Aula 11

Pillow

O Pillow é uma biblioteca usada pelo Python para processar imagens.

Como ele não é uma biblioteca padrão, nós temos que instalar ela usando o pip. Veja abaixo:

```
(.venv) C:\Users\gutoh\curso>pip install pillow
Collecting pillow
  Downloading Pillow-10.0.0-cp311-cp311-win_amd64.whl (2.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 8.0 MB/s eta 0:00:00
Installing collected packages: pillow
Successfully installed pillow-10.0.0

(.venv) C:\Users\gutoh\curso>
```

Lembre de realizar a instalação com o ambiente virtual ativado, como acima mostrado.

Agora, é possível trabalharmos com as imagens diretamente nos scripts Python. Veja um exemplo abaixo:

C:\Users\gutoh\curso\imagem.py

```
1  from PIL import Image
2
3  if __name__ == '__main__':
4      nome_arq = 'frances.jpg'
5
6      with Image.open(nome_arq) as img:
7          img.load()
8
9      img.show()
10
11     print(f'tipo da imagem {type(img)}')
12     print(f'isinstance(img, Image.Image): {isinstance(img, Image.Image)}')
13
```

No exemplo acima, realizamos a importação da biblioteca. Após, na linha 6 e 7, abrimos a imagem (da mesma forma como se estivéssemos abrindo um arquivo de texto ou json) e carregamos a imagem com o método load() e depois mostramos ela com o método show().

Veja a imagem carregada abaixo:

Como podemos ver no terminal, temos os prints do tipo da imagem e a verificação da instância da variável:

```
(.venv) C:\Users\gutoh\curso>python imagem.py
tipo da imagem <class 'PIL.JpegImagePlugin.JpegImageFile'>
isinstance(img, Image.Image): True

(.venv) C:\Users\gutoh\curso>
```

Também podemos verificar outros dados, como tipo, tamanho e modo

```
C:\Users\gutoh\curso\imagem.py
from PIL import Image

if __name__ == '__main__':
    nome_arq = 'frances.jpg'

    with Image.open(nome_arq) as img:
        img.load()

    print(f'img.format : {img.format}')
    print(f'img.size : {img.size}')
    print(f'img.mode : {img.mode}')
```

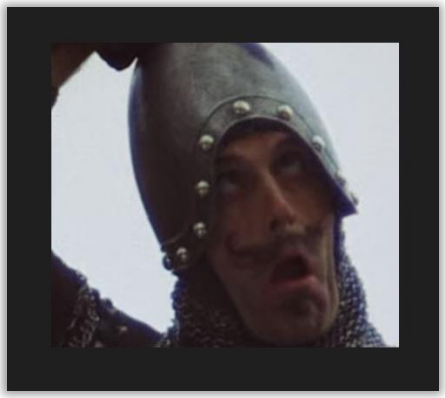
```
(.venv) C:\Users\gutoh\curso>python imagem.py
img.format : JPEG
img.size : (1127, 918)
img.mode : RGB

(.venv) C:\Users\gutoh\curso>
```

Também podemos recortar a imagem (usando um crop, como é chamado pelos editores de imagens).

Veja abaixo o código:

```
C:\Users\gutoh\curso\imagem.py
from PIL import Image

if __name__ == '__main__':
    nome_arq = 'frances.jpg'

    with Image.open(nome_arq) as img:
        img.load()

    imagem_recortada = img.crop((300, 150, 700, 500))
    print(f'imagem_recortada : {imagem_recortada.size}')

    imagem_recortada.show()
```

E teremos como resultado:



Repare que passamos uma tupla para o método crop. Aquele tupla representa as coordenadas da imagem que queremos realizar o recorte, esquerda, acima, direita e abaixo, respectivamente.

Também podemos reduzir a qualidade da imagem:

C:\Users\gutoh\curso\imagem.py

```python
from PIL import Image

if __name__ == '__main__':
    nome_arq = 'frances.jpg'

    with Image.open(nome_arq) as img:
        img.load()

    imagem_recortada = img.crop((300, 150, 700, 500))

    imagem_baixa_resolucao = imagem_recortada.resize(
        (imagem_recortada.width // 4, imagem_recortada.height // 4)
    )

    imagem_baixa_resolucao.show()
```

Acima, podemos ver que a imagem foi reduzida a tua quarta parte do tamanho. O método resize recebe como parâmetro dois valores, a altura e a largura.
Também podemos realizar essa alteração usando o método reduce. Ele recebe apenas um inteiro, que é o fator que a imagem será reduzida.

