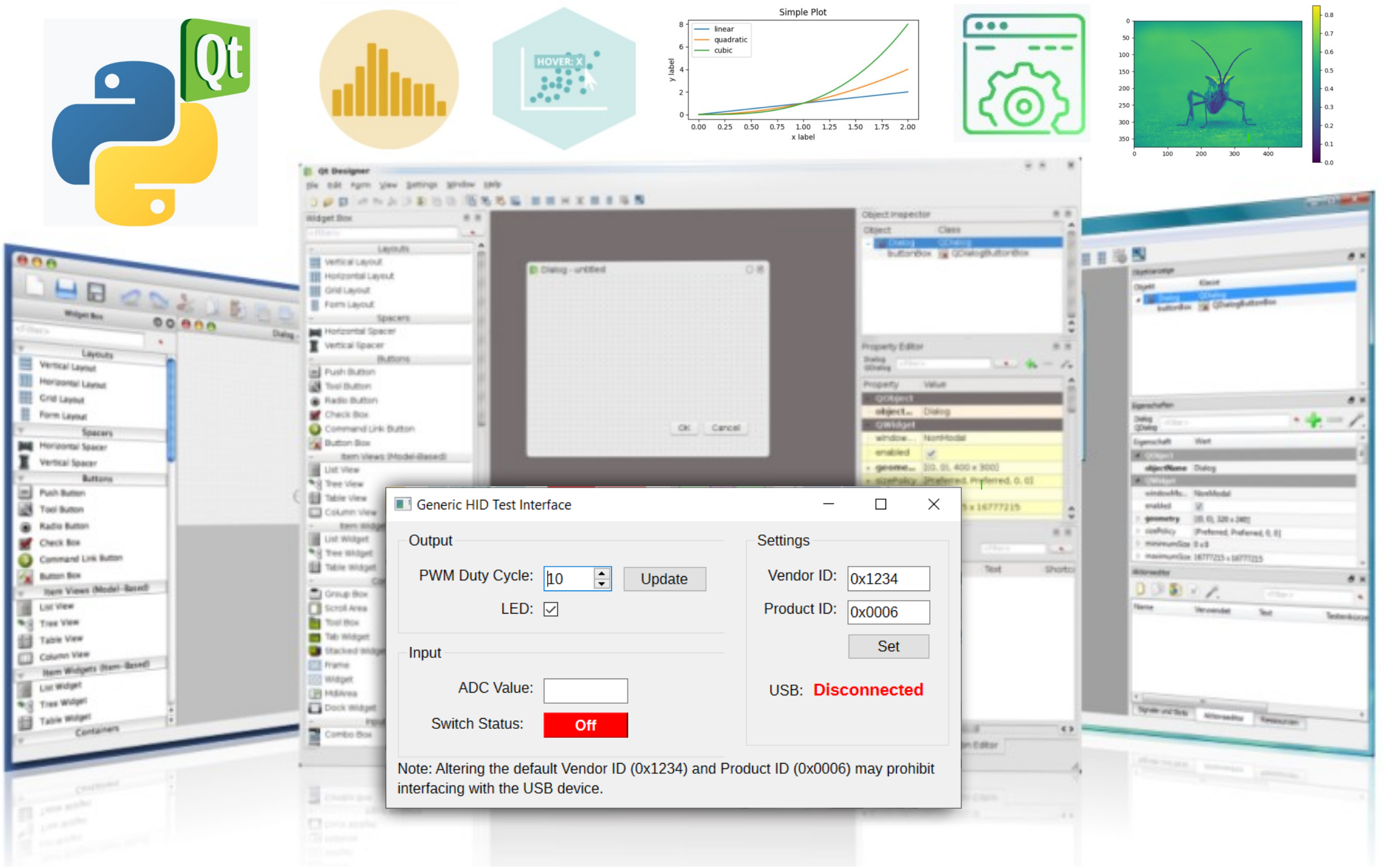
4. Python grafikus bővítmények használata



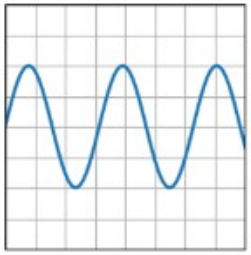
Felhasznált és ajánlott irodalom

- Mark Pilgrim/Kelemen Gábor: Ugorj fejest a Python 3-ba!
- P. Wentworth et al. (ford. Biró Piroska, Szeghalmy Szilvia és Varga Imre): Hogyan gondolkozz úgy, mint egy informatikus: Tanulás Python 3 segítségével
- Burkhard A. Meier: **Python GUI Programming Cookbook**
- Martin Fitzpatrick: Create GUI Applications with Python and Qt6 (Pyside6)
- **Matplotlib tutorial**
- **How To Plot A Line Graph In Python**
- Pillow - Python Image Library (PIL fork)
- Pillow Handbook
- Tkinter tutorial
- wxPython Tutorial
- **Pyside6 tutorial**
- **Qt6 for Python Documentation**

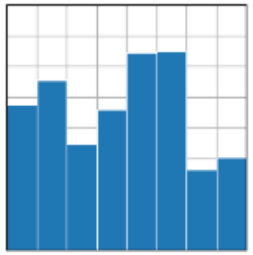
Grafikus könyvtárak és keretrendszerek

- A Python programok grafikus elemekkel való kiegészítése többféle módon és célirányban végezhető

Adatok ábrázolása (diagramok)

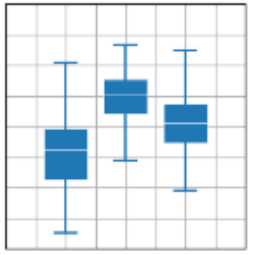


plot(x, y)



bar(x, height)

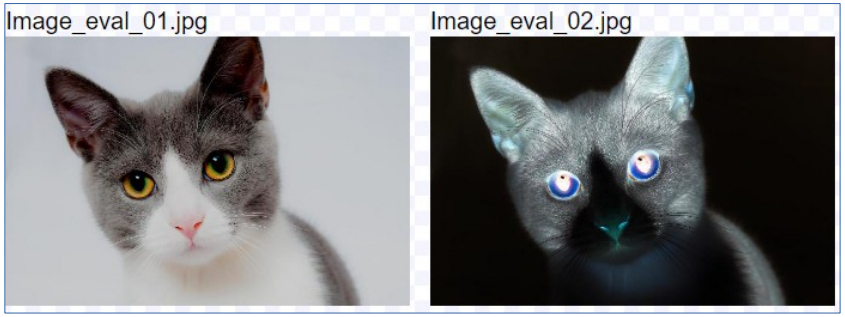
contour(X, Y, Z)



boxplot(X)

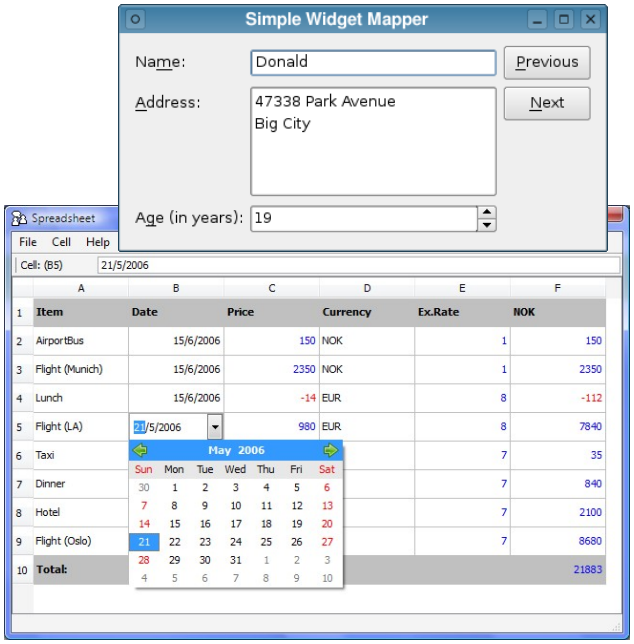
(Matplotlib, Pandas)

Médiafájlok kezelése és feldolgozása



(PIL, Pillow)

GUI alkalmazások (grafikus felhasználói felület)



(Tkinter, wxPython, PyQt)

A matplotlib kiegészítő könyvtár használata

- Telepítése:

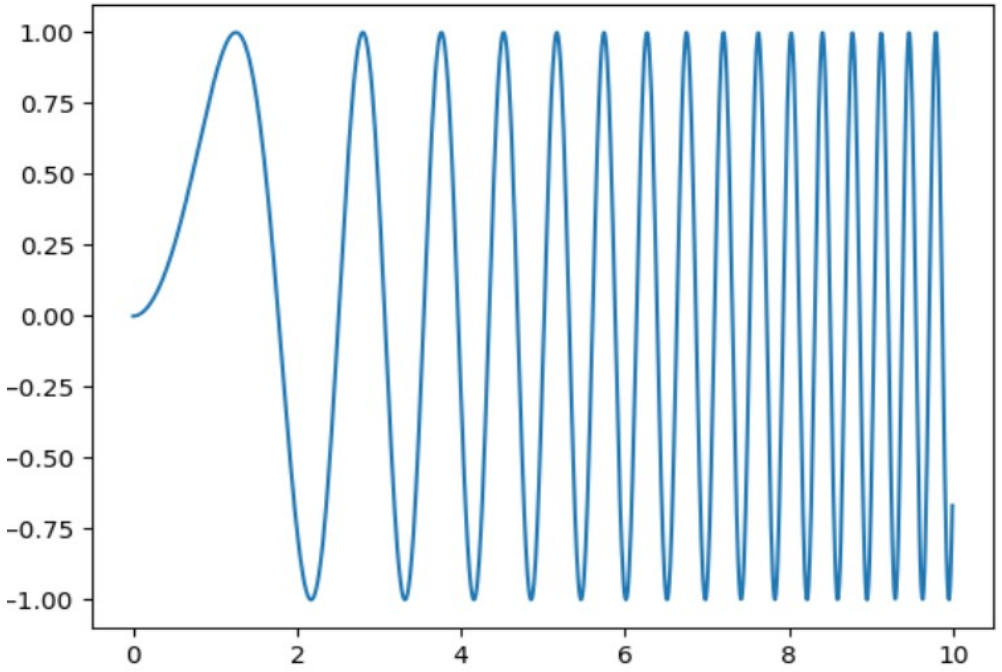
```
pip install numpy
pip install matplotlib
```

