# Základy Pythonu aneb Sbohem Octave

Kateřina Chytrá

# Obsah workshopu



Vstupní soubory ke stažení na Google Disk. Po skončení zde také všechny skripty

# Instalace python3 a pip3

GNU/Linux - Python 2 v základní distribuci

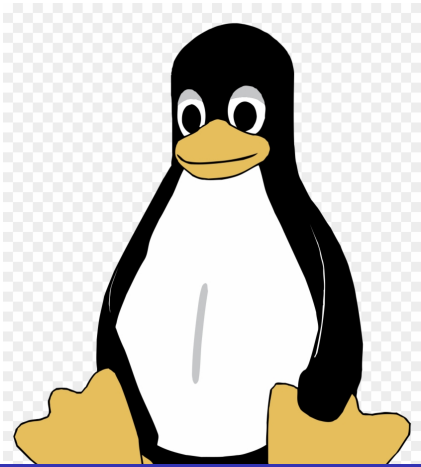
```
$ sudo apt-get install python3.7
```

Podrobný návod např. zde: `https://tecadmin.net/install-python-3-7-on-ubuntu-linuxmint/`

Pip - instalátor balíků

```
$ sudo apt-get install python3-pip
$ sudo pip3 install numpy pydicom matplotlib scikit-image scipy
```

Grafická prostředí: Spyder, Eric, ...



# Spuštění

- v terminálu

```
$ python3
```

# Spuštění

- v terminálu

```
$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>exit()
```

# Spuštění

- v terminálu

```
$ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>exit()
```

- spuštění skriptu v terminálu

```
$ python3 ScriptName.py
```

$$1+1 = ?$$

# 1+1 = ?

**Octave**

```
>>1+1
```

**$ Python**

```
>>>1+1
```

# 1+1 = ?

**Octave**

```
>>1+1
>>2
```

**$ Python**

```
>>>1+1
```

# 1+1 = ?

**Octave**

```
>>1+1
>>2
```

**$ Python**

```
>>>1+1
>>>2
```

# 1+1 = ?

**Octave**

```
>>1+1
>>2
>>[1]+[1]
```

**$ Python**

```
>>>1+1
>>>2
>>>[1]+[1]
```

# 1+1 = ?

**Octave**

```
>>1+1
>>2
>>[1]+[1]
>>2
```

**$ Python**

```
>>>1+1
>>>2
>>>[1]+[1]
```

# 1+1 = ?

**Octave**

```
>>1+1
>>2
>>[1]+[1]
>>2
```

**$ Python**

```
>>>1+1
>>>2
>>>[1]+[1]
>>>[1, 1]
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
>>> A = [1,2,3]
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
>>> A = [1,2,3]
>>>import numpy as np
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
>>> A = [1,2,3]
>>>import numpy as np
>>>B=np.asarray(A)
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
>>> A = [1,2,3]
>>>import numpy as np
>>>B=np.asarray(A)
>>>2*B
```

# Matice

## Octave

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[2,4,6]
```

## $ Python

```
>>> [1,2,3]
[1,2,3]
>>> 2*[1,2,3]
[1,2,3,1,2,3]
>>>type([1,2,3])
<class 'list'>
>>> A = [1,2,3]
>>>import numpy as np
>>>B=np.asarray(A)
>>>2*B
array([2,4,6])
```

# Matice - velikost

## Octave

```
>> A = [[1,2,3];[4,5,6]]
A =
        1 2 3
        4 5 6
>>size(A)
ans =
        2 3
```

## $ Python

```
>>> A = np.asarray([[1,2,3],[4,5,6]])
>>> print(A)
[[1,2,3]
 [4,5,6]]
>>>np.shape(A)
(2,3)
```

# Matice - indexování

**Octave**

```
>> A = [[1,2,3];[4,5,6]]
>>A(1)
ans = 1
>>A(2,1)
ans = 4
>>A(1,2:end)
ans = 2 3
>>A(1,end:-1:2)
ans = 3 2
```

**$ Python**

```
>>> A = np.asarray([[1,2,3],[4,5,6]])
>>>A[1]
array([4,5,6])
>>>A[1,0]
4
>>>A[0,1:]
array([2,3])
>>>A[0,2:0:-1]
array([3,2])
```

Octave: A[start:step:end] - "end" včetně
Python: A[start:end:step] - bez "end"

```
>>A(1,end:-1:1)
ans = 3 2 1
```

```
>>>A[0,2::-1]
array([3,2,1])
```

# Indexování - shrnutí

- hranaté závorky A[]
- začíná se od nuly
- A[start:end:step] - bez "end"
- sestupný výpis až do konce: A[end::-1]
- velikost matice: numpy.shape(A)

$ Python

```
>>> np.shape(A)
(2,3)
>>>from numpy import shape
>>>shape(A)
(2,3)
```

# Uložení/načtení matice do/z textového souboru

$ Python

```
>>>import numpy as np
>>>np.savetxt('MyMat.txt',A)
>>> B = np.loadtxt('MyMat.txt')
```

# For cyklus

## Octave - skript

```
for i=1:3:12        #start:step:stop
        i
endfor
```

# For cyklus

## Octave - skript

```
for i=1:3:12        #start:step:stop
        i
endfor
```

```
>> i=1
>> i=4
>> i=7
>> i=10
```

# For cyklus - klasický zápis

## For.py

```
1  import numpy as np
2
3  A=np.arange(4*4).reshape(4,4)
4
5  B=np.empty((2,4))
6  i=i+1
7  for row in range(0,4,2):
8      B[i] = A[row]
9      i=i+1
10 print('B=')
11 print(B)
12 print(type(B))
```

```
$ python3 For.py
A =
[[ 0 1 2 3]
[ 4 5 6 7]
[ 8 9 10 11]
[12 13 14 15]]
B =
[[ 0. 1. 2. 3.]
[ 8. 9. 10. 11.]]
<class 'numpy.ndarray'>
```

# For cyklus - na 1 řádek

## For.py

```
import numpy as np

A=np.arange(4*4).reshape(4,4)

B = [A[row] for row in range(0,4,2)]
print(B)
print(type(B))
```

```
[array([0, 1, 2, 3]), array([ 8, 9, 10, 11])]
<class 'list'>
```

# For cyklus - na 1 řádek

## For.py

```
import numpy as np
from numpy import asarray

B = asarray([A[row] for row in range(0,4,2)])

print(B)
print(type(B))
```

```
[[ 0 1 2 3]
[ 8 9 10 11]
<class 'numpy.ndarray'>
```

# For cyklus - na 1 řádek

## For.py

