Robot-lektor, avagy címsorokhoz kötődő automatizmus megalapozása

(Robot proofreader, or checking lists free of omissions and overlaps)

Kosdi Gábor, Pitlik László, (KJE)

# Bevezetés, apropó

A robot-lektor fejlesztése több, mint egy évtizedes múltra nyúlik vissza: <https://miau.my-x.hu/temp/2025tavasz/rules/>, ill. <https://miau.my-x.hu/mediawiki/index.php/Vita:CT_00>, valamint <https://miau.my-x.hu/miau2009/index.php3?x=e0&string=lektor>.

Az aktuális kihívás lényege, hogy a szakdolgozatok érdekében a plágium-ellenőrzéshez hasonlóan saját hatáskörben tudjon a Hallgató tetszőleges gyakorisággal ellenőrzéseket lefuttatni és ne a triviális szabályok be nem tartása körül forogjon a szakdolgozati egyeztetések zöme, hanem végre több idő jusson a tartalomra.

Jelen dokumentumban a MS-WORD címsorai kapcsán indul meg az alapozás arra vonatkozóan, miként lehet ezekkel automatizálható műveleteket végezni:

# Esettanulmány

**Promt: Szerinted lehetseges word-ben cimsorokat megkulonboztetni program alapjhan?**

**A ChatGPT ezt mondta:**

Igen, teljesen lehetséges **Word dokumentumban** különböző **címek** (pl. címsorok) megkülönböztetése **program alapján**! 🖥️

A programozás során a leggyakoribb módszer, hogy a dokumentumban a címsorok formázását **stílusok** segítségével kezeljük. A Microsoft Word is használ **beépített stílusokat**, mint például "Címsor 1", "Címsor 2", stb., és ezek automatikusan más formázást biztosítanak (pl. nagyobb betűméret, félkövér, stb.).

**Hogyan csinálhatod ezt Python segítségével?**

A **python-docx** nevű könyvtár segíthet ebben! 😎

* A python-docx segítségével **Word dokumentumokat hozhatsz létre, módosíthatsz és formázhatsz**, beleértve a címsorokat is.
* Az alapvető funkció, hogy alkalmazhatsz **stílusokat** a címekhez (például "Címsor 1" stílust), és megkülönböztetheted a címsorokat a szövegtől.

**1. Python telepítése:**

Először is telepítened kell a python-docx könyvtárat:

bash

MásolásSzerkesztés

pip install python-docx

**2. Alap kód Word címsorokkal:**

A következő Python kód segít neked egy Word dokumentumban a **címsorok** megkülönböztetésében:

python

MásolásSzerkesztés

from docx import Document

# Új dokumentum létrehozása

doc = Document()

# Címsor 1 - főcím

doc.add\_paragraph('Főcím', style='Heading 1')

# Címsor 2 - alkategória

doc.add\_paragraph('Alkategória', style='Heading 2')

# Szöveg, ami nem címsor

doc.add\_paragraph('Ez itt egy szöveges bekezdés, nem címsor.')

# További címsorok

doc.add\_paragraph('További részletes információk', style='Heading 3')

# Mentés

doc.save('cim\_sorok\_pelda.docx')

**Mi történik a kódban?**

1. Létrehozunk egy új Word dokumentumot.
2. **Címsorokat** adunk hozzá különböző stílusokkal ('Heading 1', 'Heading 2', stb.).
3. Hozzáadunk egy sima **szöveges bekezdést** is, ami nem címsor.
4. A végén elmentjük a dokumentumot .docx formátumban.

**3. Mit csinál a program?**

* **Címsorok megkülönböztetése**: A Heading 1, Heading 2 stílusok alkalmazásával a címsorokat kiemeljük. A Word automatikusan nagyobb betűmérettel, félkövérrel vagy más formázással jeleníti meg őket, attól függően, hogyan vannak beállítva a stílusok.
* A címek hierarchikus rendben jelennek meg (pl. "Heading 1" → főcím, "Heading 2" → alfejezet, "Heading 3" → alcím, stb.).

**4. Extra ötlet – További formázás:**

* **Szöveg színe**, **betűtípus**, **súly** (félkövér, dőlt) is programmal módosítható, ha szükséges.

