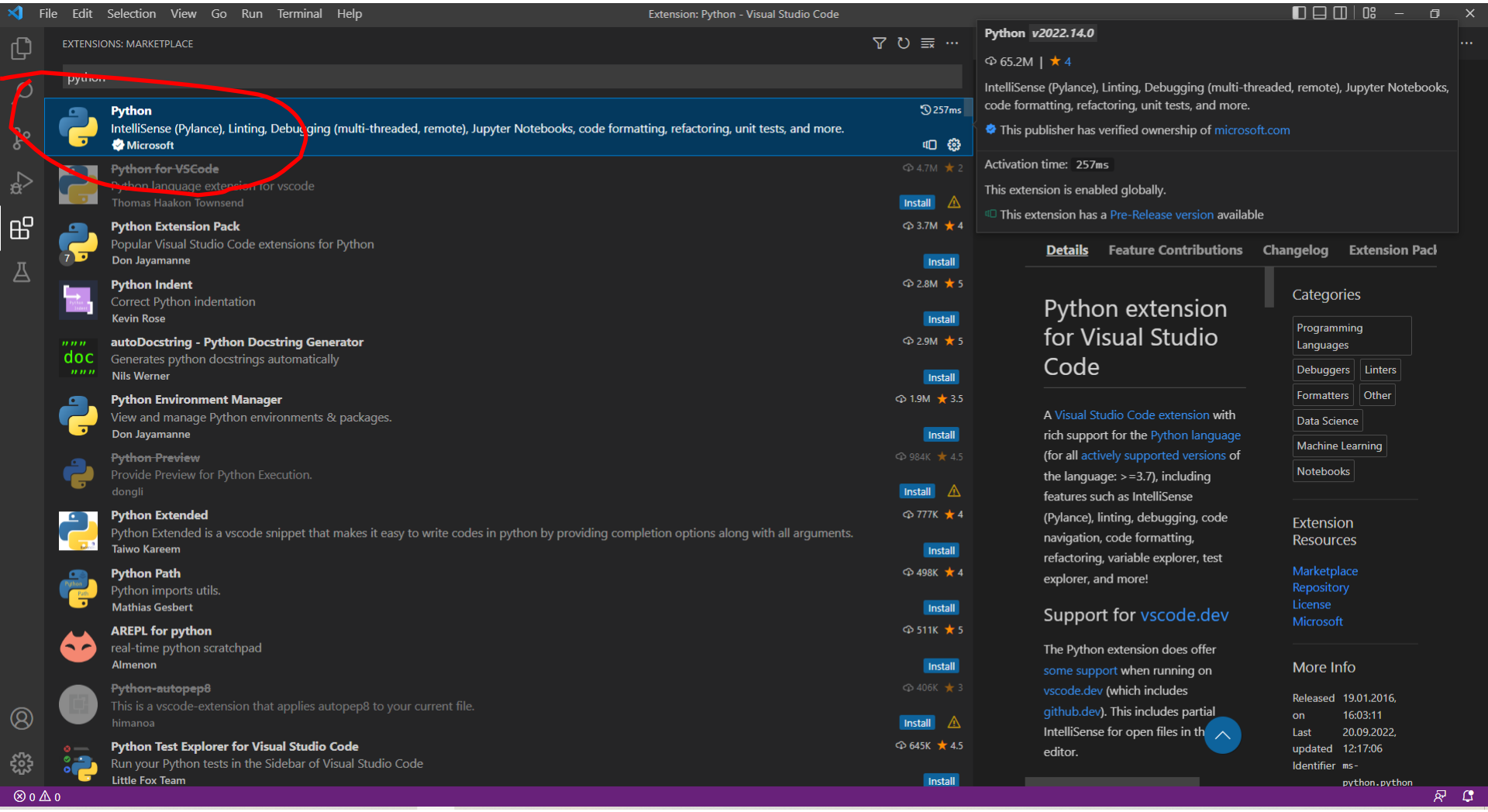
Python Tkinter

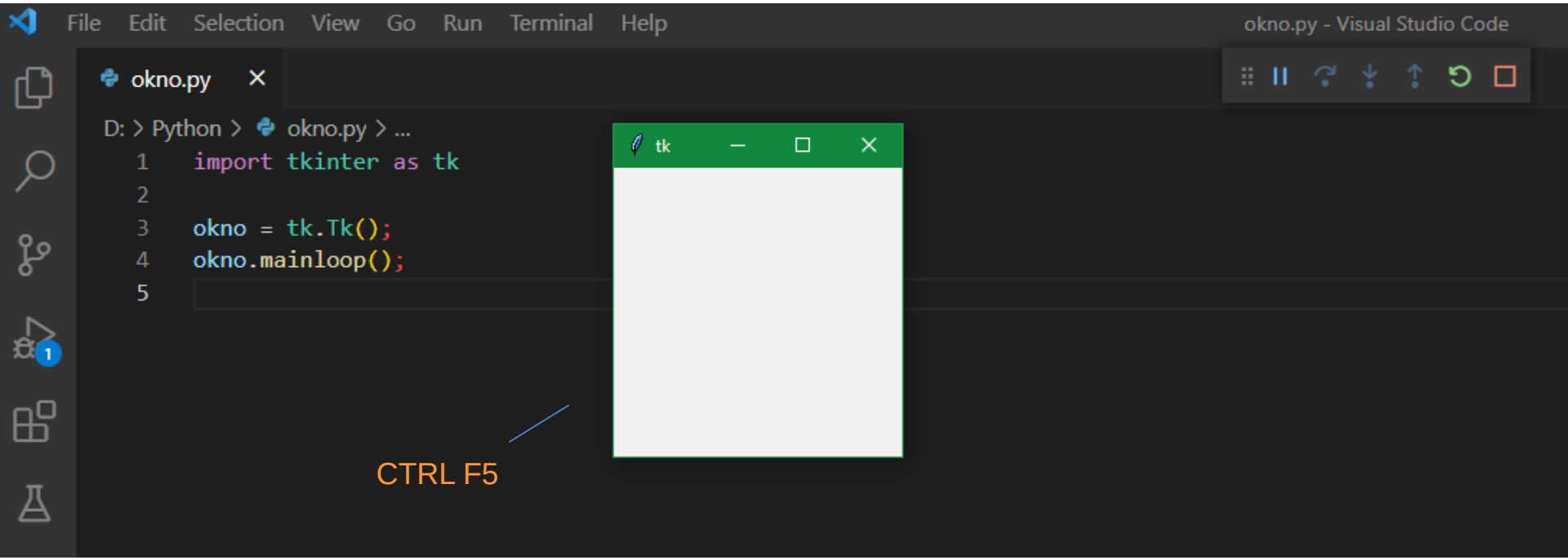
Download


Python
PSF
Docs
PyPI
Jobs
Community
python™
Donate
Search
GO
Socialize
About
Downloads
Documentation
Community
Success Stories
News
Events
Download the latest version for Windows
Download Python 3.10.7
Looking for Python with a different OS? Python for Windows,
Linux/UNIX, macOS, Other
Want to help test development versions of Python? Prereleases,
Docker images
Looking for Python 2.7? See below for specific releases

