Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Proiect: Interfețe Om-Mașină

Sistem ce permite rostirea denumirii fișierului audio ce va fi apoi redat și afișat pe ecran

Profesor coordonator: Andreea Griparis

Studenți: Jurubiță Miruna, 441A
Buzea Florin-Dorinel, 441A
Ciucu Ninel-Gabriel, 441A

București 2022

Cuprins

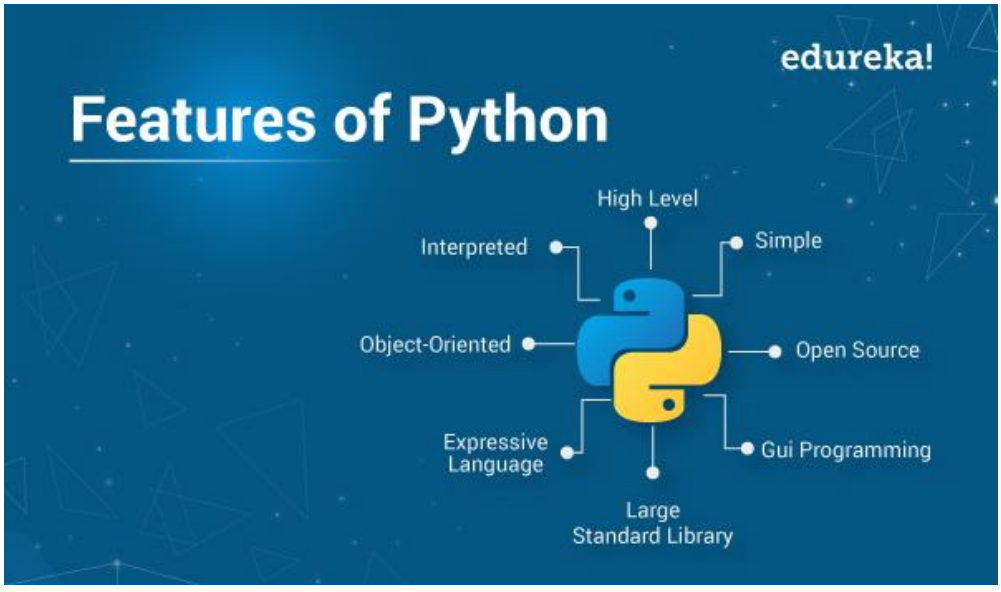
Limbajul de programare Python

Python este un limbaj de programare de nivel înalt, interpretat și orientat pe obiecte cu semantică dinamică. Structurile sale de date de nivel înalt, combinate cu tastarea dinamică, il fac foarte atractiv pentru dezvoltarea rapidă a aplicațiilor, precum și pentru utilizare ca limbaj de scriptare.

Python este unul dintre cele mai populare limbaje de programare datorită bibliotecilor sale extinse distribuite în mod gratuit și a productivității crescute pe care o oferă. Deoarece nu există pas de compilare, ciclul de editare-test-debug este extrem de rapid. Sintaxa simplă, ușor de învățat, subliniază lizibilitatea și ,prin urmare, reduce costurile de întreținere a programului. Python acceptă module și pachete, ceea ce încurajează modularitatea programului și reutilizarea codului. [1]

Python suportă multiple paradigme de programare, inclusiv programare structurată, orientată pe obiecte și funcțională. Alte paradigme pot fi utilizate prin intermediul altor extensii.[2]

Codul din Python ruleaza printr-un interpretor in timp ce la C++ este pre-compilat. Prin urmare, atunci când rulăm un fișier cu extensia ".py", este apelat interpretorul, care compilează automat scriptul în codul compilat numit și "byte code".[3]



Avantaje Python:

- Ușor de învățat, citit, scris și întreținut, de aceea este recomandat începătorilor.
- Poate rula pe diverse platforme hardware folosind aceeași interfață.
- Pot fi incluse module de nivel scăzut în interpretorul Python.
- Oferă o structură ideală și suport pentru programele mari.
- Acceptă un mod interactiv de testare și debug.
- Oferă tipuri de date dinamice la nivel înalt și acceptă verificarea tipului dinamic.
- Limbajul Python poate fi integrat cu codul de programare Java, C și C++. [4]

