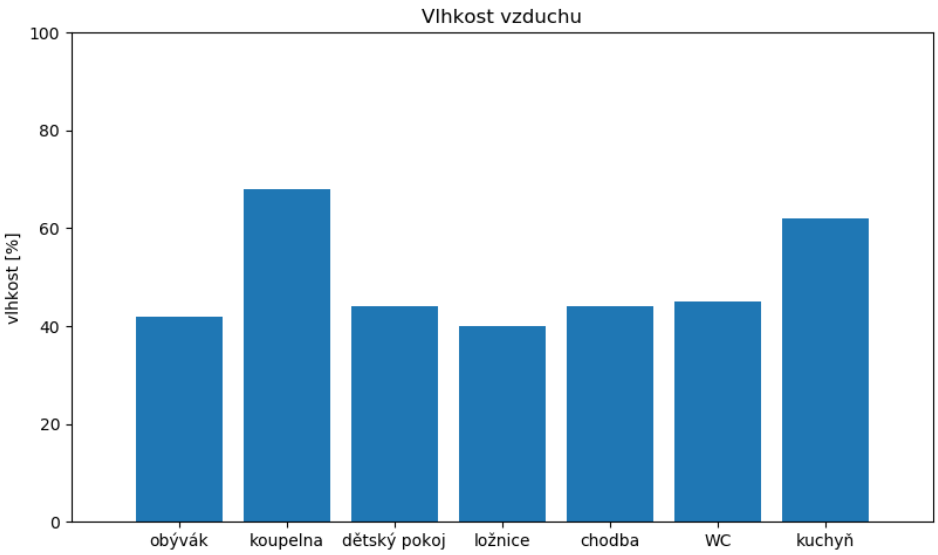
Nejprve však zobrazíme jednoduchá data, ve kterých bychom se vyznali i bez grafů. Třetí parametr ve funkci plot() říká, že chceme diskrétní hodnoty v modré barvě. ro by byly červené tečky, g* zelené hvězdičky, ys žluté čtverečky, k-- černé čárky, g^ zelené trojúhelníky apod. Pokud parametr vynecháme, vykreslí se mezi body spojnice, což by pro tento případ bylo nevhodné.

```
import matplotlib.pyplot as plt

plt.plot(['obývák', 'koupelna', 'dětský pokoj', 'ložnice', \
  'chodba', 'WC', 'kuchyň'], [42,68,44,40,44, 45, 62], 'bo')
plt.title('Vlhkost vzduchu')
plt.xlabel('místnost')
plt.ylabel('vlhkost [%]')
plt.xlim(0, 6)
plt.ylim(0, 100)
plt.show()
```



Trochu přehledněji můžeme data zobrazit jako sloupcový graf.

```
plt.bar(['obývák', 'koupelna', 'dětský pokoj', 'ložnice', \
         'chodba', 'WC', 'kuchyň'], [42,68,44,40,44,45,62])

plt.title('Vlhkost vzduchu')
plt.ylabel('vlhkost [%]')
plt.xlim(-1, 7)
plt.ylim(0, 100)
plt.show()
```



Na ose x jsme protáhli limitní body o jeden vlevo a vpravo aby graf na okrajích nebyl oříznutý.

```
pip install pandas
```

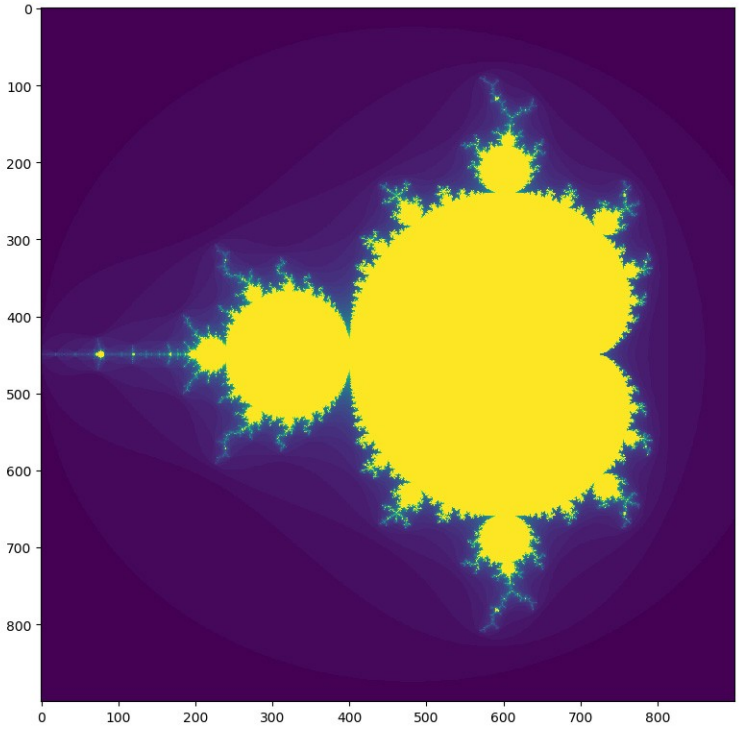
Fraktály

Fraktály mohou být děsivé. Na videu https://youtu.be/PD2XgQOyCCk se podívejte na nekonečný průlet komplexní rovinou. (Video není vhodné pro osoby s epilepsií.) Fraktály mají sice i různá praktická využití, např. k simulaci přírodních struktur a tvarů (zejm. stromů), protože připomínají v určitých detailech přírodní statistické rozložení hmoty, ale zde je uvádím jen pro zajímavost, jak lze snadno zobrazit Mandelbrotovu množinu bodů komplexní roviny.

