

Dokumentacja techniczna aplikacji pod tytułem "Kalkulator matematyczny WSB"

Twórca programu: Szymon Napiórkowski.

Spis treści:

1.Na czym polega aplikacja.

2.Sposób działania.

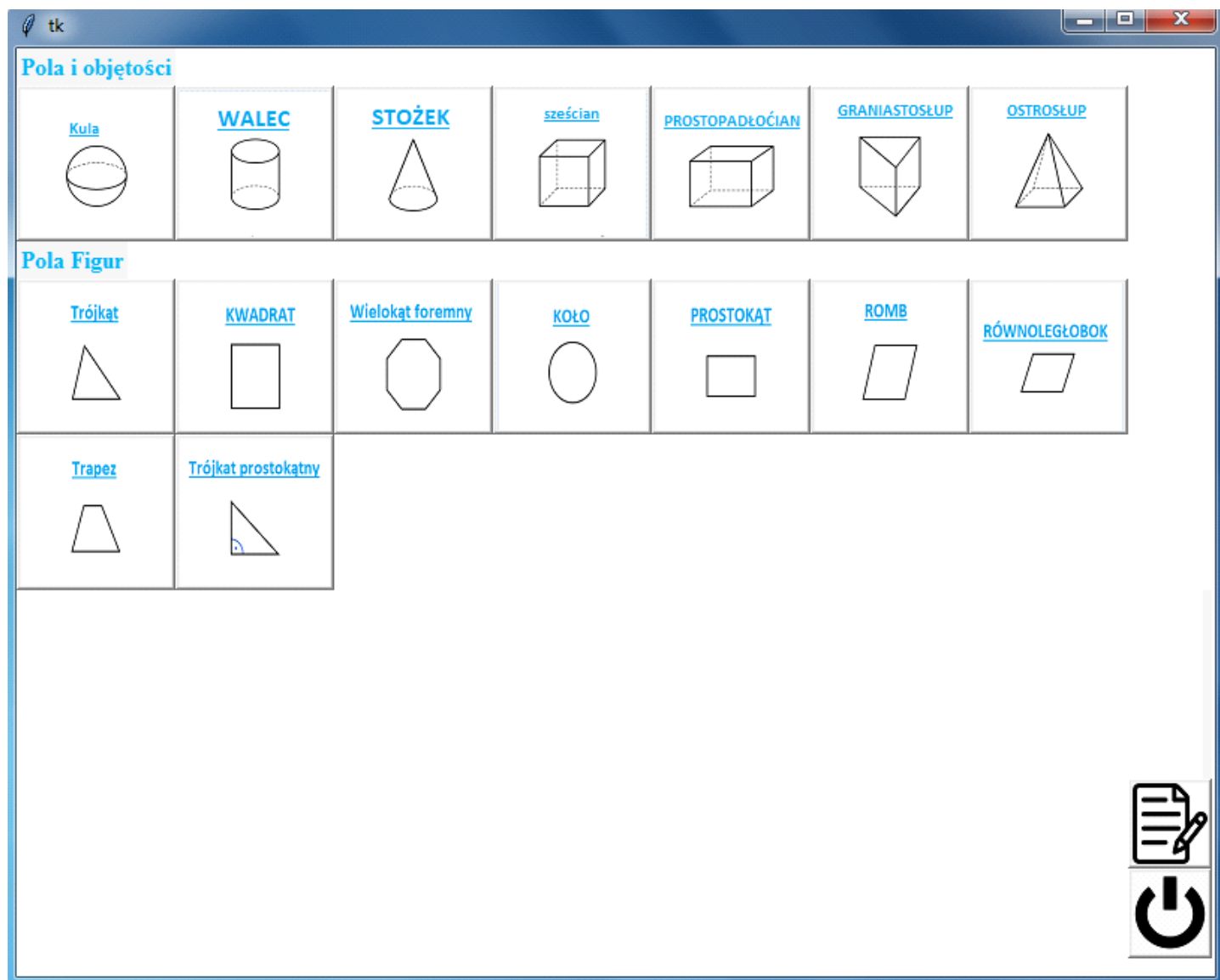
3.Zastosowane możliwości języka.

1. Na czym polega aplikacja.

1. Idea projektu.

Projekt został stworzony dla uczniów i studentów, którzy mają problemy z bryłami obrotowymi i figurami płaskimi w obliczaniu ich pól i obwodów. Dzięki tej aplikacji z łatwością po wpisaniu odpowiednich danych można wyliczyć np. pole całkowite stożka. W aplikacji znajdują się również cenne informacje takie jak wzory czy wygląd danej bryły.

Wygląd programu:

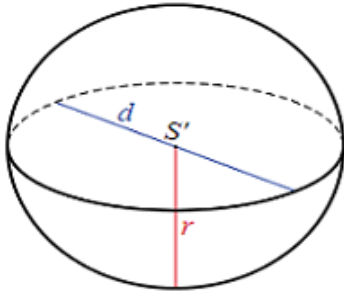


tk

Objętość i powierzchnia kuli

Kula jest określona promieniem lub średnicą, jest zbiorem punktów, które są odległe od środka maksymalnie długością równą promieniowi.

