Kurzy pro začátečníky a mírně pokročilé

prof. Karel Zaplatílek

Obsah

Předmluva

Tato kniha je určena pro ty z vás, kdo se chcete ponořit do základů programovacího jazyka python. Jste úplní začátečníci nebo jen mírně pokročilí? Pak vítejte ve světě moderního programování. Kromě toho kniha poslouží učitelům a lektorům jako „noty“ pro vedení kurzů.

Já jsem svou profesí univerzitním profesorem elektroniky a zpracování signálů. K mé odborné práci patří odpočátku programování, spojené s technikou nebo její aplikací. Proto je tato elektronická kniha založena na tzv. procedurálním (někdy též imperativním) programování. Je to zjednodušeně posloupnost příkazů, které dělají, co potřebujeme. Přesněji při tomto způsobu tvorby programů (algoritmů) nepotřebujeme vytvářet vlastní třídy, ale využíváme třídy, objekty a metody již hotové. V knize tedy nepoužívám tzv. objektově orientované programování, neboť to prostě pro dané účely není třeba.

Kniha obsahuje celkem pět kurzů od úplných začátků v jazyku python přes práci se signály a grafikou až po tvorbu grafického uživatelského rozhraní (GUI) a číselné soustavy. Volba témat vychází z mé celoživotní praxe učitele studentů (nejen) technického zaměření, kteří potřebují naprogramovat své úlohy pokud možno jednoduše a efektivně. Každodenní styk s mladými lidmi mně dává skvělou zpětnou vazbu a umožňuje vidět různé přístupy k výuce programování a jejich důsledky. To vše jsem zúročil při psaní téhto knihy.

Kurzy byly vyzkoušeny a ověřeny jak v prezenční formě v prostředích poslucháren a zasedacích místností, tak ve formě distanční s využitím internetu. Odborný obsah, formát A4 a zvolený styl jsou přizpůsobeny vedení kurzů. Právě větší formát knihy a větší použité fonty velmi usnadňují práci v pološeru zasedacích místností nebo učeben, jak ví každý, kdo někdy vedl kurzy podobného typu. Přes velká usnadnění, která nám poskytuje výpočetní technika a informační technoligie se domnívám, že naní nad tištěné podklady kurzu. Lektor tak nemusí v případě nutnosti přepínat okna na své obrazovce a způsobovat zmatek na promítacím plátně dataprojektoru, když si potřebuje osvěžit paměť nebo nalézt zapomenutý příkaz či funkci.

Lektor při vedení kurzů a školení potřebuje znát časový rozsah celého kurzu i jednotlivých kapitol a řadu podpůrných návěstí, tipů a doporučení tak, aby průběh každé kapitoly byl plynulý a aby jednotlivé kapitoly měly odbornou i časovou kontinuitu.

Neskromně se domnívám, že moje dlouholetá učitelská s lektorská praxe, spojená se zpětnou vazbou od studentů, účastníků kurzů a učitelů, skýtá záruku kvalitního pečlivě rozmyšleného a pedagogickou praxí vyzkoušeného obsahu.

Jak pracovat s publikací

Jak je zmíněno výše, obsahuje kniha pět kurzů v prostředí jazka python. Učitelé a lektoři kurzů již v obsahu naleznou kromě odborných témat také časové rozsahy pro celý daný kurz i jeho jednotlivé kapitoly. Časový rozsah každé velké kapitoly je navíc uveden přímo na jejím začátku za názvem. Doporučuji si celý kurz před vlastním vedením projít a pokusit se odhadnout časové možnosti vaše a vašich posluchačů. Sám dobře vím, že s některou skupinou čas jen letí, zatímco s jinou se spíše vleče. Proto je třeba být flexibilní, nespěchat a nechat si určitou odbornou rezervu pro případ, že vám zbyde na konci více času. To vše je součásí praxe lektora, která se vyvíjí postupně s množstvím odvedených kurzů a školení. Největší výzvou je nesnažit se spěchat a neříkat posluchačům příliš mnoho informací naráz. Spíše doporučuji pomalejší tempo s dostatkem času na provedení příkazů s jejich dobrým zažitím.

Přeji si, aby se tato kniha stala příručkou jak pro vás, kteří s pythonem začínáte, tak také pro lektory kuzrů a školení. Prakticky každý software se dnes poměrně rychle vyvíjí; python není výjimkou. Buďte proto flexibilní tam, kde to bude v budoucnu třeba.

