NumPy + Matplotlib - řešení

January 6, 2021

1 NumPy

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

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

1.1 NumPy pole

- obdoba typu `list` z Pythonu
- základní objekt, se kterým NumPy pracuje
- pouze prvky stejného typu
- fixní velikost

```
[5]: a = np.array([1, 2, 3])
     a
```

```
[5]: array([1, 2, 3])
```

```
[6]: a.dtype
```

```
[6]: dtype('int64')
```

```
[7]: np.array([1, 'ahoj', False])
```

```
[7]: array(['1', 'ahoj', 'False'], dtype='<U21')
```

```
[14]: np.arange(2, 10, dtype=int)
```

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

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

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

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

```
[16]: array([0.79001315, 0.65122915, 0.55360169, 0.56902461, 0.95363964,
             0.66821891, 0.68745137, 0.82941886, 0.10186436, 0.69028409])
```

1.2 Základní operace

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

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

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

```
[18]: 10
```

```
[19]: a[0], a[-1]
```

```
[19]: (0, 9)
```

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

```
[21]: a = list(range(10))
      a
```

```
[21]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
[22]: [x + 1 for x in a]
```

```
[22]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
[23]: b = np.arange(10)
      b
```

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

```
[24]: b + 1
```

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

```
[25]: b ** 2
```

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

1.3 Vícerozměrná pole

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

```
[26]: 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])
```

```
[27]: a.shape
```

```
[27]: (25,)
```

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

```
[28]: 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]])
```

```
[29]: b.shape
```

```
[29]: (5, 5)
```

```
[176]: b[0]
```

```
[176]: array([91, 83,  1, 10, 20, 73, 48, 84, 57, 63])
```

```
[34]: b[0, 3]
```

```
[34]: 3
```

```
[36]: b[0, :]
```

```
[36]: array([0, 1, 2, 3, 4])
```

```
[37]: b[:, 2]
```

```
[37]: array([ 2,  7, 12, 17, 22])
```

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

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

```
[39]: np.zeros(10)
```

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

```
[41]: np.zeros((3, 3), dtype=int)
```

```
[41]: array([[0, 0, 0],
             [0, 0, 0],
             [0, 0, 0]])
```

```
[43]: a = np.ones((3, 3))
      a
```

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

```
[45]: a[:, 1] = 4
      a
```

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

```
[46]: a.T
```

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

```
[48]: np.eye(4)
```

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

1.4 Otázka: Jaký bude výsledek tohoto výrazu?

```
3 * np.eye(2) + np.arange(9).reshape(3, 3)
```

a)

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

b)

```
ValueError
```

c)

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

d)

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

1.5 Užitečné funkce

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

```
[57]: 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]])
```

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

```
[58]: (0, 29, 435, 14.5)
```

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

```
[59]: (0, 29, 435, 14.5)
```

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

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

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

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

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

1.6 NumPy a lineární algebra

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

1.6.1 Příklad - soustava lineárních rovnic

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

je ekvivalentní:

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

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

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

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

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

```
[67]: x = np.linalg.solve(A, b)
      x
```

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

```
[68]: A @ x
```

```
[68]: array([1., 2.])
```

```
[71]: np.linalg.inv(A) @ b
```

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

1.7 Vizualizace dat - matplotlib

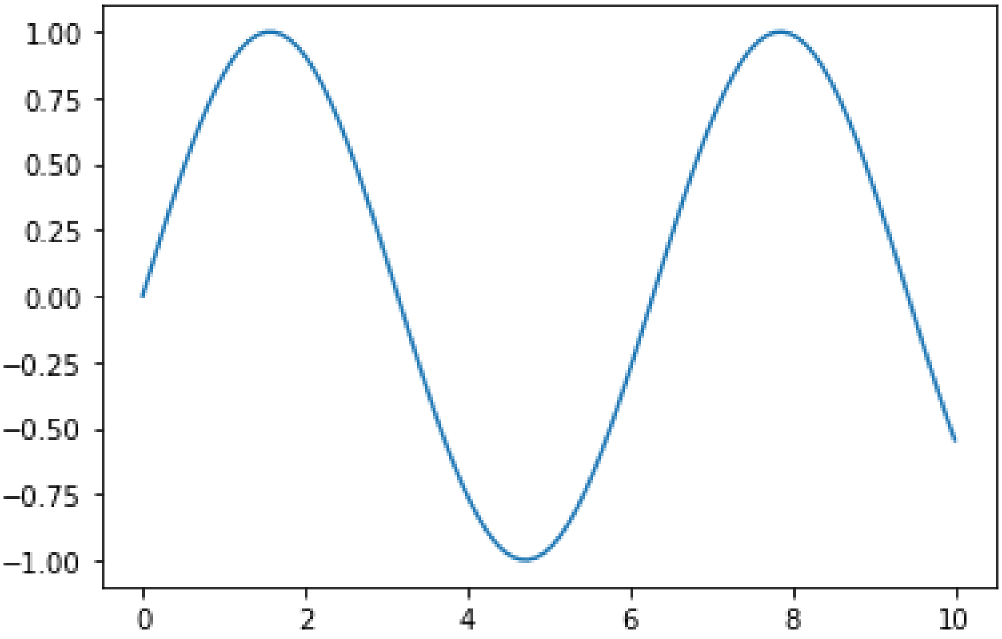
- asi nejrozšířenější modul
- podobná syntaxe jako v Matlabu
- velké možnosti nastavení, typu grafů
- pracuje nad NumPy poli