- Importálás:

```
import numpy as np     # Csak a számoláshoz kell
import matplotlib.pyplot as plt
```

- A 2022. március 24-i előadásban már bemutatott példa:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,10,0.01)
y = np.sin(x*x)
plt.plot(x,y)
plt.show()
```



- További példák: **Pyplot tutorial**,

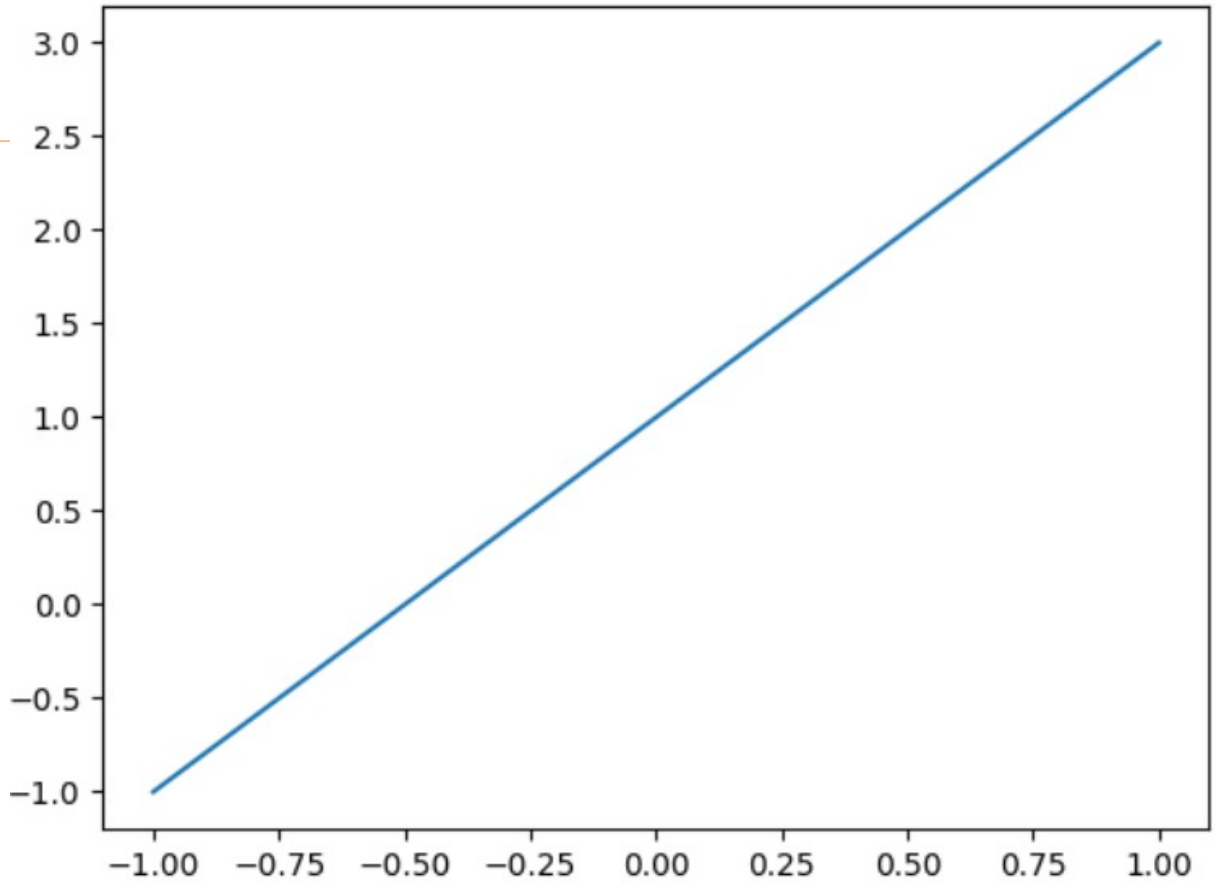
ill. **pyplot.ipynb** *(Jupyter notebook)*

Egyszerű függvény ábrázolása

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(-1, 1, 51)
y = 2*x + 1
plt.plot(x,y)
plt.show()
```

`x = np.linspace(-1,1,51)` eredménye

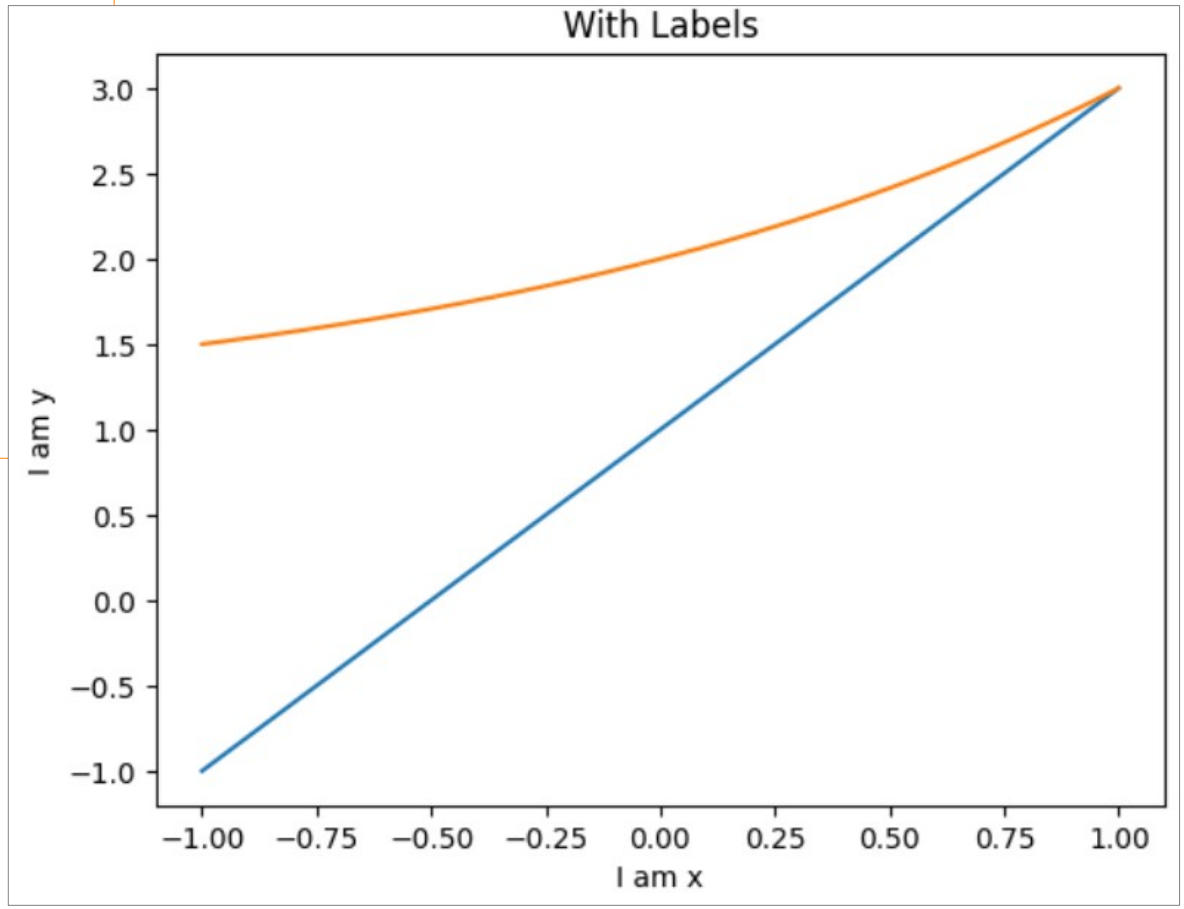
```
[-1.   -0.96 -0.92 -0.88 -0.84 -0.8  -0.76 -0.72 -0.68 -0.64 -0.6  -0.56
 -0.52 -0.48 -0.44 -0.4  -0.36 -0.32 -0.28 -0.24 -0.2  -0.16 -0.12 -0.08
 -0.04  0.    0.04  0.08  0.12  0.16  0.2   0.24  0.28  0.32  0.36  0.4
  0.44  0.48  0.52  0.56  0.6   0.64  0.68  0.72  0.76  0.8   0.84  0.88
  0.92  0.96  1.  ]
```



Ábra készítése feliratokkal

```
x = np.linspace(-1, 1, 50)
y1 = 2*x + 1
y2 = 2**x + 1

plt.figure()
plt.plot(x, y1)
plt.plot(x, y2)

plt.xlabel("I am x")
plt.ylabel("I am y")
plt.title("With Labels")

plt.show()
```