Autor

4. Práce s enticemi, množinami a slovníky (1 h)

4.1. Použití entic (tuple, tuples)

Entice je podobná seznamům, ale na rozdíl od nich **je neměnná !**

tuple_1=(1, 2, 'ahoj', [3, 4])
tuple_11 = 1, 2, 'ahoj', [3, 4] bez závorek, méně běžné
print(tuple) (1, 2, 'ahoj', [3, 4])
type(tuple_1) class ‘tuple‘
a = tuple_1[0:2]
print(a) (1, 2) indexování jako u seznamů
a, b, c, d = tuple_1 rozbalení ntice do proměnných, sami vyzkoušejte print a type
a, b, c = 4, ‘Kája’, 123 nejsou-li použity závorky, ntice se vytvářejí automaticky
tuple_1[0] = 55 chybové hlášení, **entici nelze měnit** (seznamy ANO, textové řetězce NE)
‘Ahoj‘ in tuple_1 True
sez = list(tuple_1) převedení entice na seznam, sami vyzkoušejte print a type
tuple_2 = tuple(sez) převedení seznamu zpět na entici

Protože entice je neměnná, lze vymazání některé její položky obejít tak, že se převede na seznam, položka se vymaže pomocí del a opět se převede na entici. Podobně setřídění.

4.2. Práce s množinami (set, sets)

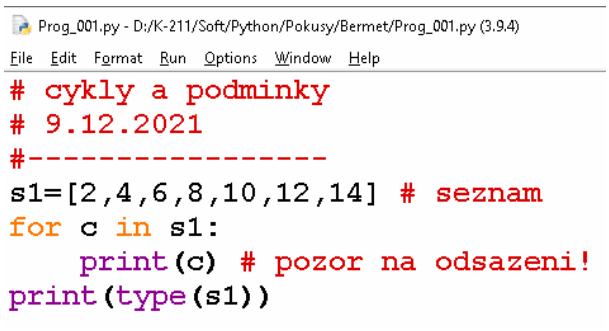
Množina je **měnitelná** a neindexovaná kolekce neuspořádaných prvků.

set_1 = {0, 1, 0, 6, 1, 5, 5}
print(set_1) {0, 1, 5, 6} **Tip:** a = set()
len(set_1) 4, délka množiny
set_1.add(8)
print(set_1) {0, 1, 5, 6, 8}
len(set_1) 5
type(set_1) class ‘set‘
set_2 = {5, 1, 6, 6}
print(set_2) {1, 5, 6}
set_3 = set_1 & set_2
print(set_3) {1, 5, 6} průnik dvou množin

5. Základní programátorské techniky (1 h)
(Cykly, podmínky a zpracování výjimek)

5.1. Cyklus „for“

Použití: opakované vykonávání posloupnosti příkazů
Cykly budeme zapisovat ne v příkazové řádce, ale ve vestavěném editoru: **File / New File**:




```
========== RESTART:
2
4
6
8
10
12
14
<class 'list'>
```

Okno se zapsaným kódem uložíme (File - Save Ass …) pod názvem Prog_001, systém doplní příponu *.py. Je vhodné si předem vytvořit na disku složku, kam budeme programy ukládat. Další zdrojové texty budeme zapisovat do okna editoru a vždy uložíme pod jiným jménem, např. Prog_002, Prog_003 …

Zapsaný text uložíme (Ctrl-S nebo File-Save) a kód spustíme díky Run-Run Module nebo pomocí F5.Odezva v příkazové řádce je na pravém obrázku. Takto budeme realizovat všechny další programy.

Upozornění: ve zdrojovém textu je vidět **odsazení** (anglicky indentation, min. jedna mezera, běžně tabulátor = 4 mezery), které je u cyklů nutné.

Další zdrojové texty s cykly:

```
# cykly a podminky
# 9.12.2021
#-----------------
s1=[2,4,6,8,10,12,14] # seznam
index=0
for c in s1:
    print(index,c) # pozor na odsazeni!
    index+=1 # index=index+1
```

```
===== RESTART:
0 2
1 4
2 6
3 8
4 10
5 12
6 14
```