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

```
[82]: xs = np.linspace(0, 10, 100)
```

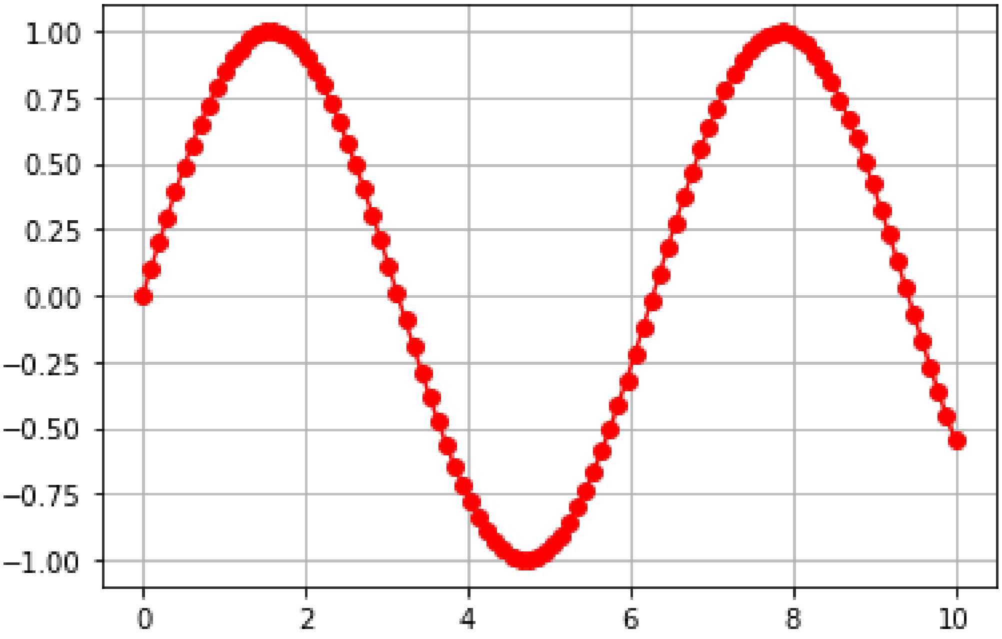
```
[83]: plt.plot(xs, np.sin(xs))
```

```
[83]: [<matplotlib.lines.Line2D at 0x7f65b2c43c40>]
```



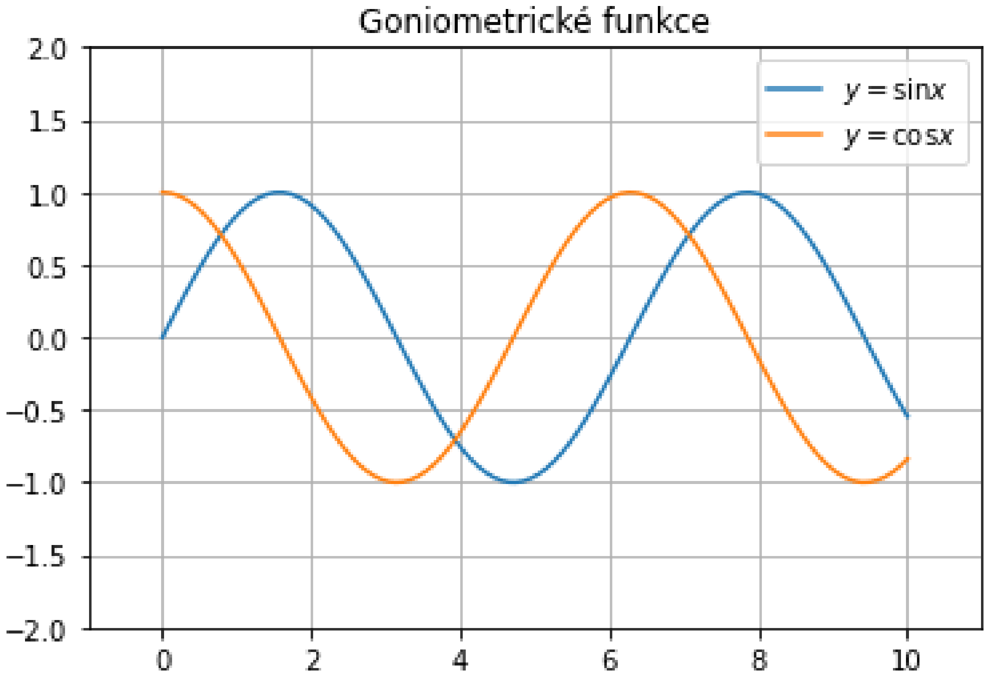
```
[84]: plt.grid()
      plt.plot(xs, np.sin(xs), '-o', color='red')
```

```
[84]: [<matplotlib.lines.Line2D at 0x7f65b2ba9550>]
```