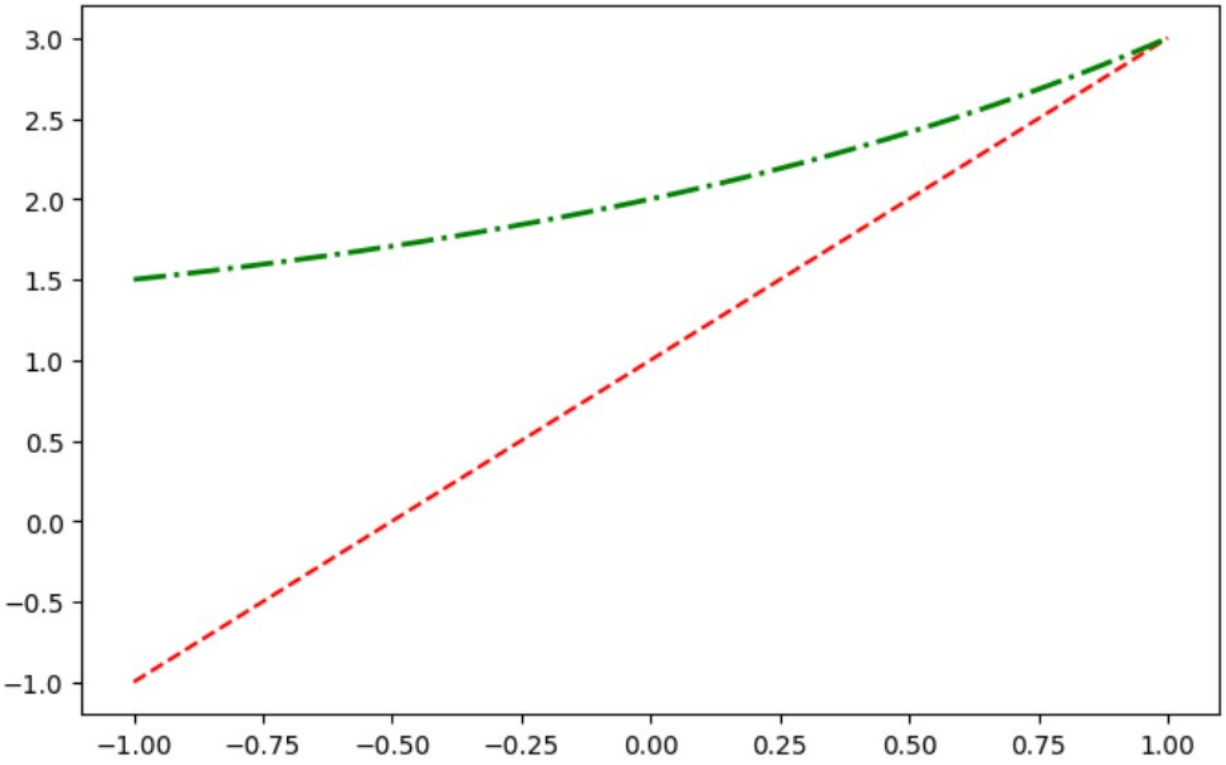
- A `plt.figure()` hívása opcionális, kihagyása esetén egy ábra és a tengelyek automatikusan létrejönnek



Ábra tulajdonságok megadása

```
x = np.linspace(-1, 1, 50)
y1 = 2*x + 1
y2 = 2**x + 1

plt.figure(num = 3,
figsize=(8, 5))

plt.plot(x, y1,
         color='red',
         linewidth=1.5,
         linestyle='--'
        )
plt.plot(x, y2,
         color='green',
         linewidth=2.0,
         linestyle='-.'
        )
plt.show()
```

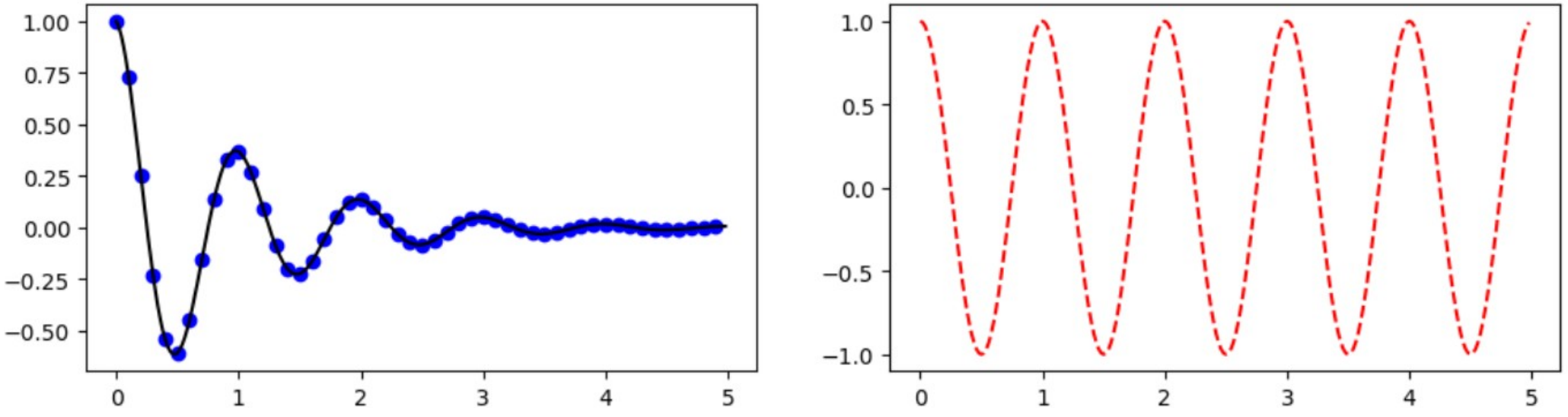
- Az ábra méretét *inch*-ben lehet megadni
- A görbék tulajdonságait (szín, vonalstílus) rövidebben is meglehet adni (lásd a következő oldalon)



Több diagram egy ábrában

```
def f(t):
    return np.exp(-t)*np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)
plt.figure(figsize=(12,3))
plt.subplot(121)
plt.plot(t1,f(t1),'bo',t2,f(t2),'k')
plt.subplot(122)
plt.plot(t2,np.cos(2*np.pi*t2),'r--')
plt.show()
```

- `subplot` három paramétere a sorok és az oszlopok száma, illetve az al-ábra sorszáma
- A vesszők elhagyhatók, így a 121 valójában 1,2,1 a 122 pedig 1, 2, 2