6. Základy práce s funkcemi (function, functions), (1 h)

```
# prace s funkcemi
#-----------------
def soucet(a, b): # nova funkce
    print('a je {}, b je {}'.format(a, b))
    return  a + b

s = soucet(10, 20) # volani funkce
print(s)
print() # prazdny radek
print(type(soucet)) # datova trida function
print(type(s)) # datova trida int
#-----------------
```

```
print('a je {}
c = a + b
return c
```

```
===== RESTART: D:\K
a je 10, b je 20
30

<class 'function'>
<class 'int'>
```

```
# prace s funkcemi
#-----------------
def radek(): # funkce pro prazdny radek
    print()
radek() # volani funkce
radek()
print('ahoj')
```

```
========= RESTART:


ahoj
```

```
# prace s funkcemi
#-----------------
def soucet(a, b): # nova funkce
    print('{} + {} = {}'.format(a, b, a+b))
    return a + b

s = soucet(a=10, b=20) # volani funkce
print('Soucet je:',s)
```

```
========= RESTART:
10 + 20 = 30
Soucet je: 30
```

7. Tvorba modulů, práce se soubory a složkami (1 h)

7.1. Tvorba a import modulů

Modul je prostý soubor *.py, obsahující užitečné příkazy. Modulem může být i složka více modulů. Níže vytvoříme zdrojový text souboru a uložíme pod názvem **Mujmodul.py** (příponu dodá systém).

```
# Muj modul
#----------
def odmocnina(cislo): # vypocet pomoci Newtonovy metody (NM)
    aproximace = cislo/2.0
    lepsi = (aproximace + cislo/aproximace)/2.0 # NM
    while lepsi != aproximace:
        aproximace = lepsi
        lepsi = (aproximace + cislo/aproximace)/2.0 # NM
    return aproximace

def soucet(a, b):
    print('{} + {} = {}'.format(a, b, a+b))
    return a + b
```

Níže vidíte různé způsoby importu našeho nového modulu a volání jeho funkcí. Při importu modulu je vytvořena složka _pycache_, kam si python ukládá tzv. *zkompilovaný bytecode*. Pokud ji smažete, bude příště opět vytvořena.

```
# Muj modul
#----------
from Mujmodul import odmocnina
s = odmocnina(36)
print(s)

from Mujmodul import soucet
s = soucet(10, 20)
print(s)
```

```
===== RESTART:
6.0
10 + 20 = 30
30
```

7.3. Práce s binárními soubory

Jsou to soubory, obsahující obrázky, fotografie, videa, zazipované a spustitelné soubory. Binární soubory nemají řádkovou organizaci a nelze je otevřít textovým editorem:

```
# prace se soubory
#-----------------
id = open('Foto.jpg','rb')
T = id.read(56) # nacte 56 znaku
print(T)
id.close()
```

```
= RESTART: D:\K-211\Soft\Python\Pokusy\Bermet\Prog_
001.py
b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x01\x01,
\x01,\x00\x00\xff\xe1\x00\x16Exif\x00\x00MM\x00*\x0
0\x00\x00\x08\x00\x00\x00\x00\x00\x00\xff\xdb\x00C\
x00\x05\x03\x04\x04\x04\x03\x05'
```

Pro práci s obrázky je třeba doinstalovat do pythonu modul (knihovnu) s názvem Pillow. K tomu účelu obsahuje instalace python soubor pip.exe.

Protože jsme při instalaci zaškrtli nabídku Add Python 3.12 to PATH, stačí nyní spustit příkazovou řádku Windows 10 (ve vyhledávacím políčku zcela vlevo dole zapíšeme cmd a potvrdíme *Enter*).

V černé příkazové řádce zapíšeme příkaz: python -m pip install Pillow a potvrdíme Enter. Modul *Pillow* (zkráceně PIL – *Python Image Library*) bude nainstalován, viz obrázek níže.