Mediul de dezvoltare Spyder

Spyder este un mediu de dezvoltare integrat (IDE) “open-source” și “cross-platform” pentru programare științifică în limbajul Python. Spyder se integrează cu o serie de pachete proeminente: NumPy, SciPy , Matplotlib , panda , IPython , SymPy și Cython precum și alte software-uri open source.[5]

Spyder este extensibil cu plugin-uri primare și terțe[6]. Include suport pentru instrumente interactive pentru inspecția datelor și încorporează instrumente de introspecție și asigurare a calității codului specifice Python. Este disponibil cross-platform prin Anaconda [7]. Anaconda este o distribuție a limbajelor de programare Python și R pentru calcul științific, care are ca scop simplificarea gestionării și implementării pachetelor.[8]

Caracteristici Spyder:

- Un editor cu evidențierea sintaxelor , introspecția , completarea codului
- Suport pentru mai multe console IPython
- Abilitatea de a explora și edita variabile dintr-o interfață grafică
- Un panou de ajutor care poate prelua și reda documentația text despre funcții, clase și metode automat sau la cerere
- Un debugger conectat la IPdb, pentru execuție pas cu pas
- Analiza codului static , realizată de Pylint
- Un Profiler de rulare , pentru a evalua codul
- Un explorator de fișiere încorporat , pentru interacțiunea cu sistemul de fișiere și gestionarea proiectelor
- O funcție „Găsiți în fișiere”, care permite căutarea completă a expresiilor regulate într-un domeniu specificat
- Un browser de ajutor online, care permite utilizatorilor să caute și să vadă documentația Python și a pachetului în interiorul IDE-ului
- Un jurnal istoric care înregistează fiecare comandă introdusă de utilizator în fiecare consolă

O consolă internă, care permite introspecția și controlul asupra propriei operațiuni [9]

Descrierea codului aplicației

```
8
9   import os
10  import tkinter as tk
11  from gtts import gTTS
12  from playsound import playsound
13  from scipy.io import wavfile
14  from matplotlib.figure import Figure
15  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
16
17
```

Fig. 1 – Librării folosite

Librării folosite:

- os – pentru lucrul cu fișiere
- tkinter – pentru interfața grafică și elementele acesteia
- gtts – pentru realizarea sintezei vocale
- playsound – pentru redarea fișierelor audio
- wavfile(din scipy.io) –pentru citirea unui fișier wav
- Figure și FigureCanvasTkAgg – pentru crearea elementului ce va conține graficul semnalului

```
81  #creare fereastra goala
82  window1 = tk.Tk()
83
84  # denumirea interfetei
85  window1.title('Plot in Tkinter')
86
87  # dimensiunile interfetei
88  window1.geometry("425x550")
89
90  # Crearea butonului care permite selectarea fisierului wav
91  btn = tk.Button(window1, height = 2, width = 16, text ='Cauta fisier', command = open_file)
92  btn.grid(row = 0 , column = 1 , columnspan = 1, sticky = 'w')
93
94  w1 = tk.Label(window1 ,width = 16, text = 'Denumire fisier:')
95  w1.grid(row = 1, column = 0,columnspan = 1, sticky = 'w')
96
97  t1 = tk.Text(window1, height = 1, width = 15)
98  t1.grid(row = 1, column = 1 ,columnspan = 2 ,sticky = 'nesw')
99
100 window1.mainloop()
101
```

Fig. 2 – Crearea ferestrei și primelor elemente ale aplicației

Prin codul de mai sus se creează întâi fereastra interfeței și se setează dimensiunea acesteia, apoi se inițializează primele elemente din fereastră , anume un buton ce va apela metoda “open_file” când este apăsat și un textbox ce va conține denumirea fișierului selectat, împreună cu un label pentru indicarea rezultatului obținut.