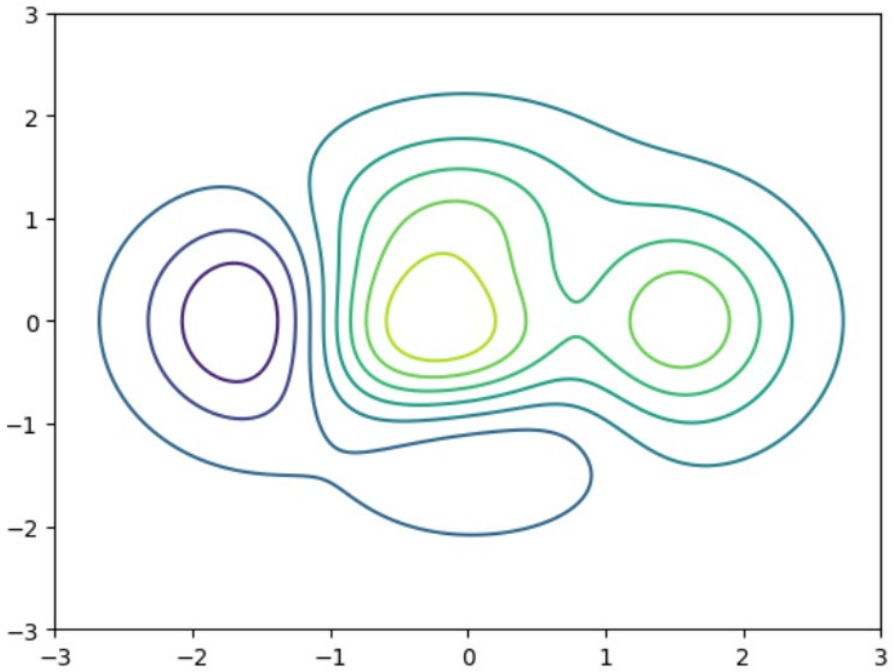


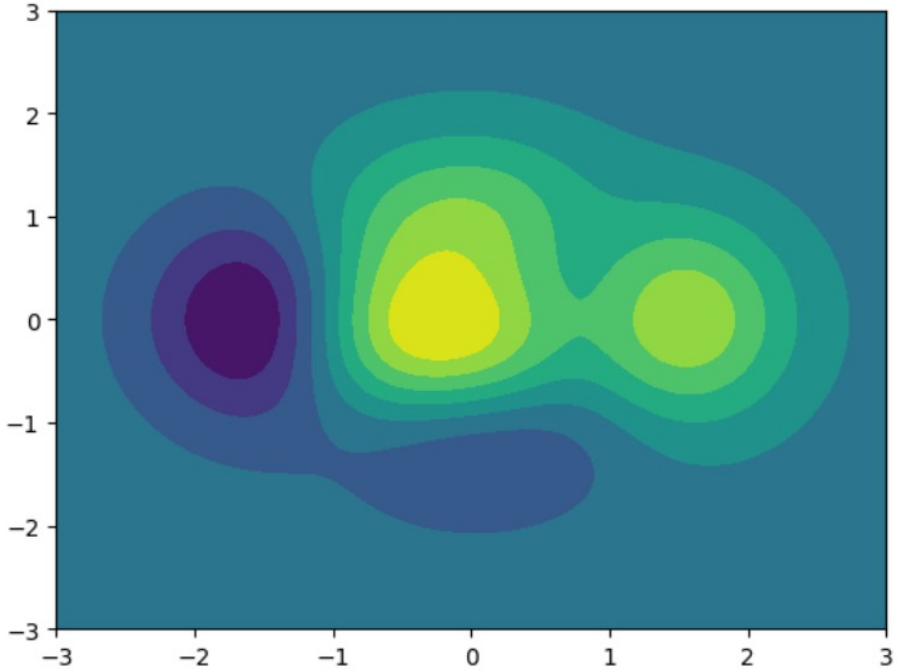
Kontúrdiagram (szintvonalas ábrázolás)

```
X, Y = np.meshgrid(np.linspace(-3, 3, 256), np.linspace(-3, 3, 256))
Z = (1 - X/2 + X**5 + Y**3) * np.exp(-X**2 - Y**2)
levels = np.linspace(np.min(Z), np.max(Z), 10)

fig, ax = plt.subplots()
ax.contour(X,Y,Z,levels=levels)
plt.show()
```

Szintvonalak száma

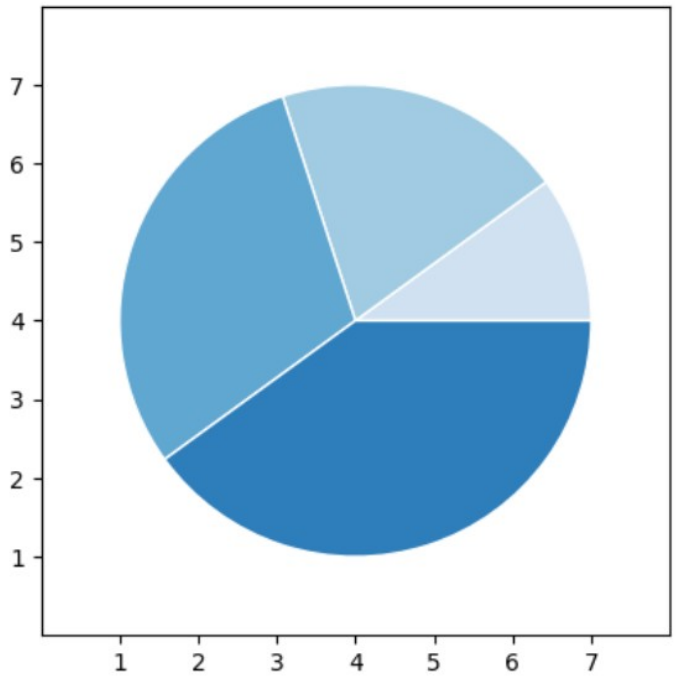
```
ax.contourf(X,Y,Z,levels=levels)
```





Tortadiagram (Pie chart)

```
x = [1, 2, 3, 4]
colors = plt.get_cmap('Blues')(np.linspace(0.2, 0.7, len(x)))
fig, ax = plt.subplots()
ax.pie(x, colors=colors, radius=3, center=(4, 4),
       wedgeprops={"linewidth":1, "edgecolor":"white"}, frame=True)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))
plt.show()
```

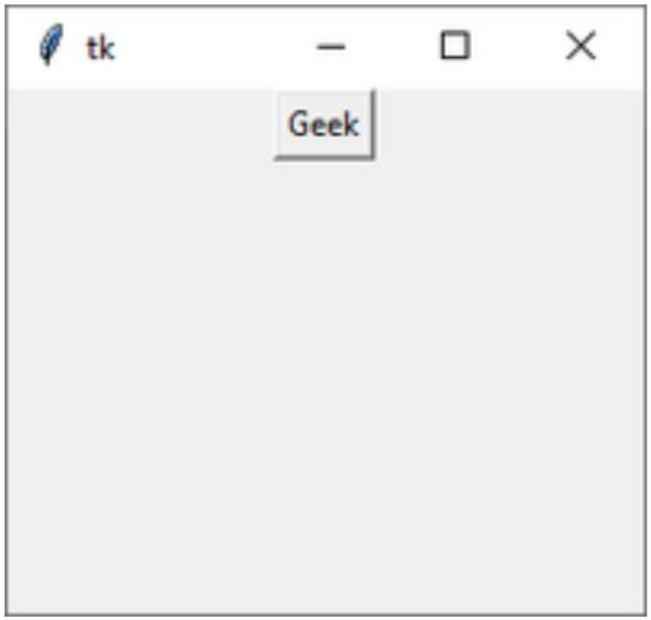


Python GUI programozás

- **GUI** (Graphical User Interface) alkalmazások fejlesztéshez Pythonban a **Tkinter** a leggyakrabban használt könyvtár, ami lényegében egy Python interfész a **Tk** GUI eszközkészlethez
- A **GUI** könyvtárak, ahogy a **Tkinter** is, különféle GUI elemeket (*widget*) biztosítanak a felhasználói kezelőfelületek kialakításához – például gombokat, menüket, beviteli mezőket és megjelenítési területeket
- A **widget** elnevezés a ***windows*** (ablak) és a ***gadget*** (bigyó) szavak összevonásából ered

```
from tkinter import *
root = Tk()             # create root window

frame = Frame(root)     # frame inside root window
frame.pack()            # geometry method

# button inside frame which is inside root
button = Button(frame, text ='Geek')
button.pack()

root.mainloop()         # Tkinter event loop
```

Output :



Tkinter widget osztályok

- Label — Rövid felirat megjelenítése a képernyőn
- Button — Kattintható nyomógombbal látja el az elkalmazást
- Canvas — Kép vagy szöveg megjelenítésére kijelölt terület
- ComboBox — Legördíthető lista és beírható szövegdoboz kombinációja
- CheckButton — Kipipálható jelölőnégyzet
- Radio Button — Választógomb (csak egy lesz aktív)
- Entry — Szövegbeviteli mező
- Frame — Widgeteket tároló konténer
- Message — Megjelenthető szöveg, ami többsoros is lehet
- Scale — Adatbeállító csúszka
- Scrollbar — Szöveg görgetésére szolgál, egy függőleges csúszkát is tartalmaz
- SpinBox — A felhasználó nyilakkal léptetheti a beírt értéket
- Text — Többsoros szöveg szerkesztésére és megjelenítésére szolgál
- Menu — Alkalmazói menü kialakítására szolgál

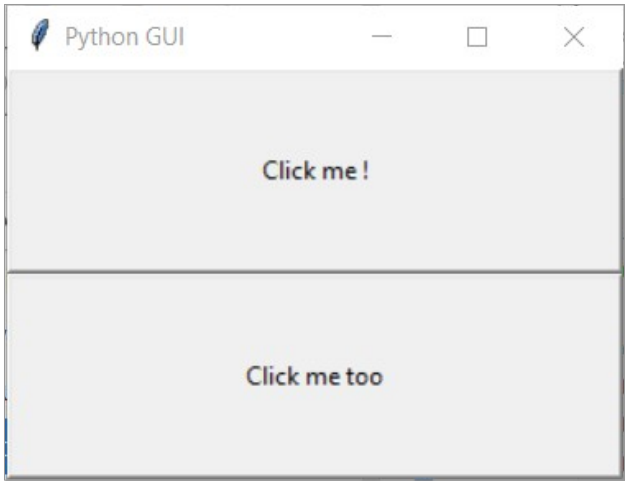
További információ: geeksforgeeks.org/what-are-widgets-in-tkinter/

Tkinter GUI alkalmazások

- **Tkinter** GUI alkalmazás létrehozásának lépései:
  - A **tkinter** modul importálása
  - A főablak (container) létrehozása
  - Adjunk widgeteket a főablakhoz
  - Alkalmazzunk eseményindítót a widgeteken
- **Tkinter** widgetek pozicionálási lehetőségei
  - **Pack()** módszer – a widgeteket a korábbiakhoz képest helyezi el (módosítók: fill, expand és side)
  - **Grid()** módszer – a konténer több sorra és oszlopra van felosztva, és a táblázat minden egyes „cellája” tartalmazhat egy widgetet.
  - **Place()** módszer – a widgeteket a programozó által meghatározott pozíciókba helyezi
- **Tkinter** eseményvezérlés
  - Visszahívási függvényt kell definiálni a kívánt akció végrehajtására
  - Hozzá kell rendelni a függvényt egy eseményhez (command paraméter)

Tkinter.Button és a pack() módszer

```
from tkinter import *
win = tk.Tk()
win.title("Python GUI")
win.geometry('300x200')
pane = Frame(win)
pane.pack(fill = BOTH, expand = True)

b1 = Button(pane, text = "Click me !")
b1.pack(fill = BOTH, expand = True)
b2 = Button(pane, text="Click me too")
b2.pack(fill = BOTH, expand = True)

win.mainloop() # Execute Tkinter
```




```
from tkinter import *
win = tk.Tk()
win.title("Python GUI")
win.geometry('400x60')
pane = Frame(win)
pane.pack(fill = BOTH, expand = True)
b1 = Button(pane,text="Click me !")
b1.pack(side = LEFT, expand = True,
fill = BOTH)
b2 = Button(pane,text="Click me too")
b2.pack(side = LEFT, expand = True,
fill = BOTH)
win.mainloop() # Execute Tkinter
```

