

# Képszerkeztő python script

Készítette:

Borics Gergő Bendegúz GPOZX9

Molnár Gábor VQHXU5

## 1. A projekt rövid leírása

A projekt célja egy olyan Python alapú, parancssoros batch képmódosító program készítése, amely képes egyszerre több képet feldolgozni egy kiválasztott mappából. A felhasználó egy mappaválasztó ablak segítségével adhatja meg a bemeneti képeket, majd különböző képmanipulációs műveletek közül választhat. A program a kiválasztott módosítást automatikusan alkalmazza az összes képre, és az eredményeket egy külön „output” mappába menti.

A támogatott műveletek közé tartozik a képek tömörítése, arányos pixel alapú átméretezése, fekete–fehér konverzió, forgatás, tükrözés, PNG és JPG formátumok közötti átalakítás, valamint a különböző módszerekkel történő képkivágás (crop). A script kifejezetten alkalmas nagy mennyiségű kép gyors, automatizált feldolgozására, és könnyen továbbfejleszthető grafikus felület vagy további funkciók irányába.

## 2. A program működésének részletes leírása

A program működésének alapja, hogy a felhasználó által kiválasztott mappában található összes képfájl automatikusan feldolgozza a megadott módosítások szerint. A folyamat több fő lépésből áll, amelyek egymásra épülve alkotják a teljes működést.

### 1) Mappa kiválasztása

A program indítás után egy grafikus mappaválasztó ablakot jelenít meg (a Tkinter askdirectory használatával). A felhasználónak itt ki kell választania azt a könyvtárat, amely a feldolgozni kívánt képeket tartalmazza. A kiválasztott mappában található képek listája automatikusan betöltődik.

A program csak a következő kiterjesztéseket fogadja el:

- .jpg
- .jpeg
- .png
- .bmp

### 2) Kimeneti mappa létrehozása

A feldolgozott képek nem írják felül az eredetieket. A program automatikusan létrehoz egy output nevű almappát a kiválasztott mappán belül, és minden eredményt ide ment.

### 3) Menü működése

A program egy többszintű menüt jelenít meg, amelyből a felhasználó kiválaszthatja, hogy milyen műveletet szeretne végezni az összes képen.

A menüben jelenleg az alábbi lehetőségek érhetők el:

- a) Képek tömörítése (JPEG quality)
- b) Képek pixel alapú (arányos) átméretezése
- c) Fekete–fehér konverzió
- d) Képek elforgatása
- e) Képek tükrözése
- f) PNG → JPG konverzió
- g) JPG → PNG konverzió
- h) Képkivágás (crop)

Minden menüpont egy teljes feldolgozási ciklust indít el, amely az összes képre lefut.

### 4) Műveletek részletezése

#### I. Képek tömörítése

A felhasználó három előre beállított tömörítési érték közül választhat:

- a) High 90%
- b) Medium 70%
- c) Low 40%

A program szükség esetén automatikusan átkonvertálja a PNG képeket JPG formátumba, mivel a JPEG tömörítési eljárást csak ezen a formátumon lehet alkalmazni. A tömörített képek új névvel mentődnek (\_compressed\_XX).

#### II. Pixel alapú arányos átméretezés

Ebben a módban a felhasználó kiválaszthatja a kívánt új szélességet:

- a) 1920px
- b) 1280px
- c) 800px

A magasság automatikusan, arányosan kerül kiszámításra, így a kép nem torzul. Az átméretezett képek (resized\_XXXXpx) néven kerülnek mentésre.

### III. Fekete–fehér konverzió

A színes képek fekete–fehér változatát a program a `PIL convert("L")` metódusával állítja elő. Az eredmény `_bw` utótaggal kerül mentésre.

### IV. Elforgatás

A felhasználó megadhatja a kívánt elforgatási szöget:

- a)  $90^\circ$
- b)  $180^\circ$
- c)  $270^\circ$

A kép tartalma ennek megfelelően elforgatásra kerül, majd `rotated` névvel kerül mentésre.

### V. Tükrözés

A tükrözés (bal–jobb irányban) a `transpose(Image.FLIP_LEFT_RIGHT)` művelettel történik. A módosított képek `_flipped` néven mentődnek.

### VI. PNG → JPG konverzió

A PNG képek átkonvertálásához először a program:

- a) RGB módba alakítja a képet
- b) JPG formátumban menti el

A konvertált fájl `png_to_jpg` utótaggal rendelkezik.