```
import numpy as np
from numpy import asarray

B = asarray([A[row] for row in range(0,4,2)])
B = asarray([row for row in A[::2]])

print(B)
print(type(B))
```

```
[[ 0 1 2 3]
[ 8 9 10 11]
<class 'numpy.ndarray'>
```

# For cyklus - na 1 řádek + maticově

## For.py

```
import numpy as np
from numpy import asarray

B = asarray([A[row] for row in range(0,4,2)])
B = asarray([row for row in A[::2]])
B = asarray(A[0:4:2])

print(B)
print(type(B))
```

```
[[ 0 1 2 3]
[ 8 9 10 11]
<class 'numpy.ndarray'>
```

# For cyklus uvnitř string

## For.py

```
for i in range(5):
    print('Mam {N:d} jablek'.format(N=i))

for i in range(5): print('Mam {N:d} jablek'.format(N=i))
```

```
Mam 0 jablek
Mam 1 jablek
Mam 2 jablek
Mam 3 jablek
Mam 4 jablek
```

# Maticové podmínky

## Conditions.py

```
A=np.arange(4) #[0,1,2,3]

B = A[A>1]
print(B)
```

```
[2 3 4]
```

# Definice funkce

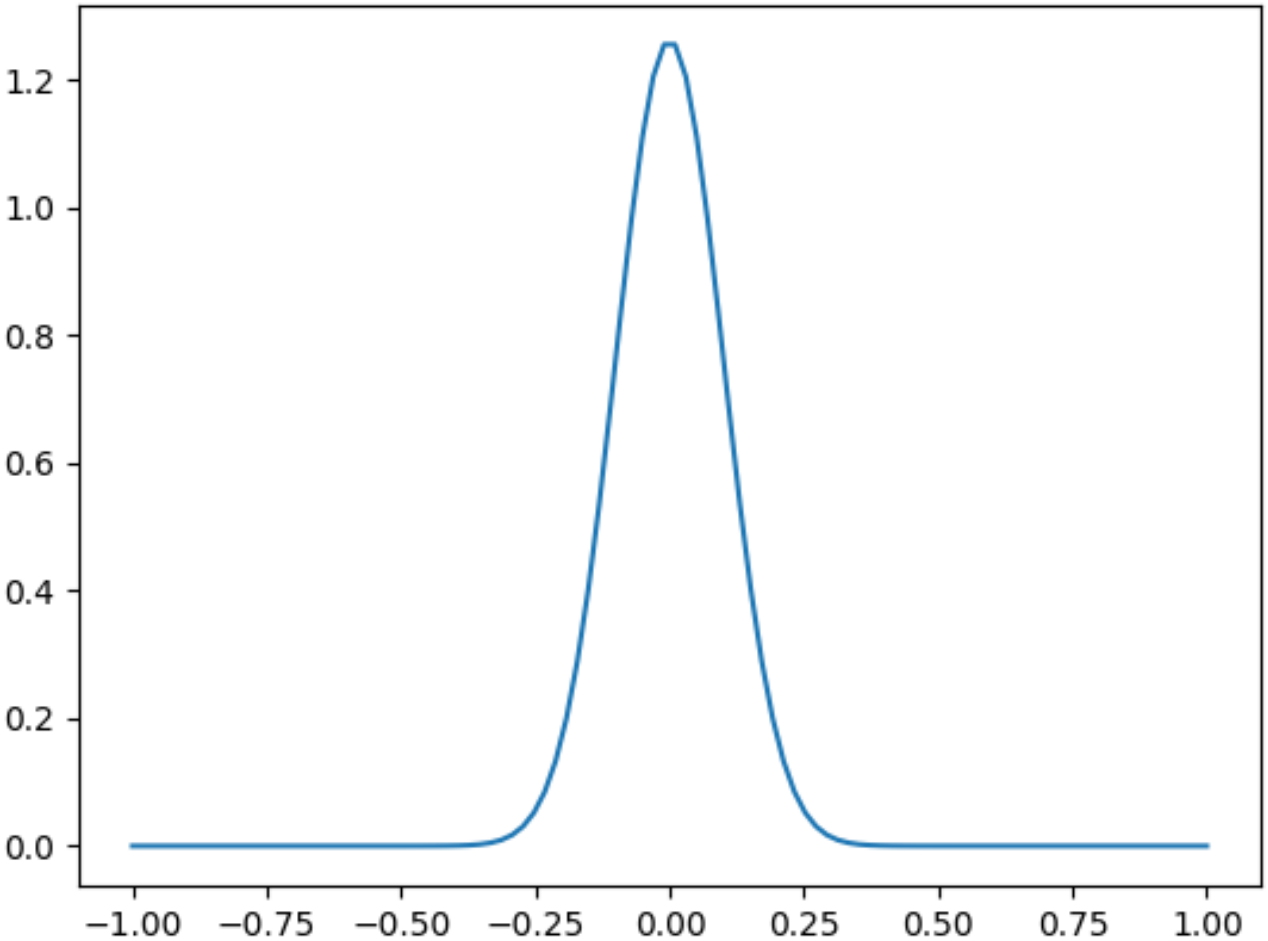
## grafy.py

```
import numpy as np
from numpy import sqrt,pi,exp

def gauss(x,mu,sigma):
    y = 1/sqrt(2*pi*sigma)*exp(-(x-mu)**2/(2*sigma**2))
    return y
```

# Tvorba grafů

## grafy.py

```
import numpy as np
import matplotlib.pyplot as plt
from numpy import sqrt,pi,exp

def gauss(x,mu,sigma):
    y = 1/sqrt(2*pi*sigma)*exp(-(x-mu)**2/(2*sigma**2))
    return y

x = np.linspace(-1,1,100)
y = gauss(x,0,0.1)