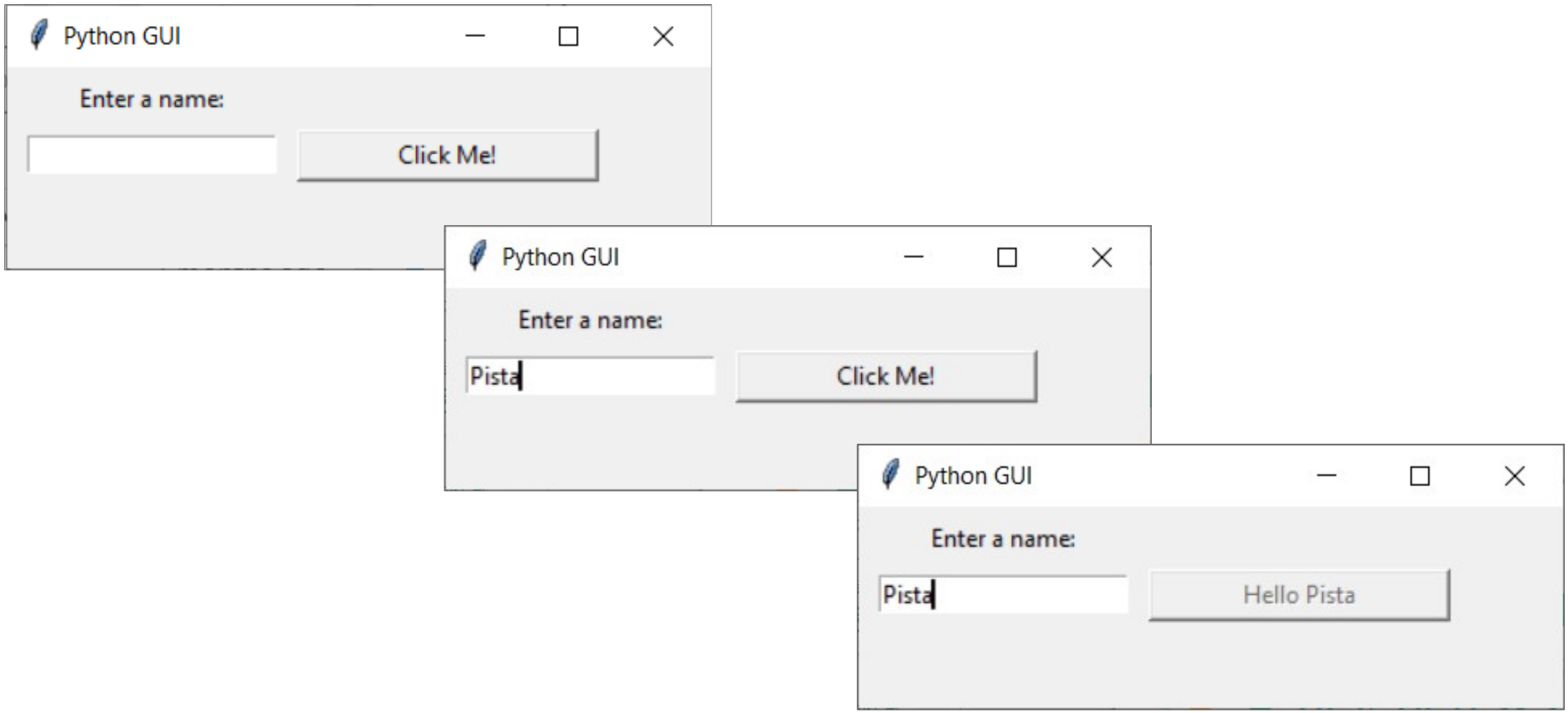
Nyomógomb kattintási esemény

```
import tkinter as tk

def click_me():                                 # Button Click Function
    action.configure(text='Hello ' + name.get())
    action.configure(state='disable')           # disable the Button Widget

win = tk.Tk()
win.title("Python GUI")
win.geometry('350x200')

# Adding a Label
tk.Label(win, text="Enter a name:").grid(column=0, row=0, pady=5)

name = tk.StringVar()                           # Adding a Textbox Entry widget
name_entered = tk.Entry(win, width=20, textvariable=name)
name_entered.grid(column=0, row=1, padx=10)

# Adding a Button
action = tk.Button(win, text="Click Me!", command=click_me, width=20)
action.grid(column=1, row=1)
action.configure(state='active')                # Enasable the Button Widget
name_entered.focus()                            # Place cursor into name Entry

win.mainloop()
```

A nyomógomb lenyomási eseményéhez hozzárendeljük az általunk definiált **click_me()** függvényt
A pozicionálás most a **grid()** módszerrel történik

Nyomógomb kattintási esemény

- A név beírása és a nyomógomb kattintása után kicserélődik a nyomógomb felirata és egyúttal letiltásra is kerül



Python Qt6 (PySide6)

- A **PyQt6**, illetve a **PySide6** a közismert **Qt6** C++ GUI keretrendszerhez készített **Python** csatolófelület
- A **PyQt6**, illetve a **PySide6** között csupán apró eltérések vannak. Ebben az előadásban mi az utóbbit használjuk
- Telepítés: `pip install PySide6`
- Tananyag: **pythonguis.com/pyside6/**
- Dokumentáció: **Qt6 for Python Documentation**

	PyQt6	PySide6
First stable release	Jan 2021	Dec 2020
Developed by	Riverbank Computing Ltd.	Qt
License	GPL or commercial	LGPL
Platforms	Python 3	Python3

Egyszerű Qt6 GUI mintapélda: app001.py

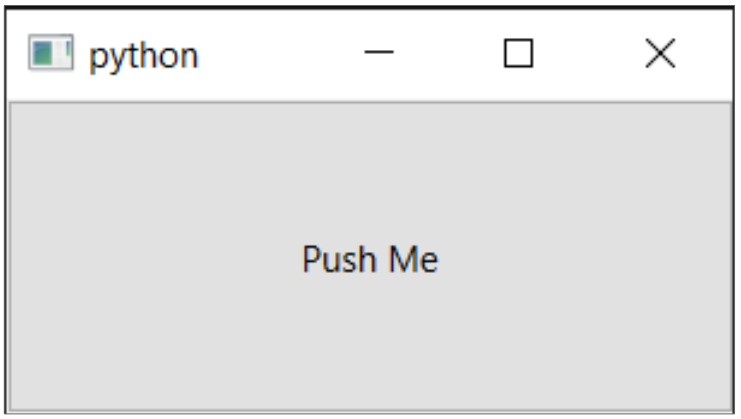
```
from PySide6.QtWidgets import QApplication, QWidget
import sys

# Egytelen QApplication példány lehet egy alkalmazásban
app = QApplication(sys.argv)



# Legalább egy Qt widget kell, ez lesz az alkalmazás fő ablaka
window = QWidget()
window.show()  # Az ablakot láthatóvá kell tenni

# Az eseménykezelő hurok indítása
app.exec()
```

- A program csak egy üres ablakot jelenít meg…
- Futtatás:
  `python app001.py`

GUI mintapélda nyomógombbal: app002.py

```
import sys
from PySide6.QtWidgets import QApplication, QPushButton

app = QApplication(sys.argv)

window = QPushButton("Push Me")
window.pressed.connect(window.close)
window.show()

app.exec()
```

- A főablak bármilyen **widget** lehet, itt most egy nyomógombot neveztünk ki
- A nyomógombnak lehet felirata, vannak eseményei (itt: *pressed*), s az eseményhez saját, vagy előre definált végrehajtandó függvényt köthetünk (connect), ami az eseménykor lefut