[90]:

```
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()
plt.savefig('image.png')
```



1.7.1 Bar plot

[93]:

```
x = np.random.randint(10, size=10)
x
```

[93]:

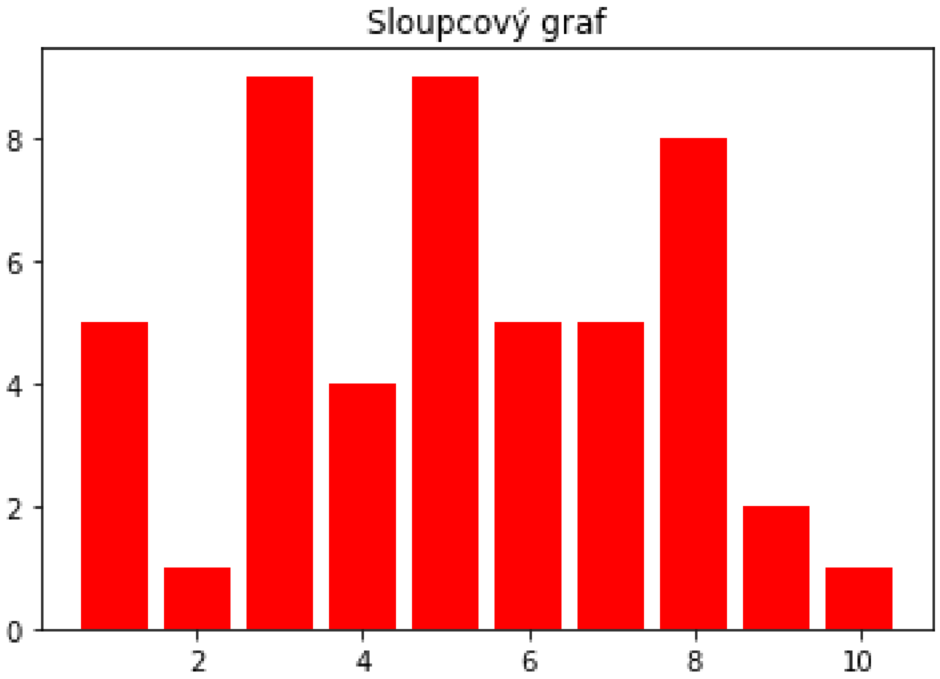
```
array([5, 1, 9, 4, 9, 5, 5, 8, 2, 1])
```

[96]:

```
plt.title('Sloupcový graf')
plt.bar(np.arange(10) + 1, x, color='red')
```

[96]:

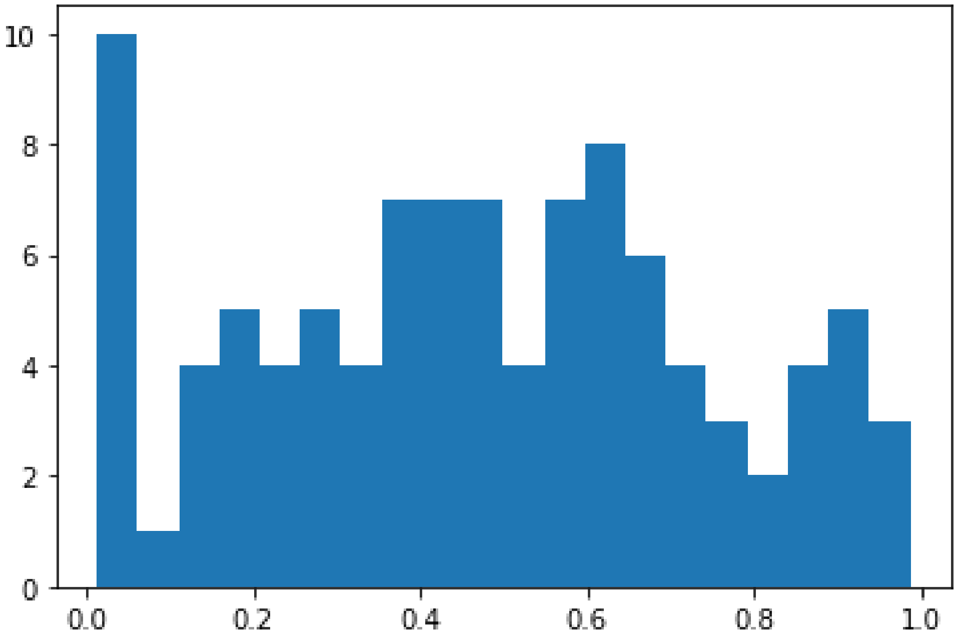
```
<BarContainer object of 10 artists>
```



