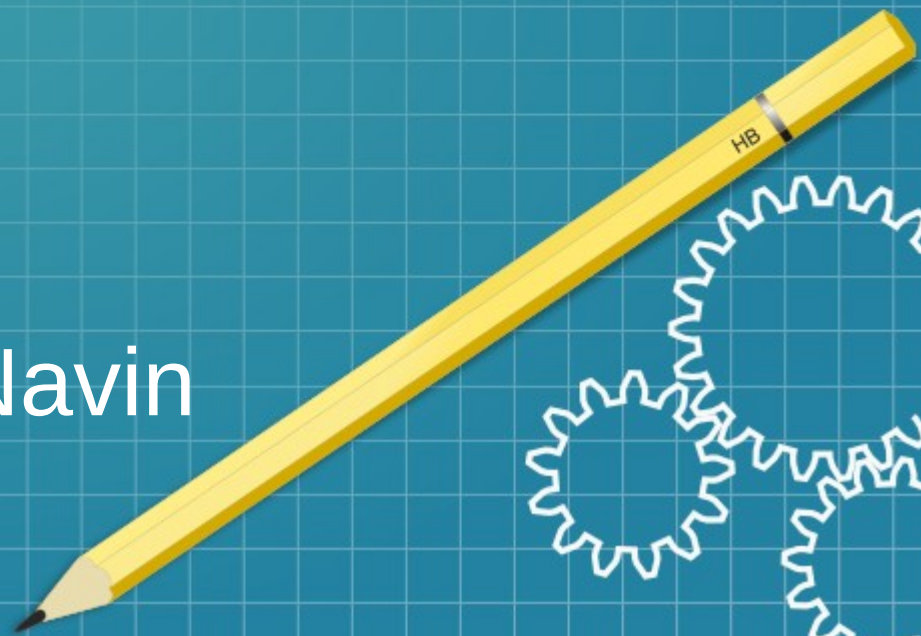
Python Programming
on
Web Application development



- Navin

Overview

- Introduction on Python
- Web Programming
- Python Web Frameworks
  - Flask
  - Django
- Demo
  - Simple Server
  - Simple Http request using form & ajax
  - Simple Server Database
  - Session Management

Python Programming

- Python is a general-purpose language.

Facts : **Why it is created ?**
In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python

Facts : **Why Python name?**
It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

Reasons to Choose Python as First Language

- Simple Elegant Syntax
- Not overly strict –
  No defining types and no semicolon
  But requires proper indentation
- Expressiveness of the language – easier to implement functionality
- Great Community and Support

PYTHON 2 vs 3

2018 DIFFERENCES

- Short version:

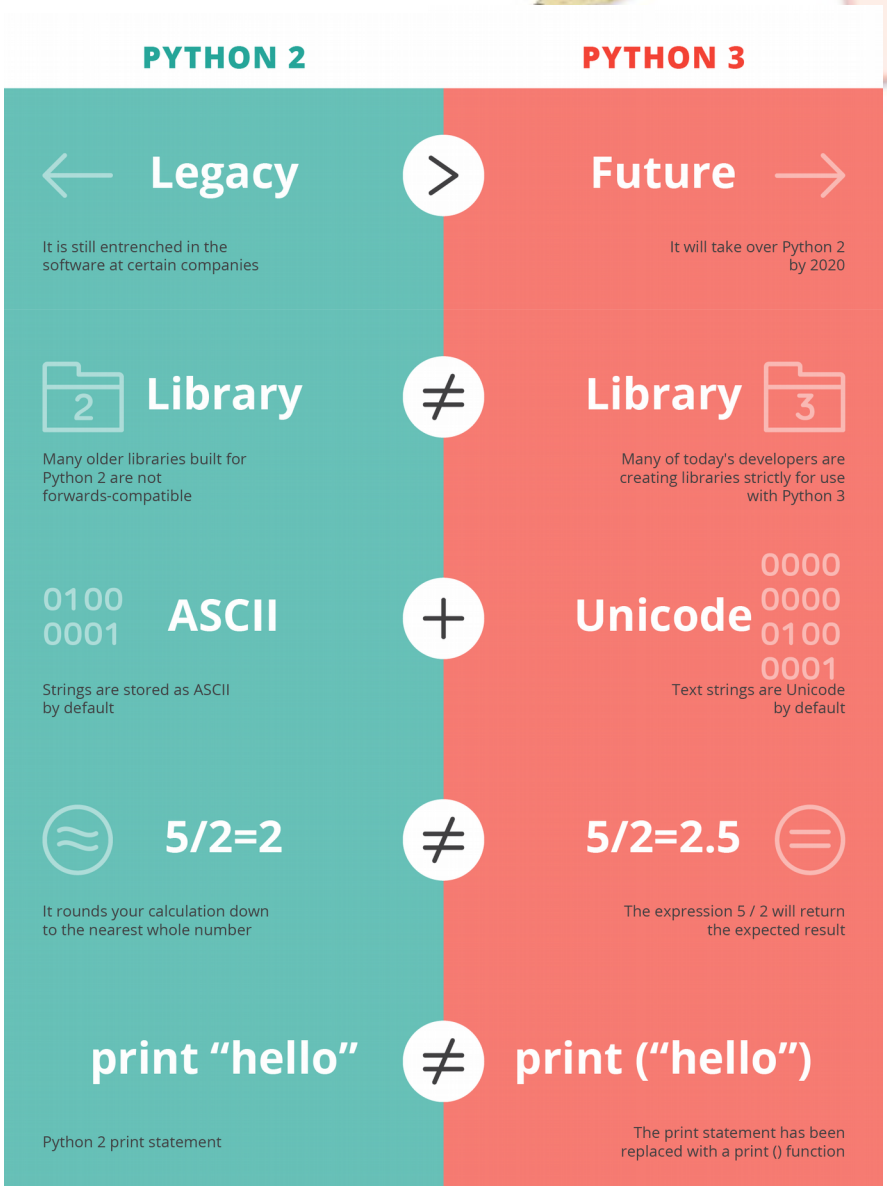
  Python 2.x is legacy
  Python 3.x is the present and future of the language