Egy másik megközelítés: app004.py

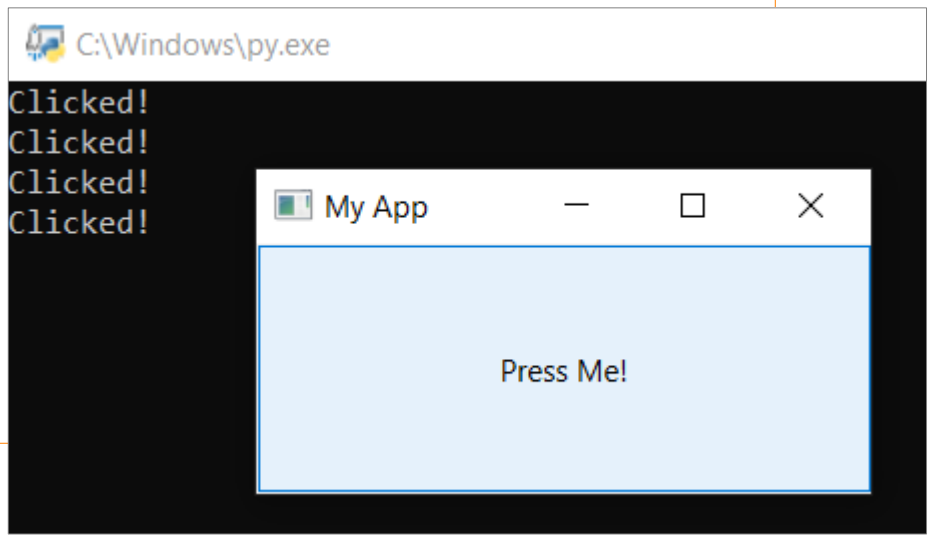
- A leszármaztatással megoldhatjuk, hogy saját inicializálással felüldefiniáljuk a fő ablak alapértelmezett tulajdonságait

```
import sys
from PySide6.QtCore import QSize, Qt
from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton

# Subclass QMainWindow to customize your application's main window
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        button.pressed.connect(self.close)
        self.setFixedSize(QSize(400, 300))
        # Set the central widget of the Window.
        self.setCentralWidget(button)

app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Saját eseménykezelő: app005.py

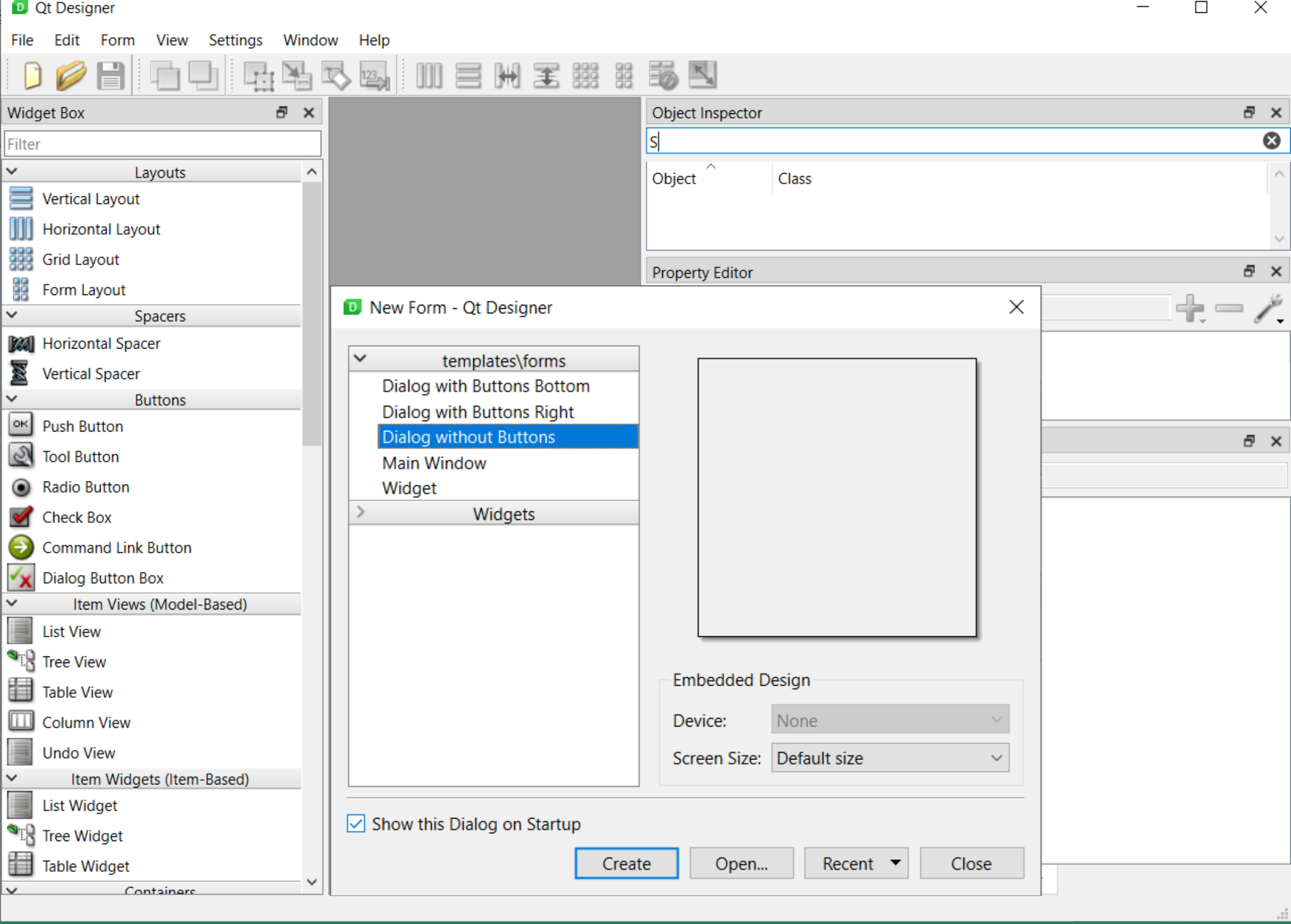
```
import sys
from PySide6.QtWidgets import QApplication, QMainWindow, QPushButton

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("My App")
        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_clicked)
        # Set the central widget of the Window.
        self.setCentralWidget(button)

    def the_button_was_clicked(self):
        print("Clicked!")