### VII. JPG → PNG konverzió

Ebben a módban a program a JPG képeket menti el PNG formátumba anélkül, hogy a tartalmat módosítaná. Az eredmény neve `_jpg_to_png` utótagot kap.

### VIII. Képkivágás (Crop)

Ez a funkció lehetővé teszi a képek méretének csökkentését a tartalom levágásával. A felhasználó három különböző módszer közül választhat:

A. Képarány szerinti kivágás (középről) A program automatikusan kiszámolja a kivágandó területet úgy, hogy a kép közepe megmaradjon, de megfelelően a választott képaránynak.

Választható arányok:

- 16:9 (Szélesvásznú)
- 4:3 (Hagyományos)
- 1:1 (Négyzet)
- 9:16 (Álló – pl. Instagram Story)
- 3:2 (Fotó szabvány)
- *Működés:* Ha a kép túl széles, a széleiből vág le; ha túl magas, az aljából és tetejéből.

B. Fix méret kivágása (középről) A felhasználó megad egy konkrét pixelméretet (pl. 500x500 px). A program a kép közepéből vágja ki ezt a területet.

- *Biztonsági funkció:* Ha a megadott méret nagyobb, mint az eredeti kép, a program kihagyja a fájlt, hogy elkerülje a hibákat.

C. Egyedi koordináták megadása Haladó mód, ahol a felhasználó manuálisan adja meg a kivágási határokat:

- Bal (left), Felső (top), Jobb (right), Alsó (bottom) értékek pixelben.
- *Koordináta-rendszer:* A kép bal felső sarka a (0,0).
- *Validáció:* A program ellenőrzi, hogy a megadott koordináták a képen belül esnek-e.

A kivágott képek fájlnevei a választott módszertől függően változnak (pl. `_crop_16x9`, `_crop_500x500`, `_crop_custom`).

## IX. Folyamat lezárása

Minden művelet végrehajtása után a program visszatér a menübe, így akár több különböző művelet is egymás után alkalmazható ugyanarra a képkészletre. A program csak akkor áll le, ha a felhasználó a Kilépés (9) opciót választja.

## 3. Használati útmutató

A program parancssoros felületen működik, de a mappa kiválasztása grafikus ablakon keresztül történik.

### 1) A program indítása

A program futtatásához Python szükséges. VS Code terminálból:

- python main.py

A program elindul, és megjelenik a mappaválasztó ablak.

## 2) Mappa kiválasztása

A program első lépésben egy grafikus ablakot jelenít meg, amelyben ki kell választani:

- azt a mappát, amely a feldolgozni kívánt képeket tartalmazza

A képek listája automatikusan betöltődik. Ha a felhasználó nem választ mappát, a program figyelmeztet és újra kéri a választást.

## 3) Művelet kiválasztása

Sikeres mappaválasztás után a program megjeleníti a műveleti menüt:

```
=== Batch Képmódosító Program ===
1. Képek tömörítése (méretcsökkentés)
2. Képek pixel átméretezése (arányos)
3. Fekete-fehér
4. Elforgatás
5. Tükrözés
6. PNG → JPG átalakítás
7. JPG → PNG átalakítás
8. Képkivágás (Crop)
9. Kilépés
Válassz: 
```

A felhasználó egy szám megadásával választhat a funkciók közül.

## 4) Feldolgozás menete

A program minden esetben **batch módban** működik, azaz:

- a kiválasztott művelet az összes képre automatikusan alkalmazódik
- a képek módosított változatai egy külön mappába kerülnek mentésre

A kimeneti mappa neve: output/

Ez automatikusan jön létre az eredeti képmappa alatt.

#### 4. Függvények részletes magyarázata

##### 1) choose\_folder() – Mappa kiválasztása

```
7   # --- Mappa kiválasztása ---
8   def choose_folder():
9       root = tk.Tk()
10      root.withdraw()
11      folder = askdirectory(title="Válassz egy mappát a képekhez")
12      if folder:
13          print("Képek betöltve innen:", folder)
14          return folder
15      else:
16          print("Nem választottál!")
17          return None
```

Feladata: Grafikus ablak segítségével kiválasztani azt a mappát, amely a feldolgozni kívánt képeket tartalmazza.

Működése:

- A Tk() létrehoz egy rejtett ablakot.
- Az askdirectory() egy Windows-mappa választó ablakot nyit meg.
- Ha a felhasználó kiválaszt egy mappát, visszaadja az elérési utat.
- Ha nem választ, None értékkel tér vissza.

