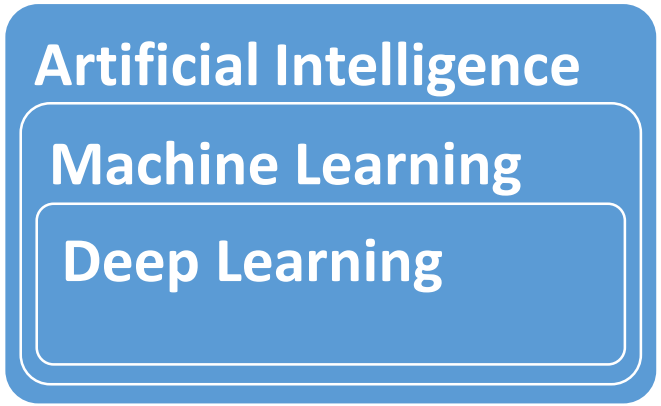
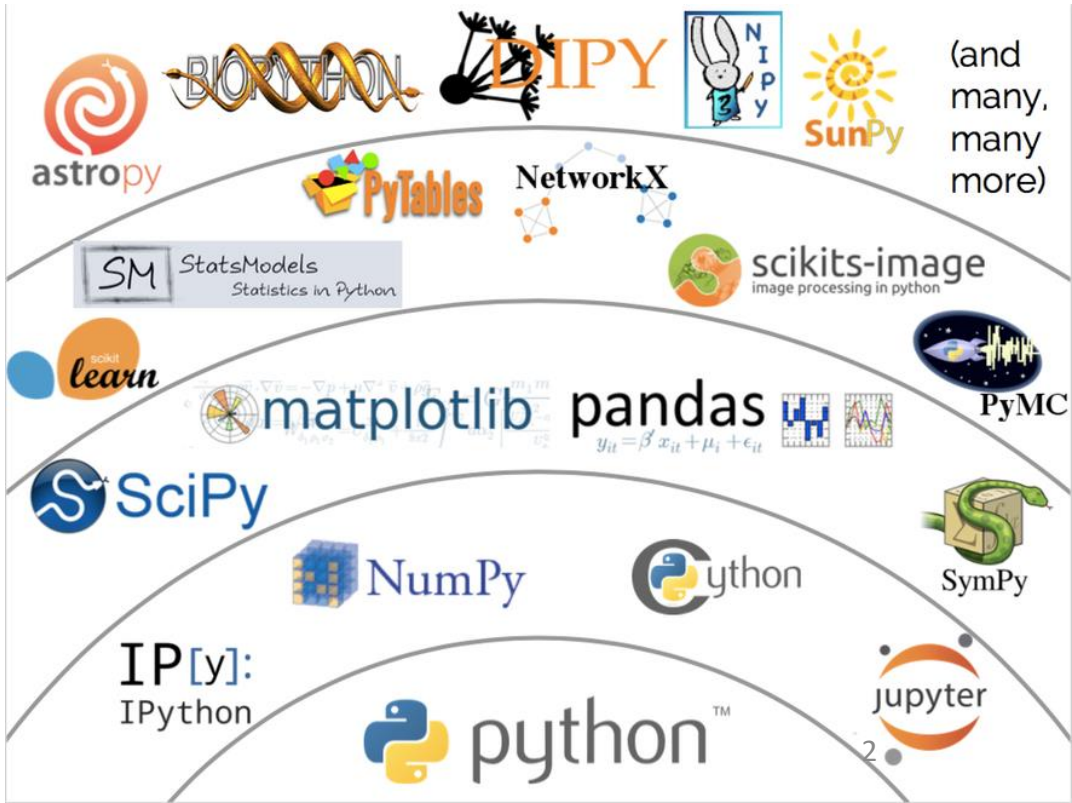
Flask
Python Web Framework

Dr. Sarwan Singh

Agenda

- Introduction – History,





One guiding principle of Python code is that

"explicit is better than implicit"

References

- Fullstackpython,

- pip install -r requirements.txt

Introduction

- Flask is a micro web framework written in Python.
- It is classified as a microframework because it does not require particular tools or libraries.
- It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.
- Developed in 2010 by Armin Ronacher as an April Fool's Day joke
- Written in python, Licensed under BSD
- Has monolithic structure and dependencies