Visual Studio Code
Python Extension



VSC uruchomienie programu okno.py



PyCharm

Version: 2022.2.2
Build: 222.4167.33
15 September 2022

System requirements

Installation instructions

Other versions

Third-party software

Download PyCharm

Windows macOS Linux

Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

Download

Free 30-day trial available

Community

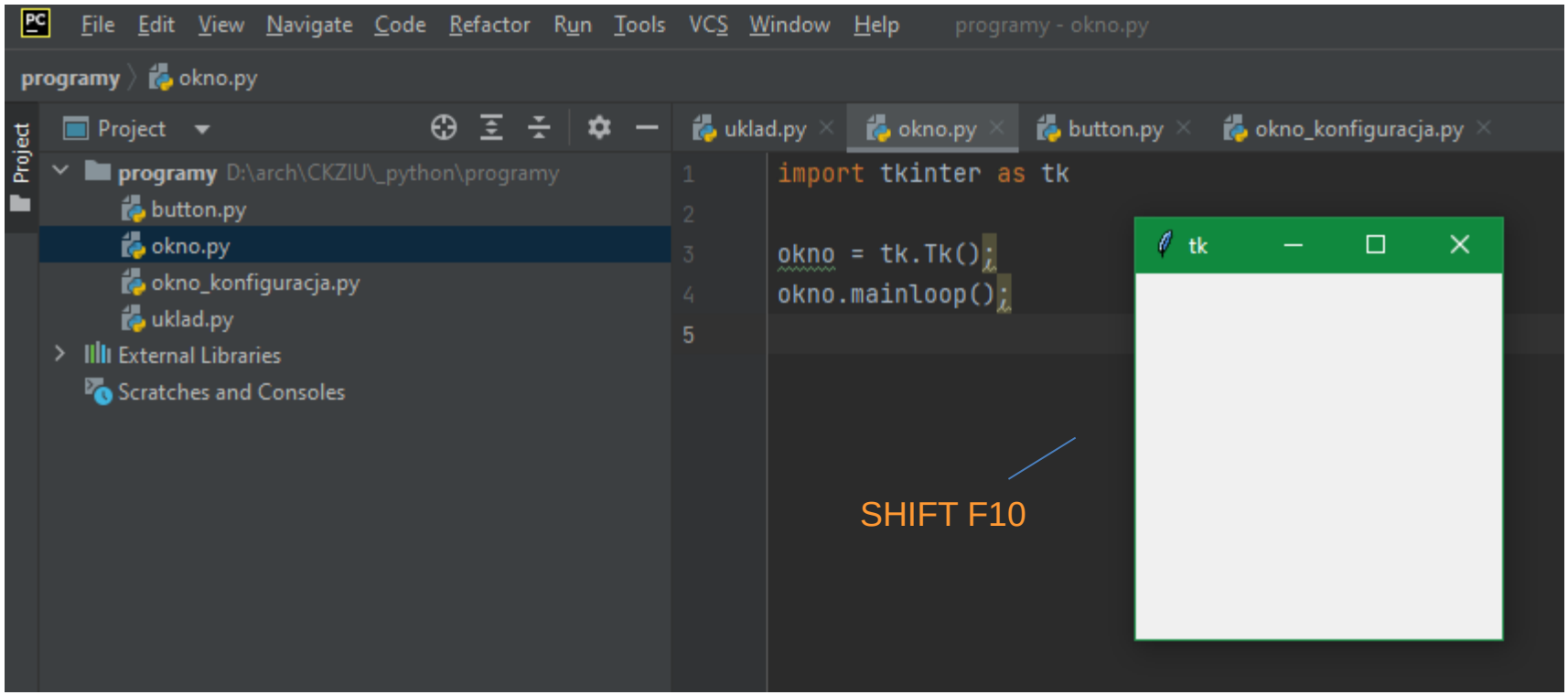
For pure Python development

Download

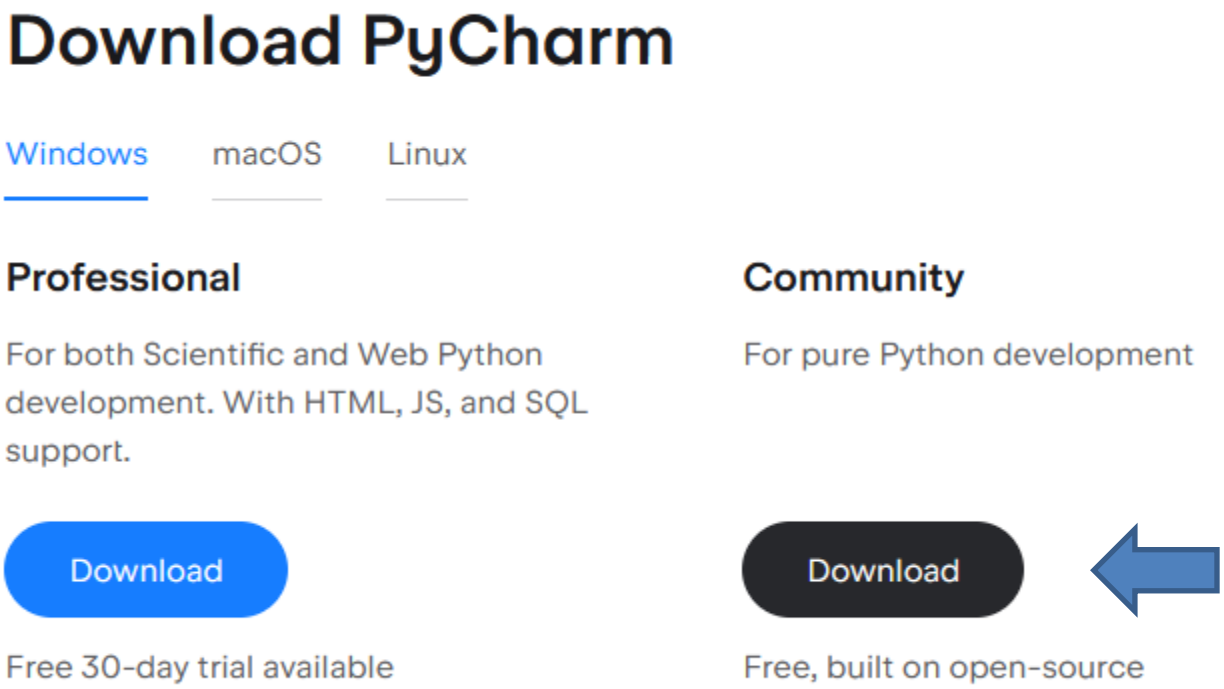
Free, built on open-source

Get the Toolbox App to download PyCharm and its future updates with ease

PyCharm
uruchomienie programu okno.py



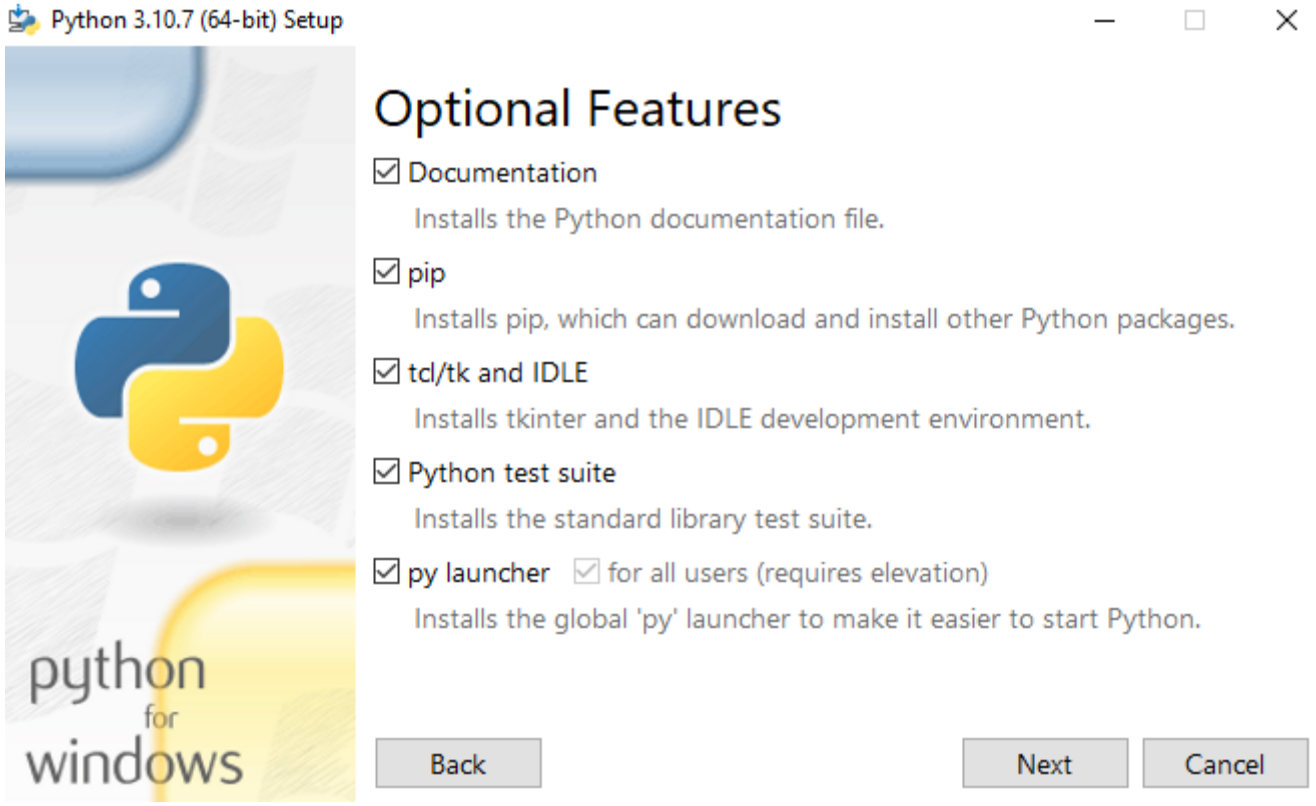
PyCharm
Zmiana interpretera



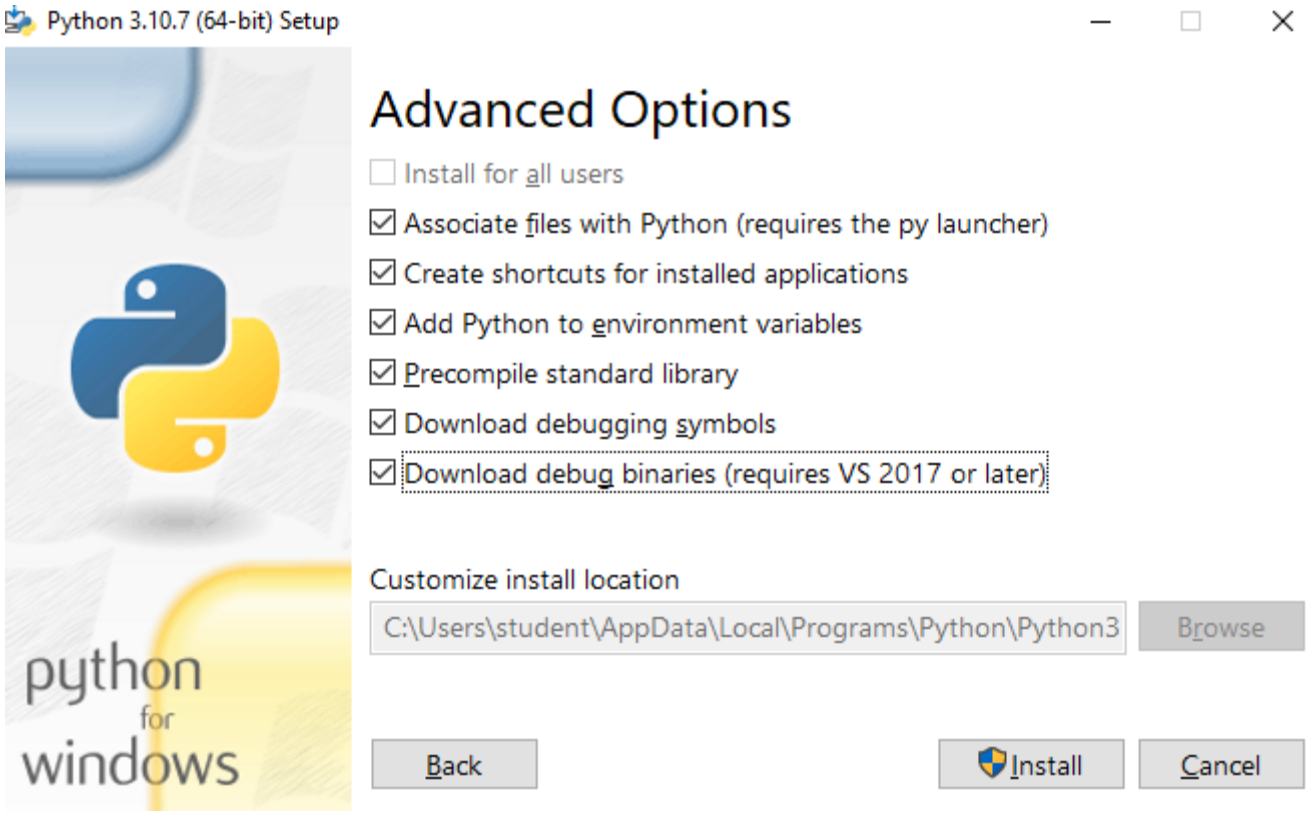
PyCharm
Zmiana interpretera

instalacja najnowszej wersję pythona (tu już jest zainstalowana)

