# NumPy + Matplotlib

December 3, 2019

## 1 NumPy

- de facto standard pro numerické výpočty v Pythonu
- velké množství dalších modulů postavených nad NumPy (SciPy, scikit-learn, pandas, …)

[142]:
```
import numpy as np
```

### 1.1 NumPy pole

- základní objekt, se kterým NumPy pracuje
- pouze prvky stejného typu
- fixní velikost

[143]:
```
np.array([1, 3, 4])
```

[143]:
```
array([1, 3, 4])
```

[144]:
```
np.arange(10)
```

[144]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[145]:
```
np.linspace(0, 1, 11)
```

[145]:
```
array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ])
```

[146]:
```
np.random.sample(10)
```

[146]:
```
array([0.62666803, 0.07045182, 0.20550107, 0.36733577, 0.38851806,
       0.03756064, 0.90113418, 0.6652082 , 0.43018563, 0.6772363 ])
```

### 1.2 Základní operace

[147]:
```
a = np.arange(10)
a
```

[147]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

[148]:
```
len(a)
```

[148]:
```
10
```

Operace se provádějí nad celým polem, není nutné používat `for` cyklus.

[149]:
```
a + 1
```

[149]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

[150]:
```
a ** 2
```

[150]:
```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

### 1.3 Vícerozměrná pole

[151]:
```
a = np.arange(25)
a
```

[151]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

[152]:
```
a.shape
```

[152]:
```
(25,)
```

[153]:
```
b = a.reshape(5, 5)
b
```

[153]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

[154]:
```
b.shape
```

[154]:
```
(5, 5)
```

[155]:
```
b[2, :]
```

[155]:
```
array([10, 11, 12, 13, 14])
```

[156]:
```
b[:, 4]
```

[156]:
```
array([ 4,  9, 14, 19, 24])
```

[157]:
```
b[3:6, 2]
```

[157]:
```
array([17, 22])
```

[158]:
```
np.zeros((3, 3))
```

[158]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

[159]:
```
x = np.ones((3, 3))
x
```

[159]:
```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

[160]:
```
x[:, 1] = 4
x
```

[160]:
```
array([[1., 4., 1.],
       [1., 4., 1.],
       [1., 4., 1.]])
```

[161]:
```
x.T
```

[161]:
```
array([[1., 1., 1.],
       [4., 4., 4.],
       [1., 1., 1.]])
```

[162]:
```
np.eye(3)
```

[162]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

[163]:
```
3 * np.eye(3) + np.arange(9).reshape(3, 3)
```

[163]:
```
array([[ 3.,  1.,  2.],
       [ 3.,  7.,  5.],
       [ 6.,  7., 11.]])
```

### 1.4 Užitečné funkce

[164]:
```
a = np.arange(30).reshape(5, 6)
a
```

[164]:
```
array([[ 0,  1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10, 11],
```

```
       [12, 13, 14, 15, 16, 17],
       [18, 19, 20, 21, 22, 23],
       [24, 25, 26, 27, 28, 29]])
```

[165]:
```
np.min(a), np.max(a), np.sum(a), np.mean(a)
```

[165]:
```
(0, 29, 435, 14.5)
```

Všechny zmíněné funkce mají parametr **axis**, který určuje, zda provést funkci přes řádky nebo sloupce.

[166]:
```
np.sum(a, axis=0)
```

[166]:
```
array([60, 65, 70, 75, 80, 85])
```

[167]:
```
np.sum(a, axis=1)
```

[167]:
```
array([ 15,  51,  87, 123, 159])
```

## 1.5 NumPy a lineární algebra

- `np.linalg`
- velké množství funkcí (determinanty, inverzní matice, vlastní hodnoty, …)

### 1.5.1 Příklad - soustava lineárních rovnic

$$x + y = 1$$
$$2x - y = 2$$

je ekvivalentní:

$$\begin{bmatrix} 1 & 1 \\ 2 & -1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