Példa arra, hogyan változtass szöveget **félkövérre** vagy **pirosra**:

python

MásolásSzerkesztés

p = doc.add\_paragraph('Ez egy fontos szöveg.')

p.bold = True # Félkövérre állítja

p.font.color.rgb = (255, 0, 0) # Vörös szín

**Miért hasznos?**

Ez a megoldás különösen akkor hasznos, ha szeretnéd **automatikusan generálni a címsorokat**, például egy jelentés, vagy dokumentáció alapján. A program automatikusan megadhatja a dokumentum szerkezetét, különböző szintű címsorokat alkalmazva, hogy jól nézzen ki és könnyen navigálható legyen.

**Ha érdekel még, hogyan formázhatsz még tovább vagy egyéb részletek kellene, kérdezz bátran!**

**Promt: A mar kesz word dokumentumban meg tudjuk kulonboztetni a cimsorokat es hasonlokat egymastol? Mondjuk megszamolni oket?**

**A ChatGPT ezt mondta:**

Igen, a már meglévő **Word dokumentumban** is meg tudjuk különböztetni a **címsorokat** és egyéb elemeket **program segítségével**! 📝

A **python-docx** könyvtár lehetővé teszi, hogy **megnézzük a dokumentumban lévő stílusokat**, és elvégezzük a kívánt műveleteket, például **megszámolhatjuk a különböző címsorokat**.

**Hogyan csinálhatjuk ezt?**

Tegyük fel, hogy van egy **már meglévő Word dokumentum** (pl. pelda.docx), és azt szeretnénk, hogy a program **megkeresse benne a címsorokat** és **megszámolja** őket.

**1. Könyvtár telepítése:**

Először is győződj meg róla, hogy a python-docx könyvtár telepítve van:

bash

MásolásSzerkesztés

pip install python-docx

**2. Kód a címsorok számolásához egy meglévő dokumentumban:**

Az alábbi kódban **megszámoljuk a különböző szintű címsorokat** (pl. "Heading 1", "Heading 2", stb.) a már meglévő Word dokumentumban:

python

MásolásSzerkesztés

from docx import Document

# Dokumentum betöltése

doc = Document('pelda.docx')

# Címsorok számlálása

heading\_1\_count = 0

heading\_2\_count = 0

heading\_3\_count = 0

# Dokumentumban végigmegyünk a bekezdéseken

for para in doc.paragraphs:

if para.style.name == 'Heading 1':

heading\_1\_count += 1

elif para.style.name == 'Heading 2':

heading\_2\_count += 1

elif para.style.name == 'Heading 3':

heading\_3\_count += 1

# Eredmények kiírása

print(f"Címsor 1: {heading\_1\_count}")

print(f"Címsor 2: {heading\_2\_count}")

print(f"Címsor 3: {heading\_3\_count}")

**3. Mi történik a programban?**

* **Dokumentum betöltése**: A Document('pelda.docx') függvénnyel betöltjük a már meglévő Word dokumentumot.
* **Stílusok keresése**: A para.style.name segítségével ellenőrizzük, hogy az adott bekezdés címsor-e.
  + Ha Heading 1, akkor **főcímet** találunk, és növeljük a heading\_1\_count számlálót.
  + Hasonlóan növeljük a másod- és harmadszintű címsorok számlálóját is.
* **Eredmények kiírása**: A végén kiírjuk a különböző típusú címsorok számát.

**4. Példa kimenet:**

Ha a dokumentumban három "Heading 1", két "Heading 2" és egy "Heading 3" van, a kimenet így nézne ki:

yaml

MásolásSzerkesztés

Címsor 1: 3

Címsor 2: 2

Címsor 3: 1

**5. Mit csinálhatunk még?**

* **További műveletek**: A dokumentumban lévő **egyéb formázott szövegeket** is kiemelhetjük, például **félkövér szöveget** vagy **dőlt betűt**.
* **Szövegek kinyerése**: A programmal kinyerhetjük a címsorok szövegeit, és akár **indexelhetjük** őket, hogy gyorsan navigáljunk a dokumentumban.

**6. További hasznos műveletek:**

Ha szeretnél más típusú elemeket is keresni a dokumentumban, például:

* **Bármilyen típusú bekezdést**, ami nem címsor
* **Táblázatokat** a dokumentumban
* **Specifikus formázásokat** (pl. félkövér szöveg)

Ezeket mind meg tudjuk keresni és külön kezelni!