A program addig ismétli a kérdést, amíg érvényes mappát nem ad meg.

##### 2) get\_images\_in\_folder(folder) – Képfájlok listázása

```
20  # --- Képek listázása ---
21  def get_images_in_folder(folder):
22      supported = (".jpg", ".jpeg", ".png", ".bmp")
23      return [f for f in os.listdir(folder) if f.lower().endswith(supported)]
```

Feladata: Betölteni az összes megfelelő formátumú képfájlt a megadott mappából.

Működése:

- Végigmegy a mappa tartalmán (os.listdir).
- Csak azokat a fájlokat veszi figyelembe, amelyek a támogatott kiterjesztések valamelyikével végződnek.
- A képfájlok listáját adja vissza.

### 3) save\_image() – Képek mentése

```
26 # --- Mentés ---
27 def save_image(img, output_folder, original_name, suffix, quality=None, ext_override=None):
28     name, ext = os.path.splitext(original_name)
29
30     # alapértelmezett formátum JPG
31     if ext_override is None:
32         new_ext = ".jpg"
33     else:
34         new_ext = ext_override
35
36     output_path = os.path.join(output_folder, f"{name}_{suffix}{new_ext}")
37
38     if quality:
39         img.save(output_path, quality=quality)
40     else:
41         img.save(output_path)
42
43     print("Mentve:", output_path)
```

Feladata: A feldolgozott képet elmenteni az output/ mappába, új fájl névvel.

Működése:

- Az eredeti fájl név alapján új fájl nevet generál a suffix hozzáadásával.
- Ha ext\_override meg van adva (pl. PNG → JPG), akkor az új formátumot használja.
- Ha a tömörítésnél quality érték van megadva, azt beállítja a képnél.
- Kírja a kimeneti fájl pontos elérési útját.

Ez biztosítja, hogy az eredeti képek ne sérüljenek.

### 4) resize\_by\_width() – Arányos pixel átméretezés

```
46 # --- Arányos átméretezés ---
47 def resize_by_width(img, target_width):
48     target_width = int(target_width)
49     orig_w, orig_h = img.size
50     scale = target_width / float(orig_w)
51     new_h = int(orig_h * scale)
52     return img.resize((target_width, new_h), Image.LANCZOS)
```

Feladata: A kép új szélességét a felhasználó választása alapján módosítani, miközben a magasság arányosan változik.

Működése:

- a) Lekérdezi az eredeti szélességet és magasságot.
- b) Kiszámítja az átméretezési arányt:

```
50 scale = target_width / float(orig_w)
```



c) A magasságot arányosan módosítja:

```
new_h = int(orig_h * scale)
```

d) Az Image.LANCZOS interpoláció nagy felbontású, minőségi átméretezést biztosít.

A kép így SOSEM torzul, megmaradnak a képarányok.

## 5) Konverziós függvények

### I. convert\_png\_to\_jpg()

```
56 def convert_png_to_jpg(img):  
57     return img.convert("RGB")  
58
```

```
186 # 6. PNG -> JPG  
187 elif choice == "6":  
188     for file in images:  
189         if file.lower().endswith(".png"):  
190             path = os.path.join(folder, file)  
191             img = Image.open(path)  
192             jpg = convert_png_to_jpg(img)  
193             save_image(jpg, output_folder, file, "png_to_jpg", ext_override=".jpg")  
194         else:  
195             print(f"{file} nem PNG -> kihagyva!")  
196
```

Feladata: A PNG formátumú képek JPG formátumba alakítása.

Miért kell? A PNG fájlok tartalmazhatnak átlátszóságot, amit a JPG nem kezel.

Ezért a kép RGB módba konvertálódik, majd JPG-ként lesz elmentve.

### II. convert\_jpg\_to\_png()

```
60 def convert_jpg_to_png(img):  
61     return img  
62
```

```
197 # 7. JPG -> PNG  
198 elif choice == "7":  
199     for file in images:  
200         if file.lower().endswith((".jpg", ".jpeg")):  
201             path = os.path.join(folder, file)  
202             img = Image.open(path)  
203             save_image(img, output_folder, file, "jpg_to_png", ext_override=".png")  
204         else:  
205             print(f"{file} nem JPG -> kihagyva!")  
206  
207     print("\nMinden kép feldolgozva!\n")  
208
```

Feladata: A JPG képet átalakítani PNG formátumra.

A PNG veszteségmentes formátum, így a JPG tartalma változatlanul kerül átmentésre, csak a formátum változik.