app = QApplication(sys.argv)
window = MainWindow()
window.show()
app.exec()
```

Minden lenyomáskor kiíratjuk, hogy: **Clicked!**



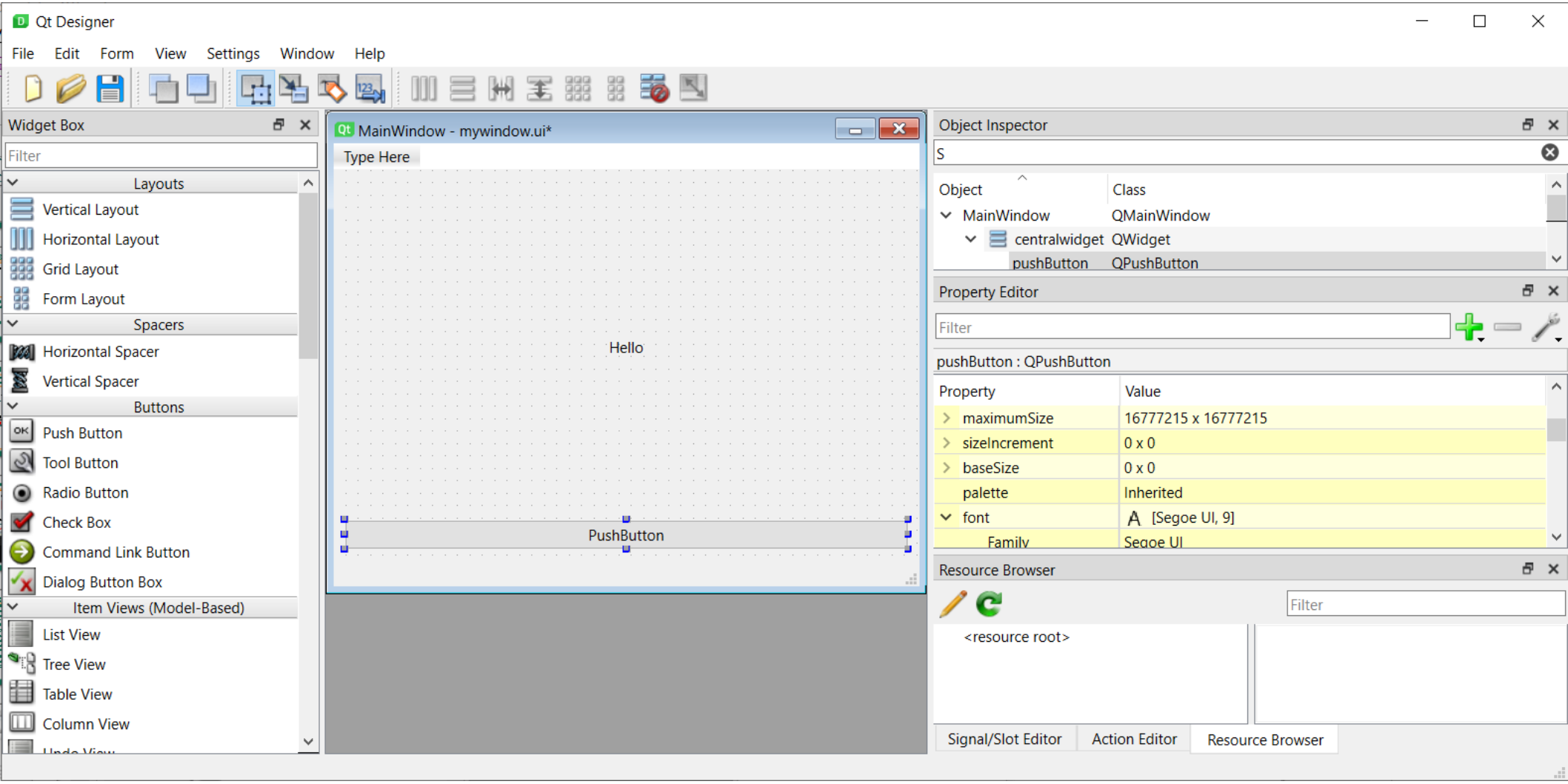
Qt Designer

- A **PySide6** telepítési mappájában található **designer.exe** program egy **Qt Designer** tervezőprogram, amellyel kényelmesen megszerkeszthetjük az alkalmazásainak komplex elrendezéseit
  (nálam **c:\Users\cserny\AppData\Local\Programs\Python\Python310\Lib\site-packages\PySide6**)
- A Python kódok generálásához az **uic.exe** segédprogramot egy **bin** almappába is be kell másolni ugyanott, mert ott keresi…
- A **Qt Designer .ui** kiterjesztésű **Xml** fájlokat készít, ezeket közvetlenül betölthetjük az alkalmazásainkba, de talán természetesebb az **.ui** állományokból Python kódot generálni, s a kapott állományt osztálykönyvtárként importálhatjuk
- A **Qt Designer** segítségével készített osztálykönyvtárakból leszármaztatással olyan objektumosztályt készíthetünk, ami saját függvényekkel tovább bővíthető – így a kinézet és a tevékenység könnyen szeparálható


Qt Designer
File Edit Form View Settings Window Help
Widget Box
Filter
Layouts
Vertical Layout
Horizontal Layout
Grid Layout
Form Layout
Spacers
Horizontal Spacer
Vertical Spacer
Buttons
Push Button
Tool Button
Radio Button
Check Box
Command Link Button
Dialog Button Box
Item Views (Model-Based)
List View
Tree View
Table View
Column View
Undo View
Item Widgets (Item-Based)
List Widget
Tree Widget
Table Widget
Containers
Object Inspector
Object
Class
Property Editor
New Form - Qt Designer
templates\forms
Dialog with Buttons Bottom
Dialog with Buttons Right
Dialog without Buttons
Main Window
Widget
Widgets
Embedded Design
Device:
None
Screen Size:
Default size
Show this Dialog on Startup
Create
Open...
Recent
Close

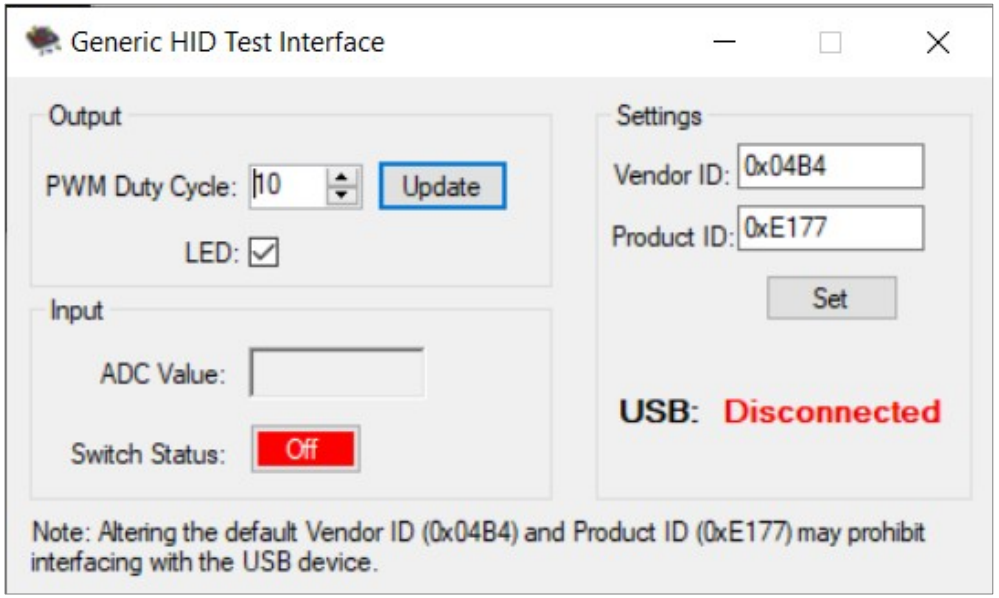
Qt Designer

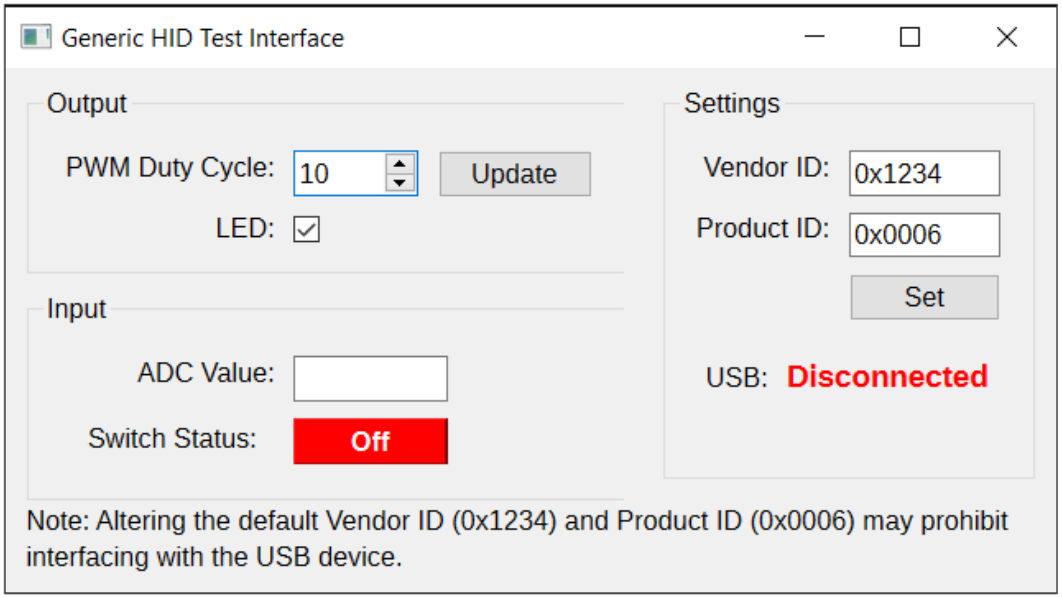
- A baloldali eszköz menüből áthúzzuk és konfiguráljuk a GUI elemeket. A tulajdonságokat a jobboldali panelon állíthatjuk be



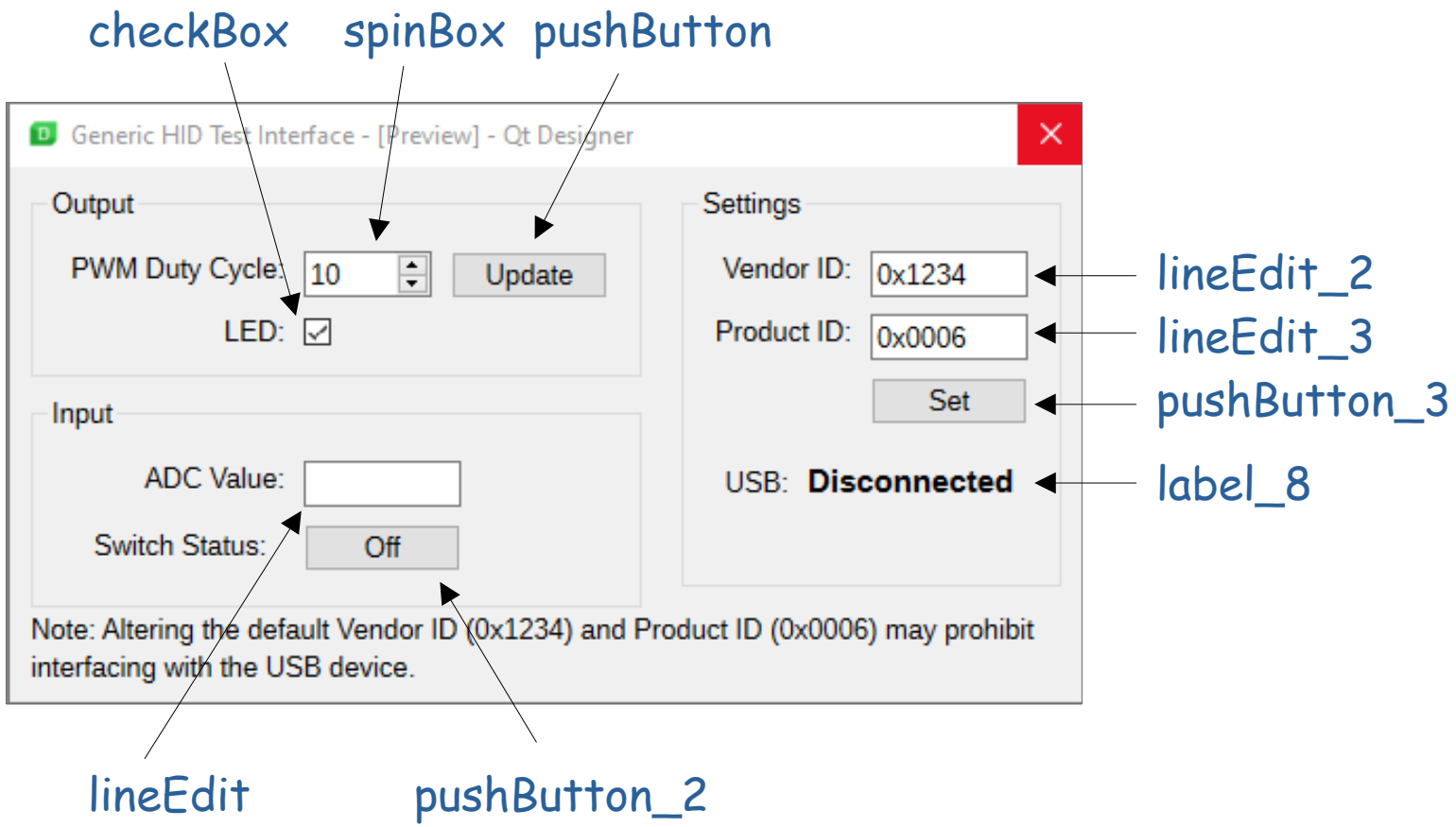
A tervezett alkalmazás

- A cél a **Cypress AN82 072** alkalmazási mintapéldában található C# alkalmazás kiváltása
- 1. lépés: az elrendezésének megtervezése **Qt Designerrel**
- 2. lépés: a működtető programrész és az USB HID eszköz kezelésének megírása
- Az ábrán baloldalt az eredeti C# alkalmazás, jobbról az általunk készített elrendezés látható





A használat során hivatkozott GUI elemek



A működtető alkalmazás demó változata

HIDTest_app.py - 2/1.

