Introduction to Flask

Brian Dickinson

University of Rochester

Fall 2019
What is Flask?

“What Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions”

- Lightweight Python framework for building web applications
- Uses Web Server Gateway Interface (WSGI)
- Relies on external packages such as Werkzeug and Jinja for all but core implementation
Why are we using Flask for this class?

- Significant development speedup over CGI
- Quick start and gradual learning curve
- Minimalist features make your first project easier
- Explicit design reinforces web programming basics
- Reliance on external packages means you can learn only as much as you need
// Hello World in Java.
public class HelloWorld {
    public static void main( String[] args ) {
        System.out.println( "Hello World" );
    }
}

# Hello World in Python.
print("Hello World")
Python Conditionals

```python
if numerical_grade >= 90:
    letter_grade = "A"
elif numerical_grade >= 80:
    letter_grade = "B"
elif numerical_grade >= 70:
    letter_grade = "C"
elif numerical_grade >= 60:
    letter_grade = "D"
else:
    letter_grade = "F"
```
# standard while loop
i = 0
while i < 10:
    print(i)
    i += 1

# standard for loop
for i in range(10):
    print(i)
# lists are most commonly declared with square brackets
a = [1, 2, 3, 4, 5]
b = ["Hello", "World"]

# more like Java’s ArrayList<Object> Arrays
a.append(6)
b.append("!")

# they can also be easily combined
c = a+b

# iteration over elements is quite simple using for loops
for e in c:
    print(e)
Dictionaries

# dictionaries are most often declared as follows
```
d = {"Hello":0, "World":1}
```

# they can also be created with the constructor
# the constructor expects an iterable of key,value tuples
```
d = dict([("Hello",0),("World",1)])
```

# dictionaries can also be iterated over quite simply
```
for k in d.keys():
    print(k)

for v in d.values():
    print(v)

for k,v in d.items():
    print(k,v)
```
# function definitions have a name and arguments
# unlike in many languages functions do not declare return types

def add(x,y):
    return x+y

# rather than using overloading, Python allows you to define
# default values for your function’s arguments

def hello(name="Stranger"):  
    print(f"Hello {name}!")

# lambda functions can also be used

d = {"Hello": 0, "World":1}  
print(sorted(d.items(), key=lambda t: t[1]))
public class Point {
    public int xCoord, yCoord;
    public Point() { xCoord = yCoord = 0; }
    public Point(int x, int y) {
        xCoord = x;
        yCoord = y;
    }
    public String toString() {
        return String.format("(%d, %d)", xCoord, yCoord);
    }
    public void shift(int xInc, int yInc) {
        xCoord += xInc;
        yCoord += yInc;
    }
}
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

    def __str__(self):
        return f"({self.x}, {self.y})"

    def shift(self, xInc, yInc):
        self.x += xInc
        self.y += yInc
#!/usr/bin/env python3

import sys

# There is nothing special about the function named 
# "main" in a Python script 

def main():
    print("Hello World!")

# In a Python script all lines are executed from top to 
# bottom. This line executes after all imports, functions, 
# and global variables are declared. This if statement 
# checks the script’s name which will only be "__main__" 
# if the script is executed. The name will be different 
# if the script is simply imported.
if __name__=="__main__":
    main()
Flask Hello World

#!/usr/bin/env python3

from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello World!"

# older style of support for direct execution instead of
# execution with the flask command line application
if __name__=='__main__':
    app.run()
#!/usr/bin/env python3

from flask import Flask
app = Flask(__name__)

@app.route('/', defaults={'name': 'Stranger'}, methods=['GET'])
@app.route('/<string:name>', methods=['GET'])
def hello(name):
    return f"Hello {name}!", 200

@app.route('/users/<int:uid>', methods=['GET'])
def users(uid):
    return f"Profile for User: {uid}" , 200

if __name__=='__main__': app.run()
Route Types

Variables passed in routes can be a variety of types.

- string - any text that does not contain a "/" character
- int - any positive integer
- float - any positive floating point value
- path - any text including text that contains a "/" character
- uuid - any valid UUID string
Running a Flask Web Server

# run server on localhost (only accessible on hosting computer)  
# FLASK_ENV=development is optional; localhost is primarily  
# used when a project is in development
export FLASK_APP=my_app.py
export FLASK_ENV=development
flask run

# run generally accessible server
export FLASK_APP=my_app.py
flask run --host=0.0.0.0

# you can replace "flask run ..." with the command
python3 -m flask run ...

# when using the "if __name__==’__main__’: app.run()" idiom  
# (the old style of deployment) you can run your server as
python3 my_app.py
Anonymous.
Flask snippets.
http://flask.pocoo.org/snippets/.

Coding Dojo.
Choosing python web frameworks: Django and flask.

Gareth Dwyer.
Flask vs. django: Why flask might be better.
Rance Necaise.
Python for java programmers.
http://python4java.necaiseweb.org/Main/TableOfContents.

Robert Picard.
Advanced patterns for views and routing.

Armin Ronacher.
Flask.
http://flask.pocoo.org/.
Pallets Team.
Flask quickstart.
http://flask.pocoo.org/docs/1.0/quickstart/.

Wikipedia.com.
Web server gateway interface.