1.7.2 Histogram

```
[104]: plt.hist(np.random.sample(100), bins=20)
```

```
[104]: (array([10.,  1.,  4.,  5.,  4.,  5.,  4.,  7.,  7.,  7.,  4.,  7.,  8.,
                 6.,  4.,  3.,  2.,  4.,  5.,  3.]),
        array([0.01256698, 0.06128679, 0.1100066 , 0.15872641, 0.20744622,
               0.25616603, 0.30488584, 0.35360565, 0.40232546, 0.45104528,
               0.49976509, 0.5484849 , 0.59720471, 0.64592452, 0.69464433,
               0.74336414, 0.79208395, 0.84080376, 0.88952357, 0.93824339,
               0.9869632 ]),
        <BarContainer object of 20 artists>)
```



1.7.3 Scatter plot

[113]:

```
xs = np.random.sample(50)
ys = np.random.sample(50)
sizes = np.random.randint(200, size=50)
colors = np.random.randint(3, size=50)
```

[109]:

```
colors
```

[109]:

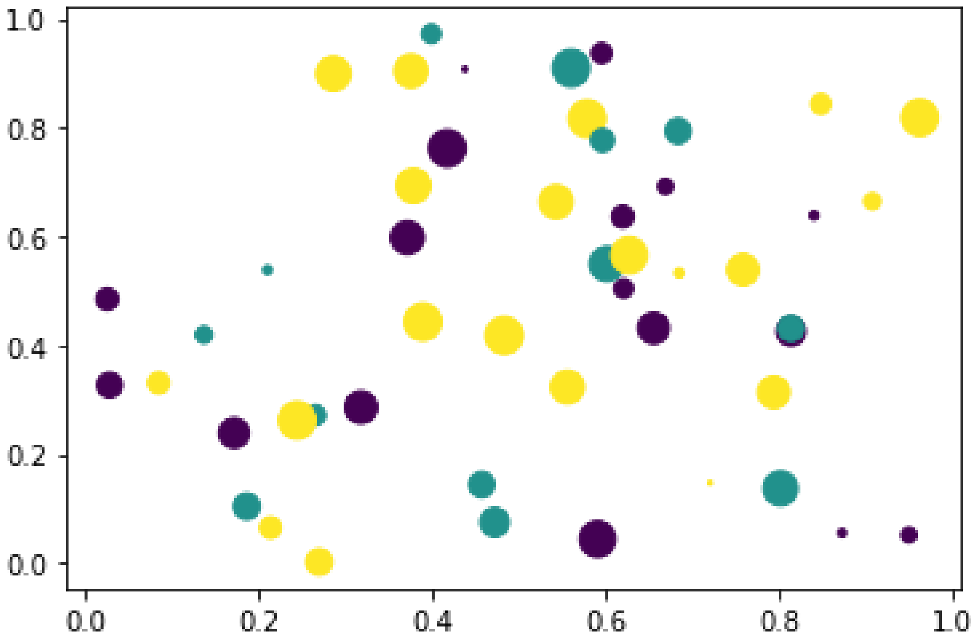
```
array([1, 2, 0, 2, 0, 0, 0, 2, 1, 1, 0, 2, 0, 1, 2, 0, 2, 0, 1, 1, 1, 2,
       1, 1, 2, 0, 0, 2, 1, 1, 2, 2, 0, 2, 0, 1, 0, 0, 1, 0, 2, 1, 0, 2,
       0, 0, 2, 2, 2, 1])
```

[116]:

```
plt.scatter(xs, ys, c=colors, s=sizes)
```

[116]:

```
<matplotlib.collections.PathCollection at 0x7f65b1085dc0>
```



1.8 Otázka: Jsou tyto příkady ekvivalentní?

```
xs, ys = np.random.sample(10), np.random.sample(10)
```

1)

```
for x, y in zip(xs, ys):
    plt.scatter(x, y)
```

2)