## 6) Crop (Képkivágó) függvények

A képkivágás logikája három, egymásra épülő függvényre bomlik a modularitás érdekében.

a) `crop_image(img, left, top, right, bottom)`

```
# --- Képkivágás (Crop) ---
def crop_image(img, left, top, right, bottom):
    """
    Képkivágás a megadott koordináták alapján.
    left, top: bal felső sarok
    right, bottom: jobb alsó sarok
    """
    return img.crop((left, top, right, bottom))
```

Feladata: Ez az alapszintű függvény, amely közvetlenül meghívja a PIL könyvtár `crop()` metódusát a pontos koordinátákkal.

Működése: A megadott 4 koordináta által határolt téglalapot adja vissza új képként.

b) `crop_center(img, crop_width, crop_height)`

```
def crop_center(img, crop_width, crop_height):
    """
    Középről kivágás a megadott szélességgel és magassággal.
    """
    orig_w, orig_h = img.size
    left = (orig_w - crop_width) // 2
    top = (orig_h - crop_height) // 2
    right = left + crop_width
    bottom = top + crop_height
    return img.crop((left, top, right, bottom))
```

Feladata: Kiszámítani a központi kivágás koordinátáit, ha csak a méretet ismerjük.

Működése:

1. Megvizsgálja az eredeti kép méretét.
2. Kiszámolja a left és top értékeket úgy, hogy a kivágott terület középre essen (az osztás maradék nélkül történik: // 2).
3. Ezután meghívja a crop funkciót a kiszámolt értékekkel.

c) crop\_aspect\_ratio(img, aspect\_w, aspect\_h)

```
def crop_aspect_ratio(img, aspect_w, aspect_h):  
    """  
    Képarány szerinti kivágás (középről).  
    Pl: 16:9, 4:3, 1:1  
    """  
  
    orig_w, orig_h = img.size  
    target_ratio = aspect_w / aspect_h  
    orig_ratio = orig_w / orig_h  
  
    if orig_ratio > target_ratio:  
        # Túl széles, levágunk oldalról  
        new_w = int(orig_h * target_ratio)  
        new_h = orig_h  
    else:  
        # Túl magas, levágunk felül/alul  
        new_w = orig_w  
        new_h = int(orig_w / target_ratio)  
  
    return crop_center(img, new_w, new_h)
```

Feladata: Képarány alapján meghatározni a maximális méretű, középre igazított kivágást.

Működése:

1. Összehasonlítja az eredeti kép arányát (orig\_ratio) a kívánt aránnyal (target\_ratio).
2. Ha a kép szélesebb a kelleténél, akkor a magasságot hagyja meg, és az új szélességet számolja ki.
3. Ha a kép magasabb a kelleténél, akkor a szélességet hagyja meg, és az új magasságot számolja ki.
4. Végül átadja a kiszámolt méreteket a crop\_center függvénynek, így a kivágás mindig középről történik.

## 7) menu() – A felhasználói menü

```
# --- Menü ---
def menu():
    print("\n=== Batch Képmódosító Program ===")
    print("1. Képek tömörítése (méretcsökkentés)")
    print("2. Képek pixel átméretezése (arányos)")
    print("3. Fekete-fehér")
    print("4. Elforogatás")
    print("5. Tükrözés")
    print("6. PNG → JPG átalakítás")
    print("7. JPG → PNG átalakítás")
    print("8. Képkivágás (Crop)")
    print("9. Kilépés")
    return input("Válassz: ")
```

Feladata:

A főmenü megjelenítése és a felhasználói választás bekérése.

Fontos:

- Minden futtatási ciklus elején meghívódik.
- A választás határozza meg, melyik műveletet indítja a program.

## 8) main() – A program vezérlőközpontja

A main() felelős a teljes működésért.

Főbb lépések:

a) Mappa kiválasztása

```
78 # --- Fő program ---
79 def main():
80
81     print("Mappaválasztás...\n")
82
83     folder = None
84     while folder is None:
85         folder = choose_folder()
```

b) Képek beolvasása

```

87     images = get_images_in_folder(folder)
88
89     if not images:
90         print("A mappában nincsenek képek!")
91         return

```

c) Output mappa létrehozása

```

92
93     output_folder = os.path.join(folder, "output")
94     os.makedirs(output_folder, exist_ok=True)
95

```

d) Menü megjelenítése és műveletek futtatása

```

96     while True:
97         choice = menu()
98
121
122         for file in images:
123             path = os.path.join(folder, file)
124             img = Image.open(path)
125