[168]:
```
A = np.array([[1, 1], [2, -1]])
A
```

[168]:
```
array([[ 1,  1],
       [ 2, -1]])
```

[169]:
```
b = np.array([1, 2])
b
```

[169]:
```
array([1, 2])
```

[170]:
```
np.linalg.solve(A, b)
```

[170]:
```
array([1., 0.])
```

### 1.6 Vizualizace dat - matplotlib

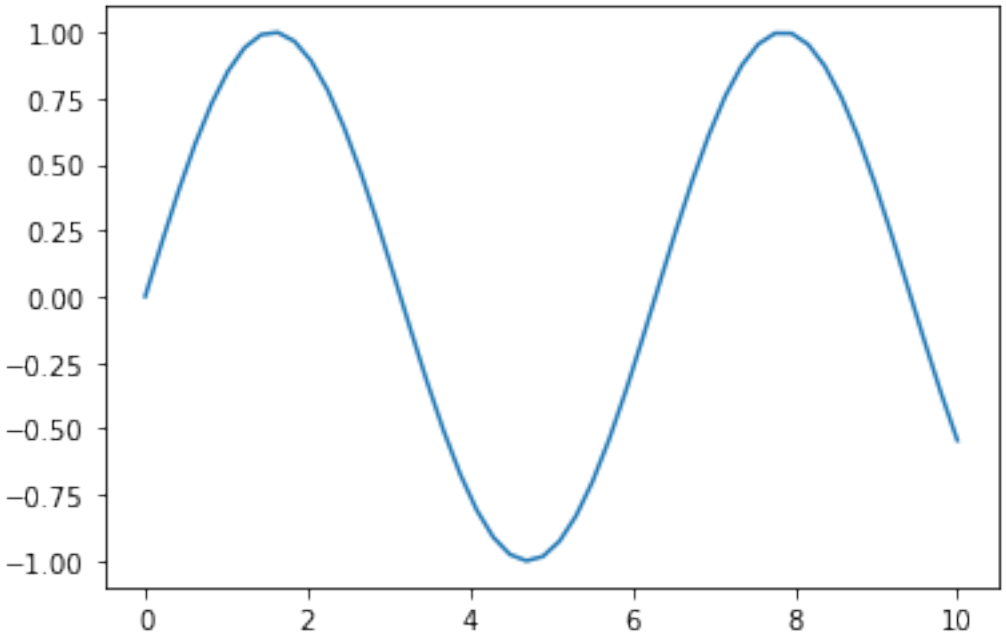
- asi nejrozšířenější modul
- podobná syntaxe jako v Matlabu
- velké možnosti nastavení, typu grafů
- pracuje nad NumPy poli