**Összegzés:**

A python-docx könyvtár segítségével könnyen **megszámolhatod a különböző típusú címsorokat**, és egyéb szövegeket kereshetsz egy meglévő Word dokumentumban, ami automatizálhatja a dokumentumok feldolgozását és elemzését.

**Szeretnél még valami mást is csinálni a dokumentummal?** Mondj bátran egy példát, és segítek tovább!

**Promt: Hogyan illesztem bele a programba a vizsgalni kivant docx-et?**

**A ChatGPT ezt mondta:**

A meglévő **Word dokumentum** (pl. pelda.docx) betöltéséhez a python-docx könyvtár segítségével nem kell túl bonyolult dolgokat tenned. Az alapvető művelet az, hogy a dokumentumot **betöltöd** a programba, majd folytathatod a szükséges műveletekkel.

**1. A dokumentum beillesztése a programba:**

A kódodban egyszerűen módosítanod kell a fájl nevét a Document konstruktorban, és az elérési utat is be kell állítani, ha a dokumentum nem ugyanabban a mappában van, mint a Python script.

**2. Kód példa a fájl betöltésére:**

Itt van a módosított kód, amelyben a kívánt **Word fájlt** illesztheted bele:

python

MásolásSzerkesztés

from docx import Document

# Itt kell megadnod a fájl nevét vagy elérési útját

doc\_path = 'pelda.docx' # Ha ugyanabban a mappában van a fájl

# doc\_path = 'C:/Users/username/Documents/pelda.docx' # Ha más mappában van

# Dokumentum betöltése

doc = Document(doc\_path)

# Címsorok számlálása

heading\_1\_count = 0

heading\_2\_count = 0

heading\_3\_count = 0

# Dokumentumban végigmegyünk a bekezdéseken

for para in doc.paragraphs:

if para.style.name == 'Heading 1':

heading\_1\_count += 1

elif para.style.name == 'Heading 2':

heading\_2\_count += 1

elif para.style.name == 'Heading 3':

heading\_3\_count += 1

# Eredmények kiírása

print(f"Címsor 1: {heading\_1\_count}")

print(f"Címsor 2: {heading\_2\_count}")

print(f"Címsor 3: {heading\_3\_count}")

**3. Mi történik ebben a kódban?**

* A doc\_path változóban megadod a **Word dokumentum fájl nevét és elérési útját**.
  + Ha ugyanabban a mappában van a script és a .docx fájl, akkor elég csak a fájl nevét írnod, mint pelda.docx.
  + Ha más mappában van a fájl, add meg az **abszolút elérési utat** (pl. C:/Users/username/Documents/pelda.docx).
* A program **betölti** a dokumentumot a Document(doc\_path) parancs segítségével.
* A program ugyanúgy végigmegy a bekezdéseken, mint korábban, és megszámolja a **címsorokat** (Heading 1, Heading 2, Heading 3), majd kiírja az eredményeket.

**4. Hogyan válassz fájlt a programban?**

A fájl kiválasztásához három megoldás van:

**1. Ha a fájl ugyanabban a mappában van a Python script-tel:**

* Ne változtasd meg az elérési utat, csak add meg a fájl nevét:

python

MásolásSzerkesztés

doc\_path = 'pelda.docx'

**2. Ha a fájl egy másik mappában van:**

* Add meg a teljes elérési utat:

python

MásolásSzerkesztés

doc\_path = 'C:/Users/username/Documents/pelda.docx' # A fájl teljes elérési útja

**3. Ha dinamikusan szeretnéd kiválasztani a fájlt:**

* Ha interaktív módon szeretnéd a fájlt kiválasztani a felhasználó számára, használhatsz egy **file picker** (fájlválasztó) ablakot a tkinter vagy más könyvtárak segítségével. Például:

python

MásolásSzerkesztés

from tkinter import Tk

from tkinter.filedialog import askopenfilename

# Fájl választása a felhasználótól

Tk().withdraw() # Ne jelenjen meg az üres Tk ablak

file\_path = askopenfilename(title="Válaszd ki a dokumentumot", filetypes=[("Word fájlok", "\*.docx")])

# Dokumentum betöltése

doc = Document(file\_path)

Ez lehetővé teszi a felhasználónak, hogy egy fájl választó ablakban válassza ki a kívánt .docx fájlt.