Wzory



$$V = \frac{4}{3}\pi r^3$$

$$P = 4 \cdot \pi r^2$$

$$d = 2 \cdot r$$

V – objętość

r – promień

P – powierzchnia

S' – środek kuli

KALKULATOR

Podaj promień:

0

Pole :

Objętość :

Oblicz

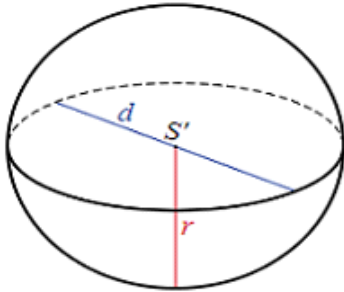
Przykładowe okienko bryły:

tk

Objętość i powierzchnia kuli

Kula jest określona promieniem lub średnicą, jest zbiorem punktów, które są odległe od środka maksymalnie długością równą promieniowi.

Wzory



$$V = \frac{4}{3}\pi r^3$$

$$P = 4 \cdot \pi r^2$$

$$d = 2 \cdot r$$

V – objętość

r – promień

P – powierzchnia

S' – środek kuli

KALKULATOR

Podaj promień:

0

Pole :

Objętość :

Oblicz

2. Sposób działania

1. Jak to działa?

Użycie aplikacji jest banalnie proste i intuicyjne. Wystarczy, że wybierzemy odpowiednią figurę i jeżeli chcemy obliczyć pole wystarczy wpisać dane, a wynik wyskoczy automatycznie.

Obliczenia:

Podaj promień

Podaj wysokość

Oblicz

Po wpisaniu danych wystarczy kliknąć oblicz.

Obliczenia:

Podaj promień

Podaj wysokość

Oblicz

Pole powierzchni podstawy stożka wynosi: 50.27

Pole powierzchni bocznej stożka wynosi: 90.62

Pole powierzchni całkowitej stożka wynosi: 140.88

Objętość stożka wynosi: 100.53

2. Język i biblioteki.

Program został stworzony w języku Python przy pomocy biblioteki Tkinter,

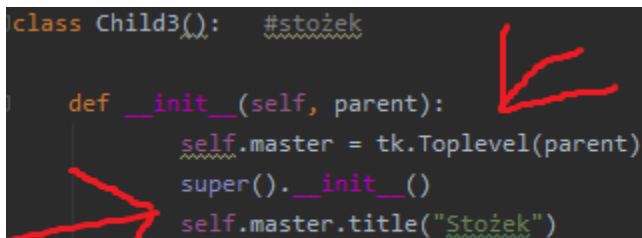
która umożliwiła stworzenie interfejsu graficznego. Wystarczyło je zimportować.

```
import tkinter as tk
from tkinter import *
import math
from tkinter import messagebox
from tkinter import scrolledtext
from abc import ABC, abstractmethod
```

Program opiera się na klasach i ich dziedziczeniu, każda bryła czy figura posiada swoją klasę tak samo jak menu programu. Dziedziczą one wygląd, wielkość czy nazwę okienka.

```
class Child3(): #stożek

    def __init__(self, parent):
        self.master = tk.Toplevel(parent)
        super().__init__()
        self.master.title("Stożek")
```



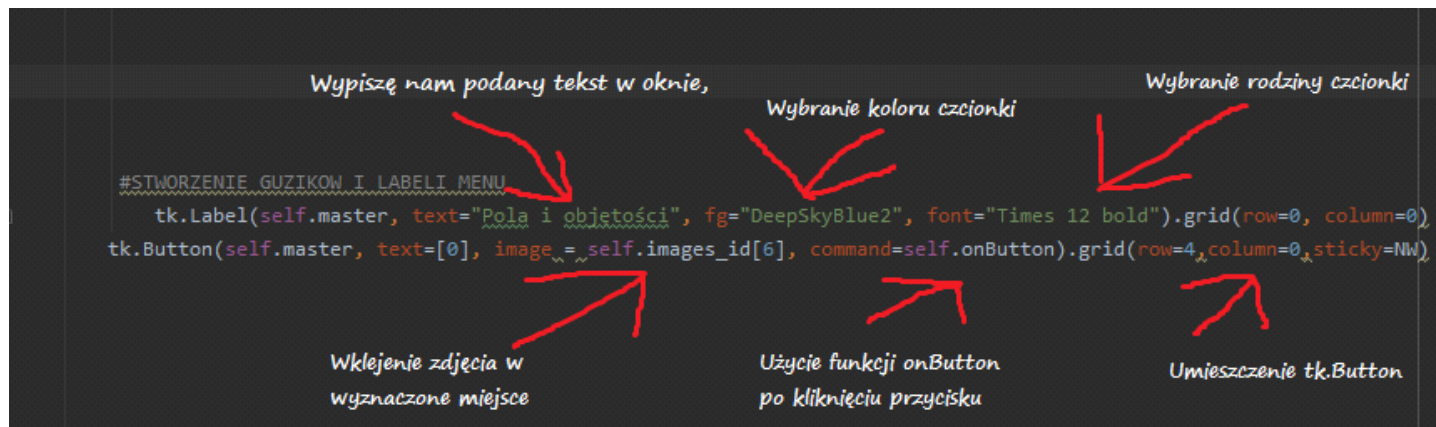
Biblioteka tkinter jest bardzo ciekawą biblioteką dzięki, której możemy stworzyć okna aplikacji ustalić jej wielkość, tytuł czy kolor tła i wiele innych.

```
class Main(ABC): #stworzenie klasy rodzica

    def __init__(self):
        self.master = tk.Tk() # bezposrednie tworzenie okna
        self.master.geometry("800x620") #wielkosc okna
        self.master.resizable(width=True, height=True) #ustawienia okna
        self.master.configure(background="white") # tlo okna
```