[171]:
```
import matplotlib.pyplot as plt
```

[172]:
```
xs = np.linspace(0, 10, 50)
```

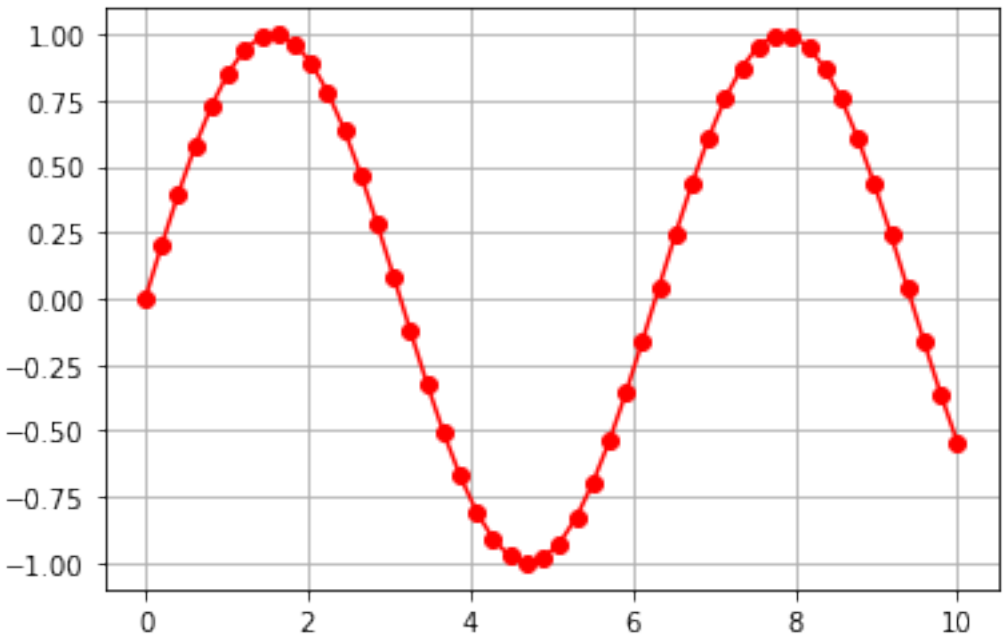
[173]:
```
plt.plot(xs, np.sin(xs))
```

[173]: [<matplotlib.lines.Line2D at 0x7fef25f51250>]



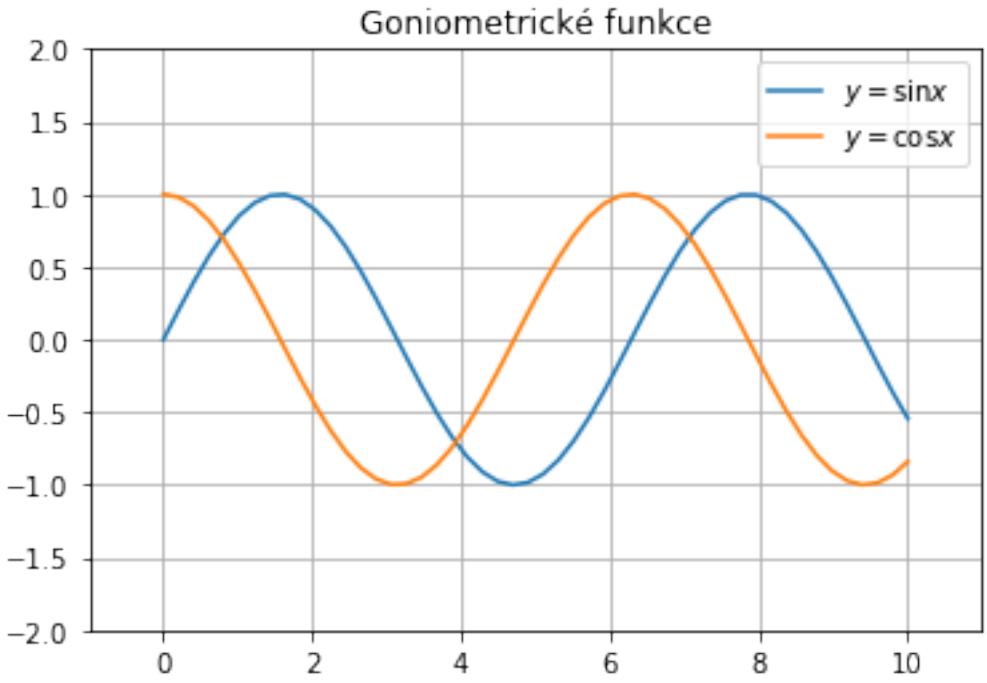

[174]:
```
plt.grid()
plt.plot(xs, np.sin(xs), '-o', color='red')
```

[174]: [<matplotlib.lines.Line2D at 0x7fef25eb2950>]




[175]:
```
plt.grid()
plt.xlim(-1, 11)
plt.ylim(-2, 2)
plt.title('Goniometrické funkce')
plt.plot(xs, np.sin(xs), label='$y = \sin{x}$')
plt.plot(xs, np.cos(xs), label='$y = \cos{x}$')
plt.legend()
```