**5. Hibaelhárítás:**

* Ha a fájl nem található, a program hibát dob, tehát győződj meg róla, hogy a fájl valóban létezik az adott helyen.
* Ha a fájl nem Word formátumban van (pl. .doc vagy más), a program nem fogja tudni betölteni.

**Összegzés:**

A dokumentum fájlt könnyen beillesztheted a programba, ha megadod annak elérési útját. A kód automatikusan végigmegy a dokumentumban található bekezdéseken, és számon tartja a különböző **címsorokat**. Ha szükséges, a fájlt dinamikusan is választhatod a felhasználó számára egy fájlválasztó ablak segítségével.

**Promt: Egy tallozasos megoldassal csinaljuk meg kerlek, talan az a legegyszerubb.**

**A ChatGPT ezt mondta:**

Rendben, készíthetünk egy **tallózós** (file picker) megoldást, amely lehetővé teszi, hogy a felhasználó egy fájlt válasszon egy grafikus felületen. Ehhez használhatjuk a tkinter könyvtárat, amely Python-ban alapértelmezett eszköz a grafikus felületek kezelésére. Ezzel a módszerrel a felhasználó könnyedén választhat egy .docx fájlt a fájlrendszeréből.

**1. A tkinter könyvtár telepítése**

A tkinter könyvtár már alapértelmezett része a Python telepítésének, tehát nem szükséges külön telepíteni, ha a Python verziód már tartalmazza.

**2. Kód a tallózóablakos fájl kiválasztáshoz**

Az alábbi kód segítségével megnyithatsz egy fájl kiválasztó ablakot, ahol a felhasználó választhat egy .docx fájlt, amit a program betölt és feldolgoz:

python

MásolásSzerkesztés

import tkinter as tk

from tkinter.filedialog import askopenfilename

from docx import Document

# Fájl kiválasztása

def file\_picker():

# Fájl kiválasztó ablak megnyitása

filepath = askopenfilename(

title="Válaszd ki a dokumentumot",

filetypes=[("Word dokumentumok", "\*.docx")]

)

# Ha a felhasználó nem választott fájlt, akkor a filepath üres lesz

if not filepath:

print("Nem választottál fájlt.")

return

return filepath

# Dokumentum betöltése és címsorok számolása

def count\_headings(doc\_path):

# Dokumentum betöltése

doc = Document(doc\_path)

# Címsorok számlálása

heading\_1\_count = 0

heading\_2\_count = 0

heading\_3\_count = 0

# Dokumentumban végigmegyünk a bekezdéseken

for para in doc.paragraphs:

if para.style.name == 'Heading 1':

heading\_1\_count += 1

elif para.style.name == 'Heading 2':

heading\_2\_count += 1

elif para.style.name == 'Heading 3':

heading\_3\_count += 1

# Eredmények kiírása

print(f"Címsor 1: {heading\_1\_count}")

print(f"Címsor 2: {heading\_2\_count}")

print(f"Címsor 3: {heading\_3\_count}")

# Fő program

if \_\_name\_\_ == "\_\_main\_\_":

# Fájl kiválasztása

file\_path = file\_picker()

if file\_path:

# Címsorok számolása

count\_headings(file\_path)

**3. Mi történik a kódban?**

1. **Fájl kiválasztása (file\_picker)**:
   * A file\_picker() függvény segítségével egy **fájl kiválasztó ablakot** nyitunk meg, amely lehetővé teszi a felhasználó számára, hogy válasszon egy .docx fájlt a fájlrendszeréből.
   * Az ablakban a fájlokat **Word dokumentumok** formátumban korlátozzuk (\*.docx), így csak a Word fájlokat tudja kiválasztani.
   * Ha a felhasználó nem választ fájlt, a kód figyelmezteti.
2. **Címsorok számolása (count\_headings)**:
   * Miután a fájl elérési útja megtörtént, a count\_headings() függvény betölti a dokumentumot, és végigmegy a bekezdéseken.
   * A program megszámolja a **különböző címsor szinteket** (Heading 1, Heading 2, Heading 3), és kiírja az eredményeket.

**4. Fájl kiválasztás:**

* Ha például van egy pelda.docx fájl a számítógépeden, a fájlválasztó ablak lehetővé teszi, hogy **böngészd és válaszd ki** a fájlt a rendszereden.