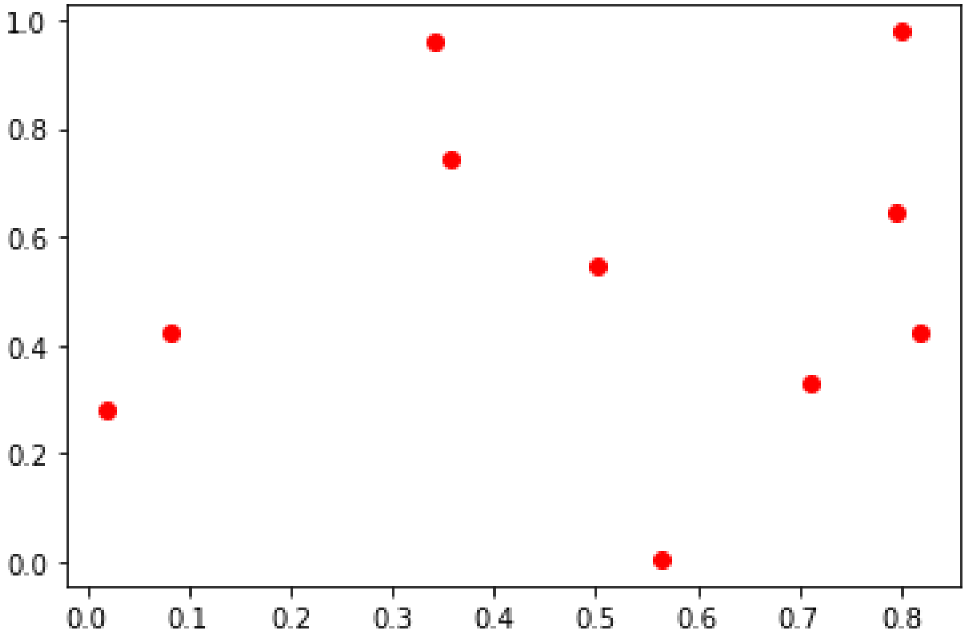
```
plt.scatter(xs, ys)
```

[119]:

```
xs, ys = np.random.sample(10), np.random.sample(10)
```

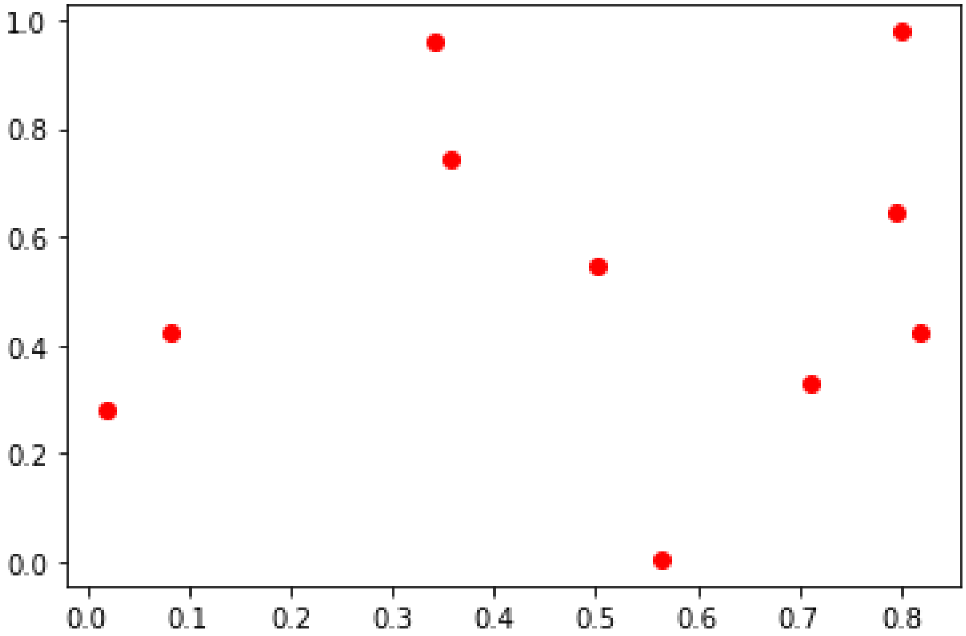
[123]:

```
for x, y in zip(xs, ys):
    plt.scatter(x, y, color='red')
```



```
[124]: plt.scatter(xs, ys, color='red')
```

```
[124]: <matplotlib.collections.PathCollection at 0x7f65b0e879a0>
```



1.9 NumPy - masky

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

```
[125]: array([[41, 75,  9, 46],
              [46, 45, 62, 37],
              [13, 70, 51, 15],
              [95, 85, 97, 82]])
```