[175]:
```
<matplotlib.legend.Legend at 0x7fef25e975d0>
```



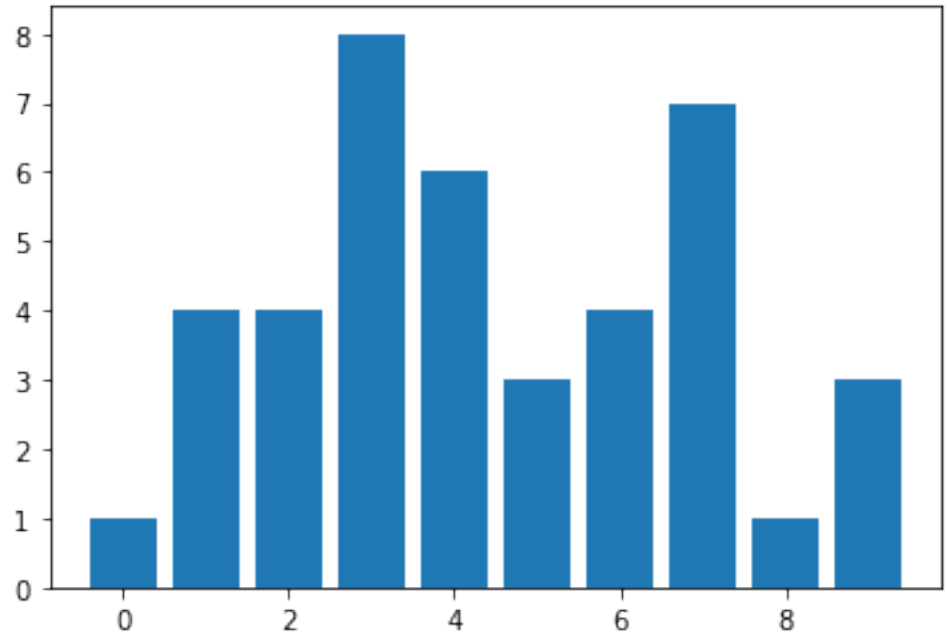

### 1.6.1 Bar plot

[176]:
```
x = np.random.randint(10, size=10)
x
```

[176]:
```
array([1, 4, 4, 8, 6, 3, 4, 7, 1, 3])
```

[177]:
```
plt.bar(np.arange(10), x)
```

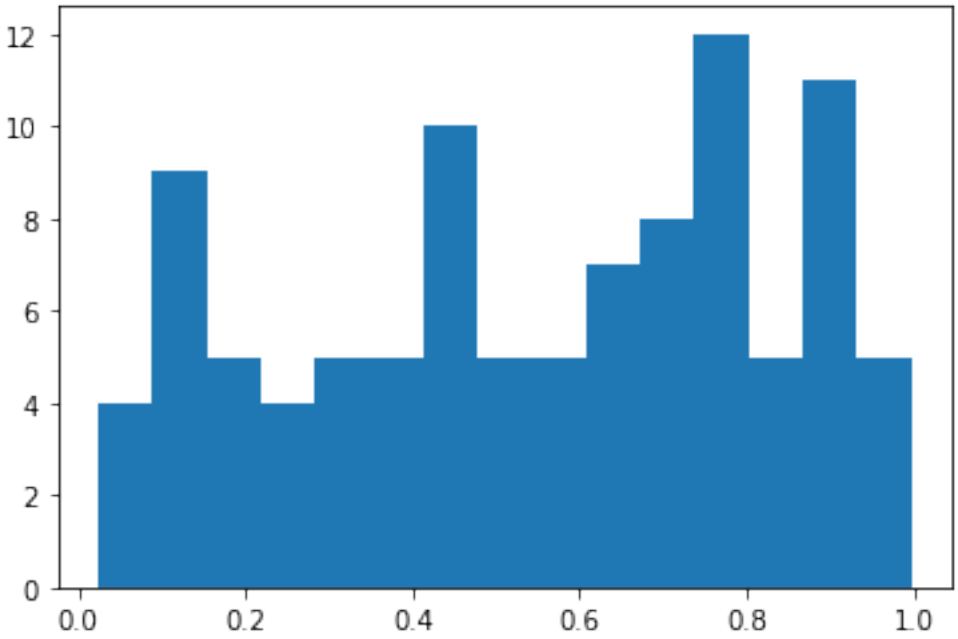
[177]:
```
<BarContainer object of 10 artists>
```




### 1.6.2 Histogram

[178]:
```
plt.hist(np.random.sample(100), bins=15)
```

[178]:
```
(array([ 4.,  9.,  5.,  4.,  5.,  5., 10.,  5.,  5.,  7.,  8., 12.,  5.,
        11.,  5.]),
 array([0.02412496, 0.08896623, 0.1538075 , 0.21864877, 0.28349004,
        0.34833131, 0.41317258, 0.47801385, 0.54285512, 0.60769639,
        0.67253765, 0.73737892, 0.80222019, 0.86706146, 0.93190273,
        0.996744  ]),
 <a list of 15 Patch objects>)
```

