Cheat Sheet for comprehensive Flask

Flask Basics

Installation

- **Install Flask**: `pip install Flask`
- **Create a Virtual Environment**: `python -m venv venv`
- **Activate Virtual Environment**:
  - **Windows**: `venv\Scripts\activate`
  - **Unix/MacOS**: `source venv/bin/activate`

Minimal Flask Application

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return 'Hello, Flask!'

if __name__ == '__main__':
    app.run(debug=True)
```

Routing

Basic Routing

- **Define Routes**: `@app.route('/path')`
- **Dynamic Routes**: `@app.route('/user/<username>')`
- **Route Methods**:
  - **GET**: `@app.route('/', methods=['GET'])`
  - **POST**: `@app.route('/submit', methods=['POST'])`

URL Building

- **url_for()**: `url_for('home')`
- **With Parameters**: `url_for('user', username='john')`

Templates

Jinja2 Templates

- **Render Templates**: `render_template('template.html')`
- **Template Inheritance**:
  - **Base Template**: `{% extends "base.html" %}`
  - **Block**: `{% block content %} ... {% endblock %}`

Template Filters

- **Common Filters**:
  - **upper**: `{{ name|upper }}`
  - **default**: `{{ value|default('default value') }}`
  - **length**: `{{ list|length }}`

Request Handling

Accessing Request Data

- **GET Parameters**: `request.args.get('key')`
- **POST Data**: `request.form['key']`
- **JSON Data**: `request.get_json()`

File Uploads

- **Upload File**:

```
from werkzeug.utils import secure_filename

@app.route('/upload', methods=['POST'])
def upload_file():
    file = request.files['file']
    file.save(secure_filename(file.filename))
    return 'File uploaded successfully'
```

Sessions and Cookies

Sessions

- **Set Session**: `session['key'] = 'value'`
- **Get Session**: `session.get('key')`

- **Delete Session**: `session.pop('key', None)`

Cookies

- **Set Cookie**: `resp.set_cookie('key', 'value')`
- **Get Cookie**: `request.cookies.get('key')`

Error Handling

Custom Error Pages

- **404 Error**:

```
@app.errorhandler(404)
def page_not_found(e):
    return render_template('404.html'), 404
```

Debugging

- **Debug Mode**: `app.run(debug=True)`
- **Logging**:

```
import logging
app.logger.setLevel(logging.DEBUG)
app.logger.debug('Debug message')
```

Blueprints

Creating Blueprints

- **Define Blueprint**:

```
from flask import Blueprint

bp = Blueprint('bp_name', __name__)

@bp.route('/')
def index():
    return 'Blueprint Index'
```

- **Register Blueprint**: `app.register_blueprint(bp, url_prefix='/prefix')`

Database Integration

SQLAlchemy

- **Initialize**:

```
from flask_sqlalchemy import SQLAlchemy

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)
```

- **Define Model**:

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
```

- **CRUD Operations**:
  - **Create**: `db.session.add(user)`
  - **Read**: `User.query.all()`
  - **Update**: `user.username = 'new_username'`
  - **Delete**: `db.session.delete(user)`

Forms and Validation

Flask-WTF

- **Install**: `pip install Flask-WTF`
- **Define Form**:

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired

class MyForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    submit = SubmitField('Submit')
```

- **Render Form**:

```
@app.route('/form', methods=['GET', 'POST'])
def form():
    form = MyForm()
    if form.validate_on_submit():
        return 'Form submitted successfully'
    return render_template('form.html', form=form)
```

Static Files

Serving Static Files

- **Static Folder**: `app = Flask(__name__, static_folder='static')`
- **URL for Static File**: `url_for('static', filename='style.css')`

Testing

Unit Testing

- **Basic Test**:

```
import unittest
from app import app

class TestApp(unittest.TestCase):
    def setUp(self):
        self.app = app.test_client()

    def test_home(self):
        response = self.app.get('/')
        self.assertEqual(response.status_code, 200)
```

Deployment

WSGI Server

- **Gunicorn**: `gunicorn -w 4 app:app`
- **Waitress**: `waitress-serve --call 'app:create_app'`

Production Configuration

- **Environment Variables**: `app.config.from_envvar('APP_SETTINGS')`
- **Secure Configuration**:

```
app.config['SECRET_KEY'] = 'your_secret_key'
app.config['SESSION_COOKIE_SECURE'] = True
```

Advanced Topics

Custom CLI Commands

- **Click Integration**:

```
import click
from flask import Flask

app = Flask(__name__)

@app.cli.command('hello')
@click.argument('name')
def hello_command(name):
    click.echo(f'Hello, {name}!')
```

Middleware

- **Custom Middleware**:

```
class MyMiddleware:
    def __init__(self, app):
        self.app = app

    def __call__(self, environ, start_response):
        # Custom logic
        return self.app(environ, start_response)

app.wsgi_app = MyMiddleware(app.wsgi_app)
```

Tips and Tricks

Debugging Tips

- **Flask Debug Toolbar**: `pip install flask-debugtoolbar`
- **Use `print()` for Quick Debugging**: `print(variable)`

Performance Optimization

- **Caching**: `from flask_caching import Cache`
- **Minimize Database Queries**: Use `join` and `select_related`

Security Best Practices

- **Use HTTPS**: `app.run(ssl_context='adhoc')`

- **Sanitize Input**: Use `escape` from `jinja2`

- **Rate Limiting**: `pip install Flask-Limiter`

Resources

Documentation

- **Flask Official Docs**:
https://flask.palletsprojects.com/

- **Jinja2 Docs**: https://jinja.palletsprojects.com/

Community and Support

- **Stack Overflow**:
https://stackoverflow.com/questions/tagged/flask

- **Flask Mailing List**:
https://groups.google.com/forum/#!forum/flask

This cheat sheet provides a comprehensive overview of Flask, covering essential features, best practices, and advanced topics. Use it as a quick reference guide for building and deploying Flask applications.

By Ahmed Baheeg Khorshid

ver 1.0