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

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

```
[129]: a[mask]
```

```
[129]: array([41, 75, 46, 46, 45, 62, 37, 70, 51, 95, 85, 97, 82])
```

```
[130]: a[a > 20]
```

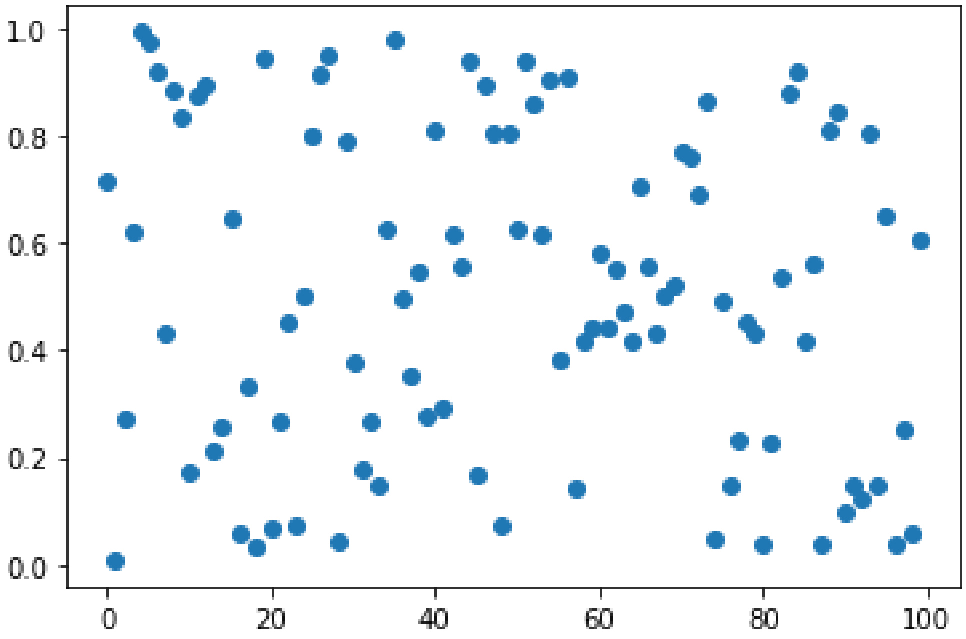
```
[130]: array([41, 75, 46, 46, 45, 62, 37, 70, 51, 95, 85, 97, 82])
```

```
[131]: a[~mask]
```

```
[131]: array([ 9, 13, 15])
```

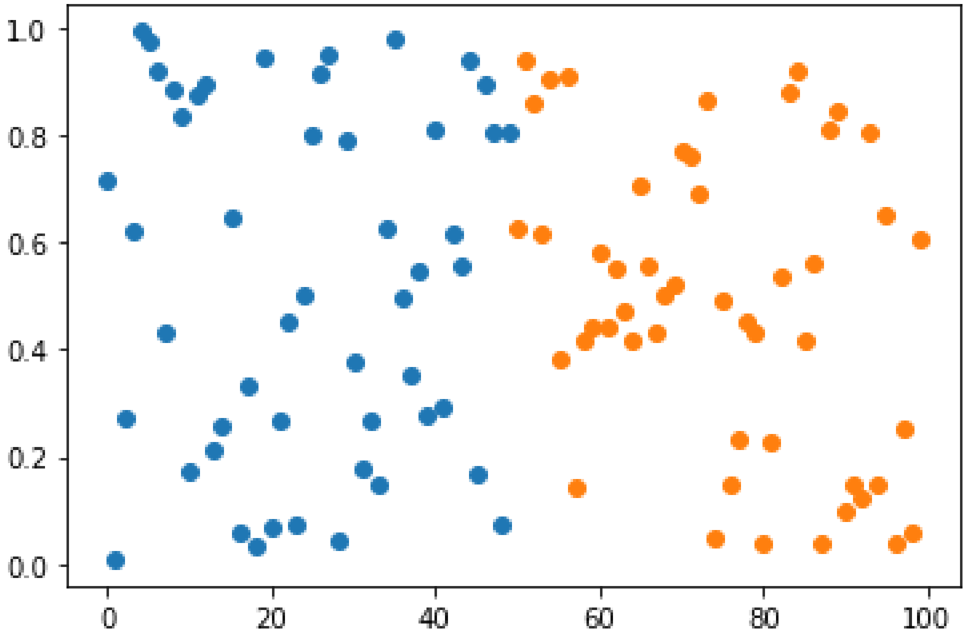
```
[132]: xs = np.arange(100)
       ys = np.random.sample(100)
       plt.scatter(xs, ys)
```

```
[132]: <matplotlib.collections.PathCollection at 0x7f65b172d0a0>
```