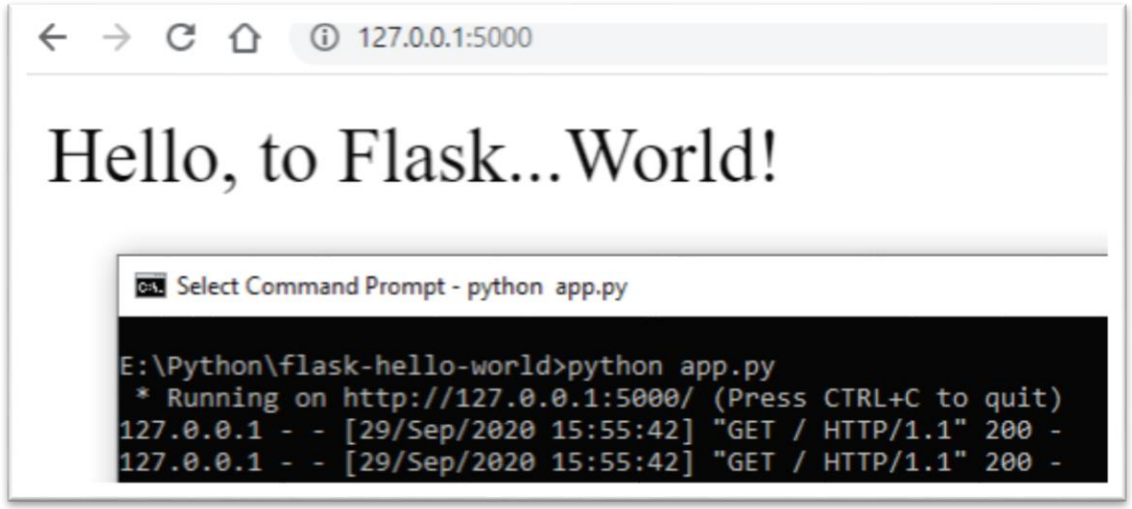
Hello world with Flask

- code shows

  "Hello, World!" on localhost port 5000 in a web browser

- when run with the

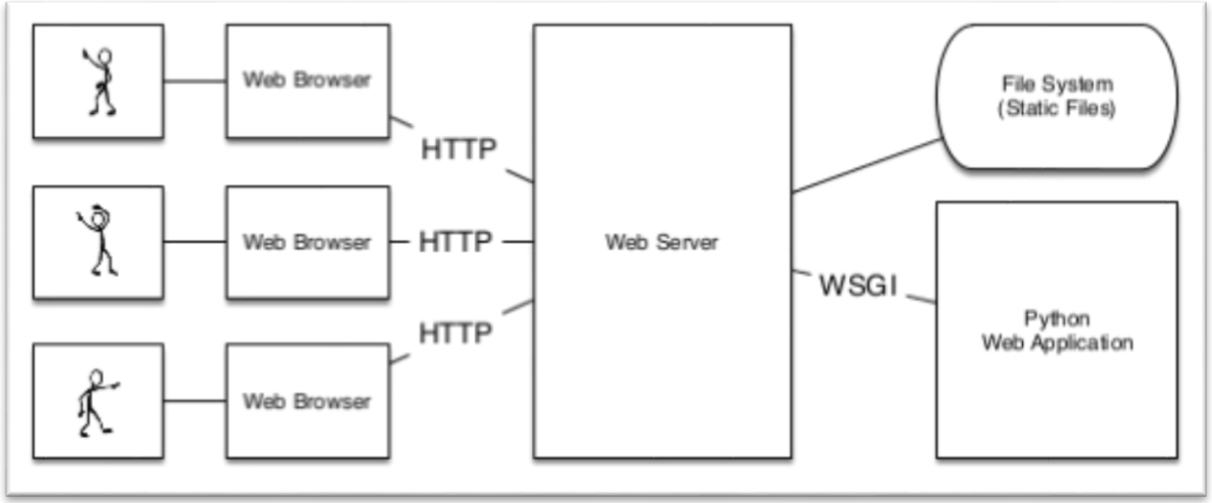
  python app.py command

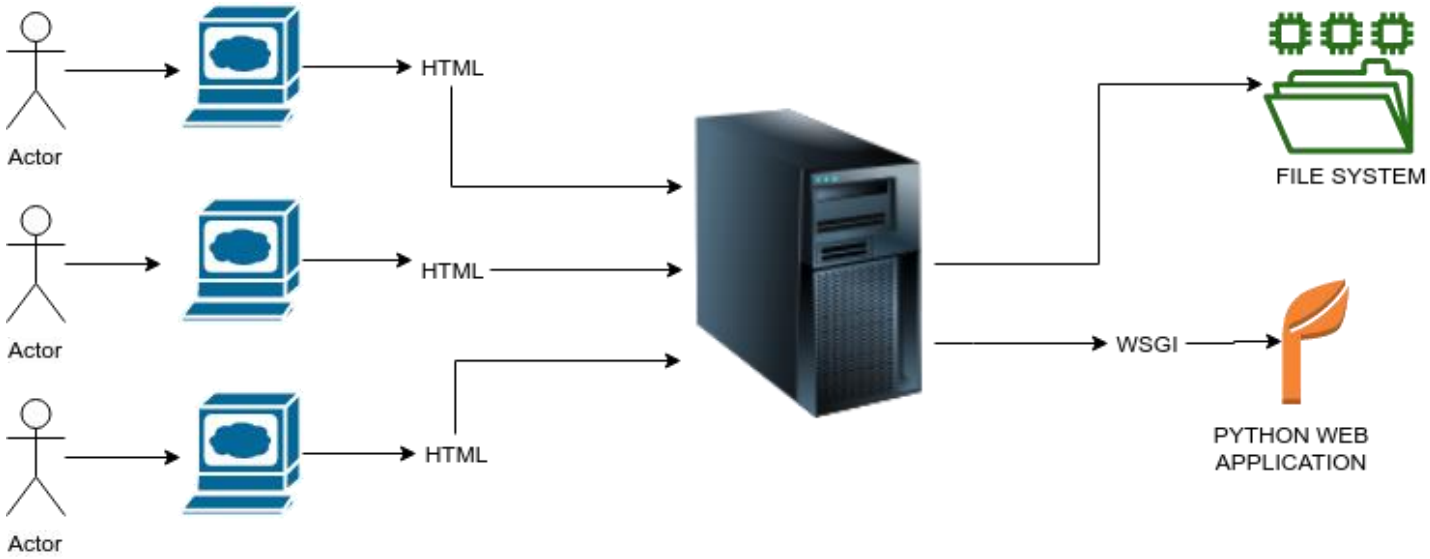



```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'

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

Web Server Gateway Interface

- WSGI is the Web Server Gateway Interface. It is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.
- Flask inherits its high WSGI usage





Flask folder structure

- Flask itself is very flexible.
- It has no certain pattern for a project folder structure, which is very good for experienced developers to organize things in their own favors.

```
project/
    __init__.py
    models/
        __init__.py
        base.py
        users.py
        posts.py
        ...
    routes/
        __init__.py
        home.py
        account.py
        dashboard.py
        ...
    templates/
        base.html
        post.html
        ...
    services/
        __init__.py
        google.py
        mail.py
```

Folder Structure

- There is no fixed folder structure.
- Django has seperate init in each folder, in flask its not compulsory
- But here we did same and unify the init process in one