plt.plot(x,y)
plt.show()
```

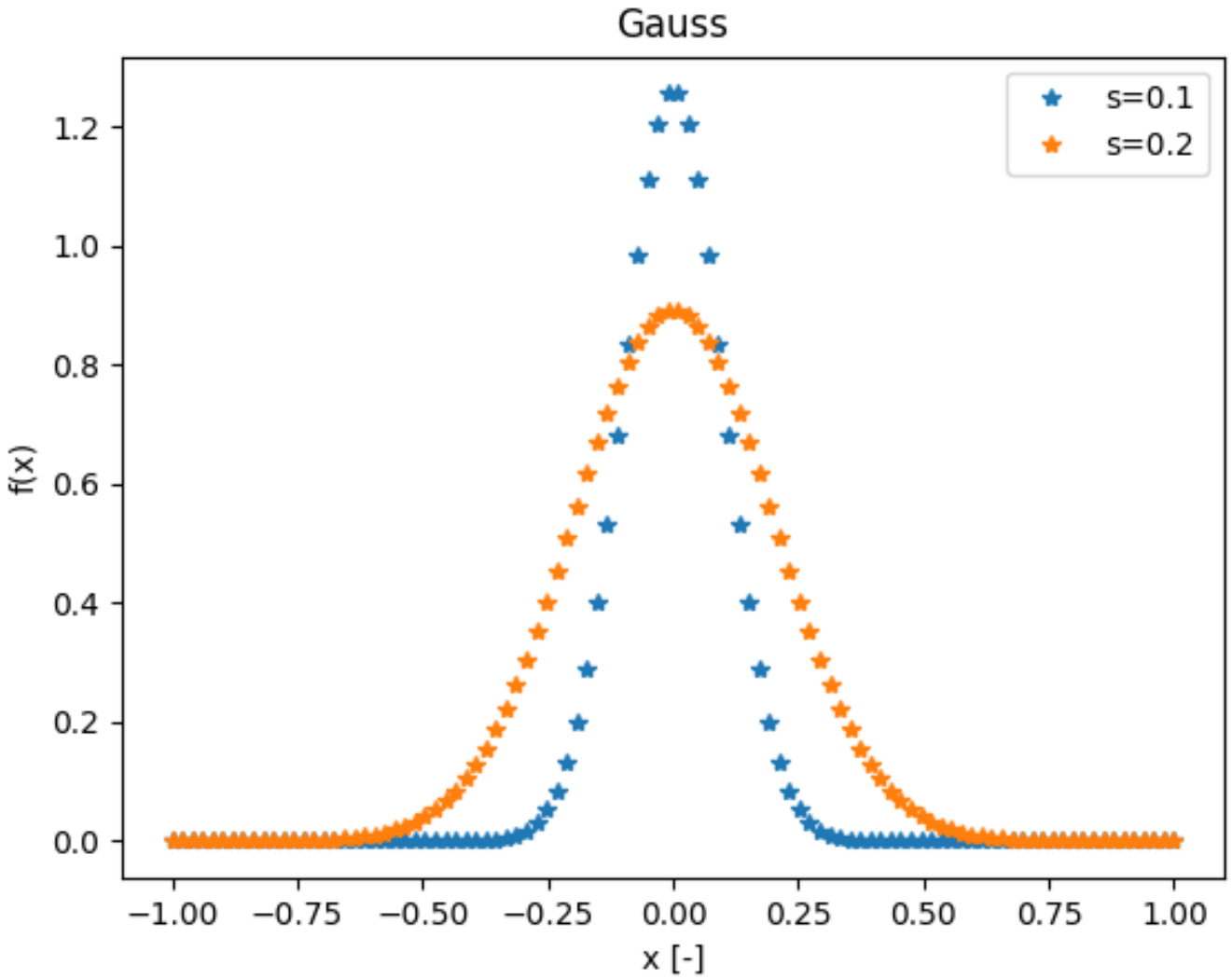

1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

# Tvorba grafů

## grafy.py

```
#(...)
x = np.linspace(-1,1,100)

y = gauss(x,0,0.1)
plt.plot(x,y,'*',label='s=0.1')

y = gauss(x,0,0.2)
plt.plot(x,y,'*',label='s=0.2')

plt.title('Gauss')
plt.xlabel('x [-]')
plt.ylabel('f(x)')

plt.legend()     #show legend
plt.savefig('gauss.png')
plt.show()
```

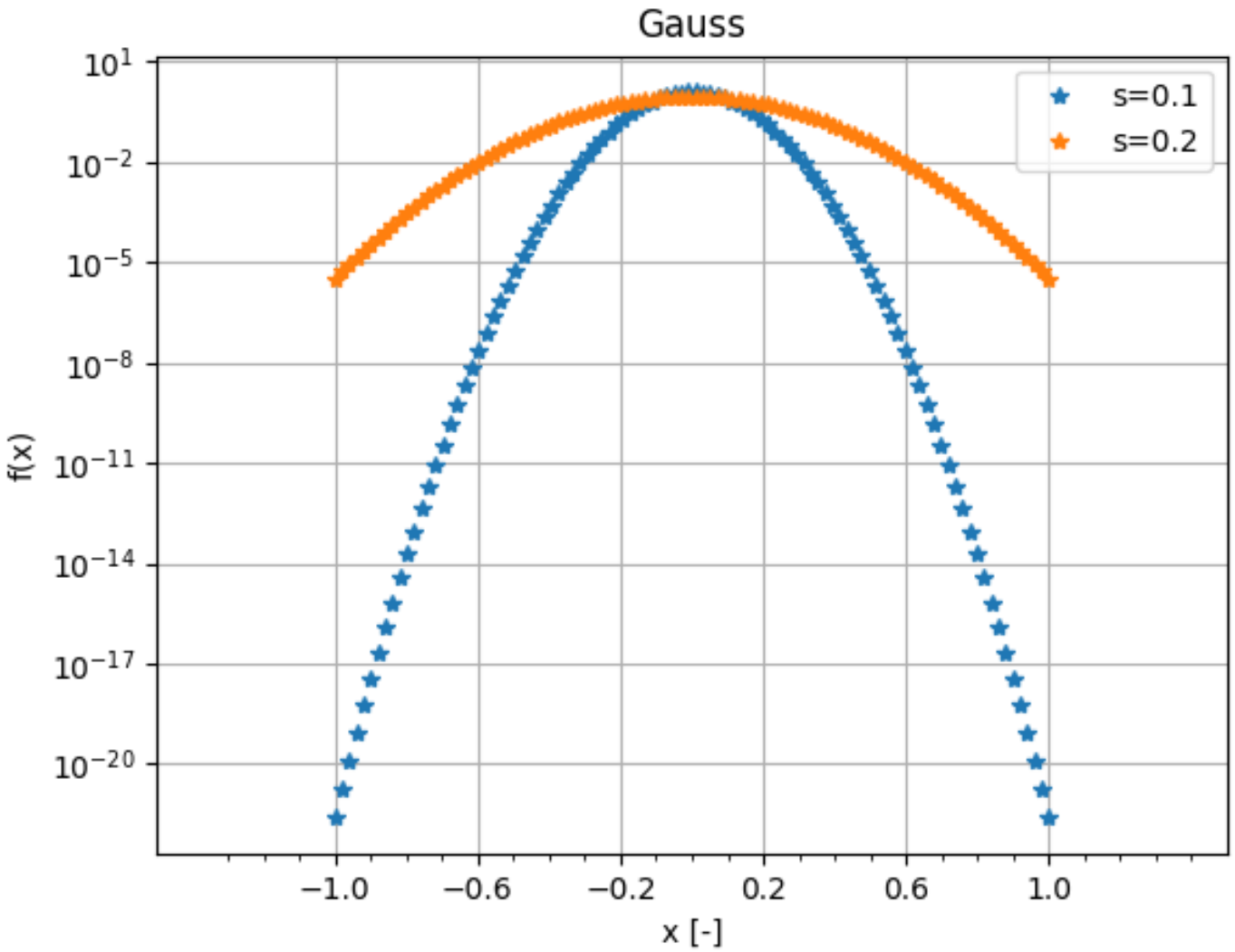
savefig() musí být před plt.show(), jinak se graf do obrázku neuloží


Gauss
s=0.1
s=0.2
1.2
1.0
0.8
0.6
0.4
0.2
0.0
f(x)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x [-]

# Tvorba grafů - nastavení osy

## grafy.py