2. Vědecko-technické výpočty a vizualizace dat. Základy práce se signály, grafika 2-D. (1,5 h)

2.1. Instalace modulů numpy a matplotliob

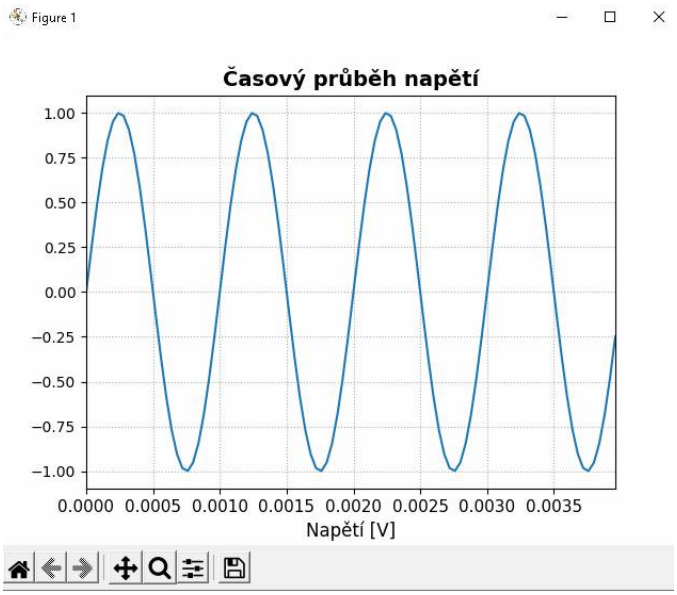
Modul Numpy je základní modul pro vědeckotechnické výpočty. V základu umožňuje vytvořit n-rozměrné homogenní pole (nejčastěji čísel) a pracovat s ním.

Modul Matplotlib je modulem pro kreslení (vizualizaci) dat a je určen pro spolupráci s *Numpy*.

Oba moduly nainstalujeme do python podobně jako modul *Pillow* v rámci kurzu 1, s. 40. Instalační příkazem ve Windows 10 (cmd) je: python -m pip install numpy

Podobně: python -m pip install matplotlib

```
# 2-D plosny graf
#----------------
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
t = np.arange(0, 4e-3, 4e-3/100) # vektor casove osy, 100 vzorku
u = np.sin(2*np.pi*1e3*t) # vetor na svisle ose, sinusovka
print(t.shape) # rozmer
plt.plot(t,u) # vytvori graf, ale nezobrazi
plt.title('Časový průběh napětí',fontsize=14,fontweight='bold')
plt.xlabel('Čas [s]',fontsize=12,fontweight='normal')
plt.xlabel('Napětí [V]',fontsize=12,fontweight='normal')
plt.grid(linestyle=':') # mrizka teckovane, '--', '-'
plt.xlim(np.amin(t),np.amax(t)) # rozsah osy t
plt.savefig('Graf_01.png') # ulozeni obrazku na disk
plt.xticks(fontsize=11) # font cisel na ose t
plt.show() # zobrazi graf
```