PyCharm
Zmiana interpretera

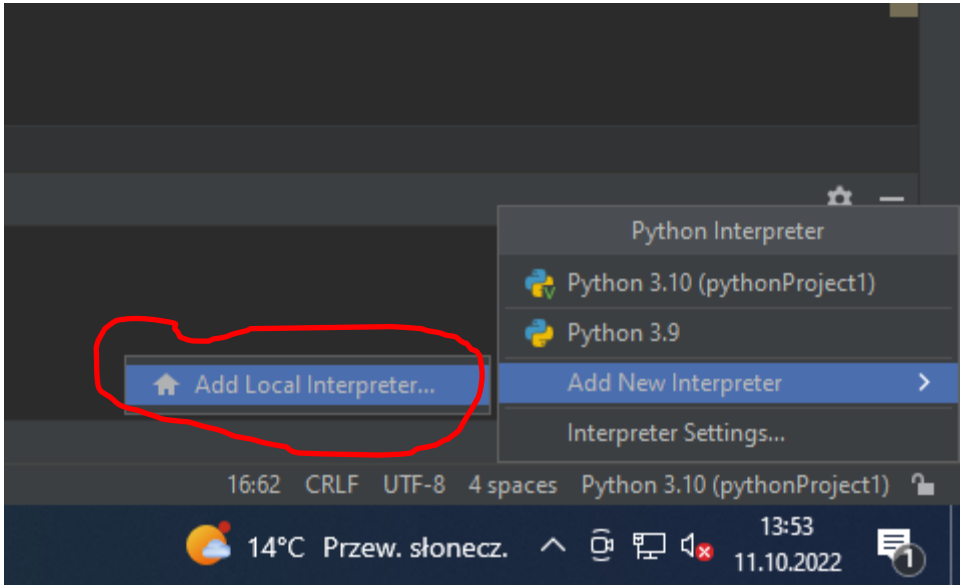


PyCharm
Zmiana interpretera

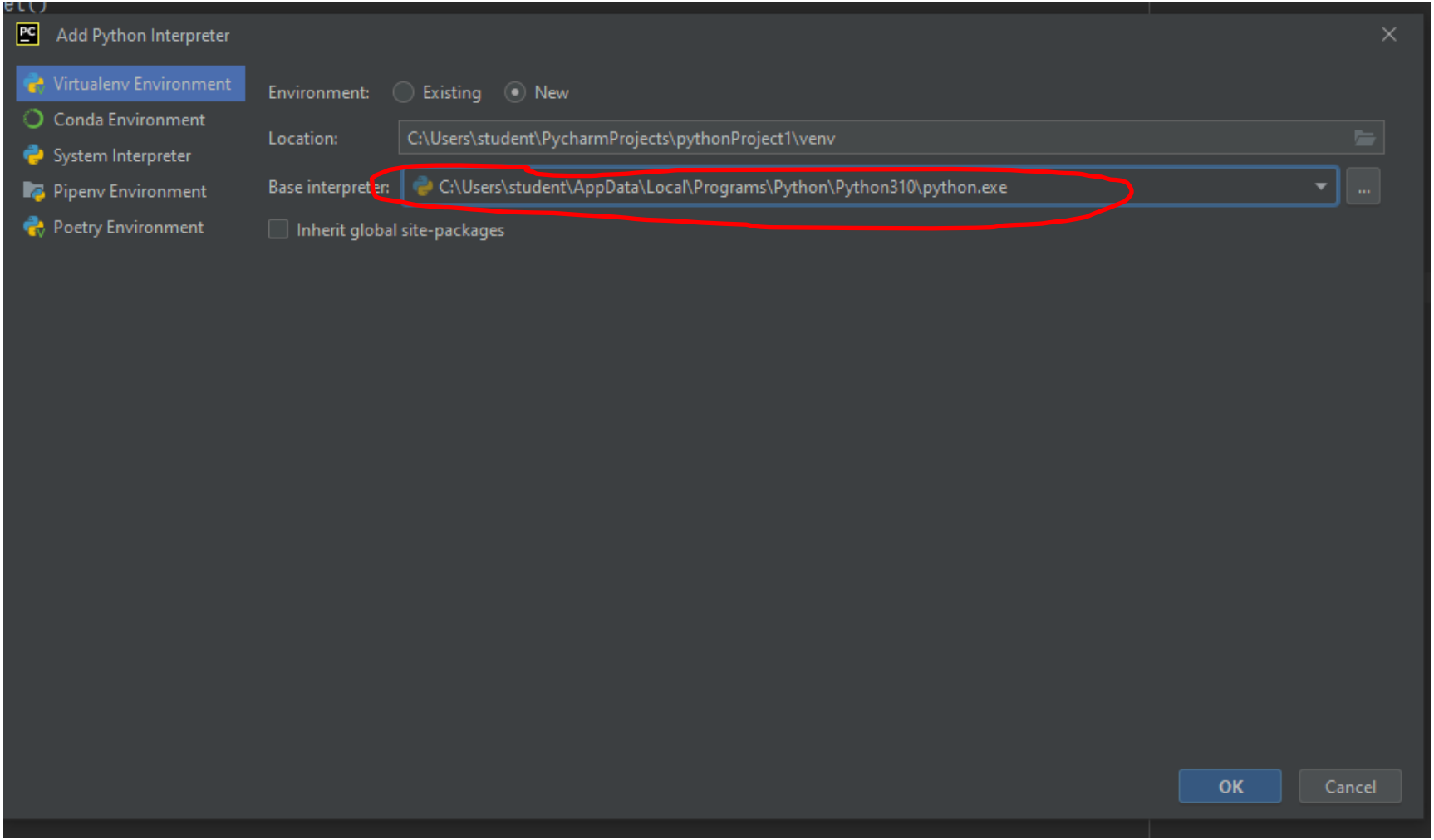


PyCharm
Zmiana interpretera

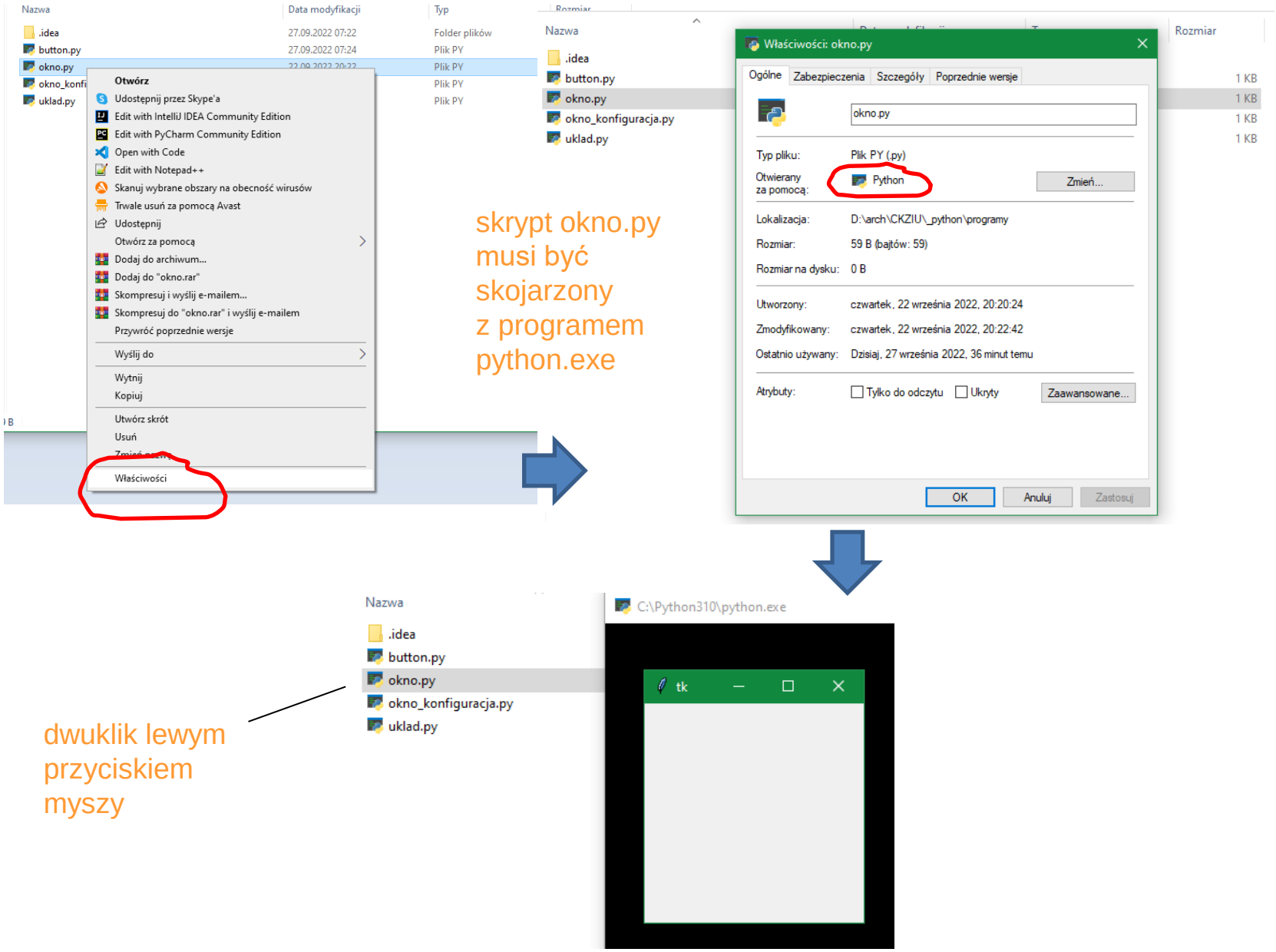
instalacja najnowszej wersję programu PyCharm



PyCharm
Zmiana interpretera

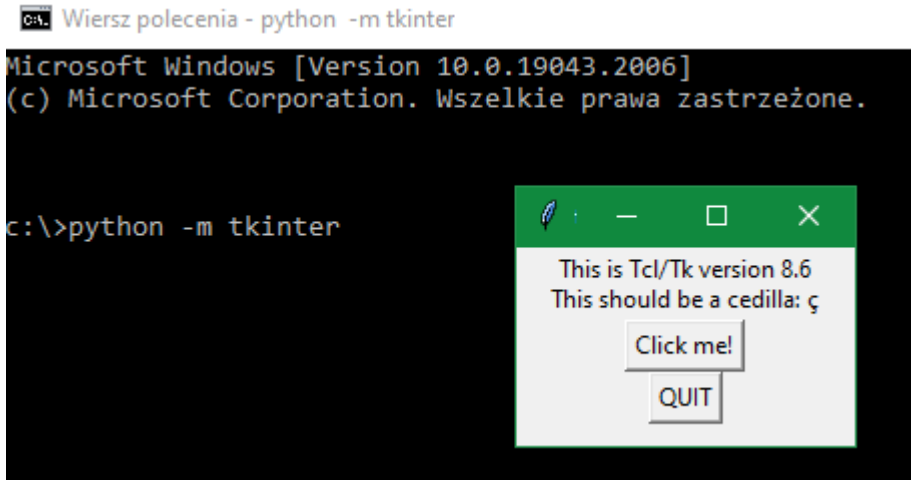


otwieranie plików .py dwuklikiem



Tkinter version

sprawdzenie, czy package Tkinter jest prawidłowo zainstalowany w systemie.



cmd

Tkinter tutorials

javatpoint

Home | Python | If Else | For Loop | Function | Array | String | Regex | List | Set | Tuple | Dictionary | Programs | Numpy | Interview Questions

- Python Tutorial
- Python Tutorial
- Python Features
- Python History
- Python Applications
- Python Install
- Python Example
- Python Variables
- Python Data Types
- Python Keywords
- Python Literals
- Python Operators
- Python Comments
- Python If else
- Python Loops
- Python For Loop
- Python While Loop
- Python Break
- Python Continue
- Python Pass
- Python Strings
- Python Lists
- Python Tuples
- Python List Vs Tuple
- Python Sets
- Python Dictionary
- Python Functions

Python Tkinter Tutorial

← Prev Next →

Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals.

Python provides the standard library Tkinter for creating the graphical user interface for desktop based applications.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

Example

```
# !/usr/bin/python3
from tkinter import *
#creating the application main window.
top = Tk()
#Entering the event main loop
```

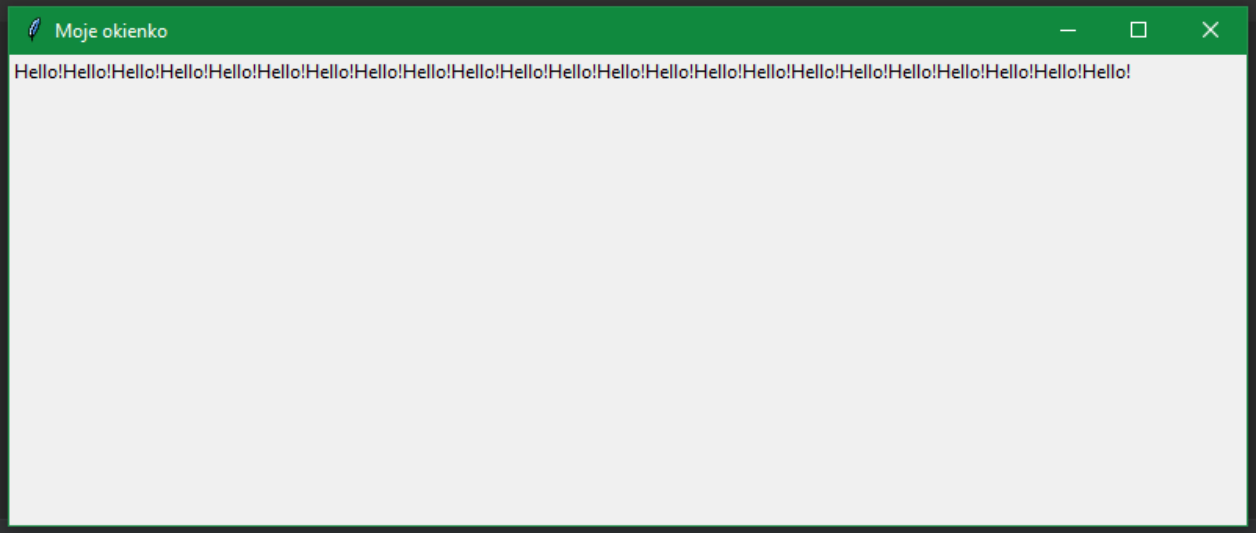
https://www.javatpoint.com/python-tkinter

https://www.tutorialspoint.com/python/python_gui_programming.htm

w3schools Python tutorial

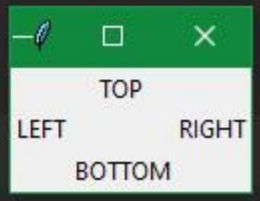
Okno

```python
import tkinter as tk

okno = tk.Tk()
# nazwa okna
okno.title("Moje okienko")
# wymiary okna
okno.geometry("400x300")
# można skalować okno wzdłuż osi x, blokada skalowania okna wzdłuż osi y
okno.resizable(True, False)
# tworzymy obiekt typu StringVar
napis = tk.StringVar()
# przypisujemy obiektowi napis tekst
napis.set("Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!Hello!")
# tworzymy Label
Label1 = tk.Label(okno, textvariable=napis)
# umiejscawiamy Label1 w oknie
Label1.grid(row=0, column=0)
## uruchomienie okna
okno.mainloop()
```



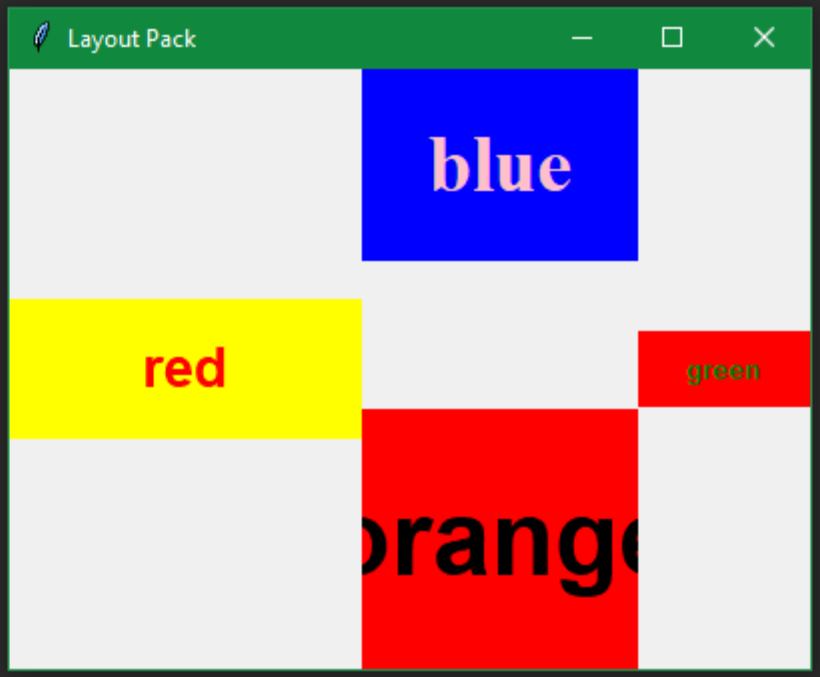
Layout pack

```
from tkinter import *
parent = Tk()

leftLabel = Label(parent, text="LEFT")
leftLabel.pack(side=LEFT)

rightLabel = Label(parent, text="RIGHT")
rightLabel.pack(side=RIGHT)

topLabel = Label(parent, text="TOP")
topLabel.pack(side=TOP)

bottomLabel = Label(parent, text="BOTTOM")
bottomLabel.pack(side=BOTTOM)

parent.mainloop()
```



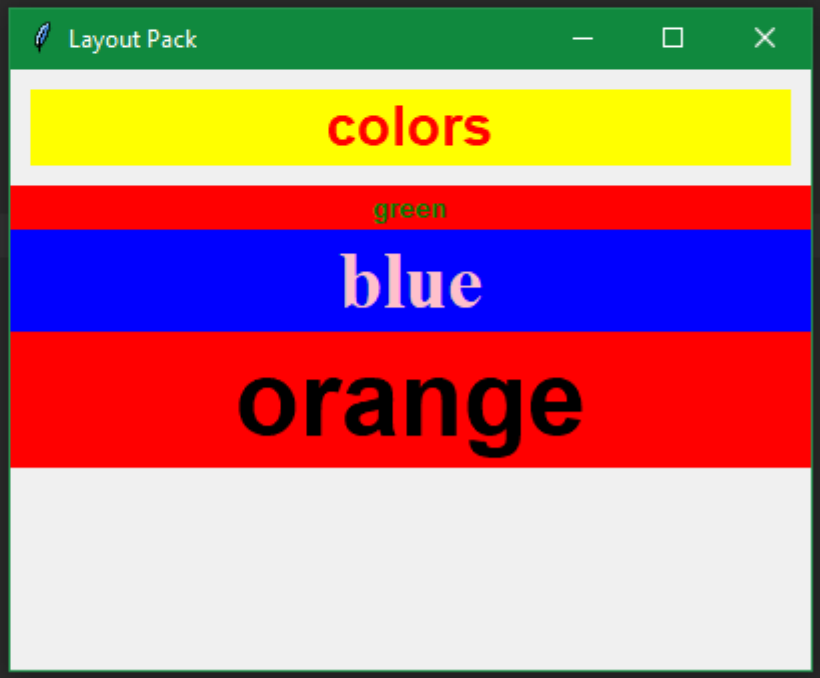
Layout pack

```python
import tkinter as tk

okno = tk.Tk()
okno.geometry("400x300")
okno.title("Layout Pack")


Label1 = tk.Label(okno, text="red", fg="red", bg="yellow", width=10, height=2, font= ('Helvetica 20 bold'))
Label2 = tk.Label(okno, text="green", fg="green", bg="red",width=10, height=2, font= ('Arial 10 bold'))
Label3 = tk.Label(okno, text="blue", fg="pink", bg="blue", width=10, height=2, font= ('"Times New Roman" 30 bold'))
Label4 = tk.Label(okno, text="orange", fg="black", bg="red", width=10, height=2, font= ('Helvetica 40 bold'))
Label1.pack(side="left")
Label2.pack(side="right")
Label3.pack(side="top")
Label4.pack(side="bottom")

okno.mainloop()
```