```
#(...)

plt.xticks(np.arange(-1,1.4,0.4))
#plt.xticks([]) #bez xticks
plt.minorticks_on() #vedlejsi mrizka

plt.xlim(-1.5, 1.5)
plt.yscale('log')
plt.grid() #zobrazeni mrizky

plt.show()
```


Gauss
s=0.1
s=0.2
f(x)
x [-]
$10^{1}$
$10^{-2}$
$10^{-5}$
$10^{-8}$
$10^{-11}$
$10^{-14}$
$10^{-17}$
$10^{-20}$
−1.0
−0.6
−0.2
0.2
0.6
1.0

# Fitování

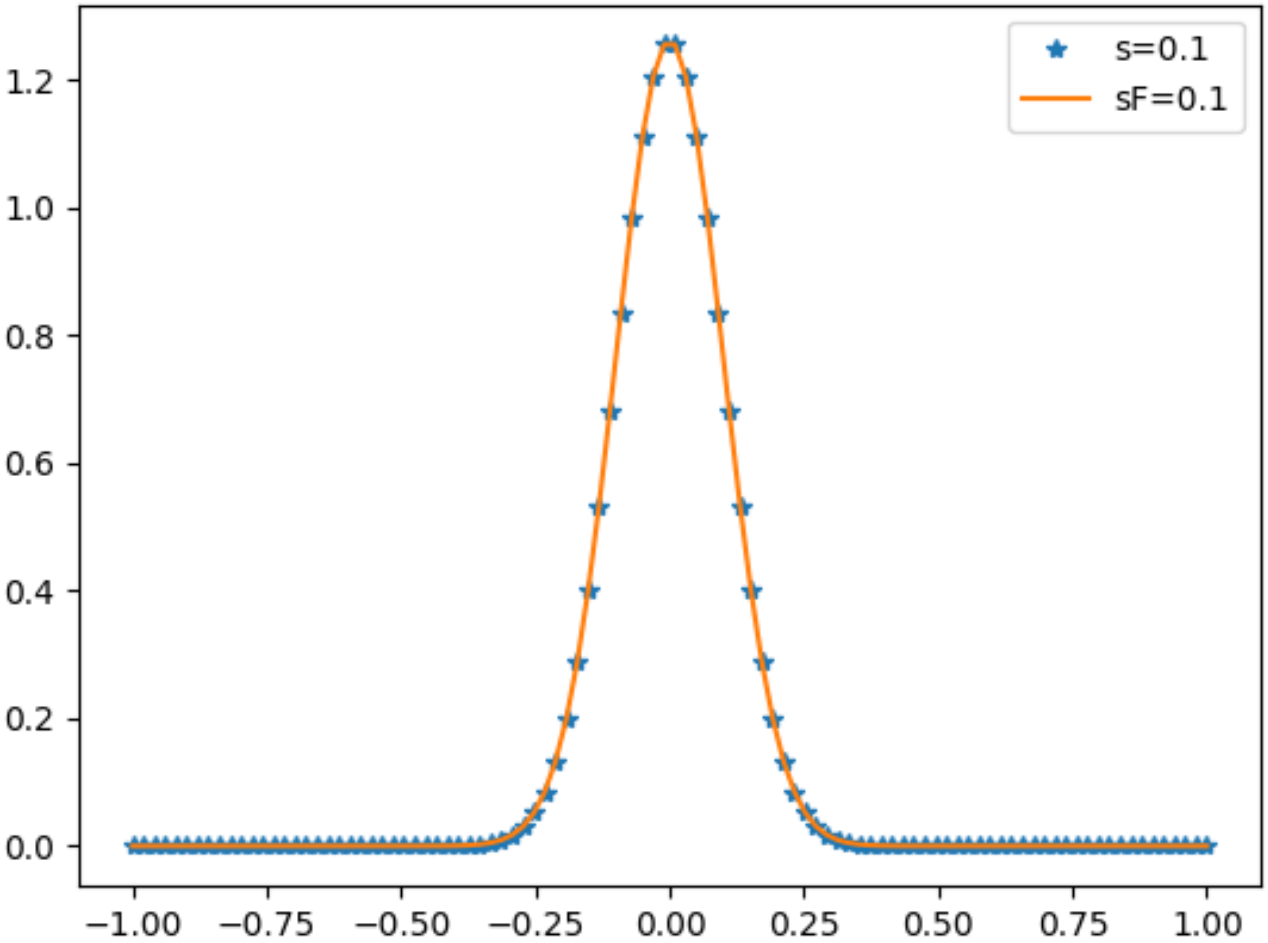
## fit.py

```
import numpy as np
from numpy import pi,exp,sqrt
from scipy.optimize import curve_fit
#-----------------------------------
def gauss(x,mu,sigma):
    y = 1/sqrt(2*pi*sigma**2)*exp(-(x-mu)**2/(2*sigma**2))
    return y
#-----------------------------------
x = np.linspace(-1,1,100)
y = gauss(x,0,0.1)

best_vals,covar = curve_fit(gauss,x,y,p0=[0,1])
muF = best_vals[0]
sF = best_vals[1]
```

# Fitování

## fit.py

```
#(...)
import matplotlib.pyplot as plt
#(...)