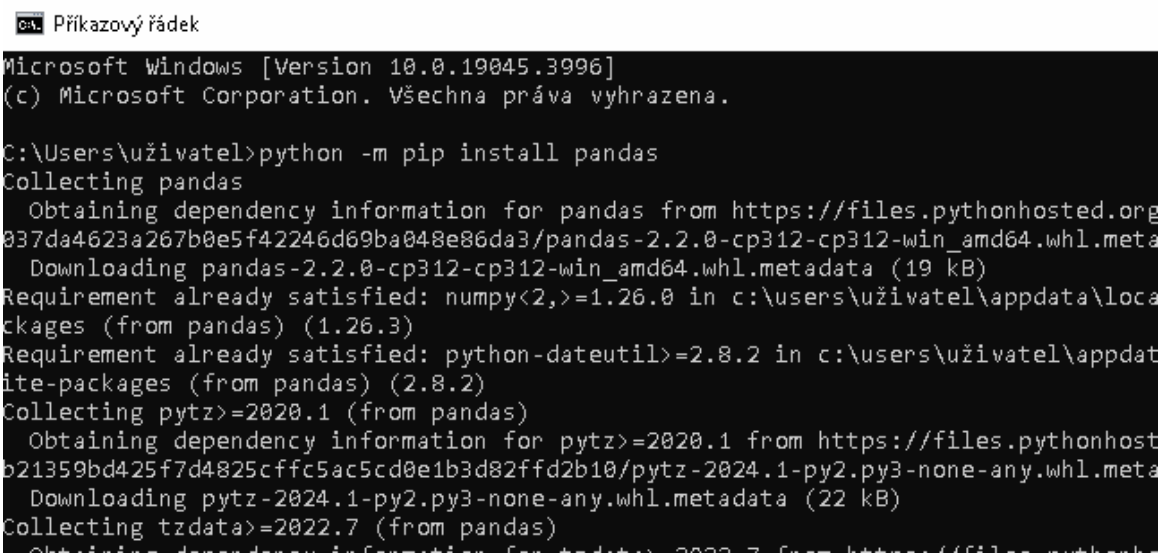


Tipy: plt.plot, plt.bar, plt.hist, plt.errorbar, plt.stem, plt.fill_between, …

3. Vědecko-technické výpočty a vi zualizace dat. Základy práce s reálnými daty, grafika 3-D. (1,5 h)

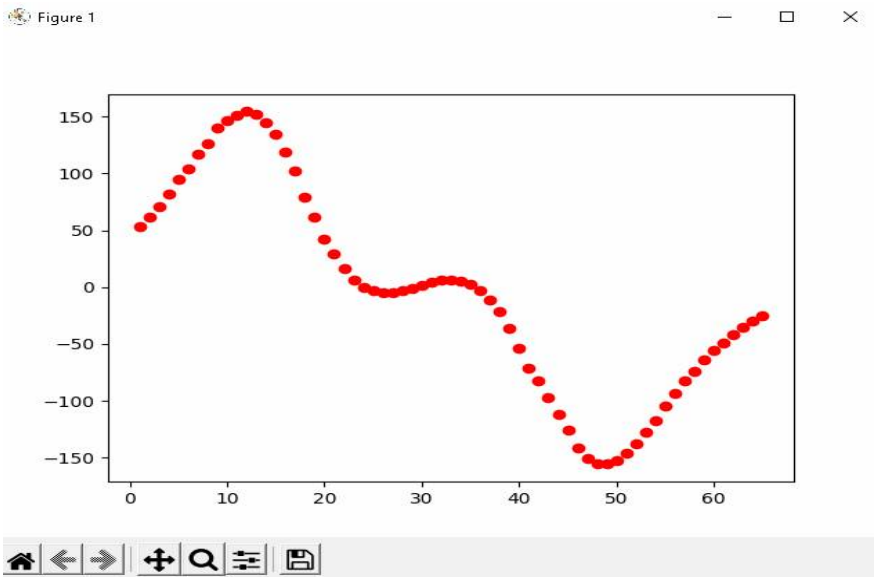
3.1. Instalace modulů *pandas* a *xlrd*

Instalace modulů Pandas a Xlrd, (práce s datovými sadami, čtení dat z MS® Excel®).
Při instalaci postupujeme analogicky jako u modulů v tématu 2.
Instalačními příkazy jsou:
python -m pip install pandas
python -m pip install xlrd



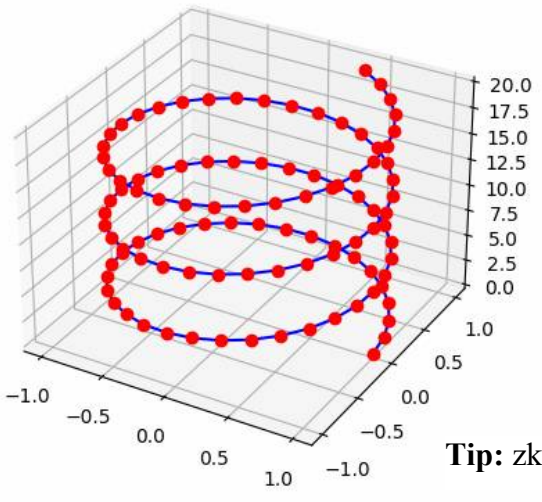
3.2. Čtení dat z MS Excel

```
# cteni dat z MS Excel
#---------------------
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_excel('Data_01.xls', sheet_name='List1',
                     skiprows=0, nrows=65, usecols='A:B') # cteni z Excel
print(data.shape)
data_1=data.iloc[:,0] # sloupec 1
data_2=data.iloc[:,1] # sloupec 2
print(data_1.shape)
plt.plot(data_1, data_2,'ro')
mat = np.column_stack((data_1, data_2)) # slouceni obou sloupcu do matice
np.savetxt('Mojedata_01.txt', mat) # ulozeni do txt souboru
plt.show()
```



3.6. Grafika 3-D

```
# carovy 3-D graf
#----------------
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure() # objekt figure (okno pro graf)
ax =plt.axes(projection='3d') # # objekt axex (platno)
osa_z = np.linspace(0, 20, 100) # 100 hodnot od 0 do 20
osa_y = np.sin(osa_z)
osa_x = np.cos(osa_z)
ax.plot3D(osa_x, osa_y, osa_z, 'blue') # plna cara
ax.plot3D(osa_x, osa_y, osa_z, 'ro') # cervene body
plt.show()
```