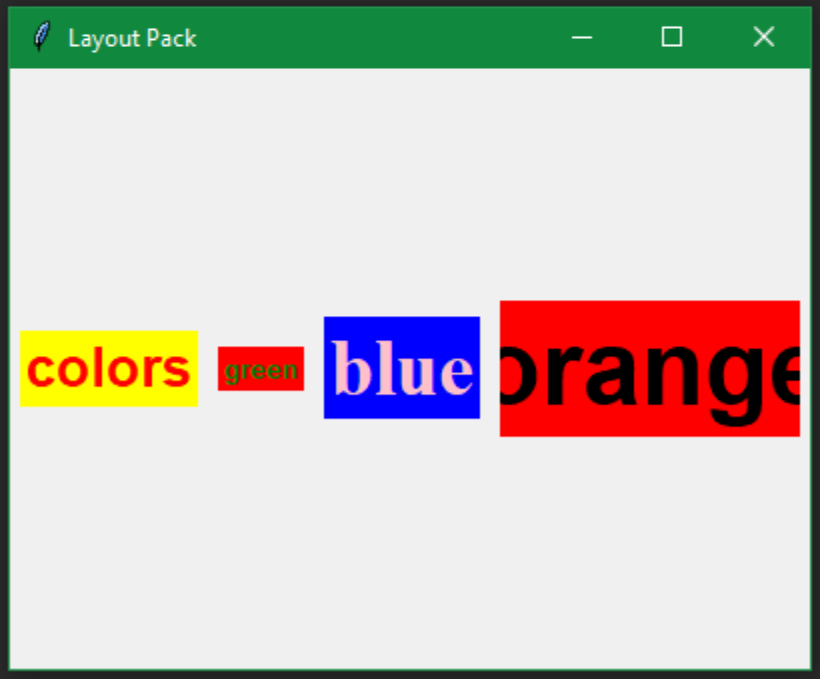
Layout pack

```
import tkinter as tk

okno = tk.Tk()
okno.geometry("400x300")
okno.title("Layout Pack")


Label1 = tk.Label(okno, text="colors", fg="red", bg="yellow", font= ('Helvetica 20 bold'))
Label2 = tk.Label(okno, text="green", fg="green", bg="red", font= ('Arial 10 bold'))
Label3 = tk.Label(okno, text="blue", fg="pink", bg="blue",  font= ('"Times New Roman" 30 bold'))
Label4 = tk.Label(okno, text="orange", fg="black", bg="red", font= ('Helvetica 40 bold'))
Label1.pack(fill=tk.X, padx=10, pady=10)
Label2.pack(fill=tk.X)
Label3.pack(fill=tk.X)
Label4.pack(fill=tk.X)

okno.mainloop()
```



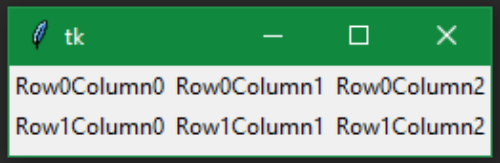
Layout pack

```python
import tkinter as tk

okno = tk.Tk()
okno.geometry("400x300")
okno.title("Layout Pack")


Label1 = tk.Label(okno, text="colors", fg="red", bg="yellow", font= ('Helvetica 20 bold'))
Label2 = tk.Label(okno, text="green", fg="green", bg="red", font= ('Arial 10 bold'))
Label3 = tk.Label(okno, text="blue", fg="pink", bg="blue",  font= ('"Times New Roman" 30 bold'))
Label4 = tk.Label(okno, text="orange", fg="black", bg="red", font= ('Helvetica 40 bold'))
Label1.pack(padx=5, pady=10, side=tk.LEFT)
Label2.pack(padx=5, pady=20, side=tk.LEFT)
Label3.pack(padx=5, pady=30, side=tk.LEFT)
Label4.pack(padx=5, pady=40, side=tk.LEFT)

okno.mainloop()
```



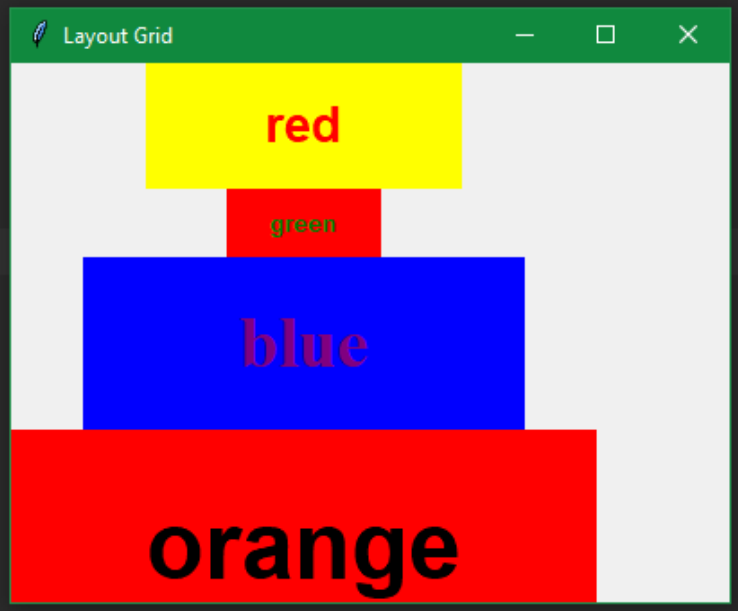
Layout grid

```
from tkinter import *
parent = Tk()

# row 1
Row0Column0 = Label(parent, text="Row0Column0")
Row0Column0.grid(row=0, column=0)
Row0Column1 = Label(parent, text="Row0Column1")
Row0Column1.grid(row=0, column=1)
Row0Column2 = Label(parent, text="Row0Column2")
Row0Column2.grid(row=0, column=2)
# row 2
Row1Column0 = Label(parent, text="Row1Column0")
Row1Column0.grid(row=1, column=0)
Row1Column1 = Label(parent, text="Row1Column1")
Row1Column1.grid(row=1, column=1)
Row1Column2 = Label(parent, text="Row1Column2")
Row1Column2.grid(row=1, column=2)

parent.mainloop()
```



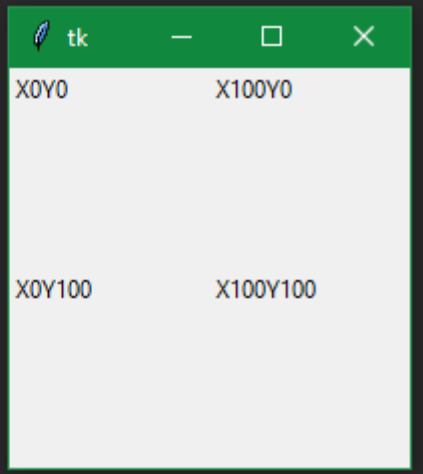
Layout grid

```python
import tkinter as tk

okno = tk.Tk()
okno.geometry("400x300")
okno.title("Layout Grid")


Label1 = tk.Label(okno, text="red", fg="red", bg="yellow", width=10, height=2, font= ('Helvetica 20 bold'))
Label2 = tk.Label(okno, text="green", fg="green", bg="red",width=10, height=2, font= ('Arial 10 bold'))
Label3 = tk.Label(okno, text="blue", fg="purple", bg="blue", width=10, height=2, font= ('"Times New Roman" 30 bold'))
Label4 = tk.Label(okno, text="orange", fg="black", bg="red", width=10, height=2, font= ('Helvetica 40 bold'))
Label1.grid(row=0, column=0)
Label2.grid(row=1, column=0)
Label3.grid(row=2, column=0)
Label4.grid(row=3, column=0)

okno.mainloop()
```



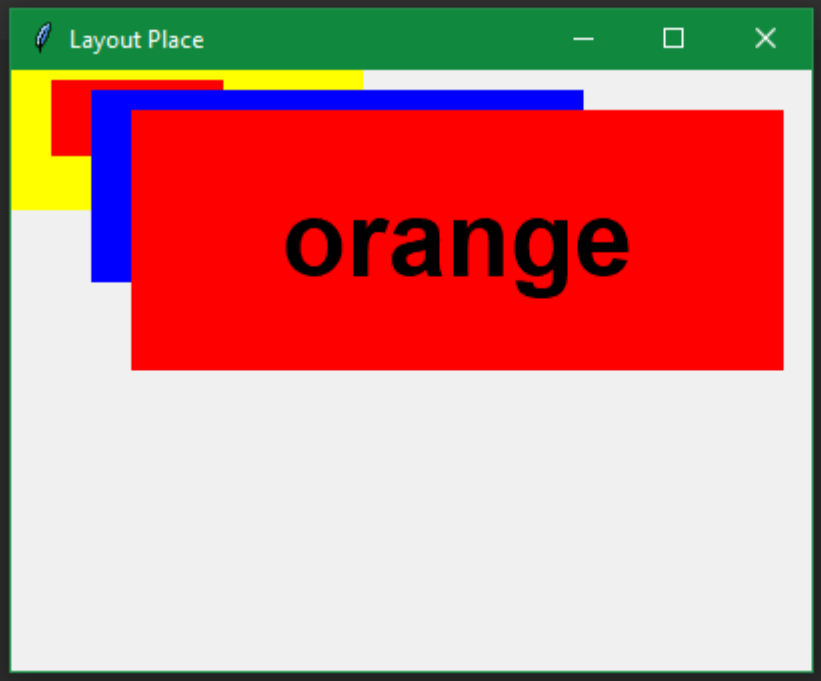
Layout place

```
from tkinter import *
top = Tk()
top.geometry("200x200")
X0Y0 = Label(top, text="X0Y0")
X0Y0.place(x=0, y=0)
X100Y0 = Label(top, text="X100Y0")
X100Y0.place(x=100, y=0)
X100Y100 = Label(top, text="X100Y100")
X100Y100.place(x=100, y=100)
X0Y100 = Label(top, text="X0Y100")
X0Y100.place(x=0, y=100)
top.mainloop()
```



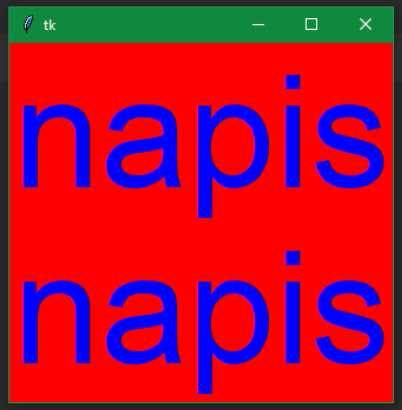
Layout place

```python
import tkinter as tk

okno = tk.Tk()
okno.geometry("400x300")
okno.title("Layout Place")


Label1 = tk.Label(okno, text="red", fg="red", bg="yellow", width=10, height=2, font= ('Helvetica 20 bold'))
Label2 = tk.Label(okno, text="green", fg="green", bg="red",width=10, height=2, font= ('Arial 10 bold'))
Label3 = tk.Label(okno, text="blue", fg="pink", bg="blue", width=10, height=2, font= ('"Times New Roman" 30 bold'))
Label4 = tk.Label(okno, text="orange", fg="black", bg="red", width=10, height=2, font= ('Helvetica 40 bold'))
Label1.place(x=0, y=0)
Label2.place(x=20, y=5)
Label3.place(x=40, y=10)
Label4.place(x=60, y=20)

okno.mainloop()
```



Label

```python
from tkinter import *
parent = Tk()

napis = Label(parent, text="napis\nnapis", background="#ff0000", foreground="#0000ff", font=("Arial", 100))
napis.grid(row=0, column=0)

parent.mainloop()
```



https://www.obliczeniowo.com.pl/496

https://www.javatpoint.com/python-tkinter-label

https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/label.html

Zmienne

```
a = StringVar()  # zmienna tekstowa - string; default ""
a = IntVar()  # liczba całkowita ; default 0
a = DoubleVar()  # liczba zmiennoprzecinkowa; default value 0.0
a = BooleanVar()  # zmienna logiczna, zwraca 0 dla False oraz 1 dla True
```

te typy zmiennych możemy łączyć z wigetami (Enter, Label itp.)