best_vals,covar = curve_fit(gauss,x,y,p0=[0,1])
muF = best_vals[0]
sF = best_vals[1]

plt.plot(x,y,'*',label='s=0.1')
plt.plot(x,gauss(x,muF,sF),label='sF='+str((sF)))
plt.legend()
plt.show()
```


s=0.1
sF=0.1
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

# DICOM

## dicom.py

```
import pydicom
filename = 'TOMO.dcm'
ds = pydicom.dcmread(filename) #ds = dataset

#print(ds)  #prints all dicom attributes
print(ds.PatientName)
if 'PixelSpacing' in ds:
    print('Pixel spacing:', ds.PixelSpacing)

if 'PixelData' in ds:
    rows = int(ds.Rows)
    cols = int(ds.Columns)
    slices = int(ds.NumberOfFrames)

    print('Image size: {rows:d} x {cols:d} x {slices:d}, {size:d}
    bytes'.format(rows=rows,cols=cols,slices=slices,
     size=len(ds.PixelData)))
```

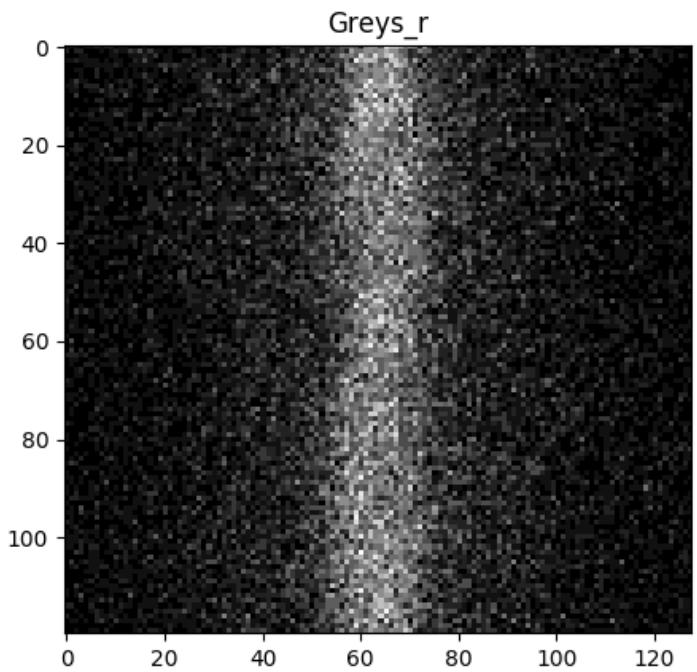
# DICOM

DATSCAN Phantom
Pixel spacing: ['3.395766', '3.395766']
Image size: 128 x 128 x 120, 3932160 bytes

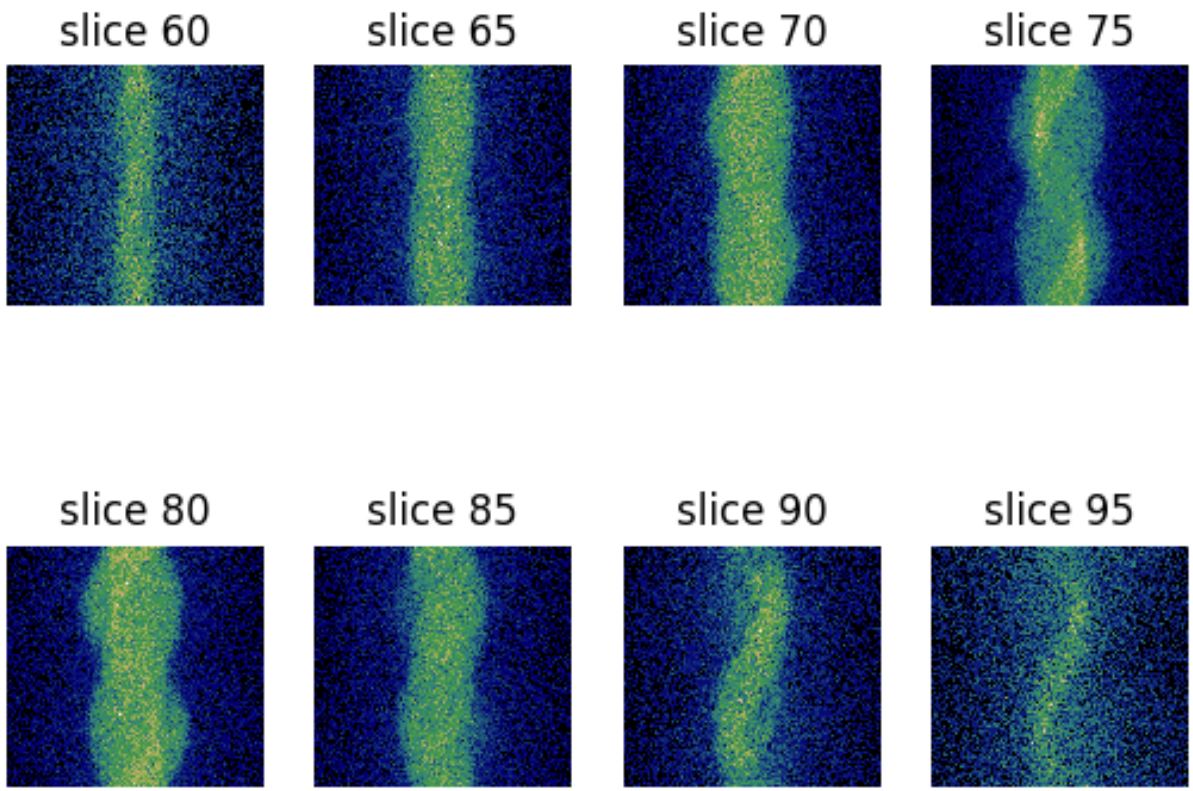
# DICOM

## dicom.py