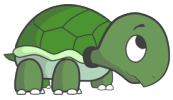
```
import numpy
import matplotlib.pyplot as plt
def mandelbrot( h,w, maxit=50 ):

    y,x = numpy.ogrid[ -1.4:1.4:h*1j, -2:0.8:w*1j ]
    c = x+y*1j
```

```
    z = c
    divtime = maxit + numpy.zeros(z.shape, dtype=int)

    for i in range(maxit):
        z = z**2 + c
        diverge = z*numpy.conj(z) > 2**2
        div_now = diverge & (divtime==maxit)
        divtime[div_now] = i
        z[diverge] = 2

    return divtime
plt.imshow(mandelbrot(900,900))
plt.show()
```

https://www.machinelearningplus.com/plots/matplotlib-tutorial-complete-guide-python-plot-examples/



Python pro děti

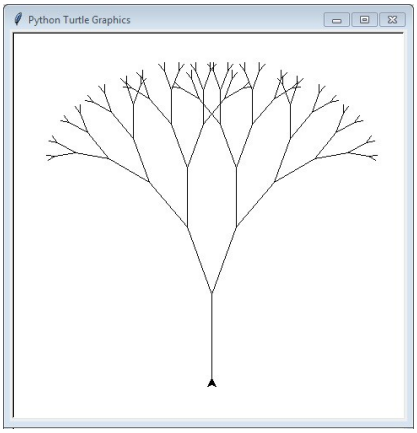


K nalákání dětí na programování lze využít celkem dobře modul turtle a modul pygame. Turtle nemusíte instalovat, je přímo v distribuci Pythonu. Máte-li děti, nebo mladší sourozence, synovce, neteře ... neváhejte je navnadit. Nemusí z nich být programátoři. Stačí, když začnou lépe chápat, jak software funguje a také jim to pomůže více přemýšlet.

V modulu turtle se kurzor při vykreslování obrázků pohybuje pomalu, jako želva, aby děti viděly, jak se vykonávají jednotlivé kroky algoritmu. Rychlost lze samozřejmě nastavit (metodou speed). Přehled *želvích* metod naleznete na stránce https://docs.python.org/3/library/turtle.html .

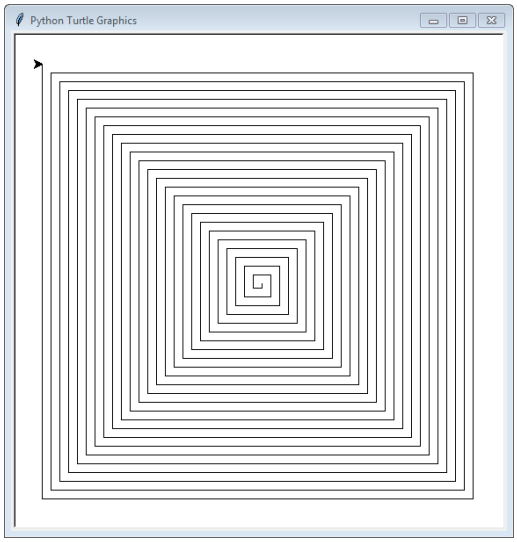
Strom ...

```
import turtle

t = turtle.Turtle()

def strom(délka_větve,t):
    if délka_větve > 2:
        t.forward(délka_větve)
        t.right(20)
        strom(délka_větve-15,t)
        t.left(40)
        strom(délka_větve-15,t)
        t.right(20)
        t.backward(délka_větve)

t.left(90)
t.up()
t.backward(100)
t.down()
strom(100,t)
```



Bludiště ...

```
import turtle

t = turtle.Pen()
t.speed(4)

for x in range(0,500,5):
    t.forward(x)
    t.right(90)
```



Náhodně umístěné kružnice

```
import random
import turtle

t = turtle.Pen()
colors = ["red", "green", "blue", "orange", "black"]
for n in range(5):
    t.pencolor(random.choice(colors))  # náhodný výběr barev
    t.pensize(3)
    velikost = random.randint(30, 40)  # velikost 30 až 40
    x = random.randrange(-turtle.window_width() // 2, \
turtle.window_width() // 2)
    y = random.randrange(-turtle.window_height() // 2, \
turtle.window_height() // 2)
    t.penup()
    t.setpos(x, y)

    for m in range(velikost):
        t.pendown()
        t.circle(m * 10)
        t.penup()
        t.setpos(x, y - m * 10)
```