```

e) Példa egy műveletre (elforgatás):

```

166         # 4. Elforgatás
167         elif choice == "4":
168             angle = int(input("Hány fokkal forgassam? (90, 180, 270): "))
169
170             for file in images:
171                 path = os.path.join(folder, file)
172                 img = Image.open(path)
173
174                 rot = img.rotate(angle, expand=True)
175                 save_image(rot, output_folder, file, f"rotated")

```

f) A ciklus addig fut, amíg a felhasználó ki nem lép a 9-as menüponttal.

```

# Kilépés
if choice == "9":
    print("Kilépés...")
    break

```

## 5. Külső források, amiket felhasználtunk vagy inspirációnak használtunk

a) Image Processing Using Pillow in Python – Real Python

Átfogó Python & Pillow tutorial, bemutat képek betöltését, átméretezését, forgatását, csatornák kezelését. Jó alap a script működésének megértéséhez.

b) Képszerkesztés és manipuláció Pythonnal a Pillow könyvtárral – IrányOnline.hu

Magyar nyelvű cikk, ami segített megérteni, hogyan manipulálhatók képek automatikusan Python-ban: formátumok, mentés, alapvető szerkesztés.

c) How to Build an Image Converter Tool With Python – Medium

Gyakorlati útmutató, hogyan lehet konvertáló / átméretező eszközt írni Python-ban, jó példa arra, hogyan szervezd a kódot.

d) Python Image Editor Using Python – GeeksforGeeks

Konkrét példák: képek betöltése, forgatása, átméretezése, szerkesztése – sok hasznos kódrészlet, amit referenciaként használtunk.

## 6. AI-használat a fejlesztés során

A projekt fejlesztése során bizonyos részeknél mesterséges intelligencia alapú eszközöket is igénybe vettünk, elsősorban segítség, ötletgyűjtés és hibaelhárítás céljából. Az alábbiakban három konkrét területet emelünk ki, ahol AI segítséget alkalmaztunk.

### 1) Információs és dokumentációs források keresése (weboldalak ajánlása)

Az AI eszközt használtuk olyan szakmai oldalak, cikkek és példák felkutatásához, amelyek relevánsak a projekt témájához — például a Python Pillow könyvtár funkcióihoz, képfeldolgozáshoz vagy konverziós technikákhoz.

Az AI ebben segített:

- hiteles forrásokat ajánlott (RealPython, GeeksForGeeks, Medium)
- leírást adott arról, hogy az adott oldal mit magyaráz
- összehasonlította, melyik cikk milyen részhez hasznos
- biztosította, hogy a dokumentáció megfelelő szakmai mélységű legyen

### 2) A mappaválasztó funkcióhoz (Tkinter askdirectory) AI által adott ötletek

A mappa kiválasztásához használt megoldás:



```

8  def choose_folder():
9      root = tk.Tk()
10     root.withdraw()
11     folder = askdirectory(title="Válassz egy mappát a képekhez")
12     if folder:
13         print("Képek betöltve innen:", folder)
14         return folder
15     else:
16         print("Nem választottál!")
17     return None

```

Az AI itt abban segített:

- A Tkinter rejtett ablak (root.withdraw()) helyes használatának megmutatásában.
- Abban, hogy ne jelenjen meg üres Tkinter-ablak a mappaválasztó előtt.
- Segített tiszta, egyszerű, beadásbarát magyarázatot adni a dokumentációhoz.
- Javaslatokat adott a felhasználóbarát működéshez (pl. mappaválasztás újraprobálása).

### 3) A képek arányos átméretezésének algoritmusai AI segítségével lett optimalizálva

```

47  def resize_by_width(img, target_width):
48      target_width = int(target_width)
49      orig_w, orig_h = img.size
50      scale = target_width / float(orig_w)
51      new_h = int(orig_h * scale)
52      return img.resize((target_width, new_h), Image.LANCZOS)

```

Az AI szerepe itt:

- Segített megtalálni a képarány megtartásához szükséges matematikai képletet.
- Rámutatott arra, hogy a target\_width és az orig\_w típusának egységesítésére szükség van.
- Ajánlotta a float(orig\_w) használatát a pontosabb osztáshoz.
- Javasolta a Image.LANCZOS interpolációs módszer használatát a jobb minőség érdekében.
- Segített elkerülni a "silent fail" hibát, amikor string típusú értékkel nincs átméretezés.