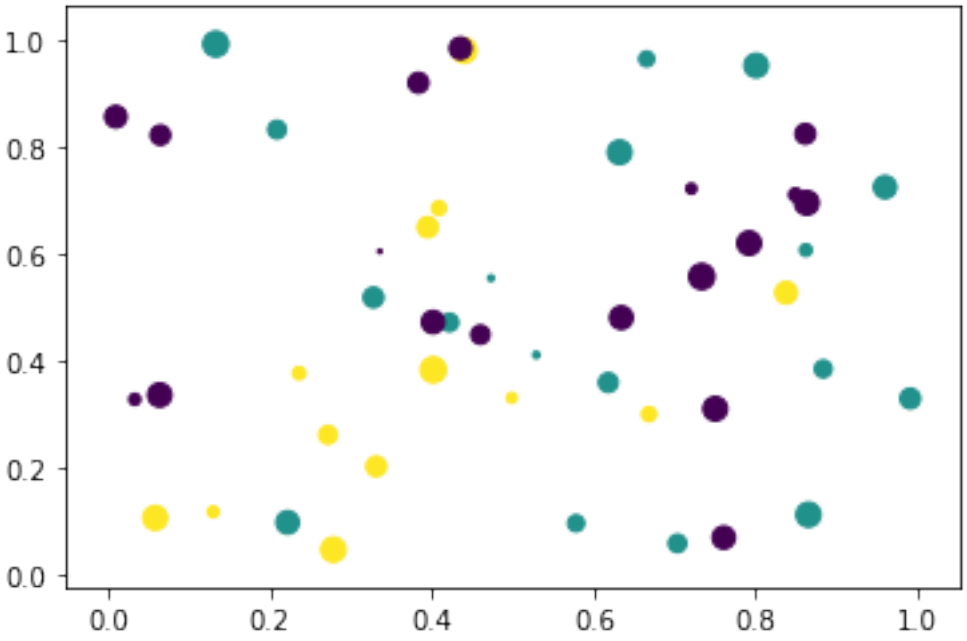



### 1.6.3 Scatter plot

[179]:
```
xs = np.random.sample(50)
ys = np.random.sample(50)
sizes = np.random.randint(100, size=50)
colors = np.random.randint(3, size=50)
```

[180]:
```
plt.scatter(xs, ys, c=colors, s=sizes)
```

[180]:
```
<matplotlib.collections.PathCollection at 0x7fef25eba6d0>
```




### 1.7 NumPy - masky

[181]:
```
a = np.random.randint(100, size=16).reshape(4, 4)
a
```

[181]:
```
array([[54, 23,  7, 62],
       [28, 53, 69, 81],
       [72, 76, 11, 63],
       [67, 40, 41, 61]])
```

[182]:
```
mask = a > 20
mask
```

[182]:
```
array([[ True,  True, False,  True],
       [ True,  True,  True,  True],
       [ True,  True, False,  True],
       [ True,  True,  True,  True]])
```

