Decoradores em Flask

Os decoradores são funções que ‘complementam’ outras funções recebidas como parâmetro com funcionalidades específicas e as retornam.
O uso de decoradores em Flask pode ocorrer de diversas formas, o exemplo que escolhi para ilustrar o uso dessas funções no Flask foi o seguinte:
Em uma API criada em flask, é necessário um token de acesso que é adquirido em um endpoint de autenticação específico, após isso, para acessar todos os outros endpoints da API é necessário o uso desse token.

Para validar o token foi criada a seguinte função:

==========

```
def token_obrigatorio(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None
        # Verificar se um token foi enviado
        if 'x-access-token' in request.headers:
            token = request.headers['x-access-token']
        if not token:
            return jsonify({'mensagem': 'Token não foi incluído'}, 401)
        # Validar o acesso consultando o banco
        try:
            resultado = jwt.decode(token, app.config['SECRET_KEY'], algorithms=["HS256"])
            autor = Autor.query.filter_by(id_autor=resultado['id_autor']).first()
        except:
            return jsonify({'mensagem': 'Token inválido'}, 401)

        return f(autor, *args, **kwargs)
    return decorated
```

==========

Desse modo, toda rota criada para algum endpoint é decorada com esse decorator, de modo que é forçado o uso do token em todas as rotas.

Como por exemplo:

==========

```
@app.route('/postagem/<int:id_postagem>', methods=['GET'])
@token_obrigatorio
def obter_postagem_por_indice(autor, id_postagem):
    postagem = Postagem.query.filter_by(id_postagem=id_postagem).first()
    postagem_atual = {}
    try:
```

```
        postagem_atual['titulo'] = postagem.titulo
    except:
        pass
    postagem_atual['id_autor'] = postagem.id_autor

    return jsonify({'postagens': postagem_atual})
```

==========

Esse é um exemplo do uso de generators em Flask.