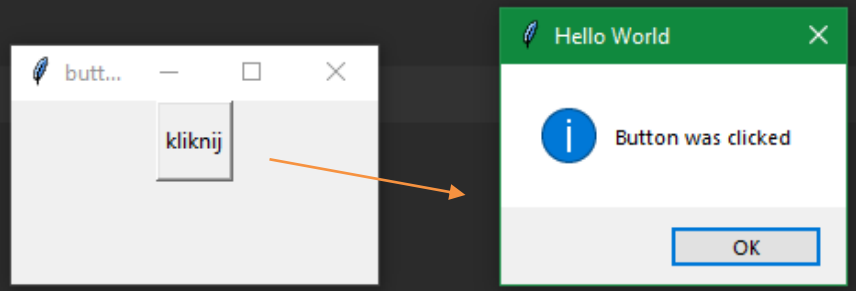
https://python-course.eu/tkinter/variable-classes-in-tkinter.php

Button

```python
import tkinter as tk
from tkinter import messagebox

# tworzymy okno
okno = tk.Tk()
okno.title("button")
okno.geometry("200x100")

def fun():
    messagebox.showinfo("Hello World", "Button was clicked")

btn = tk.Button(okno, text="kliknij", command=fun, pady=10)
btn.pack()

okno.mainloop()
```

definicja funkcji fun,
która nie ma parametrów

przypisanie zdarzenia
kliknięcia do funkcji fun



Button

import modułu partial

```
import tkinter as tk
from tkinter import messagebox
from functools import partial

# tworzymy okno
okno = tk.Tk()
okno.title("button")
okno.geometry("200x100")

def fun(liczba):
    messagebox.showinfo("Hello World", "liczba: " + str(liczba.get()))

liczba = tk.IntVar(okno,10)

fun = partial(fun, liczba)
btn = tk.Button(okno, text="kliknij", command=fun, pady=10)
btn.pack()

okno.mainloop()
```

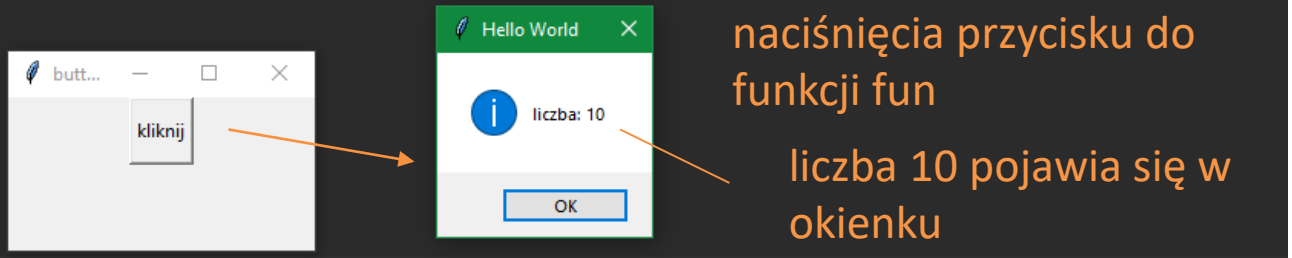
definicja funkcji fun, która ma jeden parametr liczba

konwersja liczby całkowitej na string

definicja zmiennej liczba typu całkowitego o wartości 10

przekazanie zmiennej liczba do funkcji fun za pomocą funkcji partial

przypisanie zdarzenia naciśnięcia przycisku do funkcji fun



liczba 10 pojawia się w okienku

Button

```python
import tkinter as tk
from tkinter import messagebox

# tworzymy okno
okno = tk.Tk()
okno.title("button")
okno.geometry("200x100")

def fun(liczba):
    messagebox.showinfo("Hello World", "liczba: " + str(liczba.get()))

liczba = tk.IntVar(okno,10)

btn = tk.Button(okno, text="kliknij", command=lambda: fun(liczba), pady=10)
btn.pack()

okno.mainloop()
```

dzięki użyciu **funkcji anonimowej** nie trzeba stosować modułu partial przy przekazywaniu do niej argumentów

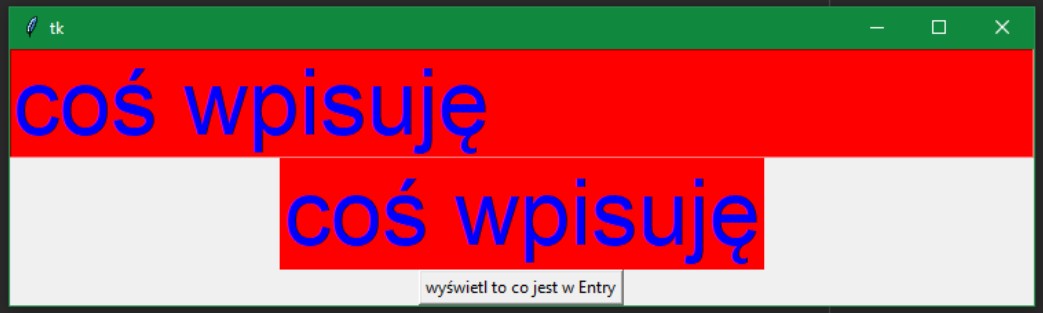
lambda to funkcja anonimowa (bez nazwy), która może przyjmować dowolną liczbę parametrów, ale w ciele ma tylko jedno wyrażenie:
lambda <parametry> : <wyrażenie>

Entry

```python
from tkinter import *
import tkinter as tk
from functools import partial


# funkcja wyświetlająca to, co jest wpisane w Entry (w zmiennej a)
def wyswietl(a):
    napis.config(text=a.get())


parent = Tk()
# zmienna a do przechowywania tego co jest wpisane w Entry
a = tk.StringVar()

# kontrolka Entry z przypisaniem zmiennej a
entry = Entry(parent, background="#ff0000", foreground="#0000ff", font=("Arial", 50), textvariable=a)
entry.grid(row=0, column=0)

# kontrolka Label do wyświetlania tego co jest wpisane w Entry (w zmiennej a)
napis = Label(parent, background="#ff0000", foreground="#0000ff", font=("Arial", 50))
napis.grid(row=1, column=0)

# przekazanie zmiennej a jako parametr funkcji wyswietl
wyswietl = partial(wyswietl, a)
# button do uruchomioenia wyświetlenia
button = tk.Button(parent, text="wyświetl to co jest w Entry", command=wyswietl)
button.grid(row=3, column=0)

parent.mainloop()
```

funkcja wyswietl może korzystać ze zmiennej a

zmienna a typu string



przypisanie wartości wpisywanej w Entry do zmiennej a

przekazanie zmiennej a do funkcji wyswietl za pomocą funkcji partial

https://www.obliczeniowo.com.pl/496

https://www.javatpoint.com/python-tkinter-entry

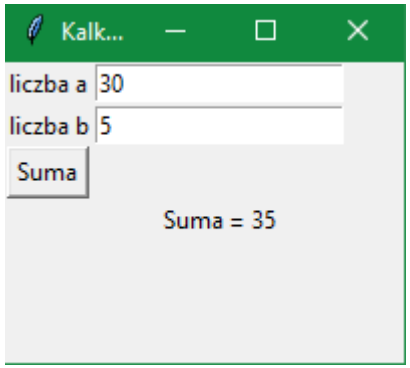
https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/entry.html

kalkulator 1.0

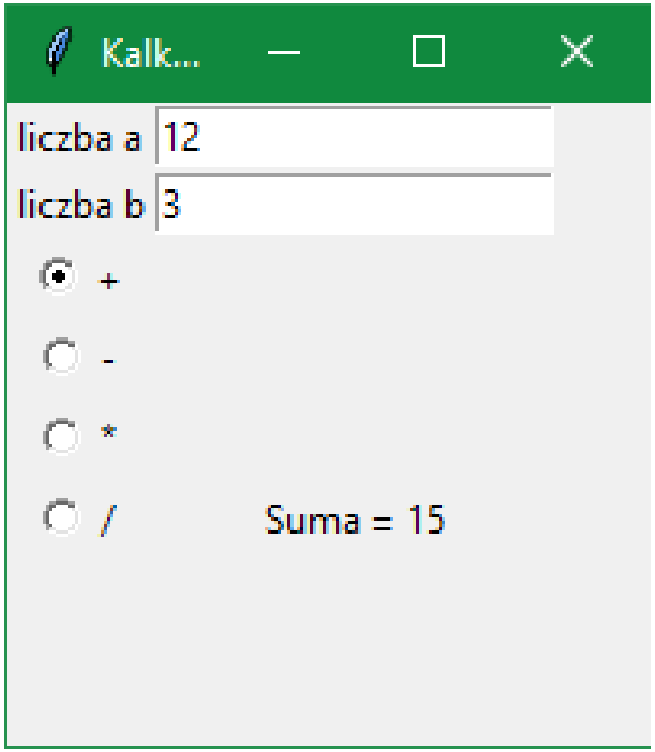

```python
import tkinter as tk
from functools import partial


# definicja funkcji sumującej
def oblicz_sume(liczba1, liczba2):
    # sumowanie argumentów z konwersją z StringVar do integera
    suma = int(liczba1.get()) + int(liczba2.get())
    # wyswietlenie rezultatu w labelu
    labelRezultat.config(text="Suma = %d" % suma)


# tworzymy okno
okno = tk.Tk()
okno.geometry('200x150')
okno.title('Kalkulator')

# zainicjowane zmienne do przechowywania wartości
a = tk.StringVar(okno, '12')
b = tk.StringVar(okno, '3')

# labele z napisami
labelLiczbaA = tk.Label(okno, text="liczba a")
labelLiczbaA.grid(row=1, column=0)
labelLiczbaB = tk.Label(okno, text="liczba b")
labelLiczbaB.grid(row=2, column=0)
labelRezultat = tk.Label(okno)
labelRezultat.grid(row=7, column=2)
```

kalkulator 1.0


```
# pola wejściowe z przypisaniem wartości pola do zmiennych a oraz b (textvariable=a)
entryLiczbaA = tk.Entry(okno, textvariable=a)
entryLiczbaA.grid(row=1, column=2)
entryLiczbaB = tk.Entry(okno, textvariable=b)
entryLiczbaB.grid(row=2, column=2)

# przekazanie argumentów do funkcji oblicz_sume (liczb z pól entry)
oblicz_sume = partial(oblicz_sume, a, b)

# przycisk uruchamiający funkcję oblicz_sumę, do której zostały przekazane liczby z pól entry
button = tk.Button(okno, text="Suma", command=oblicz_sume)
button.grid(row=3, column=0)

# uruchomienie okna
okno.mainloop()
```



kalkulator 2.0



wybór operatora za pomocą kontrolki Radiobutton

kalkulator 2.0

instrukcja wyboru match jest dostępna od wersji 3.10

instrukcja wyboru match (switch) dzięki której wykonujemy określone działanie na liczbach w zależności od wyboru Radiobuttona

funkcja operator obsługująca kontrolki Radiobuttons

```
import tkinter as tk
from builtins import float
from functools import partial


def operator(liczba1, liczba2):
    choice = radio.get()
    match choice:
        case 1:
            # sumowanie argumentów z konwersją z StringVar do integera
            suma = int(liczba1.get()) + int(liczba2.get())
            # wyswietlenie rezultatu w labelu
            labelRezultat.config(text="Suma = %d" % suma)
        case 2:
            # różnica argumentów z konwersją z StringVar do integera
            roznica = int(liczba1.get()) - int(liczba2.get())
            # wyswietlenie rezultatu w labelu
            labelRezultat.config(text="Różnica = %d" % roznica)
        case 3:
            # iloczyn argumentów z konwersją z StringVar do integera
            iloczyn = int(liczba1.get()) * int(liczba2.get())
            # wyswietlenie rezultatu w labelu
            labelRezultat.config(text="Iloczyn = %d" % iloczyn)
        case 4:
            if liczba2.get() != "0":
                # iloraz argumentów z konwersją z StringVar do integera
                iloraz = float(liczba1.get()) / float(liczba2.get())
                # wyswietlenie rezultatu w labelu
                labelRezultat.config(text="Iloraz = %f" % iloraz)
            else:
                labelRezultat.config(text="nie wolno dzielić przez zero")
```

kalkulator 2.0


```
# tworzymy okno
okno = tk.Tk()
okno.geometry('200x200')
okno.title('Kalkulator')

# zainicjowane zmienne do przechowywania wartości
a = tk.StringVar(okno, '12')
b = tk.StringVar(okno, '3')

# labele z napisami
labelLiczbaA = tk.Label(okno, text="liczba a")
labelLiczbaA.grid(row=1, column=0)
labelLiczbaB = tk.Label(okno, text="liczba b")
labelLiczbaB.grid(row=2, column=0)
labelRezultat = tk.Label(okno)
labelRezultat.grid(row=7, column=2)

# pola wejściowe z przypisaniem wartości pola do zmiennych a oraz b (textvariable=a)
entryLiczbaA = tk.Entry(okno, textvariable=a)
entryLiczbaA.grid(row=1, column=2)
entryLiczbaB = tk.Entry(okno, textvariable=b)
entryLiczbaB.grid(row=2, column=2)
```

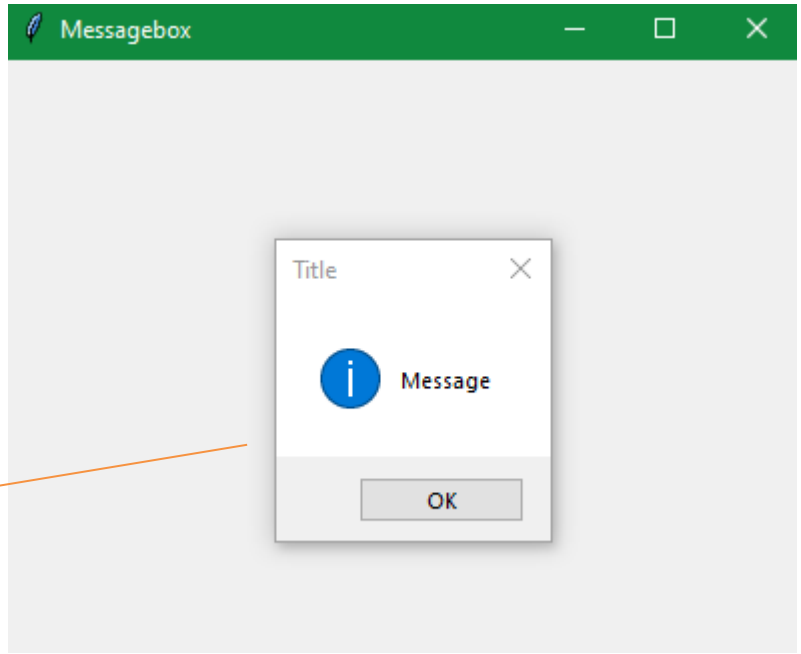
kalkulator 2.0


```
# wybór operatora

# zmienna ustawiana przez radiobuttony
radio = tk.IntVar()

# przekazanie argumentów do funkcji oblicz_sume (liczb z pól entry)
operator = partial(operator, a, b)

radioButton1 = tk.Radiobutton(text="+", variable=radio, value=1, command=operator)
radioButton1.grid(row=4, column=0)
radioButton2 = tk.Radiobutton(text="-", variable=radio, value=2, command=operator)
radioButton2.grid(row=5, column=0)
radioButton3 = tk.Radiobutton(text="*", variable=radio, value=3, command=operator)
radioButton3.grid(row=6, column=0)
radioButton4 = tk.Radiobutton(text="/", variable=radio, value=4, command=operator)
radioButton4.grid(row=7, column=0)

# uruchomienie okna
okno.mainloop()
```

definicje Radiobutton

messagebox

```python
import tkinter as tk
from tkinter import messagebox

okno = tk.Tk()
# nazwa okna
okno.title("Messagebox")
# wymiary okna
okno.geometry("400x300")

# chowanie głównego okna
#okno.withdraw()

# Message Box
messagebox.showinfo("Title", "Message")

# pokazywanie schowanego okna
#okno.deiconify();

## uruchomienie okna
okno.mainloop()
```



https://docs.python.org/3.9/library/tkinter.messagebox.html?highlight=messagebox#module-tkinter.messagebox