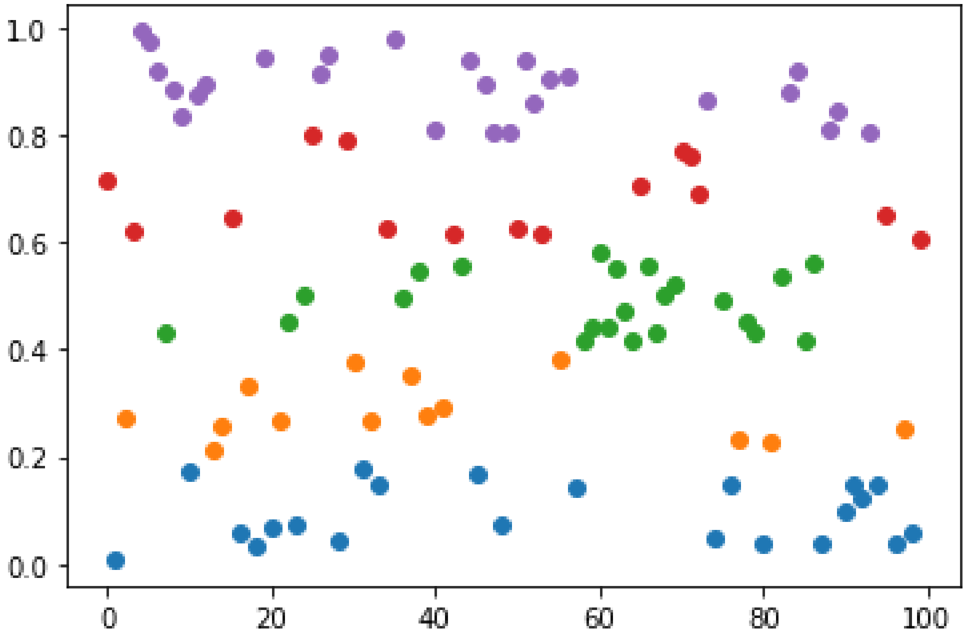
[135]:
```
mask = xs < 50
plt.scatter(xs[mask], ys[mask])
plt.scatter(xs[~mask], ys[~mask])
```

[135]:
```
<matplotlib.collections.PathCollection at 0x7f65b0e3ef40>
```



[136]:

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



1.9.1 Příklad - náhodnostní výpočet π

[154]:

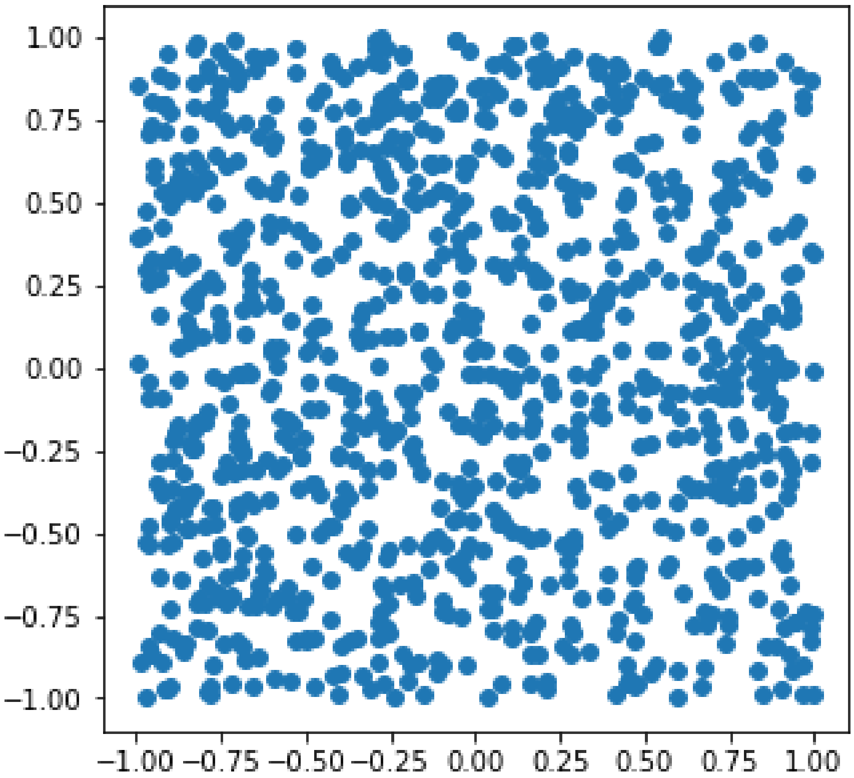
```
xs = np.random.sample(100000) * 2 - 1
ys = np.random.sample(100000) * 2 - 1
```

