Acta et Commentationes, Sciences of Education, nr. 4(38), 2024 ISSN 1857-0623
p. 7-14 E-ISSN 2587-3636

CZU: 004.42:37.018.43:004.65 DOI: 10.36120/2587-3636.v38i4.7-14

ABORDĂRI METODICE ÎN AUTOMATIZAREA SARCINILOR CU PYTHON

Maria PAVEL, dr., conf. univ.
https://orcid.org/0000-0003-4803-6398
Dorin PAVEL, dr., conf. univ.
https://orcid.org/0000-0002-9600-1360
Catedra Informatică și Tehnologii Informaționale
Universitatea Pedagogică de Stat „Ion Creangă” din Chișinău

Rezumat. În lucrare se abordează metodic subiectul automatizării sarcinilor în era digitală, începând cu descrierea conceptului și finalizând cu exemple practice de automatizare a unor sarcini de rutină cu ajutorul limbajului de programare Python, în contextul studierii acestui limbaj de către studenții specialităților informatice. Exemplele contribuie la o mai bună înțelegere de către studenți a aplicabilității diverselor module Python ce pot fi implementate în automatizarea sarcinilor.
Cuvinte cheie: studierea limbajului de programare Python, automatizarea sarcinilor, modul Python, importarea modulelor.

METHODOLOGICAL APPROACHES IN AUTOMATING TASKS WITH PYTHON

Abstract. The paper methodically approaches the topic of automating tasks in the digital age, starting with the description of the concept and ending with practical examples of automating routine tasks with the help of the Python programming language, in the context of studying this language by computer science students. The examples contribute to a better understanding by students of the applicability of various Python modules that can be implemented in automating tasks.
Keywords: studying the Python programming language, task automation, Python module, importing modules.

Introducere

În era digitalizării intense a majorității activităților umane este inevitabilă realizarea unui șir de operații cu ajutorul tehnologiilor software și hardware, care au fost de fapt inventate de omenire anume pentru a facilita existența acesteia. Începând cu abacul cu bile și terminând cu cel mai performant robot, dea lungul istoriei omul a fost întotdeauna preocupat de a crea instrumente care l-ar ajuta să preia anumite sarcini de muncă. În epoca modernă această preocupare s-a orientat către inteligența artificială pentru diverse tehnologii robotizate, ce realizează operații de rutină sau mai complexe care permit evitarea greșelilor sau anumitor riscuri pentru oameni. Astfel, au fost create mașinării robotizate pentru industria alimentară care automatizează producerea și ambalarea, pentru industria auto care produc și asamblează diverse piese, dar și pentru activitatea intelectuală a omului în care se automatizează sarcini plictisitoare pentru oameni, ce se repetă și provoacă oboseală.

Automatizarea sarcinilor este o preocupare din ce în ce mai intensă a diferitor politici ce presupun pe de o parte facilitarea muncii oamenilor, eliberarea acestora de sarcini de rutină, evitarea erorilor, iar pe de altă parte presupun protejarea de riscuri în cazul unor sarcini foarte complexe. Aceste politici consideră, pe lângă avantajele evidente, și dezavantajele automatizării sarcinilor, cum ar fi: încrederea aproape totală în tehnologiile bazate pe Inteligența artificială, pierderea anumitor competențe sau calificări umane, realizarea fără implicare conștientă în anumite procese, nevoia de recalificare profesională etc. Astfel, Uniunea Europeană recomandă ca atunci „când se introduc sisteme robotizate bazate pe IA pentru automatizarea sarcinilor, ar trebui să se adopte o abordare de tipul *human-in-command*. Astfel, lucrătorii sunt capabili să utilizeze automatizarea în avantajul lor, păstrând în același timp controlul asupra procesului de lucru" [1]. Totodată, în cadrul Uniunii Europene se desfășoară campanii multi-lingvistice de informare cu diverse resurse disponibile online, precum campania „Condiții de muncă sigure și sănătoase în era digitală", printre prioritățile căreia este și automatizarea sarcinilor pentru muncitorii din cadrul uniunii [2]. În suportul aceste companii vin și un șir de studii de caz publicate din agricultură, medicină, industria auto etc., precum: roboți populari în peste 45 de țări, care realizează sarcini automatizate de curățare a grajdurilor din ferme, aplicații bazate pe inteligența artificială ce se utilizează în medicina de diagnosticare a colonoscopiei, producerea robotizată a recipientelor din plastic prin injectare și extrudat etc. [3].