```
# project/__init__.py
from flask import Flask


def create_app()
    from . import models, routes, services
    app = Flask(__name__)
    models.init_app(app)
    routes.init_app(app)
    services.init_app(app)
    return app
```

Creating web application

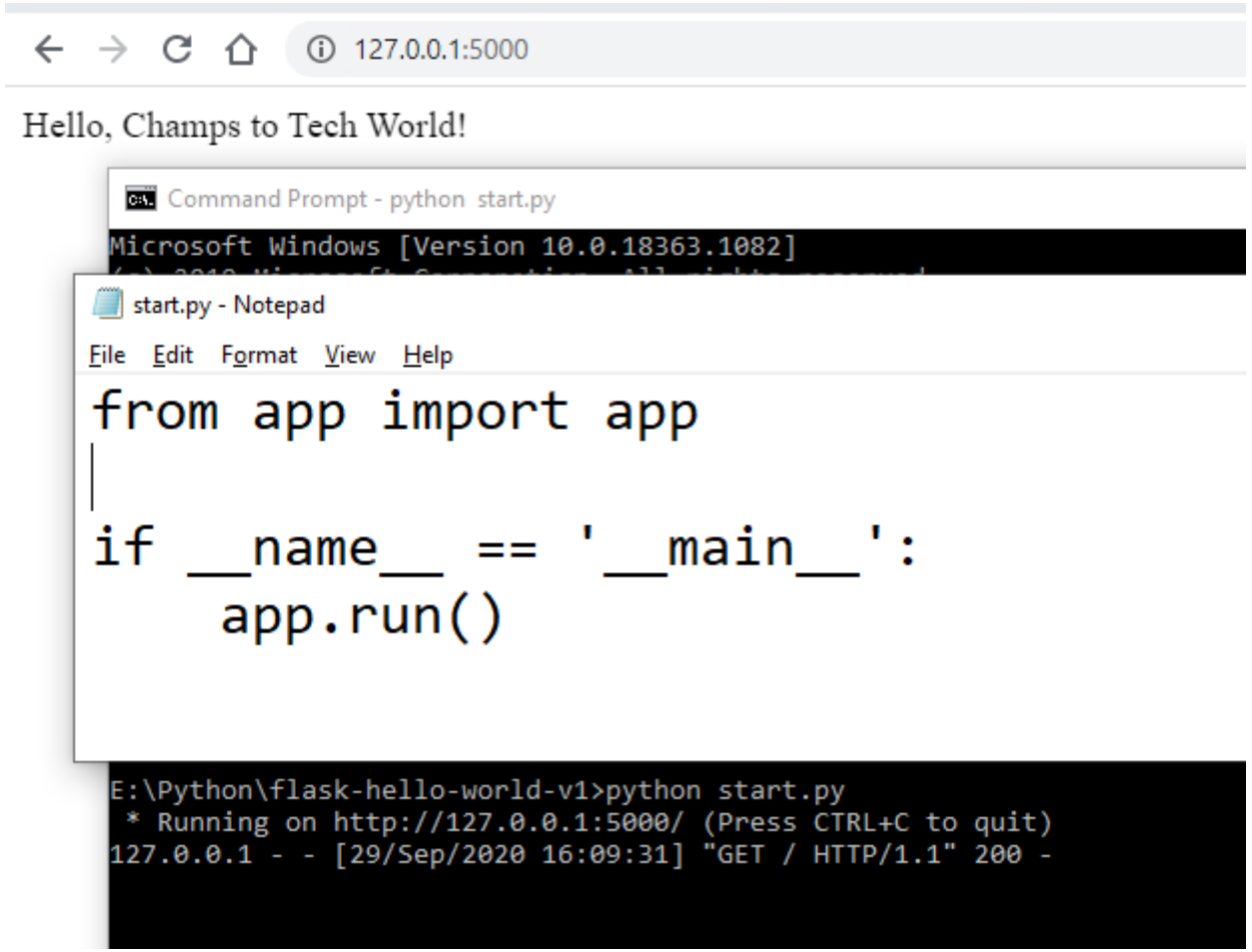
```
app/
    __init__.py
    templates/
    models/
    controllers/
    (or other names)
```

- The __name__ variable passed to the Flask class is a Python predefined variable, which is set to the name of the module in which it is used.
- The application then imports the routes module,

```
from flask import Flask

app = Flask(__name__)

from app import routes
```

- The app package is defined by the app directory and the __init__.py script, and is referenced in the from app import routes statement.
- The app variable is defined as an instance of class Flask in the __init__.py script, which makes it a member of the app package.

```
from app import app

@app.route('/')
@app.route('/index')
def index():
    return "Hello, World!"
```


127.0.0.1:5000
Hello, Champs to Tech World!
Command Prompt - python start.py
Microsoft Windows [Version 10.0.18363.1082]
start.py - Notepad
File Edit Format View Help
from app import app
if __name__ == '__main__':
app.run()
E:\Python\flask-hello-world-v1>python start.py
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [29/Sep/2020 16:09:31] "GET / HTTP/1.1" 200 -