Principalele metode folosite: “open_file”, “ denumire”, “afiseaza” , “sunet”

```
def open_file():
    global data
    global filepath

    t1.delete('1.0',tk.END)


    filepath = tk.filedialog.askopenfilename(title='select', filetypes= [("all wav format", ".wav")])

    t1.insert(tk.END, os.path.basename(filepath))
    fs,data = wavfile.read(filepath)
```

Fig. 3 – Metoda “open_file”. Funcționalitatea de selectare a unui fișier

Inițial în metoda open_file se folosește o funcție din libraria tkinter pentru deschiderea unei ferestre de dialog pentru selectarea unui fișier de tip wav, iar numele acestui fișier este preluat și introdus în textbox-ul definit anterior. Variabilele data și filepath vor fi definite global deorece vor fi folosite în alte funcții ale aplicației.

```
31
32      # butonul care va afisa forma de unda a semnalului din fiserul selectat
33      btn2 = tk.Button(master = window1, command = afiseaza,
34                               height = 2, width = 16, text = "Afiseaza")
35      btn2.grid(row = 3,column = 0, columnspan = 1, sticky = 'nesw')
36
37      #butonul actionat pentru rostirea denumirii fisierului
38      btn3 = tk.Button(master = window1, command = denumire,
39                               height = 2, width = 16, text = "Rosteste denumire ")
40      btn3.grid(row = 3,column = 1, columnspan = 1,sticky = 'nesw')
41
42      #butonul acitonat pentru redarea fisierului selectat
43      btn4 = tk.Button(master = window1, command = sunet,
44                               height = 2, width = 16, text = "Reda sunetul")
45      btn4.grid(row = 3, column = 2, sticky = 'nesw')
46
```

Fig. 4 – Metoda “open_file” . Crearea elementelor.

Tot în interiorul funcției se vor crea restul elementelor interfeței, astfel aceste elemente devenind vizibile abia după selectarea fișierului wav dorit. Interfața se populează cu alte 3 butoane intitulate “Afiseaza” , “Rosteste denumirea” , “Reda sunetul” ce vor apela funcțiile “afisare” , “denumire” și respectiv “sunet” când vor fi apăsate.

```
def denumire():
    mytext = t1.get('1.0',tk.END)
    myspeech = gTTS(mytext, lang = 'en', slow = False)
    myspeech.save('denumire.mp3')
    os.system('')
    playsound('denumire.mp3', True)
    os.remove('denumire.mp3')
```

Fig. 5 – Metoda “denumire”

Prin metoda “denumire” se realizează sinteza vocală a denumirii fișierului, se creează astfel un fișier mp3 și cu funcția „playsound” , din librăria cu același nume, se va reda fișierul creat. Se folosește metoda “remove” din librăria “os” pentru stergerea fișierului după ce a fost redat, astfel făcându-se posibilă apăsarea repetată a butonului.

```
def sunet():
    playsound(filepath)
```

Fig. 6 – Metoda “sunet”

Funcția “sunet” folosește variabila “filepath” ,declarată de tip global în metoda “open_file”, și funcția “playsound” folosită și mai sus, pentru a reda fișierul selectat.

```
def afiseaza():

    #label
    w2 = tk.Label(window1,width = 15, text = 'Forma semnalului:')
    w2.grid(row = 4, column = 0, sticky = 'w')

    # crearea figurii care va contine graficul semnalului
    fig = Figure(figsize = (4.25, 4), dpi = 100)

    # adaugarea subgraficului
    plot1 = fig.add_subplot(111)

    # afisarea semnalului
    plot1.plot(data)

    # crearea panzei (canvas) care va contine figura
    canvas = FigureCanvasTkAgg(fig, master = window1)
    canvas.draw()

    # pozitionarea panzei
    canvas.get_tk_widget().grid(row = 5, column = 0, columnspan = 10)
```

Fig. 7 – Metoda “afiseaza”

În metoda “afiseaza” se crează o etichetă sugestivă pentru următorul element , acesta fiind o pânză creată prin apelarea metodei “FigureCanvasTkAgg”, iar aceasta va conține o figură creată prin funcția “Figure” , unde se va adăuga graficul formei de undă al semnalului fișierului wav selectat.