```
#(...)
sinogram = ds.pixel_array[:,:,:] #[:,slice,:]
plt.imshow(sinogram[:,60,:],cmap = plt.cm.Greys)
plt.show()
```

# Subplots

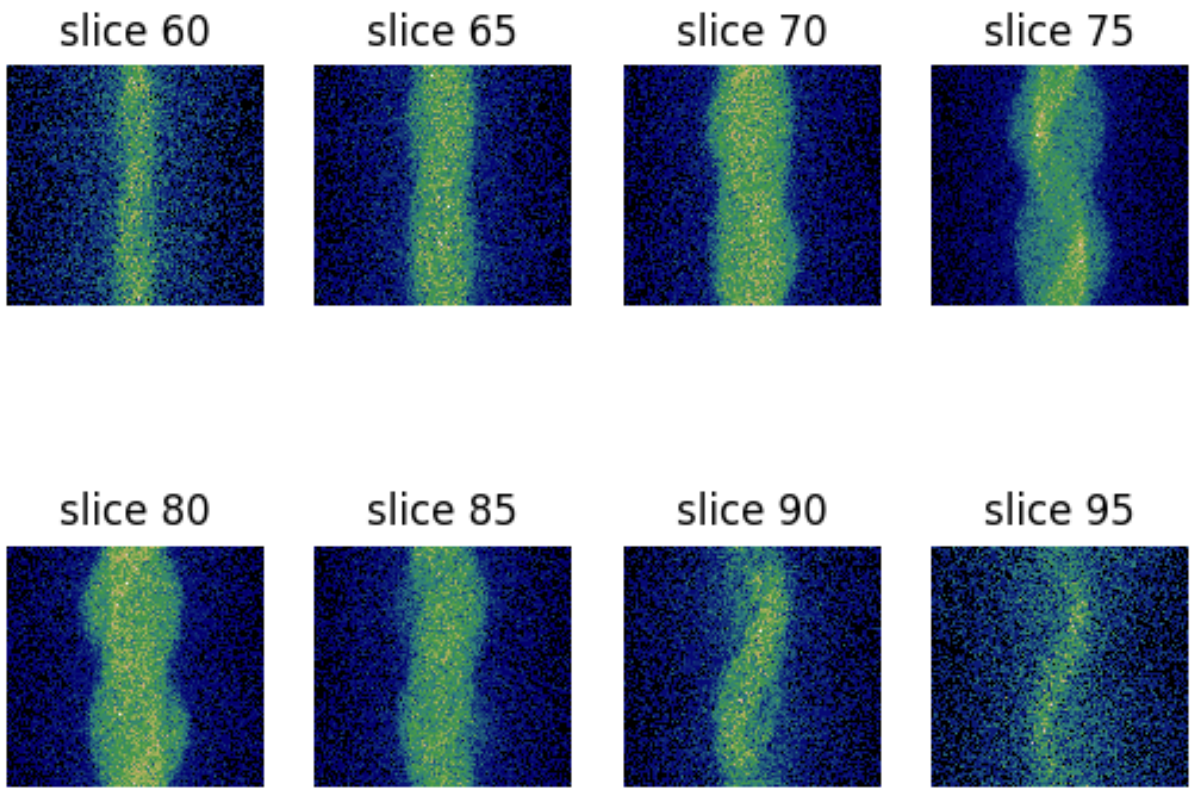

slice 60
slice 65
slice 70
slice 75
slice 80
slice 85
slice 90
slice 95

# Subplots

## dicom.py

```
#(...)

fig,axes = plt.subplots(2,4,sharey=True)
ax = axes.ravel() #1-D array

rez = 60
for item in ax:
    item.imshow(sinogram[:,rez,:],cmap = plt.cm.gist_earth)
    item.set_title('slice '+str(rez))
    item.axis('off')
    rez=rez+5

plt.show()
```

# Subplots


slice 60
slice 65
slice 70
slice 75
slice 80
slice 85
slice 90
slice 95

# Image processing - scikit-image

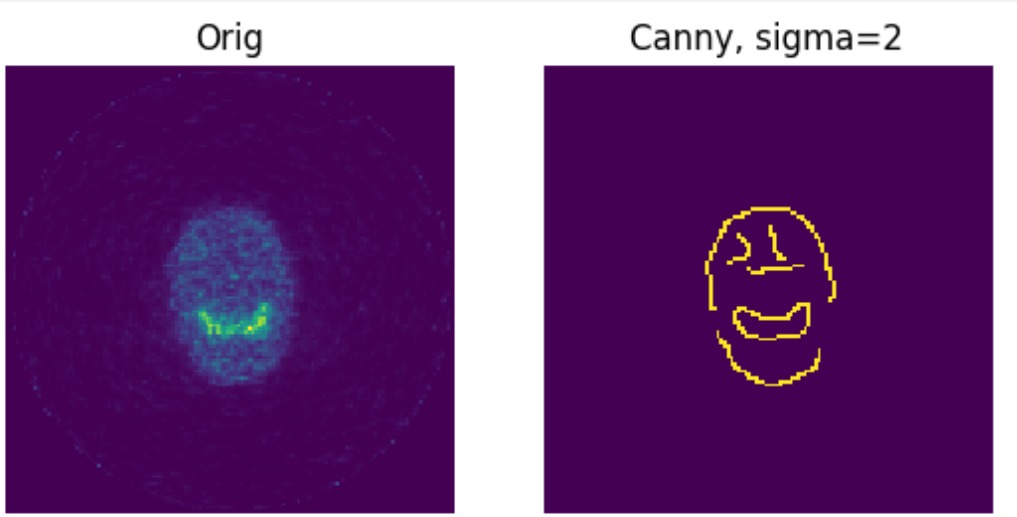
## image.py

```
import numpy as np
import matplotlib.pyplot as plt

from skimage import filters
from skimage.measure import regionprops    #teziste snimku

filename = 'TOMOosem.txt'
I0 = np.loadtxt(filename)
I0= I0.reshape((128,128,128))

#Binarizace
threshold_value = filters.threshold_otsu(I0)
binIm = (I0 > threshold_value).astype(int)