[144]:

```
plt.figure(figsize=(5, 5))
plt.scatter(xs, ys)
```

[144]:

```
<matplotlib.collections.PathCollection at 0x7f65b0d26b80>
```

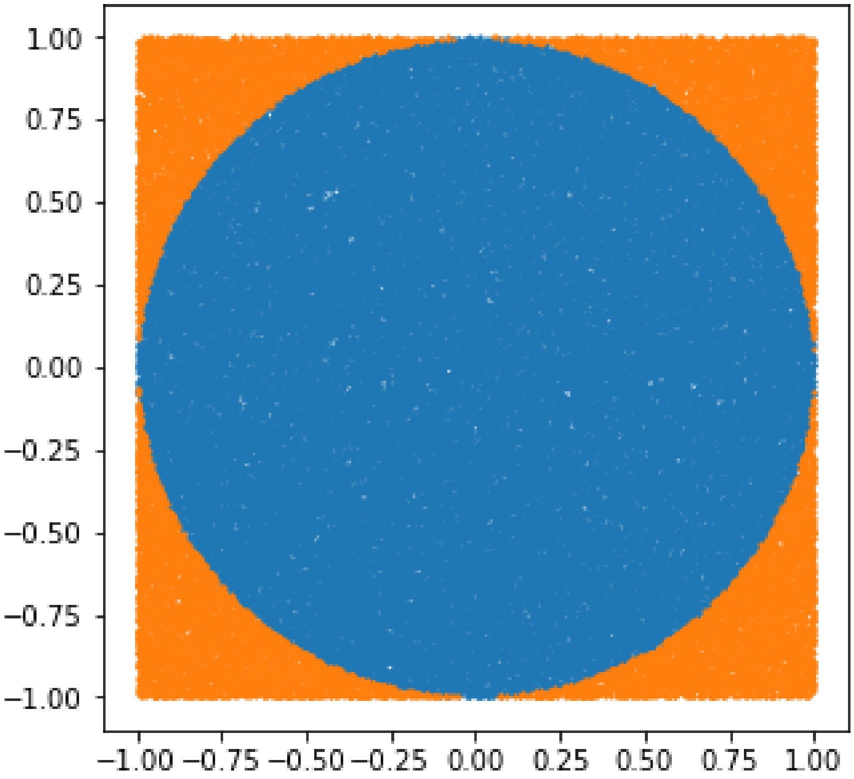


Jednotkový kruh je množina bodů, pro které platí $x^2 + y^2 \leq 1$.

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

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

```
[156]: <matplotlib.collections.PathCollection at 0x7f65b032fa60>
```



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

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

```
[157]: 4 * np.sum(mask) / len(mask)
```

```
[157]: 3.14304
```

1.10 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 (?)

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

```
[158]: array([[91, 83,  1, 10, 20, 73, 48, 84, 57, 63],
              [ 5, 22, 26,  3, 55, 31, 32, 31, 36, 85],
              [ 6, 85, 85, 74, 35, 67, 19, 91, 16, 85],
              [92, 92, 17, 57, 30, 39, 85, 26, 74, 84],
              [80, 52, 60, 95, 48, 78, 96, 21, 59, 99]])
```

```
[163]: np.savetxt('data.csv', a, fmt='%03d')
```

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

```
[164]: array([[91, 83,  1, 10, 20, 73, 48, 84, 57, 63],
              [ 5, 22, 26,  3, 55, 31, 32, 31, 36, 85],
              [ 6, 85, 85, 74, 35, 67, 19, 91, 16, 85],
              [92, 92, 17, 57, 30, 39, 85, 26, 74, 84],
              [80, 52, 60, 95, 48, 78, 96, 21, 59, 99]])
```

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

```
[167]: b = np.load('data.npy')
       b
```

```
[167]: array([[91, 83,  1, 10, 20, 73, 48, 84, 57, 63],
              [ 5, 22, 26,  3, 55, 31, 32, 31, 36, 85],
              [ 6, 85, 85, 74, 35, 67, 19, 91, 16, 85],
              [92, 92, 17, 57, 30, 39, 85, 26, 74, 84],
              [80, 52, 60, 95, 48, 78, 96, 21, 59, 99]])
```

1.11 NumPy - rychlost

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

```
235 µs ± 1.74 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

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

```
2.31 µs ± 22.2 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

1.11.1 Součet dvou seznamů/polí

[170]:

```
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:

[171]:

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

```
191 ms ± 1.17 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[172]:

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

```
143 ms ± 583 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

[174]:

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

```
117 ms ± 763 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

NumPy

[175]:

```
%%timeit
c_np = a_np + b_np
```

```
920 µs ± 5.48 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```