Interfața aplicației

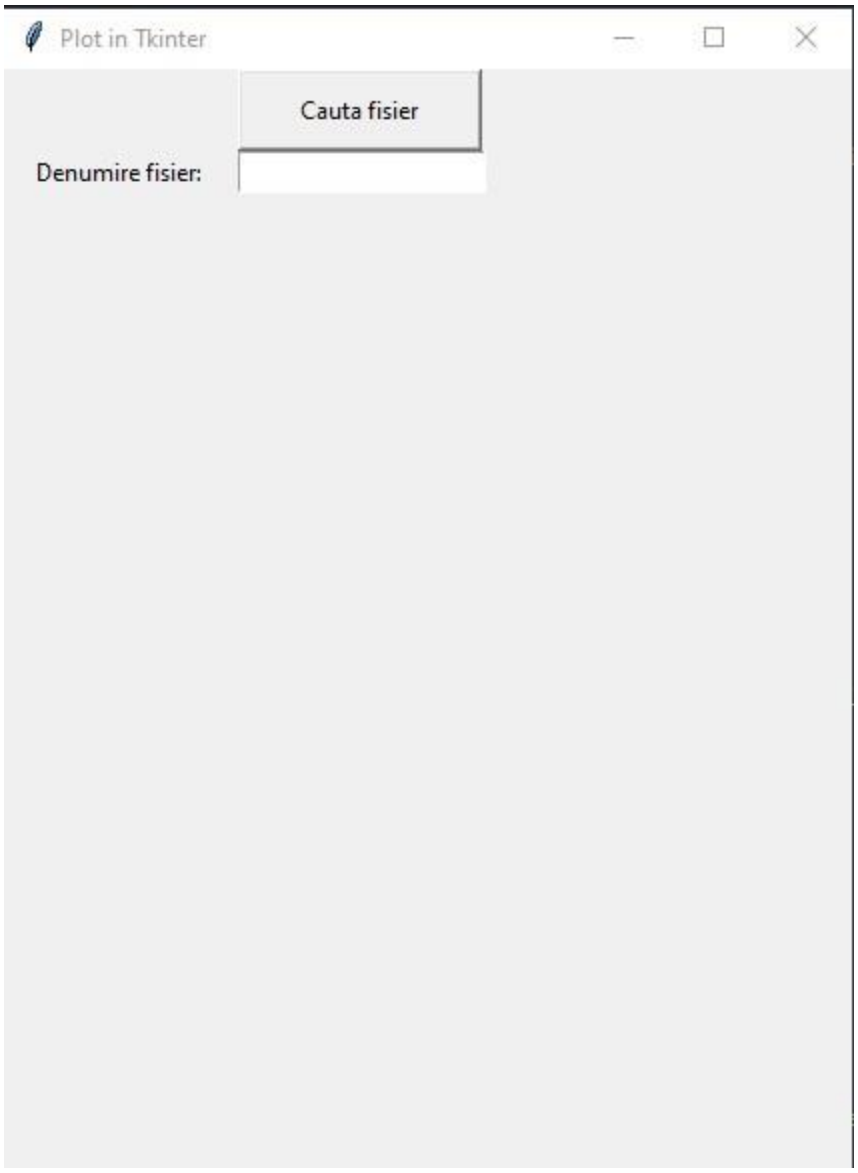


Fig. 8 – Interfața aplicației înaintea selectării unui fișier

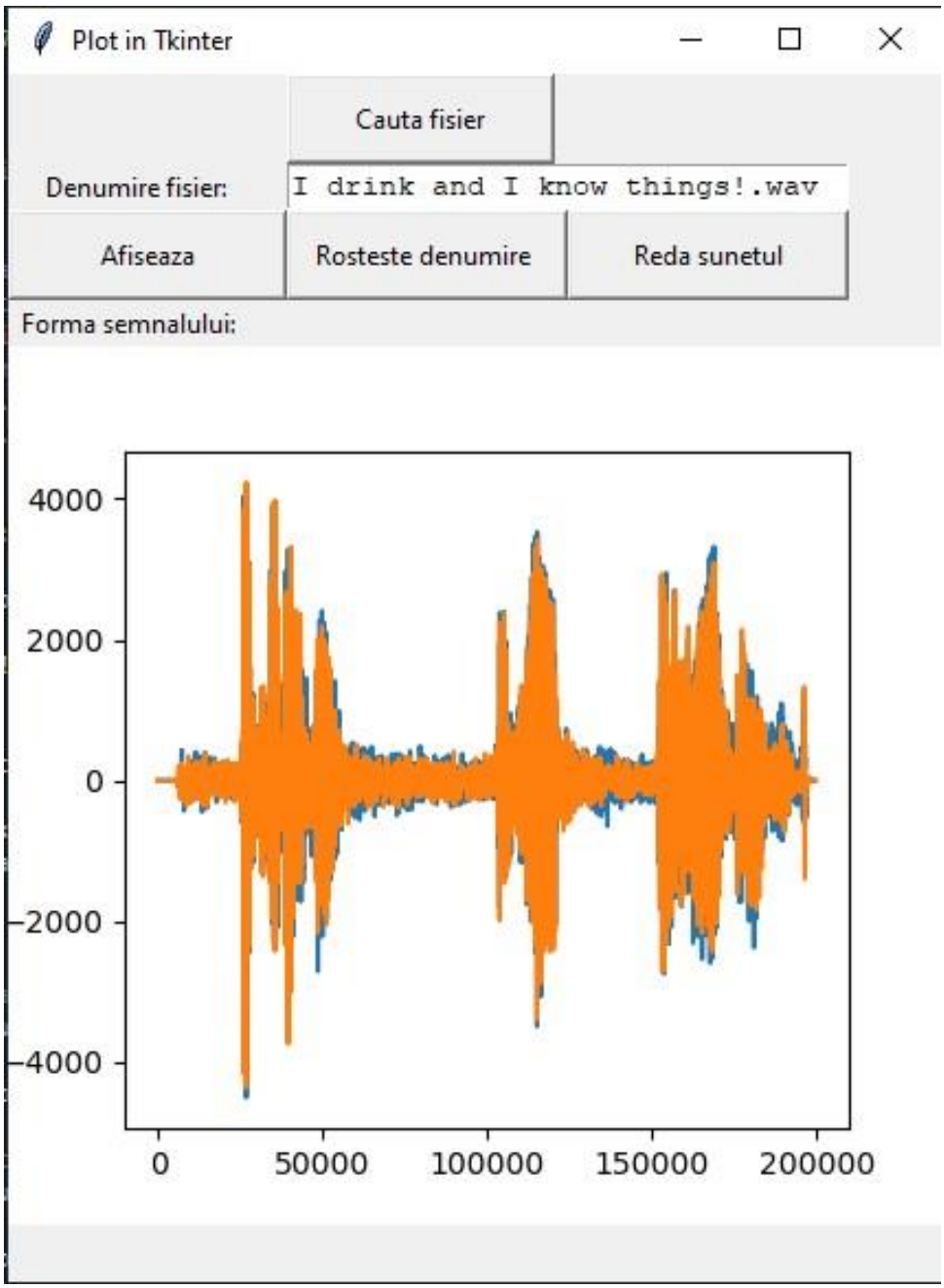


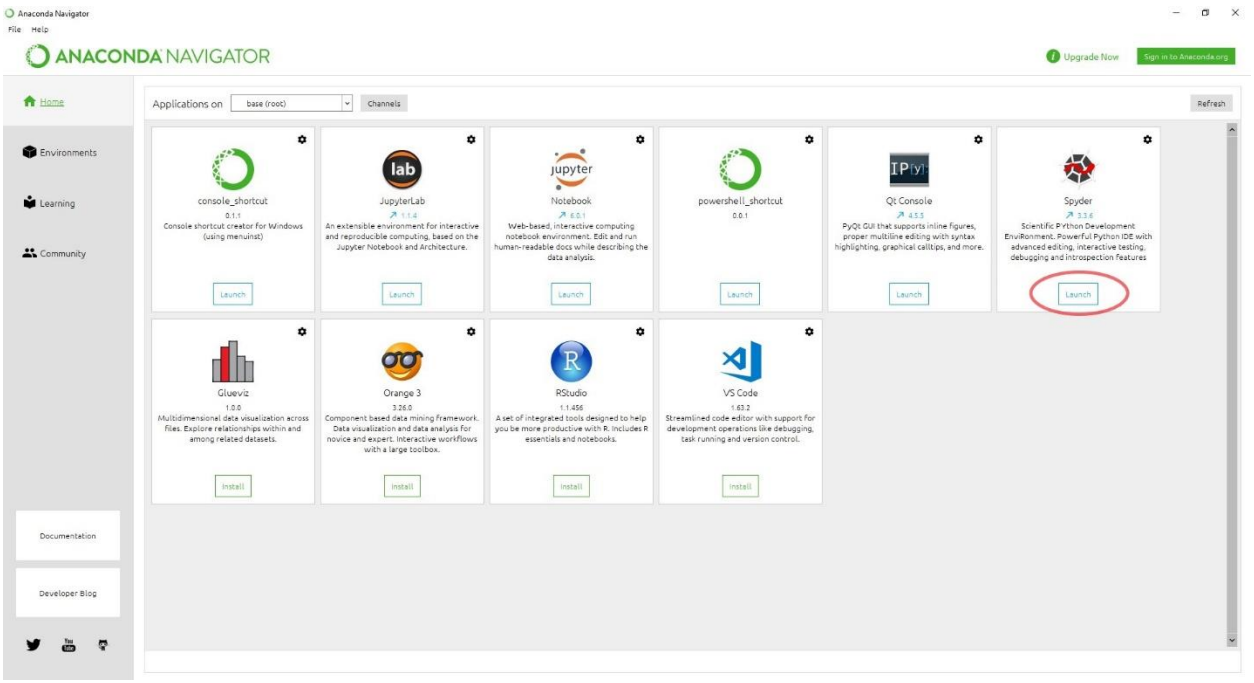
Fig. 9 – Interfața aplicației după selectarea unui fișier și după afișarea formei de undă.

Instalare, rulare și utilizare

Anaconda:

• Descărcați executabilul potrivit platformei dumneavostră (https://www.anaconda.com/products/individual) și parcurgeți instalarea cu setările implicite.

• După instalare deschideți Anaconda Navigator și veți vedea toate mediile de dezvoltare și bibliotecile instalate (în tab-ul Environments).