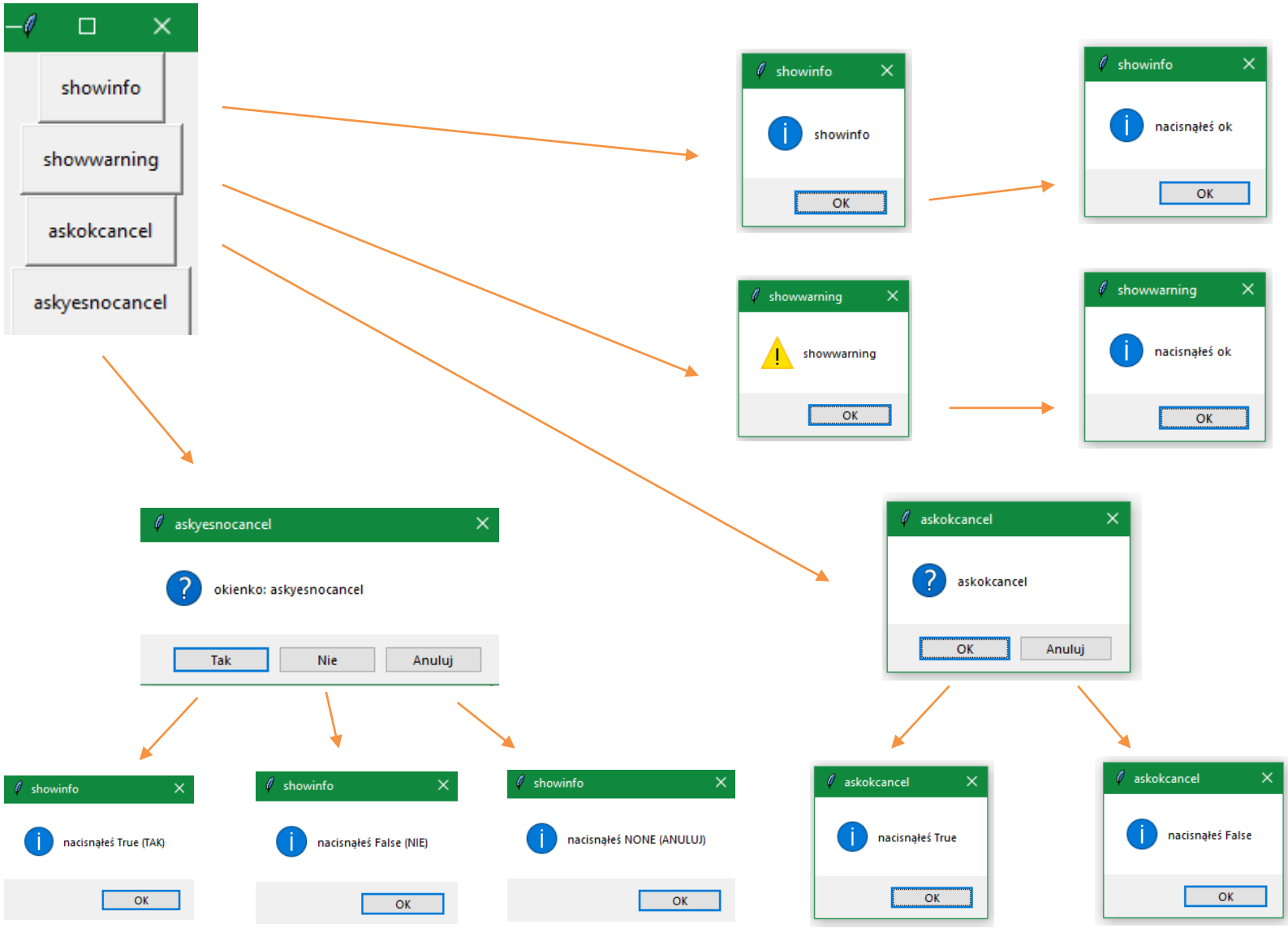
messagebox

różne rodzaje okienek informacyjnych

```python
import tkinter as tk
from tkinter import messagebox

## obsługa buttonów pokazujących poszczególne okienka
def btnShowInfo():
    odp = messagebox.showinfo("showinfo", "showinfo")
    messagebox.showinfo("showinfo", "nacisnąłeś " + odp)

def btnShowWarning():
    odp = messagebox.showwarning("showwarning","showwarning")
    messagebox.showinfo("showwarning", "nacisnąłeś " + odp)

def btnAskOkCancel():
    odp = messagebox.askokcancel("askokcancel", "askokcancel")
    messagebox.showinfo("askokcancel", "nacisnąłeś " + str(odp))

# tworzymy okno
okno = tk.Tk()
# nazwa okna
okno.title("Messagebox")
# wymiary okna
okno.geometry("400x300")

# Message Boxes
buttonShowInfo = tk.Button(okno,text="showinfo",command=btnShowInfo)
buttonShowInfo.grid(row="0", column="0")

buttonShowWarning = tk.Button(okno,text="showwarning",command=btnShowWarning)
buttonShowWarning.grid(row="1", column="0")

buttonAskOkCancel = tk.Button(okno,text="askokcancel",command=btnAskOkCancel)
buttonAskOkCancel.grid(row="2", column="0")

## uruchomienie okna
okno.mainloop()
```

wypisujemy to co zwrócił messagebox

messagebox


showinfo
showwarning
askokcancel
askyesnocancel
showinfo
showinfo
OK
nacisnąłeś ok
OK
showwarning
showwarning
OK
nacisnąłeś ok
OK
askyesnocancel
okienko: askyesnocancel
Tak
Nie
Anuluj
askokcancel
askokcancel
OK
Anuluj
showinfo
nacisnąłeś True (TAK)
OK
showinfo
nacisnąłeś False (NIE)
OK
showinfo
nacisnąłeś NONE (ANULUJ)
OK
askokcancel
nacisnąłeś True
OK
askokcancel
nacisnąłeś False
OK

messagebox

```python
from functools import partial
import tkinter as tk
from tkinter import messagebox


okno = tk.Tk()

# funkcje obsługujące buttony
def showinfo(nazwa):
    messagebox.showinfo("showinfo", nazwa.get())

def showwarning(nazwa):
    messagebox.showwarning("showwarning", nazwa.get())

def askokcancel(nazwa):
    # okienko zwraca True albo False
    odp = messagebox.askokcancel("askokcancel", nazwa.get())
    # jeśli odp == True
    if odp:
        messagebox.showinfo("showinfo", "nacisnąłeś OK")
    else:
        messagebox.showinfo("showinfo", "nacisnąłeś ANULUJ")

def askyesnocancel(nazwa):
    # okienko zwraca True, False albo None
    odp = messagebox.askyesnocancel("askyesnocancel", nazwa.get())
    if odp == None:
        messagebox.showinfo("showinfo", "nacisnąłeś NONE (ANULUJ)")
    elif odp == True:
       messagebox.showinfo("showinfo", "nacisnąłeś True (TAK)")
    else:
        messagebox.showinfo("showinfo", "nacisnąłeś False (NIE)")
```

```python
# definicje nazw okien
nazwashowinfo = tk.StringVar(okno, "okienko: showinfo")
nazwashowwarning = tk.StringVar(okno, "okienko: showwarning")
nazwaaskokcancel = tk.StringVar(okno, "okienko: askokcancel")
nazwaaskyesnocancel = tk.StringVar(okno, "okienko: askyesnocancel")

# przekazujemy nazwę okna do funkcji uruchamianej przez button
showinfo = partial(showinfo, nazwashowinfo)
btn1 = tk.Button(okno, text="showinfo", command=showinfo, pady=10, padx=10)
btn1.pack()

showwarning = partial(showwarning, nazwashowwarning)
btn2 = tk.Button(okno, text="showwarning", command=showwarning, pady=10, padx=10)
btn2.pack()

askokcancel = partial(askokcancel, nazwaaskokcancel)
btn3 = tk.Button(okno, text="askokcancel", command=askokcancel, pady=10, padx=10)
btn3.pack()

askyesnocancel = partial(askyesnocancel, nazwaaskyesnocancel)
btn4 = tk.Button(okno, text="askyesnocancel", command=askyesnocancel, pady=10, padx=10)
btn4.pack()

okno.mainloop()
```

link

```python
from tkinter import *
import webbrowser

okno = Tk()
okno.geometry("600x160")

def open_url(url):
    webbrowser.open_new_tab(url)

label1= Label(okno, text= "pracownia.palacmlodziezy.lodz.pl", cursor= "hand2", fg= "white", bg= "DodgerBlue", font= ('Arial 18'))
label2= Label(okno, text= "www.palacmlodziezy.lodz.pl", cursor= "hand2", fg= "white", bg="Tomato", font= ('Arial 18'))
label1.pack(pady=30)
label2.pack()

url1= 'https://pracownia.palacmlodziezy.lodz.pl'
url2= 'https://www.palacmlodziezy.lodz.pl'

label1.bind("<Button-1>", lambda e:open_url(url1))
label2.bind("<Button-1>", lambda e:open_url(url2))
okno.mainloop()
```

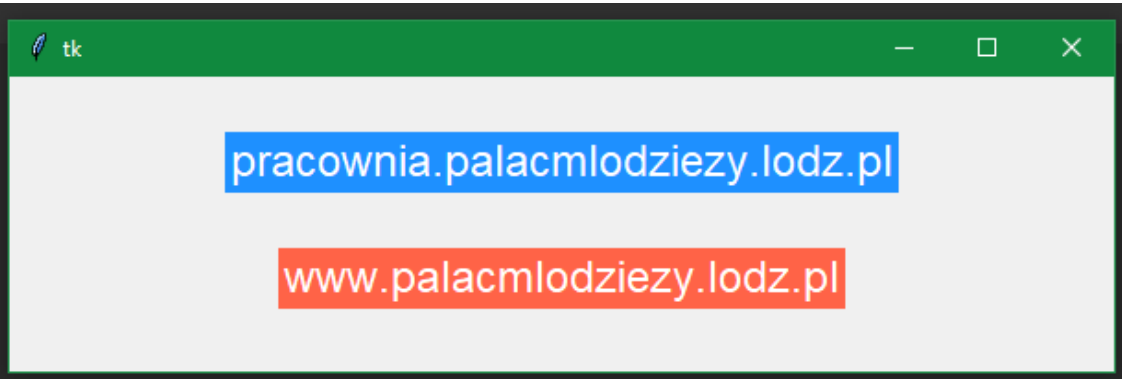
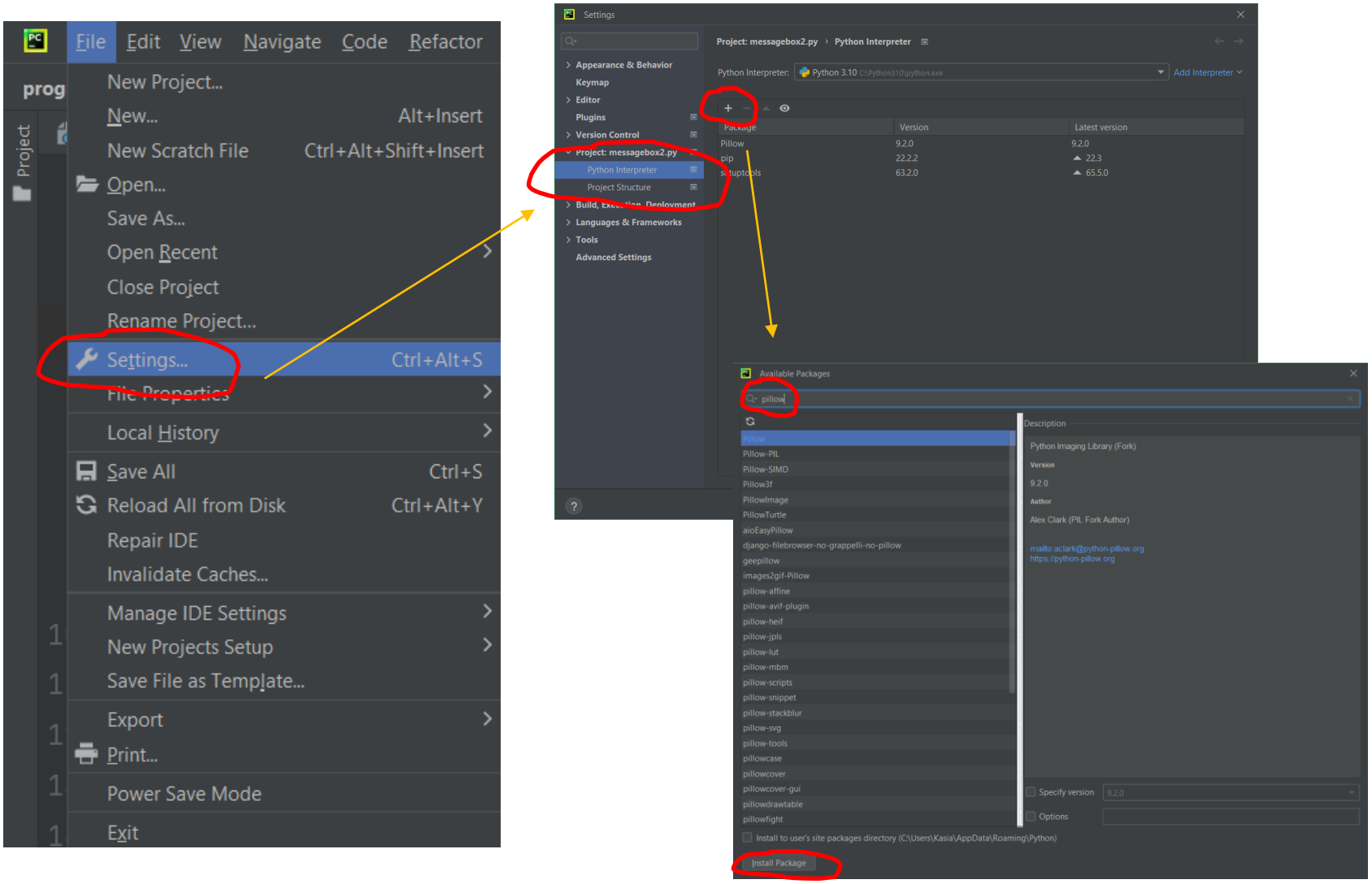


image – instalacja pillow

aby zainstalować bibliotekę pillow do obsługi zdjęć należy wpisać w linii komend:
py –m pip install pillow