Depois de alterada, podemos salvar a imagem como método save. Veja abaixo:

C:\Users\gutoh\curso\imagem.py

```python
from PIL import Image

if __name__ == '__main__':
    nome_arq = 'frances.jpg'

    with Image.open(nome_arq) as img:
        img.load()

    imagem_recortada = img.crop((300, 150, 700, 500))

    imagem_recortada.save('foto-perfil.jpg')
```

Isso vai gerar um novo arquivo de imagem com o nome especificado.

Há também outros métodos de manipulação de uma imagem, como:

- transpose: passando Image.FLIP_TOP_BOTTOM vai deixar a imagem de cabeça para baixo;
    - outras opções incluem: FLIP_LEFT_RIGHT, ROTATE_90 (180 ou 270), TRANSPOSE e TRANSVERSE;
- rotate: rotaciona a imagem de acordo com o valor inteiro passado, representando os graus de rotação;
- filter: onde podemos aplicar um filtro na imagem, como o Blur;

Mais usos podem ser encontrados nesses dois links:

- Documentação
- Processando Imagens com Pillow

Django

Modelos

Os modelos são criados no Django para que possamos fazer com que os dados da nossa aplicação sejam salvos em um banco de dados.

Para fazer isso, vamos alterar o arquivo **/linguagem_cpp/models.py**. Esse arquivo é o responsável por ter a classe que vai controlar os dados daquele aplicativo.

Veja como vai ficar a nossa classe, usando como modelo uma estrutura parecida com a que fizemos no arquivo **/utils/versoes/fabrica.py**:

C:\Users\gutoh\curso\linguagem_cpp\models.py

```
10 from django.db import models
 9
 8 class Versao(models.Model):
 7     titulo = models.CharField(max_length=100)
 6     versao = models.IntegerField()
 5     slug = models.SlugField()
 4     descricao = models.CharField(max_length=200)
 3     data_lanc = models.DateField()
 2     eh_publicado = models.BooleanField(default=False)
 1     foto = models.ImageField(upload_to='imagens/Y%/%m/%d/')
11
```

Por enquanto, nosso código da classe ficará assim.

Sobre o código acima:

- o campo slug será usado para criarmos a url com base em algum texto;
- vários campos são semelhantes aos usados no banco de dados;

Agora, finalmente, vamos nos livrar daquela mensagem que sempre vemos ao executarmos o servidor!

```
(.venv) C:\Users\gutoh\curso>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): adm
in, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
July 03, 2023 - 17:17:01
Django version 4.2.2, using settings 'curso.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Para isso, temos que executar o comando abaixo:

```
(.venv) C:\Users\gutoh\curso>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial ... OK
  Applying auth.0001_initial ... OK
  Applying admin.0001_initial ... OK
  Applying admin.0002_logentry_remove_auto_add ... OK
  Applying admin.0003_logentry_add_action_flag_choices ... OK
  Applying contenttypes.0002_remove_content_type_name ... OK
  Applying auth.0002_alter_permission_name_max_length ... OK
  Applying auth.0003_alter_user_email_max_length ... OK
  Applying auth.0004_alter_user_username_opts ... OK
  Applying auth.0005_alter_user_last_login_null ... OK
  Applying auth.0006_require_contenttypes_0002 ... OK
  Applying auth.0007_alter_validators_add_error_messages ... OK
  Applying auth.0008_alter_user_username_max_length ... OK
  Applying auth.0009_alter_user_last_name_max_length ... OK
  Applying auth.0010_alter_group_name_max_length ... OK
  Applying auth.0011_update_proxy_permissions ... OK
  Applying auth.0012_alter_user_first_name_max_length ... OK
  Applying sessions.0001_initial ... OK