Automatizarea sarcinilor de birou reprezintă principala funcție a unui șir de aplicații dedicate, foarte populare în gestionarea afacerilor. De la aplicații simple, ce realizează una sau câteva sarcini, precum managementul poștei electronice, până la platforme complexe ce întrunesc sute de aplicații pentru automatizarea eficientă, rapidă, cu un risc redus de erori, instrumentele de automatizare a fluxului de operații sunt implementate cu succes într-o afacere modernă. O simplă căutare pe internet, furnizează zeci de soluții complexe, ce solicită efort financiar, precum: Albato, Pabbly, Workato, Power Automate (de la Microsoft), Kissflow și altele; sau soluții simple, gratuite, precum: TeamWork, Airtable, Todoist, SmartSheet, Trello etc. Aceste instrumente nu necesită cunoștințe de programare pentru utilizatori, însă pentru specialiștii în tehnologii informaționale este important să cunoască principiile și structura de elaborare, limbajele de programare precum Python, bibliotecile necesare etc.

Metode, mijloace

Studenții specialităților informatice studiază mai multe limbaje de programare de diferite niveluri și paradigme, care le-ar permite să realizeze majoritatea activităților din viața profesională, precum: C, C++, C#, Java, Java Script etc. Însă, în ultimul timp, tot mai popular printre programatori este limbajul Python, datorită versatilității sale în dezvoltarea aplicațiilor, inclusiv de inteligență artificială (IA), numeroaselor biblioteci și module pe

care le conține. Pe lângă modulele și bibliotecile de bază, Python este implementat cu succes în dezvoltarea aplicațiilor esențiale în diverse domenii ale IA, așa ca:

- ✓ *Machine Learning*, în care se apelează la modulele: *scikit-learn*, pentru învățare automată; *XGBoost* și *LightGBM*, pentru modele de gradient boosting; *TensorFlow* și *PyTorch*, utilizate în special pentru modele de învățare profundă (deep learning); *Keras*, interfață de nivel înalt pentru *TensorFlow*, folosită pentru dezvoltarea rapidă a modelelor de deep learning.
- ✓ *Prelucrarea limbajului natural* (NLP), domeniu în care se implementează modulele: *NLTK*, procesarea limbajului natural, cu instrumente pentru tokenizare, stemming, și analiza sintactică; *spaCy*, pentru aplicații comerciale; transformers, dezvoltată de Hugging Face și utilizată pentru lucrul cu modele avansate, precum BERT, GPT și alte arhitecturi Transformer.
- ✓ *Computer Vision*, în care se apelează la modulele: *OpenCV*, robustă pentru procesarea imaginilor și a videoclipurilor; *Pillow*, pentru manipularea imaginilor; *PyTorch* și *TensorFlow*, utilizate și pentru dezvoltarea modelelor de recunoaștere a obiectelor, clasificarea imaginilor etc. [4].
- ✓ *Manipularea și vizualizarea datelor*, domeniu în care se implementează modulele: *NumPy*, pentru calcul numeric, esențială pentru manipularea matricilor și a altor structuri de date numerice; *Pandas*, pentru manipularea datelor în format tabelar (DataFrames); *Matplotlib* și *Seaborn*, pentru vizualizarea datelor; *Plotly*, pentru vizualizări interactive.
- ✓ *Inteligență artificială simbolică*, în care se lucrează cu modulul *PyBrain*, pentru rețele neuronale și alte metode de învățare automată.
- ✓ *Optimizare*, în care se apelează la modulele: *DEAP*, pentru algoritmi evolutivi și *PyGAD* pentru algoritmi genetici și optimizare.
- ✓ Roboți conversaționali și asistenți virtuali, domeniu în care se implementează modulele: *Rasa NLU* (Natural Language Understanding), platformă pentru dezvoltarea de chatbot-uri avansate; *ChatterBot*, pentru crearea chatbot-urilor simple; *Dialogflow* API (prin gRPC), pentru IA conversațională.
- ✓ *Reinforcement Learning* (RL), în care se lucrează cu modulul *OpenAI Gym* pentru dezvoltarea și testarea algoritmilor de învățare RL și modulul Stable-Baselines3, ce reprezintă o implementare standard a algoritmilor RL [5].