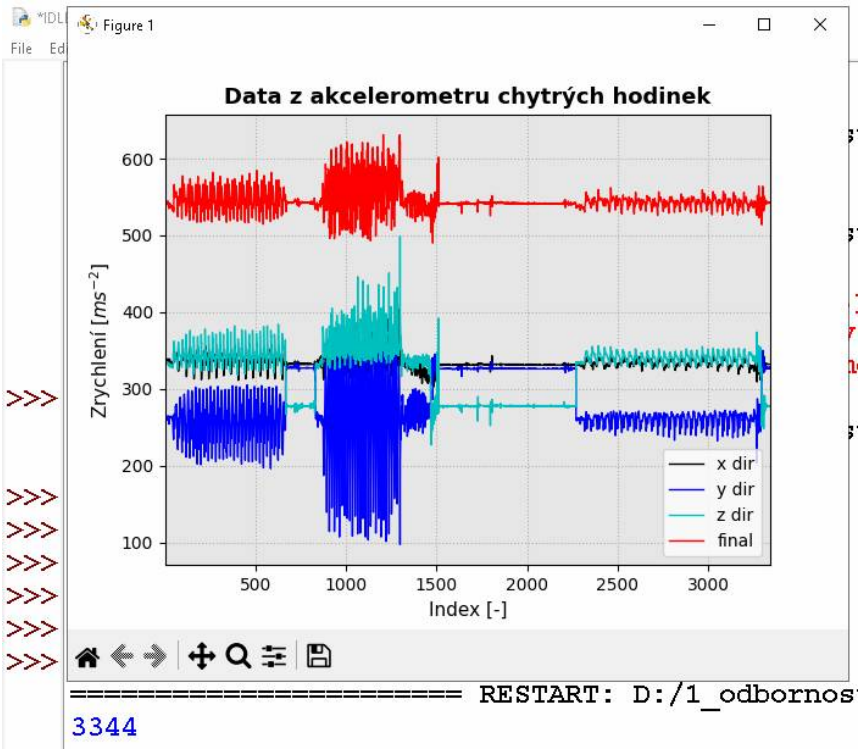


Tip: zkuste myší rotovat s grafem.

Kurz 3/5
Práce s daty z reálné praxe (3x1 h)

3. Data z akcelerometru chytrých hodinek (1 h)

```
# cteni dat z krokomeru
#-----------------------------
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#-------------
# cteni dat
#-------------
soubor='Krokomer.csv'
data = pd.read_csv(soubor,header=None) # v data je tzv. dataframe
data1=data.iloc[:,0] # pristup k danemu sloupci
data2=data.iloc[:,1]
data3=data.iloc[:,2]
dataall=np.sqrt(data1**2+data2**2+data3**2)
N=len(data1)
index=np.arange(1,N+1,dtype=int) # vektor poradovych indexu
print(N)
#---------
# kresleni
#---------
#data.plot() # kresli vsechny tri sloupce
#plt.show()
fig1 = plt.figure() # nove okno pro graf
ax1=plt.gca()
plt.plot(index, data1,'k-',linewidth=1)
plt.plot(index, data2,'b-',linewidth=1)
plt.plot(index, data3,'c-',linewidth=1)
plt.plot(index, dataall,'r-',linewidth=1)
ax1.set_facecolor([0.9,0.9,0.9]) # barva platna v RGB
plt.legend(['x dir','y dir','z dir','final'], loc='lower right') # legenda
plt.xlabel('Index [-]', fontsize=11)
```



Seznam modulů a příkazy pro jejich instalaci v příkazové řádce Windows (cmd):

numpy: python -m pip install pyarrow

Datové soubory, použité v kurzu 3:

Klima_FALUN.txt
Klima_LINKOEPING.txt
Vyhybka_1.txt
Vyhybka_2.txt
Hydraulika.txt
Krokomer.csv