(.venv) C:\Users\gutoh\curso>
```

Esse comando, vai preparar o banco de dados para receber as nossas classes

Agora, vamos executar outro comando para que as nossas classes sejam usadas no banco de dados:

```
(.venv) C:\Users\gutoh\curso>python manage.py makemigrations
Migrations for 'linguagem_cpp':
  linguagem_cpp\migrations\0001_initial.py
    - Create model Versao

(.venv) C:\Users\gutoh\curso>
```

Esse comando vai gerar um arquivo que vai ficar salvo na pasta especificada.

Esse arquivo contém todo o código que será usado pelo Django para criar as nossas tabelas no banco de dados.

C:\Users\gutoh\curso\linguagem_cpp\migrations\0001_initial.py

```python
# Generated by Django 4.2.2 on 2023-07-03 20:25

from django.db import migrations, models


class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='Versao',
            fields=[
                ('id', models.BigAutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('titulo', models.CharField(max_length=100)),
                ('versao', models.IntegerField()),
                ('slug', models.SlugField()),
                ('descricao', models.CharField(max_length=200)),
                ('data_lanc', models.DateField()),
                ('eh_publicado', models.BooleanField(default=False)),
                ('foto', models.ImageField(upload_to='imagens/Y%/%m/%d/')),
            ],
        ),
    ]
```

Repare no nome do arquivo. Esse nome e o todo o código dentro foi gerado automaticamente. Eles **NUNCA** devem ser alterados.

Agora, executaremos novamente o comando abaixo. Ele vai realizar a criação das tabelas para nosso aplicativo.

```
(.venv) C:\Users\gutoh\curso>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, linguagem_cpp, sessions
Running migrations:
  Applying linguagem_cpp.0001_initial ... OK

(.venv) C:\Users\gutoh\curso>
```

Agora, quando nossos modelos criados serão gerados dentro do arquivo de banco de dados db.sqlite3.

IMPORTANTE: sempre que qualquer alteração for feita nos arquivos models.py de qualquer aplicativo, somos obrigados a reexecutar os dois comandos:

- python manage.py makemigrations
- python manage.py migrate

Exercícios para Praticar

1. Carregue uma imagem para uma variável usando o método Image.open().
2. Mostre a imagem carregada usando o método Image.show().
3. Recorte a imagem carregada para obter uma região retangular específica usando o método Image.crop().
4. Altere o tamanho da imagem carregada para 800x600 pixels usando o método Image.resize().
5. Transponha a imagem carregada verticalmente usando o método Image.transpose().
6. Rotacione a imagem carregada em 90 graus no sentido horário usando o método Image.rotate().
7. Salve a nova imagem resultante após o redimensionamento usando o método Image.save().
8. Carregue uma imagem chamada "foto.png" para uma variável usando o método Image.open().
9. Mostre a imagem carregada usando o método Image.show().
10. Recorte a imagem carregada para obter uma região retangular específica usando o método Image.crop().
11. Altere o tamanho da imagem carregada para 500x500 pixels usando o método Image.resize().
12. Transponha a imagem carregada horizontalmente usando o método Image.transpose().
13. Rotacione a imagem carregada em 180 graus no sentido anti-horário usando o método Image.rotate().
14. Salve a nova imagem resultante após o redimensionamento com o nome "nova_foto.png" usando o método Image.save().
15. Carregue uma imagem chamada "paisagem.jpg" para uma variável usando o método Image.open().
16. Mostre a imagem carregada usando o método Image.show().
17. Recorte a imagem carregada para obter uma região retangular específica usando o método Image.crop().
18. Altere o tamanho da imagem carregada para 1200x800 pixels usando o método Image.resize().
19. Transponha a imagem carregada verticalmente usando o método Image.transpose().
20. Rotacione a imagem carregada em 45 graus no sentido horário usando o método Image.rotate().
21. Salve a nova imagem resultante após o redimensionamento com o nome "nova_paisagem.jpg" usando o método Image.save().

Repita os exercícios anteriores com diferentes imagens e parâmetros para obter mais variações e prática.

1. Crie as classes para seus aplicativos.
2. Realize a migração para o banco de dados.
3. Abra o arquivo db.sqlite3 e veja seu conteúdo.