Domeniile enumerate constituie subiecte de conținut pentru cursurile universitare studiate la programele de licență în informatică, precum: Robotică, Inteligență Artificială, Sisteme expert, Programare Logică, Algoritmi genetici, Rețele Neuronale etc. Iar pregătirea inițială pentru familiarizarea studenților cu paradigma, sintaxa, structurile și modulele Python, se face în cadrul unui curs, predat de obicei la anul I, licență, cu diferite titulaturi: *Programare în Python* (USM), sau *Inițiere în limbajul de programare Python*

(UPSC). Acest curs este important pentru a facilita abordarea subiectelor complexe de Inteligență Artificială, evitând astfel alocarea timpului pentru cunoștințe elementare de sintaxă din contul cursurilor dedicate, enumerate anterior. Totodată, în cadrul cursului *Inițiere în limbajul de programare Python* poate fi explicitată utilitatea modulelor Python, prin valorificarea unor exemple de automatizare a sarcinilor.

Un exemplu de sarcină ce poate fi automatizată cu un cod Python, este expedierea de mesaje cu atașamente personalizate către un număr mare de destinatari. Deși există un șir de instrumente pe piața digitală care realizează această sarcină, precum: Mailchimp, YAMM de la Google, MailMerge365 de la Microsoft ș.a., fiecare cu avantajele și dezavantajele sale (fie sunt contra cost, fie gratuit permit doar expedierea mesajelor fără atașamente, fie limitează doar la un anumit număr de mesaje etc.), crearea unui instrument propriu, necostisitor (deoarece Python și toate modulele lui sunt gratuite), va forma la studenți competențe de programare avansate și va contribui la o mai bună înțelegere a aplicabilității modulelor Python.

Rezultate

Unul din modulele principale ce permit crearea de obiecte SMTP (Simple Mail Transfer Protocol) pentru conexiunea cu orice client Internet în așteptare, este *smtplib*, care se va importa direct și din care se va utiliza funcția SMTP [6]:

```
server = smtplib.SMTP(smtp_server, smtp_port)
```

Ulterior pentru obiectul *server* se vor utiliza doar metodele *sendmail()* și *SMTP.quit*(), pentru a expedia un mesaj de la expeditor către destinatar cu un conținut textual opțional și, respectiv, pentru a termina sesiunea de comunicare și a închide conexiunea:

```
server.sendmail(sender_email, destinatar["email"], mesaj.as_string())
server.quit()
```

O altă librărie pentru expedierea mesajelor electronice este *email* [7], care spre deosebire de *smtplib*, nu expediază efectiv mesajele, ci manipulează, analizează și construiește mesaje complexe cum ar fi cele cu atașamente, prin intermediul obiectelor MIME (Multipurpose Internet Mail Extensions). Modulul dat include și un submodul separat *email.mime* ce conține obiecte pentru gestionarea părților mesajului: MIMEText, *MIMEImage*, *MIMEBase*, *MIMEMultipart*. Un alt submodul important atunci când se expediază atașamente la mesaj este *encoders*. Prin urmare, pentru sarcina propusă, codul de program va importa modulele și submodulele menționate:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
```

Expedierea efectivă a emailurilor se realizează cu ajutorul codului:

```
for destinatar in destinatari:
    mesaj = MIMEMultipart()
    mesaj["From"] = sender_email
    mesaj["To"] = destinatar["email"]
    mesaj["Subject"] = "Certificates, STEAM Conference, November 01 - 02, 2024"
    mesaj.attach(MIMEText(f'Dear participant {destinatar["nume"]},'+r'''
thank you for the communication in the plenary of the
International Scientific Conference
„Inter /transdisciplinary approaches in the teaching of the real sciences, (STEAM concept)”,
we wish you success and hope for further collaboration.
We are sending attached certificates of participation.
The conference proceedings can be accessed at:
https://drive.google.com/file/d/16fsxM0yo6MKSzbkgfOJDyenGuMeLF_DU/view?usp=sharing
The conference proceedings will be accessible on the UPSC website and ibn.idsi.md.
Have a nice day.''', "plain"))
```

După cum se observă din codurile de mai sus, mesajul este construit cu ajutorul listei de dicționare *destinatari* (elementele acesteia se accesează ciclic), valorile cărora sunt numele destinatarului *destinatar["nume"]* și adresa de email a acestuia *destinatar["email"]*:

```
destinatari = [
    {
        "nume": "LT",
        "email": "lt@yahoo.com",
        "certificat": "p24.jpg"
    },
    {
        "nume": "VGH",
        "email": "vgh@gmail.com",
        "certificat": "p25.jpg"
    },
    {
        "nume": "GGH",
        "email": "g_gh@yahoo.com",
        "certificat": "p26.jpg"
    },
    {
        "nume": "LS",
        "email": "ls@gmail.com",
        "certificat": "p27.jpg"
    },
    ...
    ]
```

Mesajul emailului inclus într-un șir de caractere cu format, ca argument al metodei *MIMEText*, din codul de mai sus, va fi personalizat anume prin apelul formatat al variabilei *destinatar["nume"]*, ce reprezintă valoarea cu cheia *nume* din dicționarele componente ale listei de destinatari. Fiecare dicționar mai conține și o valoare cu cheia *certificat*, ce conține numele fișierului de atașat. Pentru a atașa acest fișier, se va utiliza codul de program:

```
    certificat = destinatar["certificat"]
    with open(certificat, "rb") as f:
        part = MIMEBase("application", "octet-stream")
        part.set_payload(f.read())
        encoders.encode_base64(part)
        part.add_header(
            "Content-Disposition", f"attachment; filename={certificat}"
        )
        mesaj.attach(part)
```

Deoarece fișierul atașat în exemplul prezentat este de tip *.jpg*, atunci se utilizează metoda *encoders.encode_base64* pentru atașamente binare (fapt pentru care fișierul se deschide cu modul *"rb"*).

Pentru ca acest cod de program să poată fi implementat este important de pregătit datele necesare pentru expedierea emailurilor, îndeosebi în cazul unui număr foarte mare de destinatari (pentru exemplul prezentat s-au expediat peste 850 de mesaje cu fișiere de tip *.jpg* ca atașamente, certificate nominale de participare la o conferință), pentru a completa valorile dicționarelor din listă. Dacă participanții la conferință au completat un formular de participare, atunci lista acestora se va salva într-un fișier Excel, din care se vor extrage doar coloanele necesare: numele participanților și adresa de email a acestora. În cazul când numele și prenumele sunt câmpuri completate de participanți separat, atunci acestea pot fi unite într-un singur câmp, cu ajutorul funcției CONCAT(). De asemenea, se completează și coloana ce conține numele fișierului nominal ce trebuie atașat. Deoarece fișierele au fost generate tot prin automatizare, preluând numele destinatarilor din aceeași listă, atunci numele acestora sunt asemănătoare, diferențiindu-se doar prin numărul de ordine atașat. Acest lucru poate fi utilizat în avantajul automatizării, generând o listă de nume în Excel, prin glisare, apoi prin concatenare, poate fi anexată extensia *.jpg*. Aceste trei coloane, ce conțin informația necesară pentru construirea dicționarelor din lista de destinatari, pot fi copiate într-un fișier textual Word, după care tabelul cu date poate fi convertit în text cu ajutorul instrumentului *Convert to Text*  . Separatorul valorilor din fiecare rând a celor trei coloane, se va utiliza ulterior ca separator pentru partiționarea textului citit din acest fișier.

Un astfel de fișier pregătit, va fi utilizat pentru citire cu ajutorul codului Python, ce va automatiza construirea listei de destinatari, pentru a facilita scrierea părții corespunzătoare de program:

```
with open('nume.txt', 'r') as file:
    lines = file.readlines()
output = []
for line in lines:
    parts = line.strip().split(',')
    if line.strip() == "":
        continue
    name = parts[0].strip()
    email = parts[1].strip()
    certificat = parts[2].strip()
    email = email.replace(" ", "")
    if email:
        output.append({"nume": name, "email": email, "certificat": certificat})
for entry in output:
    print(entry)
import json
with open('output.json', 'w') as json_file:
    json.dump(output, json_file, indent=4)
```

Codul dat este ușor de construit de către studenți, deoarece la tema *Fișiere Python*, se studiază modurile de deschidere (aici fișierul *"nume.txt"* se deschide cu modul de citire *"r"*), metodele de citire din fișier (aici se citește întregul fișier ca o listă de linii cu ajutorul metodei *readlines*), metodele de scriere în fișier, metodele de manipulare a șirurilor de caractere (aici se utilizează metoda *strip* pentru a elimina spațiile goale din fața și după șirul citit, iar metoda *split* separă șirul în fragmente delimitate de virgulă: *strip().split(',')*). Lista obținută în variabila *output*, se va scrie pentru comoditate într-un fișier json (JavaScript Object Notation) cu ajutorul metodei *dump*, care convertește (serializează) dicționarul *output* într-un format JSON și îl scrie în fișier. Pentru a accesa această metodă se va importa modulul JSON [8].

Conținutul fișierului obținut sub formă de listă de dicționare, se va copia și se va insera direct în codul programului de expediere a mesajelor, în locul listei model exemplificate mai sus.

Datele adresei de email de pe care se vor expedia mesajele personalizate cu atașamentele nominale se vor indica în fragmentul de cod de configurare a SMTP, înainte de a descrie lista destinatarilor:

```
smtp_server = "smtp.gmail.com"
smtp_port = 587
sender_email = "steamconferencefmti@gmail.com"
sender_password = "zvmo jqml aipm plod"
```

După cum se poate vedea, *sender_password* nu poate fi parola reală a contului de email a expeditorului, deoarece în scopul securizării, *gmail* nu va permite acest lucru. Pentru aceasta se vor urma pașii sugerați de *Google*, se va obține un cod de acces și o parolă specială valabilă doar pentru program, care va fi atribuită variabilei corespunzătoare, conform codului din imaginea de mai sus.

La ultima etapă, se va testa programul pentru una sau câteva adrese de email personale, iar după asigurarea că totul se realizează exact cum ne-am propus, se va executa programul pentru întreaga listă de destinatari. Procesul de expediere poate fi urmărit în contul de email al expeditorului, la secțiunea mesajelor trimise (numărul de mesaje trimise indicat de *gmail* va crește interactiv):

Finalizarea execuției programului va fi semnalată de apariția promptului de inserare >>> în linia de comenzi Shell Python. Totodată, trebuie menționat că fișierul ce conține codul programului trebuie amplasat în același dosar cu fișierele ce urmează a fi atașate, astfel evitând eroare de neidentificare a acestora.

Astfel, automatizarea sarcinii de expediere a emailurilor personalizate cu atașamente nominale, este un exemplu elocvent pentru studenți, ce va valorifica cunoștințele acestora de programare în Python pentru un exemplu practic, care economisește foarte mult timp

expeditorului și contribuie la evitarea erorilor de pierdere a unei adrese, scrierea greșită a vreunui nume, atașarea eronată a altul fișier, decât cel destinat etc.

Concluzii

Implicarea studenților informaticieni în elaborarea și testarea de programe Python ce valorifică diverse module din bibliotecile limbajului și care realizează o activitate practică de automatizare a sarcinilor, cum ar fi cea de expediere a mesajelor electronice, va contribui la formarea și dezvoltarea unor competențe de programare necesare studierii altor cursuri axate pe subiecte complexe de Inteligență Artificială, bazate pe codificarea în Python.

Articol realizat în cadrul proiectului de cercetări științifice „Metodologia implementării TIC în procesul de studiere a științelor reale din perspectiva conceptului STEAM și Inteligenței Artificiale”, codul 040101, din cadrul Programului instituțional de cercetare (2024-2027), aprobat prin Ordin MEC nr. 102 din 01.02.2024

Bibliografie

1. Agenția Europeană pentru Securitate și Sănătate în Muncă. Automatizarea sarcinilor. Disponibil online: https://healthy-workplaces.osha.europa.eu/ro/about-topic/priority-area/automation-tasks
2. Agenția Europeană pentru Securitate și Sănătate în Muncă. Automatizarea sarcinilor în era digitală: explorarea oportunităților și provocărilor. Disponibil online: https://osha.europa.eu/ro/highlights/automation-tasks-digital-age-exploring-opportunities-and-challenges
3. Agenția Europeană pentru Securitate și Sănătate în Muncă. Case studies. Disponibil online: https://healthy-workplaces.osha.europa.eu/en/tools-and-publications/case-studies?f%5B0%5D=publications_priority_area%3A1710
4. PAVEL, Maria, PAVEL, Dorin. STEAM și Computer Vision. În: *Materialele Conferinței științifice internaționale „Abordări inter/transdisciplinare în predarea științelor reale (concept STEAM)*”, ediția a IV-a. 01-02 noiembrie 2024. Chișinău: IDIS „Viitorul”, 2024, pp. 320-327. ISBN 978-5-86654-132-4 (PDF). 552 p. Online: https://drive.google.com/file/d/1eTq-XoN3qO70gjaSsvK2fCFMAx0DHQQG/view
5. Python Tutorial. Disponibil online: https://www.w3schools.com/python/default.asp
6. The Python Standard Library. smtplib — SMTP protocol client. https://docs.python.org/3/library/smtplib.html#
7. The Python Standard Library. email — An email and MIME handling package. https://docs.python.org/3/library/email.html
8. The Python Standard Library. json — JSON encoder and decoder. https://docs.python.org/3/library/json.html#module-json