#Teziste snimku
properties = regionprops(binIm, I0)
center_of_mass = properties[0].centroid
```

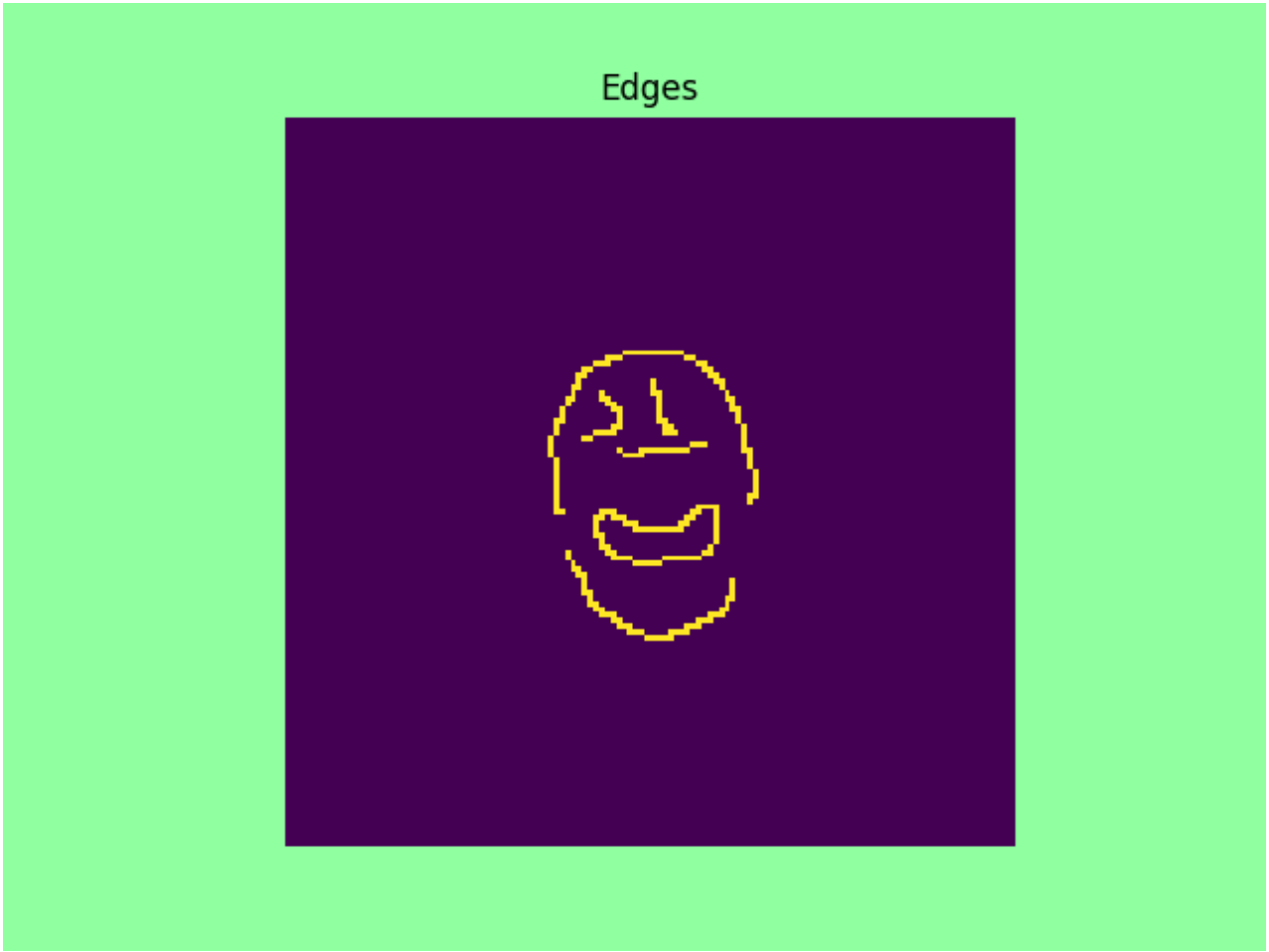
# Scikit-image - detekce hran

## image.py

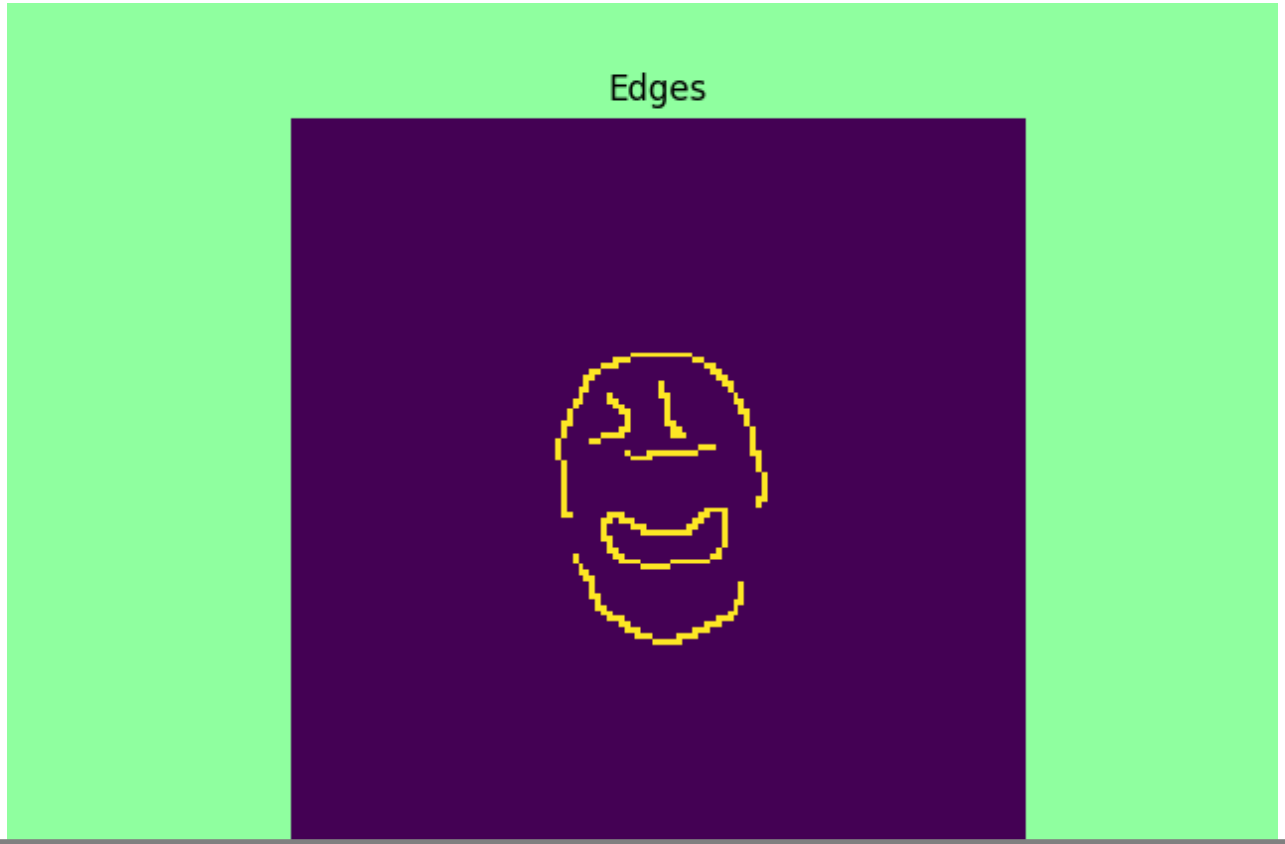
```
#(...)
from skimage.feature import canny
#(...)
#edge detection
Is = I0[int(round(center_of_mass[0])),:,:]
edges = canny(Is, sigma=2)