Mamy również możliwość zmiany czcionki na inną czy rozmieszczenie napisów

w odpowiednich miejscach.



3.Zastosowane możliwości języka.

W aplikacji zostało użyte wiele klas, funkcji czy pętli pokaże kilka z nich żeby wyjaśnić ich działanie.

Tak jak na zdjęciu poniżej widać użycie pętli `for`, w celu skrócenia kodu gdyż zdjęć w programie zostało użytych bardzo dużo. Stworzenie tablicy pozwoliło nie hierarchizację danych zdjęć i szybszy oraz łatwiejszy dostęp do nich.

```
##stworzenie tablicy z plikami zdjęć figur
images_names = ["walec.png", "stozek.png", "szescian.png", "prostopadloscian.png", "graniastoslup.png", "ostroslup.png", "kula.png", "quit.png", "kole.png", "kwadrat.png"]
self.images_id = []
for im_name in images_names: ##stworzenie petli ktora bedzie wypisywac po kolei indeksy kazdego zdjecia
    self.images_id.append(tk.PhotoImage(file=im_name))
```

Przykładowe użycie tkinter w połączeniu z banalną funkcją:

```
def zamknij(self):  
    self.master = tk.Tk()    #funkcja do zamknięcia aplikacji  
    self.master.destroy()    #proste użycie biblioteki dzięki której  
    exit()                   # możemy wyłączyć aplikację
```

Stworzenie kalkulatora było największym wyzwaniem połączenie biblioteki tk oraz math nie było łatwe. Trzeba było przekonwertować tk.IntVar(), który był stringiem w float dzięki czemu można było obliczyć cokolwiek. Funkcje typu oblicz2 liczą nasze pole kuli następnie formatują stringu podane przez użytkownika, aż w końcu wypisują wynik po wciśnięciu przycisku oblicz.

```
#STWORZENIE OKNA ENTRY DO WPISYWANIA WARTOŚCI  
number1 = tk.IntVar()  
tk.a = Entry(self.master, textvariable=number1, width=8)  
tk.a.grid(column=0, row=18, sticky=NW)  
a = float(tk.a.get()) #ZAMIANIENIE ZE STRINGA NA FLOAT ENTRY  
  
wynik3 = tk.Label(self.master, text="Objętość : ", bg="white", fg="navy", font="none 13 bold")  
wynik3.grid(column=0, row=20, sticky=NW)  
  
def oblicz2(): #wzór na pole kuli  
    kulson2 = 4/3 * math.pi * float(tk.a.get())*float(tk.a.get())*float(tk.a.get())  
    wynik4 = f"Objętość : {round(kulson2, 2)}" #formatowanie stringow  
    wynik3.configure(text=wynik4) #wypisanie wyniku w odpowiednim miejscu  
  
    kulson = 4 * math.pi * float(tk.a.get()) * float(tk.a.get())  
    wynik = f"Pole : {round(kulson, 2)}"  
    wynik2.configure(text=wynik)  
  
przycisk = Button(self.master, text="Oblicz", command=oblicz2, bg="white", fg="navy",  
                  font="none 12 bold").grid(column=0, row=21, sticky=NW) #przycisk oblicz
```


Tk.Label służy to usadowienia odpowiednich napisów typu "Obliczenia: " na swoich miejscach ta funkcja ma również możliwość wklejenie w te miejsce zdjęcia czy zmiana koloru tła czy czcionki.

```
tk.Label(self.master, image=img1).grid(column=0, row=0, sticky=E)

tk.Label(self.master, text="Obliczenia: ", bg="white", fg="midnight blue",
        font="none 13 bold").grid(column=0, row=14, sticky=NW)
tk.Label(self.master, text="Podaj promień", bg="white", fg="midnight blue",
        font="none 13 bold").grid(column=0, row=15, sticky=NW)
tk.Label(self.master, text="Podaj wysokość", bg="white", fg="midnight blue",
        font="none 13 bold").grid(column=0, row=15, sticky=S)
```