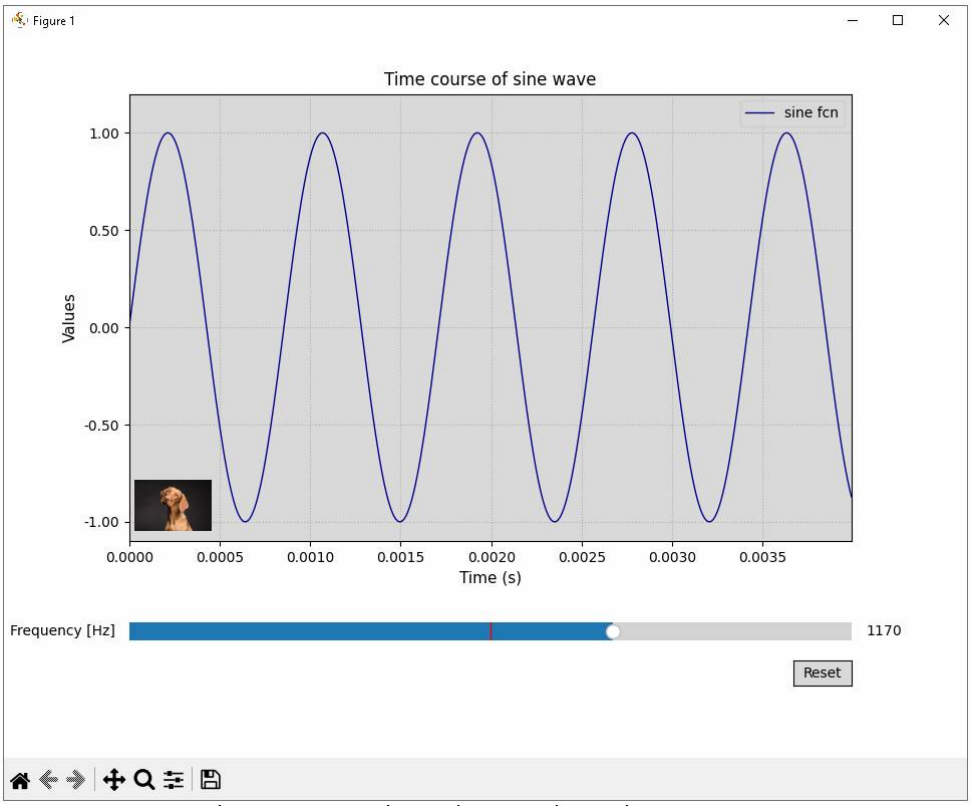
Kurz 4/5
Základy tvorby grafického uživatelského rozhraní (GUI) (1,5 + 1 h)

1. Tvorba GUI s využitím vestavěného modulu *tkinter* (1,5 h)

```
# python kurz 4
# tvorba GUI, modul tkinter
#---------------------------------
import tkinter as tk
from tkinter import ttk # lepsi styly widgetu nez tkinter
from PIL import ImageTk, Image # kvuli obrazku nize
import webbrowser
#----------------------------
#--- hlavni okno GUI ---
#----------------------------
top=tk.Tk() # hlavni okno GUI
top.title('python kurz 4, tvorba GUI') # nazev hlavniho okna GUI
top.geometry('400x300') # velikost hlavniho okna GUI
#top.config(bg='yellow') # barva hlavniho okna GUI (backgroudcolor)
#-----------------------------------------------
#--- alokace nekterych promennych ---
```

2. Ovládání ferekvence funkce sinus tahovým potenciometrem s využitím modulu *matplotlib* (1 h)

```
# python kurz 4
# ovladani frekvence grafu funkce sinus
# pomoci Scale (slider) a Button v matplotlib
#-------------------------------------------------------
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.widgets import Button, Slider
from matplotlib.ticker import StrMethodFormatter # pro pocet desetinnych mist
from PIL import ImageTk, Image # kvuli obrazku nize
#----------------------
#--- definice funkci ---
#----------------------
```



Kurz 5/5
Číselné soustavy (1,5 + 1 h)

1. Dvojková (binární) soustava (1,5 h)

Základ dvojkové soustravy: 2
Číslice (symboly): 0 1

Ilustrační příklad:

0	1	0	1	1	0	1	1
$0*2^7$	$1*2^6$	$0*2^5$	$1*2^4$	$1*2^3$	$0*2^2$	$1*2^1$	$1*2^0$

8 bitů = 1 Byte (bajt, slovo)
součet: $2^6 + 2^4 + 2^3 + 2^1 + 2^0 = 91$

Příklad 1.1

Převeďte desítkové (dekadické) číslo +458 na dvojkové (binární).
Symbolicky:

$$(+458)_{10} \rightarrow X_2$$

a) analyticky (papír a tužka) metodou dělení číslem 2 (tzv. Euklidův algoritmus):

```
458:2=229  zbytek 0  (LSB)
229:2=114  zbytek 1
114:2= 57  zbytek 0
 57:2= 28  zbytek 1
 28:2= 14  zbytek 0
 14:2=  7  zbytek 0
  7:2=  3  zbytek 1
  3:2=  1  zbytek 1
  1:2=  0  zbytek 1  (MSB)
```

python: **int(458/2), 458//2, 458%2**
print(int(458/2),458%2)
229 0

LSB: *Least Significant Bit* (nejméně významný bit)
MSB: *Most Significant Bit* (nejvýznamnější bit)