[183]:
```
a[mask]
```

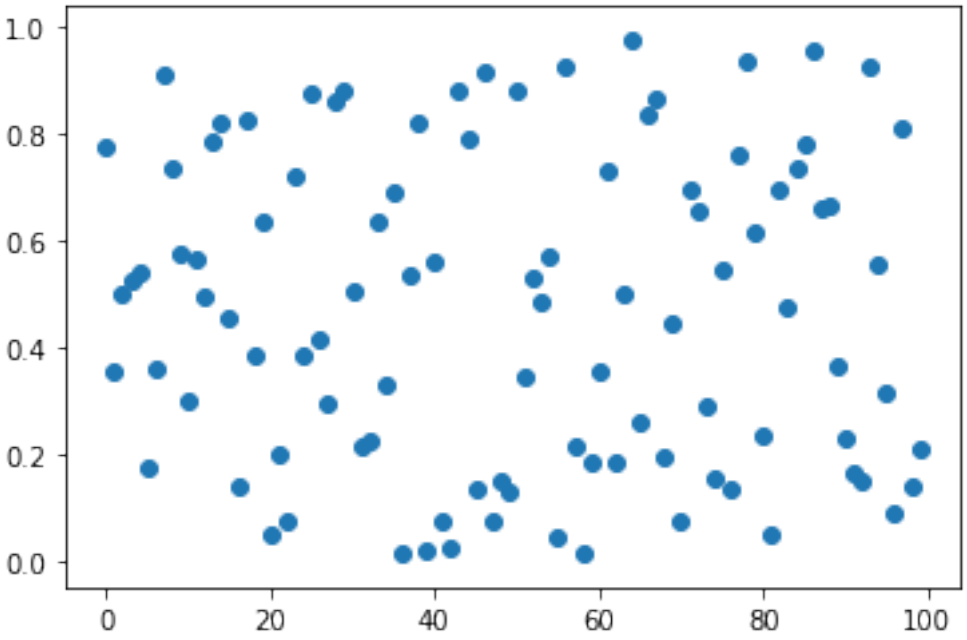
[183]:
```
array([54, 23, 62, 28, 53, 69, 81, 72, 76, 63, 67, 40, 41, 61])
```

[184]:
```
a[~mask]
```

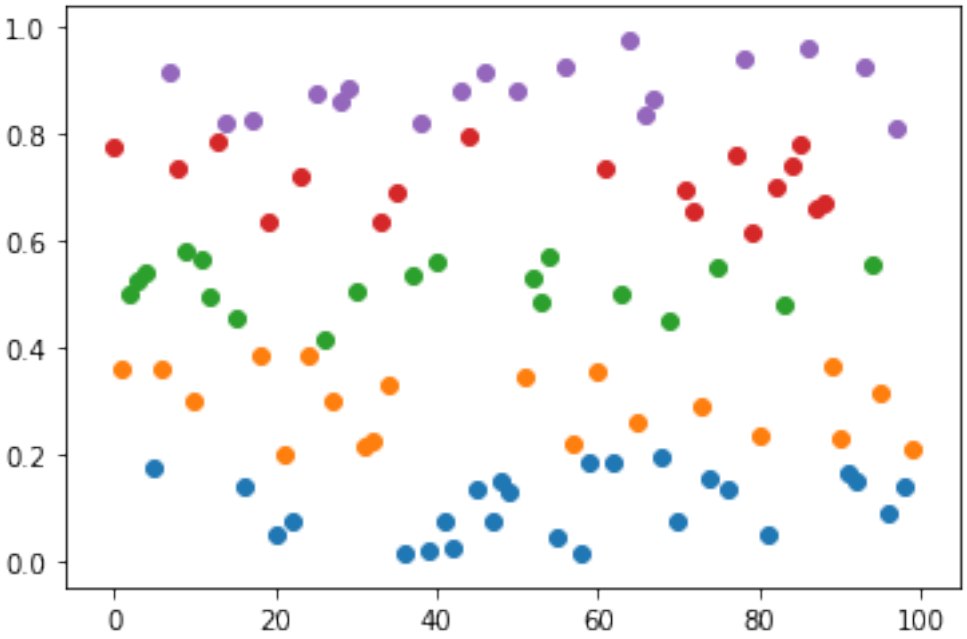
[184]:
```
array([ 7, 11])
```

[185]:
```
xs = np.arange(100)
ys = np.random.sample(100)
plt.scatter(xs, ys)
```

[185]: <matplotlib.collections.PathCollection at 0x7fef25c884d0>




[186]:
```
for threshold in np.linspace(0, 1, 6):
    mask = (ys > threshold) & (ys < threshold + 0.2)
    plt.scatter(xs[mask], ys[mask])
```