plt.imshow(edges)
plt.show()
```



# Scikit-image - vygenerování obrázků



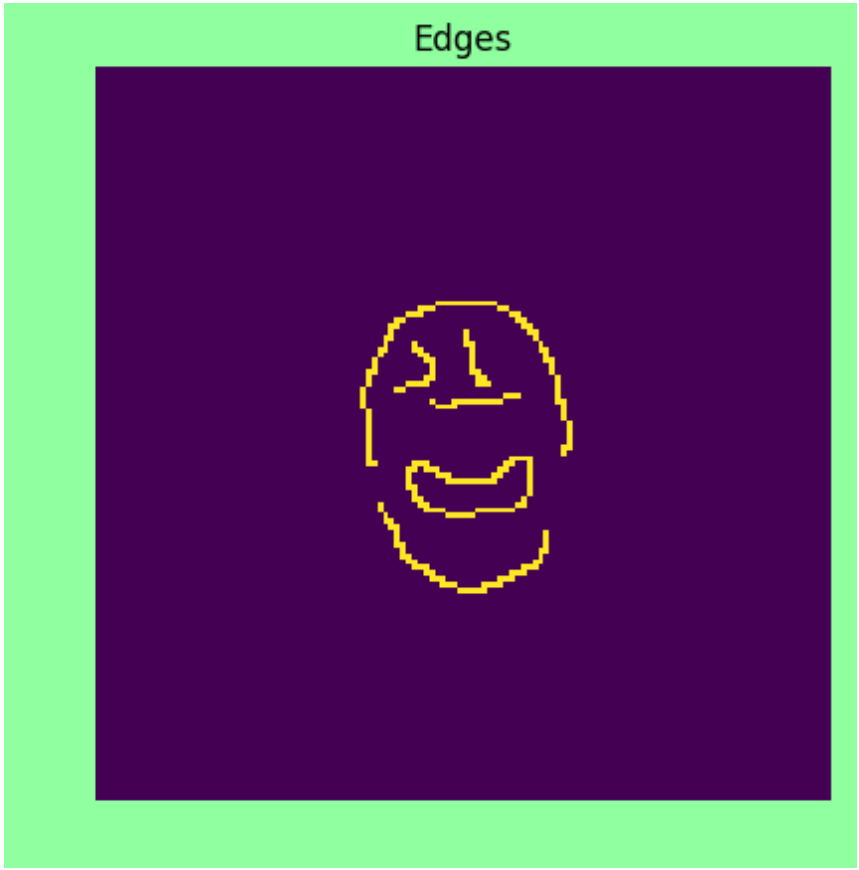
# Scikit-image - vygenerování obrázků



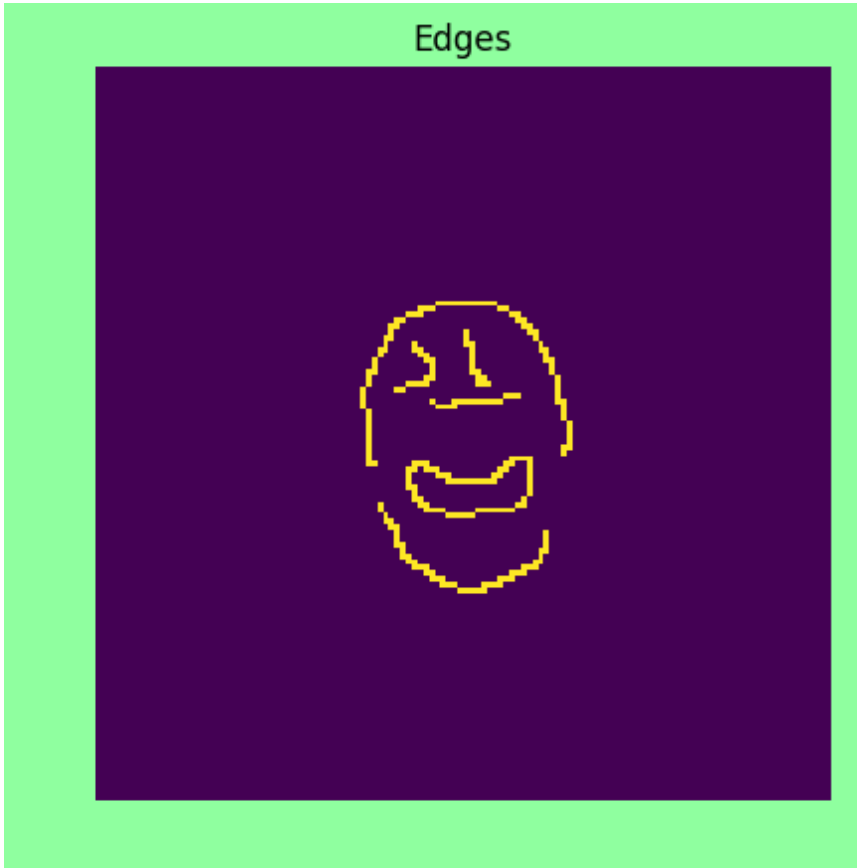

## image.py

```
#(...)
fig.patch.set_facecolor('xkcd:mint green')
plt.savefig('edges.png',facecolor=fig.get_facecolor())
plt.show()
```

# Scikit-image - vygenerování obrázků



# Scikit-image - vygenerování obrázků




## image.py

```
#(...)
plt.savefig('edges.png',bbox_inches='tight',pad_inches=0.1)
plt.show()
```

# Pár poznámek na závěr

- v knihovně numpy - matematické operace defaultně po prvcích
- maticové násobení np.dot(m1,m2)
- nastavení relativní cesty

```
import os.path
myPath = os.path.abspath(os.path.dirname(__file__))
```

- komentář přes více řádků

```
'''
Tohle vsechno
chci zakomentovat
'''
```

# Mini-kvíz

# Mini-kvíz

```
>>> 'C'+'S'+'F'+'M'
```

1. Výstup kódu výše je
   (a) chybová hláška - string nelze sčítat
   (b) string: .....(doplňte)

# Mini-kvíz

```
>>> 'C'+'S'+'F'+'M'
```

1. Výstup kódu výše je
   (a) chybová hláška - string nelze sčítat
   (b) string: .....(doplňte)
2. Doplňte kód namísto otazníků, tak abyste dostali tento výstup. Použijte proměnnou name.

```
>>> name = 'KRF CSFM'
>>> ???
CSFM
```

# Děkuji za pozornost!

Doporučuji tutoriál : `https://pythonic.eu/fjfi/`