**5. Hogyan futtathatod:**

1. **Mentsd el a kódot egy .py fájlba** (pl. count\_headings.py).
2. Futtasd a Python scriptet:
   * Nyisd meg a **terminált** vagy **command prompt**-ot.
   * Navigálj a fájl helyére, majd futtasd a következő parancsot:

bash

MásolásSzerkesztés

python count\_headings.py

1. A program elindul, és megnyílik egy **fájl választó ablak**, ahol kiválaszthatod a Word fájlt. Miután kiválasztottad a fájlt, a program megszámolja a címsorokat és kiírja az eredményeket.

**6. Eredmény kiírása:**

Ha a kiválasztott dokumentumban van néhány címsor, a kimenet így nézhet ki:

yaml

MásolásSzerkesztés

Címsor 1: 3

Címsor 2: 2

Címsor 3: 1

**7. Mi következik?**

Most már könnyedén választhatsz ki egy Word dokumentumot a fájlkezelő segítségével, és megszámolhatod a címsorokat a dokumentumban. Ha bármilyen más műveletet szeretnél hozzáadni, szólj, és segítek!

Let’s make it easy and fun! 😊

**Promt: Szoval ez a kimenete az example.docx-re: Példa dokumentum generálva: example.docx Címsor 1: 1 Címsor 2: 1 Címsor 3: 1**

Igen, pontosan! Az általad generált example.docx dokumentumban három különböző címsor található:

1. **Heading 1**: "Main Topic"
2. **Heading 2**: "Subtopic 1"
3. **Heading 3**: "Detail 1"

Ezért a kimenet azt mutatja, hogy egy-egy cím szerepel a dokumentumban az első, második és harmadik szinten, tehát:

yaml

MásolásSzerkesztés

Címsor 1: 1

Címsor 2: 1

Címsor 3: 1

Ez a várt eredmény, mivel mindegyik címsor pontosan egyszer szerepel a dokumentumban.

Ha szeretnél más típusú tesztet végezni, vagy más dokumentumot használni a programhoz, bátran szólj, és segítek.

Python-kód: pl. <https://miau.my-x.hu/miau/323/robot_lektor/example.docx>

from docx import Document

from tkinter.filedialog import askopenfilename

# FĂˇjl kivĂˇlasztĂˇs (szintĂ©n tallĂłzĂˇs)

def file\_picker():

filepath = askopenfilename(

title="VĂˇlaszd ki a dokumentumot",

filetypes=[("Word dokumentumok", "\*.docx")]

)

return filepath

# Dokumentum betĂ¶ltĂ©se Ă©s cĂ­msorok szĂˇmolĂˇsa

def count\_headings(doc\_path):

doc = Document(doc\_path)

# CĂ­msorok szĂˇmlĂˇlĂˇsa

heading\_1\_count = 0

heading\_2\_count = 0

heading\_3\_count = 0

# Dokumentumban vĂ©gigmegyĂĽnk a bekezdĂ©seken

for para in doc.paragraphs:

if para.style.name == 'Heading 1':

heading\_1\_count += 1

elif para.style.name == 'Heading 2':

heading\_2\_count += 1

elif para.style.name == 'Heading 3':

heading\_3\_count += 1

print(f"CĂ­msor 1: {heading\_1\_count}")

print(f"CĂ­msor 2: {heading\_2\_count}")

print(f"CĂ­msor 3: {heading\_3\_count}")

# PĂ©lda dokumentum generĂˇlĂˇsa

def generate\_example\_doc():

doc = Document()

doc.add\_heading('Main Topic', level=1)

doc.add\_paragraph('This is some introduction text under Main Topic.')

doc.add\_heading('Subtopic 1', level=2)

doc.add\_paragraph('This is some content under Subtopic 1.')

doc.add\_heading('Detail 1', level=3)

doc.add\_paragraph('This is detailed content under Detail 1.')

doc.save('example.docx')

print("PĂ©lda dokumentum generĂˇlva: example.docx")

# FĹ‘ program

if \_\_name\_\_ == "\_\_main\_\_":

generate\_example\_doc()

# FĂˇjl kivĂˇlasztĂˇsa

file\_path = file\_picker()

if file\_path:

count\_headings(file\_path)