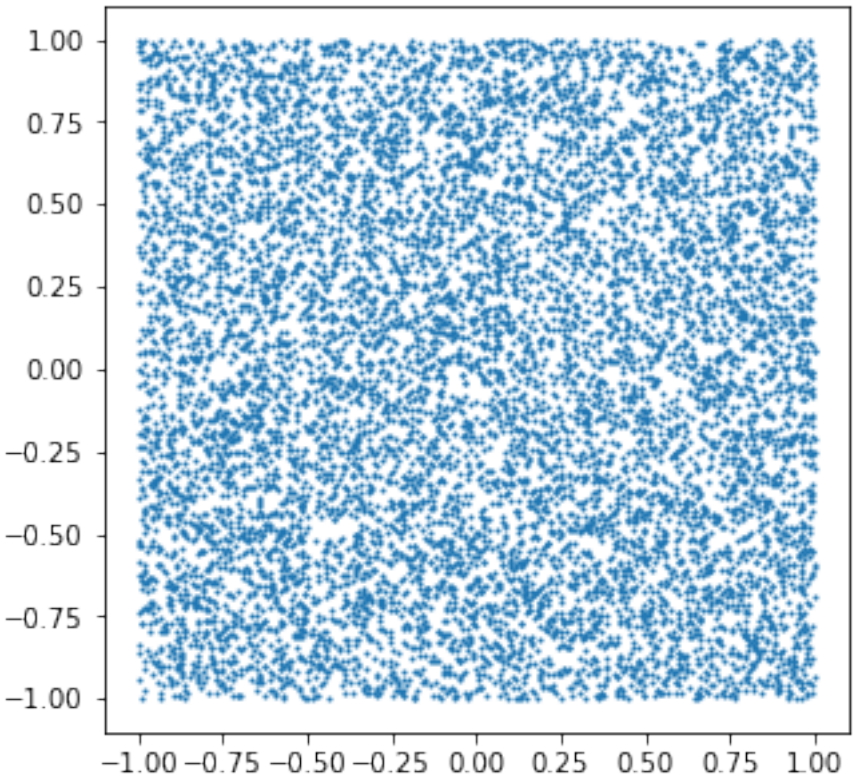



### 1.7.1 Příklad - náhodnostní výpočet $\pi$

[187]:
```
xs = np.random.sample(10000) * 2 - 1
ys = np.random.sample(10000) * 2 - 1
```

[188]:
```
plt.figure(figsize=(5, 5))
plt.scatter(xs, ys, s=1)
```

[188]:
```
<matplotlib.collections.PathCollection at 0x7fef25b6c810>
```

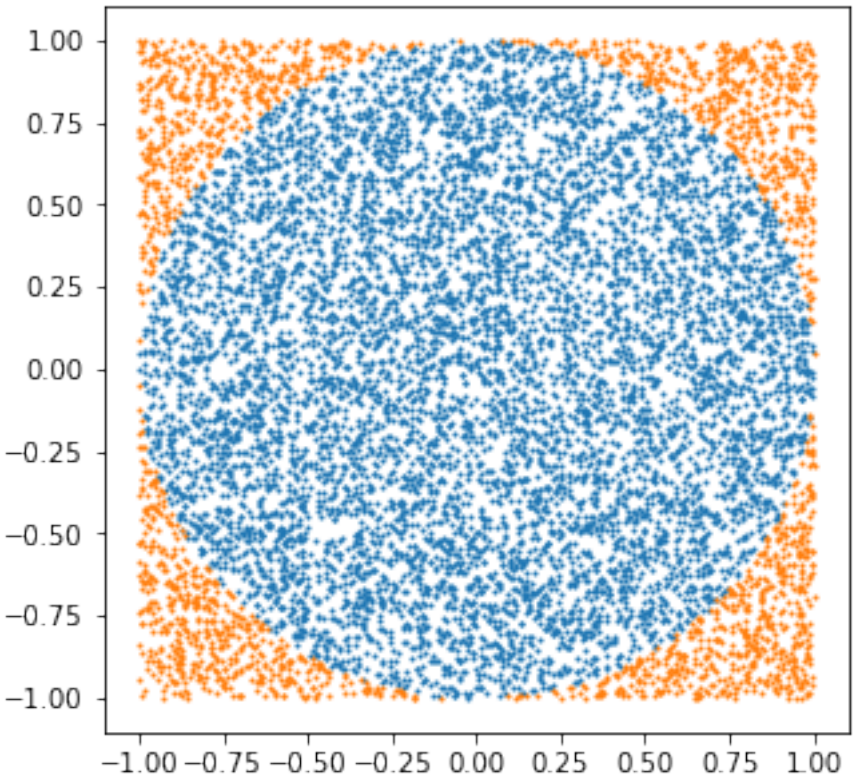



Jednotkový kruh je množina bodů, pro které platí $x^2 + y^2 \leq 1$.