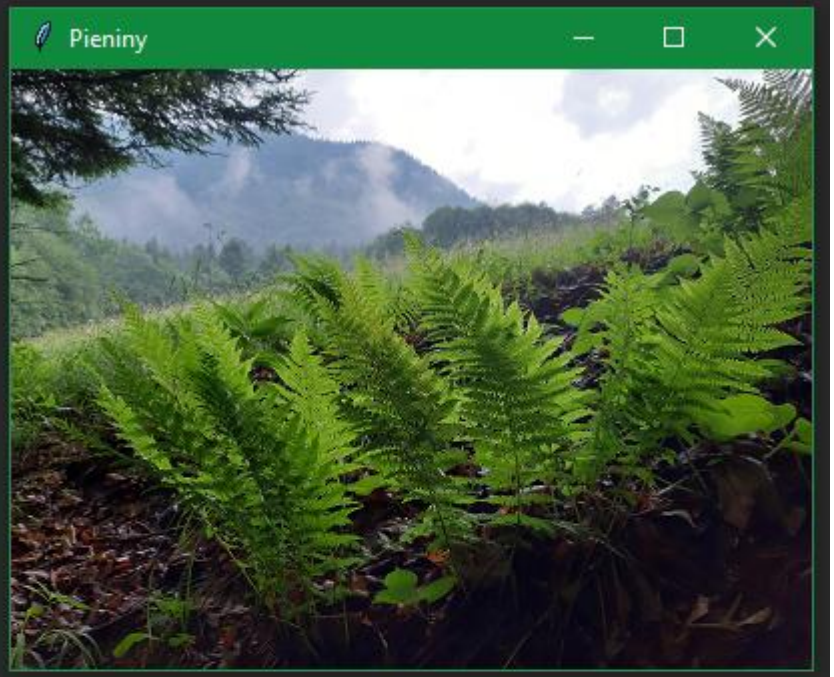
```
C:\>py -m pip install pillow
Collecting pillow
  Downloading Pillow-9.2.0-cp310-cp310-win_amd64.whl (3.3 MB)
     ---------------------------------------- 3.3/3.3 MB 13.1 MB/s eta 0:00:00
Installing collected packages: pillow
Successfully installed pillow-9.2.0

[notice] A new release of pip available: 22.2.2 -> 22.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

instalacja pillow w PyCharm



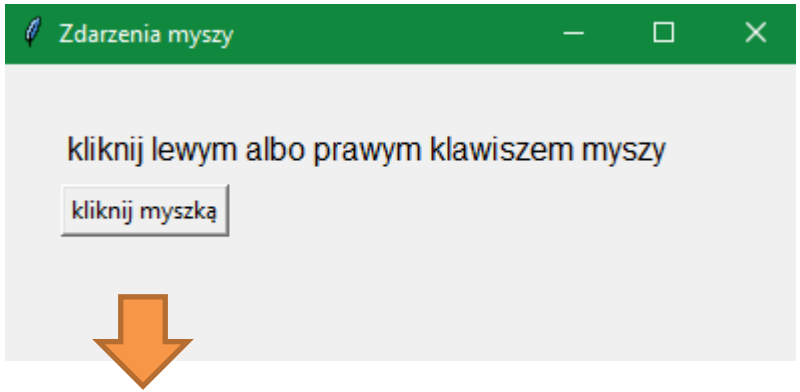
image

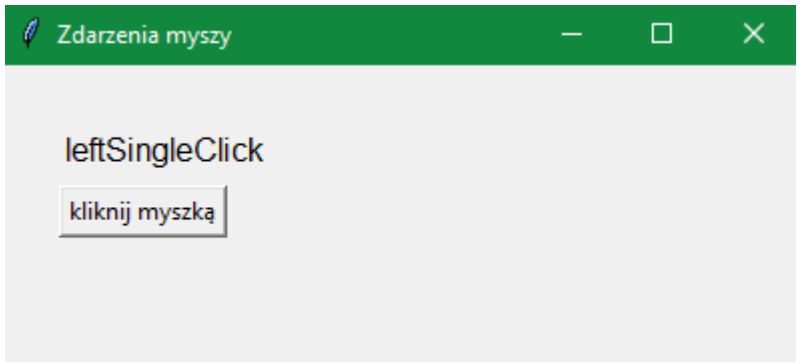
```python
import tkinter as tk
from PIL import ImageTk, Image

# tworzymy okno
okno = tk.Tk()
okno.geometry('400x300')
okno.title('Pieniny')

# przypisujemy zdjęcie do labela
img = ImageTk.PhotoImage(Image.open("Pieniny.png"))
picture = tk.Label(okno, image=img)
picture.pack(side="bottom", fill="both", expand="yes")

# uruchomienie okna
okno.mainloop()
```

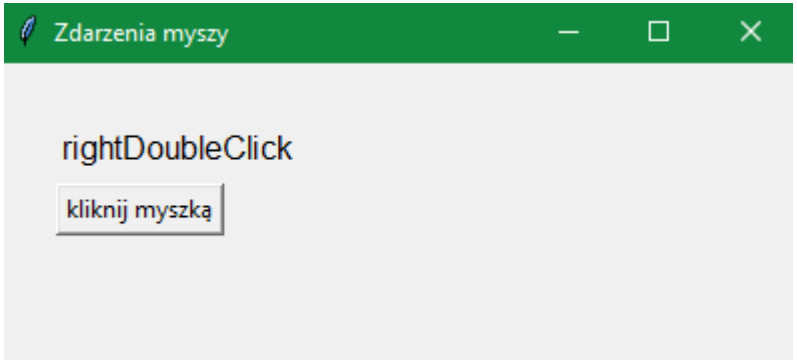


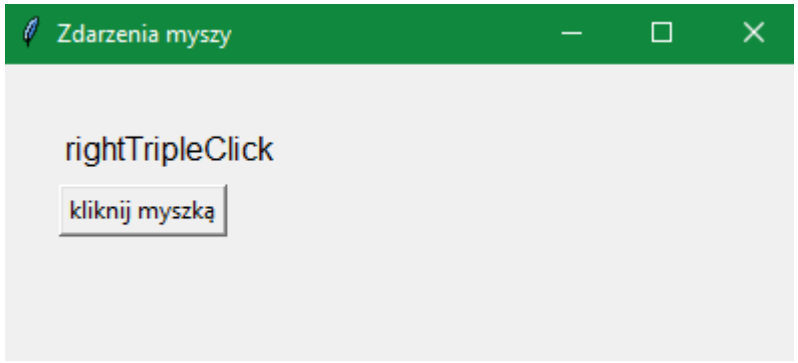
mouse events

kliknięcie lewym bądź prawym klawiszem myszy wypisuje stosowny komunikat









https://python-course.eu/tkinter/events-and-binds-in-tkinter.php

mouse events

```python
import tkinter as tk

# funkcje zwiazane ze zdarzeniami myszy
def leftSingleClick(event):
    # printowanie informacji w konsoli
    print("leftSingleClick")
    # wypisujemy stosowną informację o zaistniałym zdarzeniu
    labelInfo.config(text="leftSingleClick")

def leftDoubleClick(event):
    print("leftDoubleClick")
    labelInfo.config(text="leftDoubleClick")

def leftTripleClick(event):
    print("leftTripleClick")
    labelInfo.config(text="leftTripleClick")

def rightSingleClick(event):
    print("rightSingleClick")
    labelInfo.config(text="rightSingleClick")

def rightDoubleClick(event):
    print("rightDoubleClick")
    labelInfo.config(text="rightDoubleClick")

def rightTripleClick(event):
    print("rightTripleClick")
    labelInfo.config(text="rightTripleClick")
```

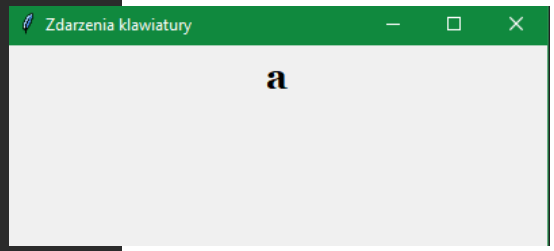
mouse events

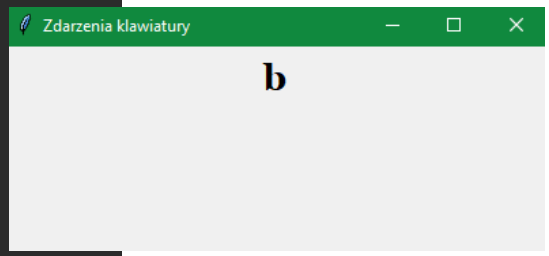
```
okno = tk.Tk()
# nazwa okna
okno.title("Zdarzenia myszy")
# wymiary okna
okno.geometry("400x150")

# miejsce wyświetlania informacji
labelInfo = tk.Label(okno, text="kliknij lewym albo "
            "prawym klawiszem myszy", font="Arial,50")
labelInfo.place(x=30, y=30)

# tworzymy przycisk do sprawdzania zdarzeń myszy
btn = tk.Button(okno, text='kliknij myszką')
btn.place(x=30, y=60)

# pojedyńcze wcisnięcie lewego klawisza myszy
# zbindowane (związane) z funkcją leftSingleClick
btn.bind('<Button-1>', leftSingleClick)
# podwójne wcisnięcie lewego klawisza myszy
btn.bind('<Double-1>', leftDoubleClick)
# potrójne wcisnięcie lewego klawisza myszy
btn.bind('<Triple-1>', leftTripleClick)
# pojedyńcze wcisnięcie prawego klawisza myszy
btn.bind('<Button-3>', rightSingleClick)
# podwójne wcisnięcie prawego klawisza myszy
btn.bind('<Double-3>', rightDoubleClick)
# potrójne wcisnięcie prawego klawisza myszy
btn.bind('<Triple-3>', rightTripleClick)
btn.mainloop()

## uruchomienie okna
okno.mainloop()
```

keyboard events

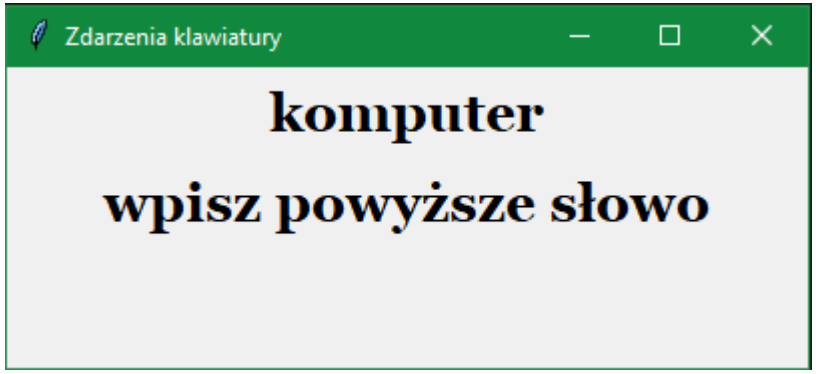
```python
import tkinter as tk

# funkcja wyświetlająca symbol klawisza
def pressdKey(event):
    labelInfo.config(text=event.keysym)

okno = tk.Tk()
# nazwa okna
okno.title("Zdarzenia klawiatury")
# wymiary okna
okno.geometry("400x150")

labelInfo = tk.Label(okno, text="symbol klawisza", font=('Georgia 20 bold'))
labelInfo.pack(pady=4)

# bindujemy naciśnięcie klawisza Enter
okno.bind('<Return>', pressdKey)

# bindujemy naciśnięcie klawisza a
okno.bind('<a>', pressdKey)

# bindujemy naciśnięcie klawisza b
okno.bind('<b>', pressdKey)

## uruchomienie okna
okno.mainloop()
```

symbol naciśniętego klawisza wyświetla się w labelu

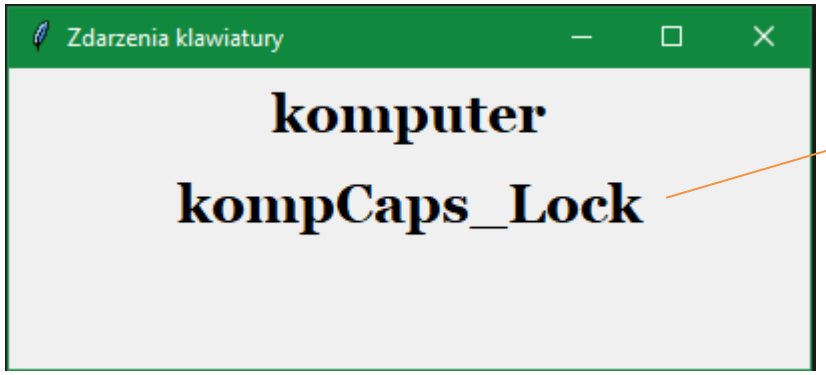




keyboard events



użytkownik ma wpisać słowo *komputer*



właściwie naciśniete litery oraz bieżąca są wyświetlane

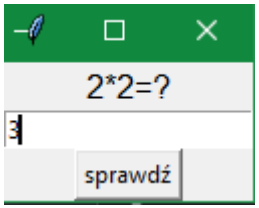
gdy użytkownik wpisze poprawnie wyraz program kończy pracę

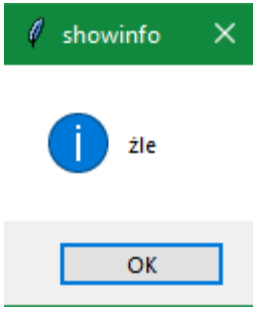
keyboard events

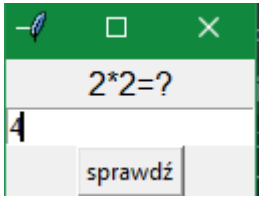
```python
import tkinter as tk