```
import sys
from PySide6 import QtWidgets

from ui_HIDTest import Ui_Dialog


class MainWindow(QtWidgets.QMainWindow, Ui_Dialog):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.setupUi(self)
        #self.setWindowTitle("USB generic HID demo")
        self.label_8.setStyleSheet("color: red")
        self.pushButton_2.setStyleSheet("color: white; background-color: red")
        self.pushButton.clicked.connect(self.on_update)
        self.pushButton_3.clicked.connect(self.on_set)
        self.checkBox.stateChanged.connect(self.on_check)

    def on_update(self):
        pwm_data = int(self.spinBox.value())
        # Just echoing the data ...
        self.lineEdit.setText("0x{0:04X}".format(pwm_data))
```

A működtető alkalmazás demó változata

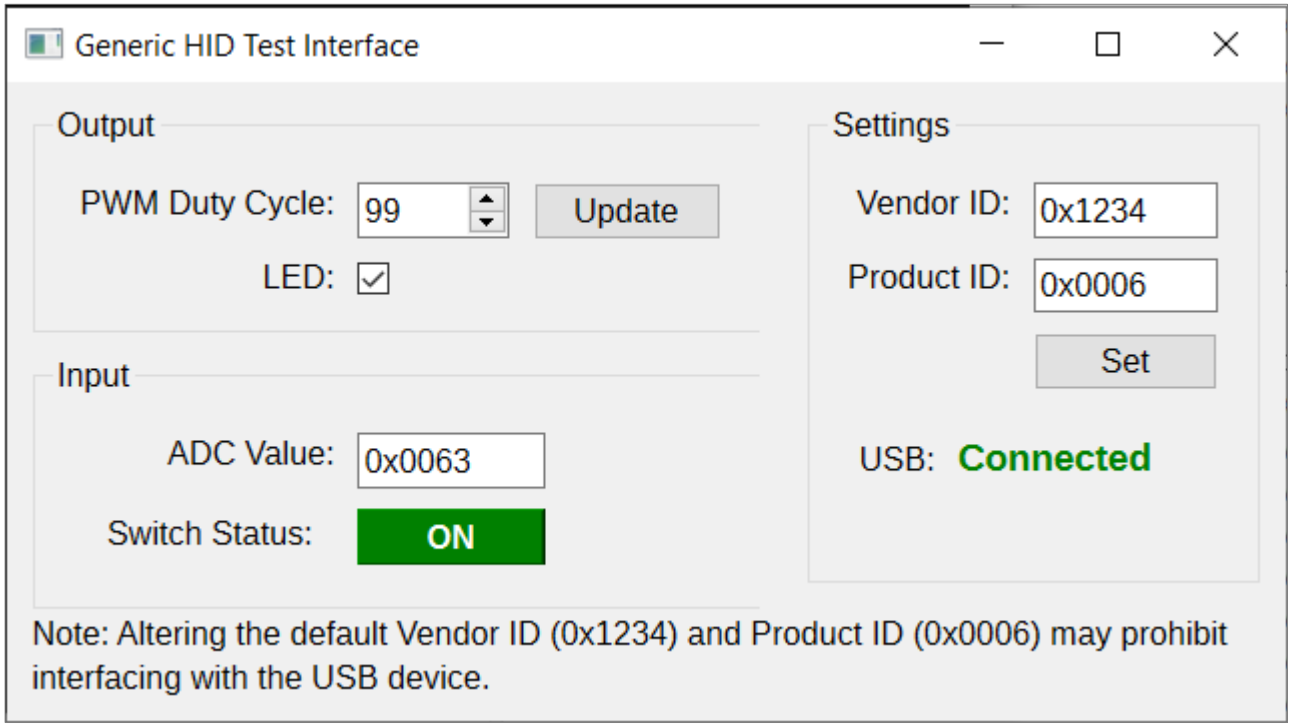
HIDTest_app.py - 2/1.

```
  def on_set(self):
    vId = int(self.lineEdit_2.text(), 16)
    pId = int(self.lineEdit_3.text(), 16)
    if ((vId == 0x1234) and (pId == 0x0006)):
      self.label_8.setText("Connected")
      self.label_8.setStyleSheet("color: green")
    else:
      self.label_8.setText("Disconnected")
      self.label_8.setStyleSheet("color: red")

  def on_check(self,state):
    if state:
      self.pushButton_2.setStyleSheet("color:white;background-color: green")
      self.pushButton_2.setText("ON")
    else:
      self.pushButton_2.setStyleSheet("color: white; background-color: red")
      self.pushButton_2.setText("Off")

app = QtWidgets.QApplication([])
window = MainWindow()
window.show()
app.exec()
```

Válámi ván, dé ném áz igázi...

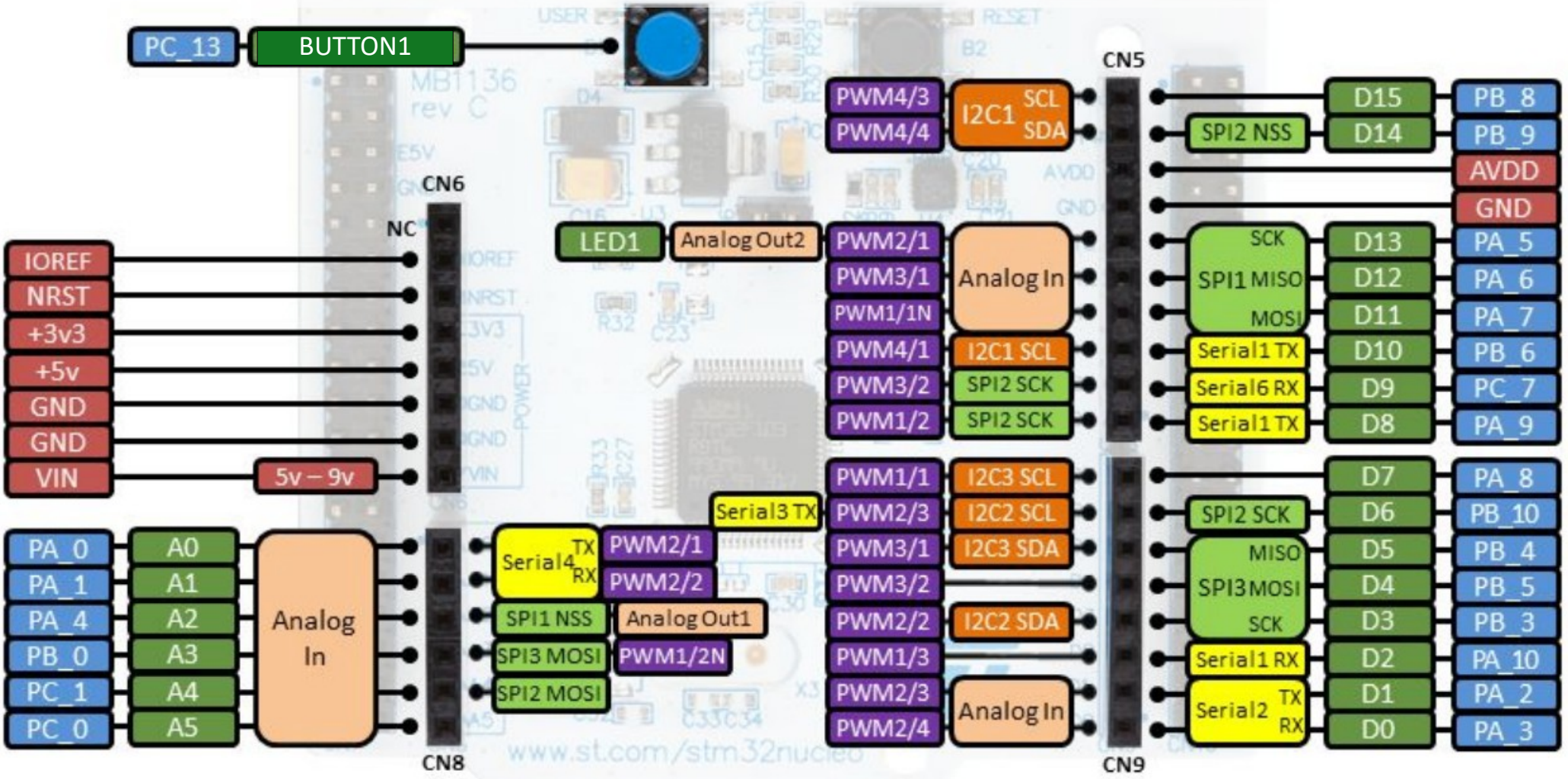
- A bemutatott programmal némi funkcionalitást már mutat a program, de nem ez az igazi, elvárt tevékenység...

(folytatás következik!)



Arduino kompatibilis kivezetések

- Kivezetés azonosításra a kék, illetve a sötétzöld címkék használhatók (pl. PA_5, D13, vagy LED1)



Morpho kivezetések

Nucleo F446RE

Morpho Headers