[189]:
```
mask = xs ** 2 + ys ** 2 <= 1
```

[190]:
```
plt.figure(figsize=(5, 5))
plt.scatter(xs[mask], ys[mask], s=1)
plt.scatter(xs[~mask], ys[~mask], s=1)
```

[190]:
```
<matplotlib.collections.PathCollection at 0x7fef25b02dd0>
```




$$\frac{\text{počet bodů}}{\text{počet bodů v kruhu}} \tilde{=} \frac{(2r)^2}{\pi r^2} = \frac{4r^2}{\pi r^2} = \frac{4}{\pi}$$

$$\pi \tilde{=} \frac{4 \cdot \text{počet bodů v kruhu}}{\text{počet bodů}}$$

[191]:
```
4 * np.sum(mask) / len(mask)
```

[191]:
```
3.138
```

### 1.8 NumPy vstup a výstup

- textový
  - `np.savetxt` a `np.loadtxt`
  - pracuje se standardním CSV
  - potřeba nastavit způsob uložení a načtení
- binární
  - `np.save` a `np.load`
  - rychlejší, menší velikost

[192]:
```
a = np.random.randint(100, size=50).reshape(5, 10)
a
```

[192]:
```
array([[31, 30, 49, 83,  7, 14, 30, 30, 53, 92],
       [77, 50, 38, 98, 48, 81, 58, 93, 60, 19],
       [29, 51, 41, 55, 84, 61, 96, 22, 72, 95],
       [30, 47, 13, 74, 98, 58, 67, 48, 61, 69],
       [17, 99, 27, 25, 91, 64, 93, 56, 33, 18]])
```

[193]:
```
np.savetxt('data.csv', a, fmt='%d')
```

[194]:
```
np.loadtxt('data.csv', dtype=np.int)
```

[194]:
```
array([[31, 30, 49, 83,  7, 14, 30, 30, 53, 92],
       [77, 50, 38, 98, 48, 81, 58, 93, 60, 19],
       [29, 51, 41, 55, 84, 61, 96, 22, 72, 95],
       [30, 47, 13, 74, 98, 58, 67, 48, 61, 69],
       [17, 99, 27, 25, 91, 64, 93, 56, 33, 18]])
```

[195]:
```
np.save('data.npy', a)
```

[196]:
```
np.load('data.npy')
```

[196]:
```
array([[31, 30, 49, 83,  7, 14, 30, 30, 53, 92],
       [77, 50, 38, 98, 48, 81, 58, 93, 60, 19],
       [29, 51, 41, 55, 84, 61, 96, 22, 72, 95],
       [30, 47, 13, 74, 98, 58, 67, 48, 61, 69],
       [17, 99, 27, 25, 91, 64, 93, 56, 33, 18]])
```

## 1.9 NumPy - rychlost

[197]:
```
%%timeit
[x ** 2 for x in range(1000)]
```

```
185 µs ± 967 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

[198]:
```
%%timeit
np.arange(1000) ** 2
```

```
2.46 µs ± 23.4 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

### 1.9.1 Součet dvou seznamů/polí

[199]:
```
a_np = np.random.randint(100, size=10 ** 6)
b_np = np.random.randint(100, size=10 ** 6)

a_py = list(a_np)
```

```
b_py = list(b_np)
```

Python - tři způsoby:

[204]:
```
%%timeit
c_py = []
for i in range(len(a_py)):
    c_py.append(a_py[i] + b_py[i])
```

```
167 ms ± 1.53 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[201]:
```
%%timeit
c_py = []
for x, y in zip(a_py, b_py):
    c_py.append(x + y)
```

```
124 ms ± 636 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[202]:
```
%%timeit
c_py = [x + y for x, y in zip(a_py, b_py)]
```

```
99.7 ms ± 223 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

NumPy

[203]:
```
%%timeit
c_np = a_np + b_np
```

```
913 µs ± 6.16 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```