# funkcja przycisk
def przycisk(event):
    global i
    global wypisz
    global slowo
    # symbol klawisza nacisniętego przez użytkownika
    symbol = event.keysym
    # lokalna zmienna wpis
    # (zatrzymuje lokalnie tylko poprawne symbole)
    # zmienna wypisz przetrzymuje wpisane poprzednio litery
    wpis = wypisz
    # jeśli skoczą się litery w słowie kończymy gre
    if i >= len(slowo):
        labelInfo.config(text="Game Over")
    else:
        # jeśli użytkownik nacisnął właściwy klawisz
        if symbol == slowo[i]:
            # dopisujemy go do zmiennej wypisz
            wypisz = wypisz +  symbol
            # inkrementujemy index
            i = i + 1
            # wypisujemy poprawne symbole
            labelInfo1.config(text=wypisz)
        else:
            # jeśli użytkownik nacisnął niewłaściwy klawisz
            # wypisujemy właściwe symbole razem z niewłaściwym
            wpis = wypisz + symbol
            labelInfo1.config(text=wpis)

# poprawne znaki wprowadzone przez użytkownika
wypisz = ""
# index znaku w słowie
i = 0
# słowo do wpisania przez użytkownika
slowo = "komputer"

okno = tk.Tk()
okno.title("Zdarzenia klawiatury")
okno.geometry("400x150")

# label ze słowem do wpisania
labelInfo = tk.Label(okno, text=slowo, font=('Georgia 20 bold'))
labelInfo.pack(pady=4)

# label ze znakami wpisanymi przez użytkownika
labelInfo1 = tk.Label(okno, text="wpisz powyższe słowo", font=('Georgia 20 bold'))
labelInfo1.pack(pady=4)

# zbindowanie (połaczenie) zdarzenia naciśnięcia przycisku na klawiaturze z funkcją przycisk
okno.bind("<Key>", przycisk)

okno.mainloop()
```

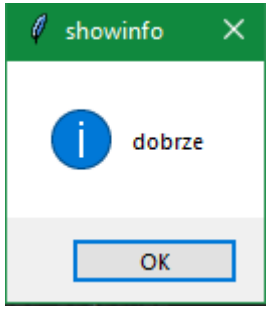
zagadka

```python
import tkinter as tk
from functools import partial
from tkinter import messagebox

# funkcja sprawdzająca odpowiedź użytkownika
def sprawdz(a):
    if a.get() == "4":
        messagebox.showinfo("showinfo", "dobrze")
    else:
        messagebox.showinfo("showinfo", "źle")

okno = tk.Tk()
# nazwa okna
okno.title("Zagadka")


# miejsce wyświetlania zagadkii
labelInfo = tk.Label(okno, text="2*2=?", font="Arial 12")
labelInfo.grid(row=0, column=0)

# komponent Entry do pobrania odpowiedzi użytkownika
a = tk.StringVar()
entryLiczbaA = tk.Entry(okno, textvariable=a)
entryLiczbaA.grid(row=1, column=0)

# przycisk do sprawdzenia odpowiedzi użytkownika
sprawdz=partial(sprawdz,a);
button = tk.Button(okno, text="sprawdź", command=sprawdz)
button.grid(row=3, column=0)

## uruchomienie okna
okno.mainloop()
```



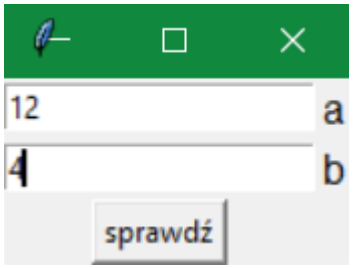


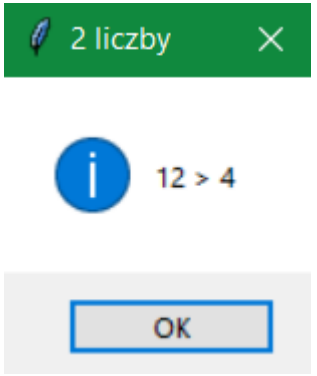


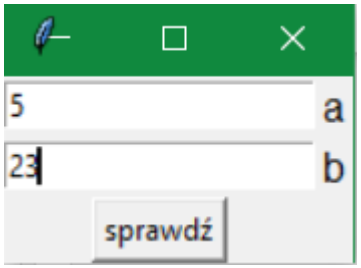


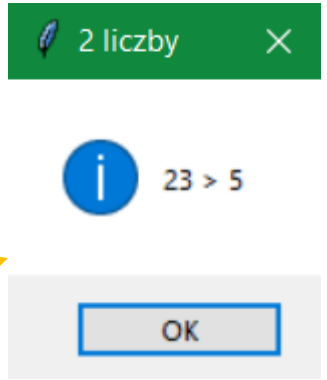
2 liczby

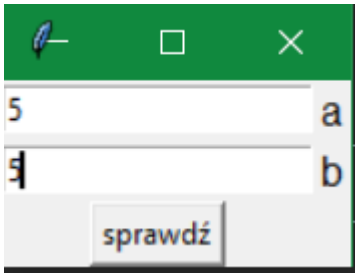
```python
import tkinter as tk
from functools import partial
from tkinter import messagebox



# funkcja porównująca 2 liczby
def sprawdz(a , b):
    if int(a.get()) > int(b.get()):
        messagebox.showinfo("2 liczby", a.get() + " > " + b.get())
    elif int(b.get()) > int(a.get()):
        messagebox.showinfo("2 liczby", b.get() + " > " + a.get())
    else:
        messagebox.showinfo("2 liczby", b.get() + " == " + a.get())

okno = tk.Tk()
# nazwa okna
okno.title("2 liczby")
# label do wyświetlenia nazwy a
labelInfo = tk.Label(okno, text="a", font="Arial 12")
labelInfo.grid(row=0, column=1)

# label do wyświetlenia nazwy b
labelInfo = tk.Label(okno, text="b", font="Arial 12")
labelInfo.grid(row=1, column=1)

# komponent Entry do wpisania pierwszej liczby
a = tk.StringVar()
entryLiczbaA = tk.Entry(okno, textvariable=a)
entryLiczbaA.grid(row=0, column=0)

# komponent Entry do wpisania drugiej liczby
b = tk.StringVar()
entryLiczbaB = tk.Entry(okno, textvariable=b)
entryLiczbaB.grid(row=1, column=0)

# przekazanie 2 argumentów (liczb a oraz b) do funkcji sprawdz
sprawdz=partial(sprawdz, a, b)
# przycisk uruchamiający funkcję porównującą liczby
button = tk.Button(okno, text="sprawdź", command=sprawdz)
button.grid(row=3, column=0)
## uruchomienie okna
okno.mainloop()
```

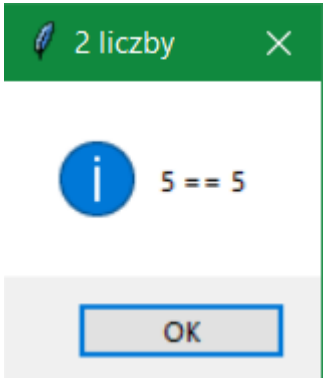




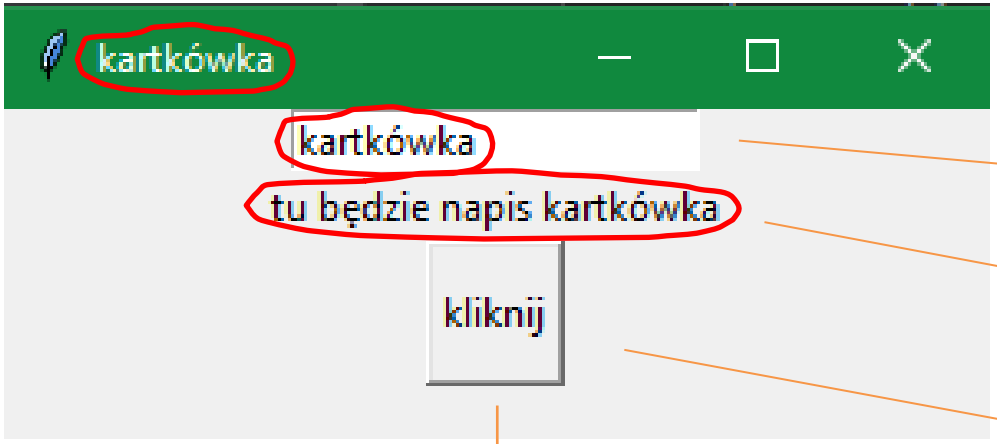








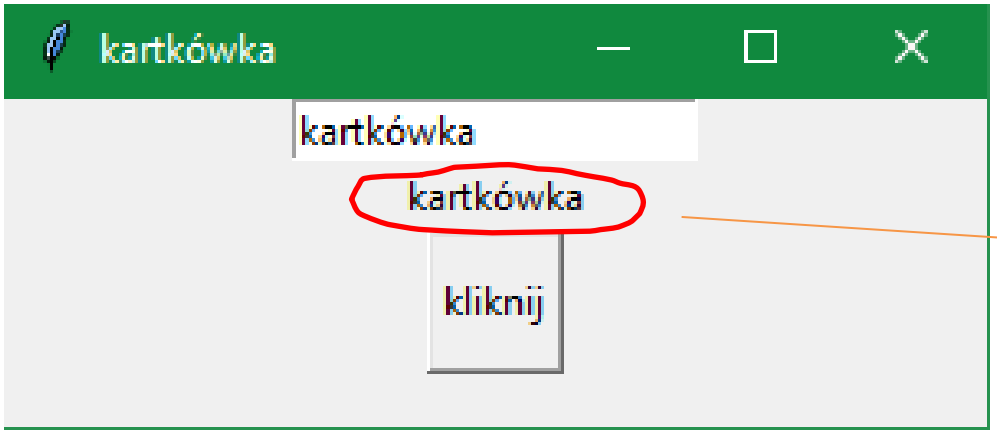
kartkówka



Entry

Label

Button



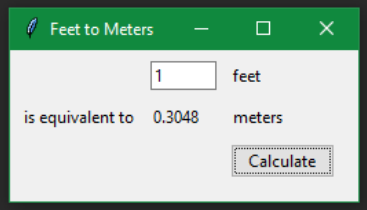
po naciśnięciu buttona zmiana napisu w Labelu

kartkówka

```
import tkinter as tk
from functools import partial

# tworzymy okno
okno = tk.Tk()
okno.title("kartkówka")
okno.geometry("300x100")

# funkcja z parametrem napis uruchamiana po naciśnięciu buttona
def fun(napis):
    # zmiana tekstu wyświetlanego przez komponent label na treść zmiennej napis
    label.config(text=napis.get())

# definicja zmiennej napis z wartością początkową "kartkówka"
napis = tk.StringVar(okno,"kartkówka")

# komponenet Entry z przypisaną zmienną napis
entry = tk.Entry(okno, textvariable=napis)
entry.pack()

# komponent Label z początkowym napisem
label = tk.Label(okno, text="tu będzie napis kartkówka")
label.pack()

# funkcja fun z parametrem napis
fun = partial(fun, napis)

# komponent button, po naciśnięciu wywoływana jest funkcja fun z parametrem napis
btn = tk.Button(okno, text="kliknij", command=fun, pady=10)
btn.pack()

okno.mainloop()
```

FeetToMeters

```
from tkinter import *
from tkinter import ttk

def calculate(*args):
    try:
        value = float(feet.get())
        meters.set(int(0.3048 * value * 10000.0 + 0.5)/10000.0)
    except ValueError:
        pass

root = Tk()
root.title("Feet to Meters")

mainframe = ttk.Frame(root, padding="3 3 12 12")
mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1)

feet = StringVar()
feet_entry = ttk.Entry(mainframe, width=7, textvariable=feet)
feet_entry.grid(column=2, row=1, sticky=(W, E))

meters = StringVar()
ttk.Label(mainframe, textvariable=meters).grid(column=2, row=2, sticky=(W, E))

ttk.Button(mainframe, text="Calculate", command=calculate).grid(column=3, row=3, sticky=W)

ttk.Label(mainframe, text="feet").grid(column=3, row=1, sticky=W)
ttk.Label(mainframe, text="is equivalent to").grid(column=1, row=2, sticky=E)
ttk.Label(mainframe, text="meters").grid(column=3, row=2, sticky=W)

for child in mainframe.winfo_children():
    child.grid_configure(padx=5, pady=5)

feet_entry.focus()
root.bind("<Return>", calculate)
```



https://tkdocs.com/tutorial/firstexample.html

// tkinter reference
https://anzeljg.github.io/rin2/book2/2405/docs/tkinter/