Výsledek: $(+458)_{10} \rightarrow (111001010)_2$

b) řešení v python:

```
b1=bin(458)
print(b1)
    0b111001010
type(b1)
    <class 'str'>

print('{:b}'.format(458))
    111001010
print('{0:016b}'.format(458))
    0000000111001010

b2=int(bin(458).replace('0b',''))
print(b2)
    111001010
b3=format(458,'b')
print(b3)
    111001010
print(f'{458:b}')
    111001010
print(f'{458:016b}')
    0000000111001010
```

```
>>> b1=bin(458)
>>> print(b1)
0b111001010
>>> type(b1)
<class 'str'>
```

```
>>> b1=bin(458)
>>> b2=b1[2:]
>>> print('Číslo 458 dvojkově:',b2)
Číslo 458 dvojkově: 111001010
```

Příklad 1.2

Převeďte desítkové (dekadické) číslo +13 na dvojkové (binární) na 8 bitů (1 Byte).
Symbolicky:

$$(+13)_{10} \rightarrow X_2$$

a) analyticky (papír a tužka) metodou dělení číslem 2 (tzv. Euklidův algoritmus):

```
13:2= 6   zbytek 1 (LSB)      python: int(13/2), 13//2, 13%2
 6:2= 3   zbytek 0                    print(int(13/2),13%2)
 3:2= 1   zbytek 1
 1:2= 0   zbytek 1 (MSB)
```

Výsledek: $(+13)_{10} \rightarrow (1101)_2$

b) řešení v python:

```
b1=bin(13)                          b2=int(bin(13).replace('0b',''))
print(b1)                           print(b2)
    0b1101                              1101
type(b1)                            b3=format(13,'b')
    <class 'str'>                   print(b3)
                                        1101
print('{:b}'.format(13))            print(f '{13:b}')
    1101                                1101
print('{0:016b}'.format(13))        print(f '{13:08b}')
    0000000000001101                    00001101
```

```
>>> b1=bin(13)
>>> print(b1)
0b1101
>>> type(b1)
<class 'str'>
```

```
>>> d1=13
>>> b1=bin(d1)
>>> b2=b1[2:]
>>> print('Číslo',d1,'dvojkově:',b2)
Číslo 13 dvojkově: 1101
```

Příklad 1.3

Převeďte desítkové číslo -71 na dvojkové (binární) na 16 bitů (2 Bytes).
Symbolicky:

$$(-71)_{10} \rightarrow X_2$$

a) analyticky s využitím tzv. dvopjkového doplňku (*two's complement*),

- vyjádříme nejprve číslo kladné dělím 2:

$$(+71)_{10} \rightarrow X_{21}$$

```
71:2= 35  zbytek 1 (LSB)      python: int(71/2), 71//2, 71%2
35:2= 17  zbytek 1                    print(int(71/2),71%2)
17:2=  8  zbytek 1
 8:2=  4  zbytek 0
 4:2=  2  zbytek 0
 2:2=  1  zbytek 0
 1:2=  0  zbytek 1 (MSB)
```

Dílčí výsledek: $(+71)_{10} \rightarrow (0000000001000111)_2$

• znegujeme všechny číslice (symboly):

$(1111111110111000)_2$

• přičteme číslo 1 k LSB (zcela vpravo):

$(1111111110111001)_2$

Výsledek: $(-71)_{10} \rightarrow (1111111110111001)_2$

b) symbolické vyjádření záporného čísla

Výsledek: $(-71)_{10} \rightarrow (-0000000001000111)_2$

c) řešení v python: **b1=bin(-71)** **print(f '{-71:016b}')**
print(b1) -000000001000111
-0b1000111

```
>>> d1=-71
>>> b1=bin(d1)
>>> print('Číslo',d1,'dvojkově:',b1)
Číslo -71 dvojkově: -0b1000111
>>>
>>> print(f'{d1:016b}')
-000000001000111
```

Poznámka: python vnitřně používá pro záporná čísla dvojkový doplně. Protože však pro celá čísla má nekonečnou přesnost, dává tam symbolicky znaménko -.

Literatura a odkazy

Vybrané webové stránky

https://www.rapidtables.com/convert/ a níže: Hex/decimal/octal/binary converter
https://www.python.org
https://python.cz
https://www.fullstackpython.com
https://realpython.com

Publikace

Na českém i světovém trhu je k dostání množství tištěné i elektronické literatury.