• Apăsați butonul ”Launch”/”Install” pentru deschiderea mediului de dezvoltare dorit (Spyder)

Instalarea bibliotecilor neceasare:

- Descridere „Anaconda Prompt” (din Start) – Run as Administrator
- conda install matplotlib
- conda install -c anaconda tk
- conda install -c anaconda scipy
- pip install gTTS
- pip install playsound

Se deschide fișierul proiectIOM_Echipa_16.py și se rulează programul făcând click pe iconița “Run file” sau apăsând tasta F5.

Bibliografie:

[1] https://www.python.org/doc/essays/blurb/

[2] https://www.python.org/about

[3] https://en.wikipedia.org/wiki/Spyder_(software)

[4] https://intellipaat.com/blog/advantages-and-disadvantages-of-python/

[5]https://web.archive.org/web/20141010221143/http://web.ics.purdue.edu:80/~smit1447/blog/?p=24

[6]https://web.archive.org/web/20131024165518/http://code.google.com/p/spyderlib/wiki/SpyderPlugins

[7]https://web.archive.org/web/20130820121204/http://fedora.cz/seznameni-s-python-ide-spyder/

[8]https://web.archive.org/web/20200419034550/https://www.anaconda.com/media-kit/

[9] https://docs.spyder-ide.org/overview.html

Note de laborator – Prof. dr. ing. Andreea Griparis