- Stable Version:

  Python2 – 2.7
  Python3 – 3.6

- __Future__ module

  Python 3.x introduced some Python 2-incompatible keywords and features that can be imported via the in-built __future__ module in Python 2.



Getting Started

Installation:

- Python :
  Visit : https://www.python.org/downloads/
  Download Python 2.7 version

  Note : For Ubuntu User, Python will be in-built package. To check version please enter following command
  $ python --version

- IDE (interactive development envirnoment):
  Any IDE is suitable – Sublime Text, VS Code, gedit

  $ sudo add-apt-repository -y ppa:webupd8team/sublime-text-2
  $ sudo apt-get update
  $ sudo apt-get install sublime-text

  Note : For Window User, IDLE program will be installed during python installation.

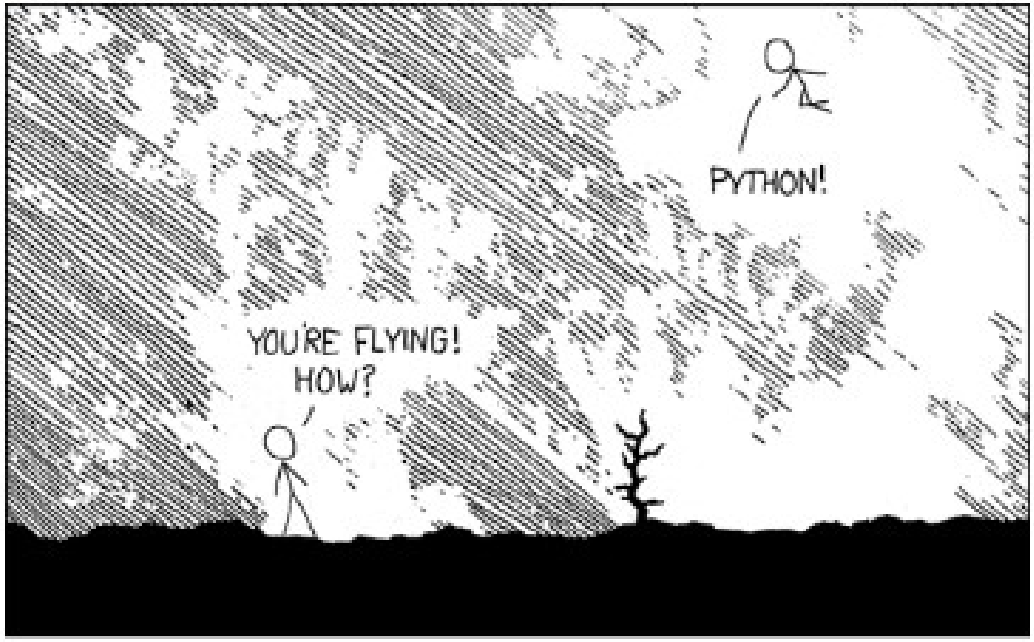
Check if Python Works

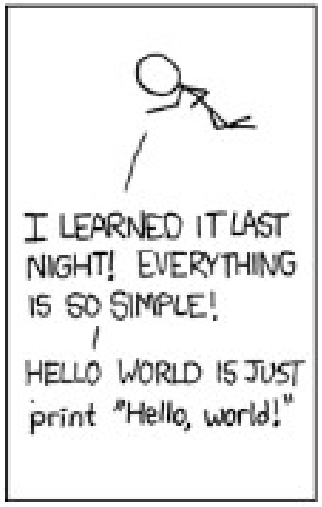
```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

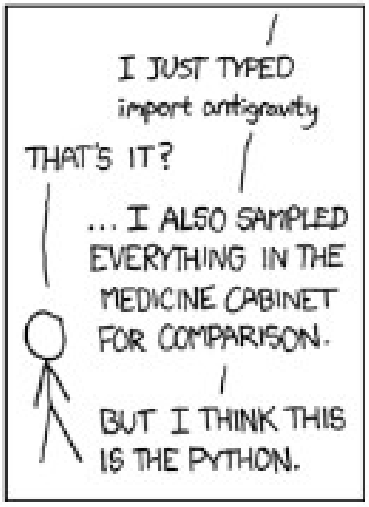
```
>>> import __hello__
Hello World...
```

More Facts on Python

```
import antigravity
```







Import Antigravity ->

Redirect to
http://xkcd:333

Web Programming

- Web programming, also known as web development, is the creation of dynamic web applications.

Is it best form of Coding?

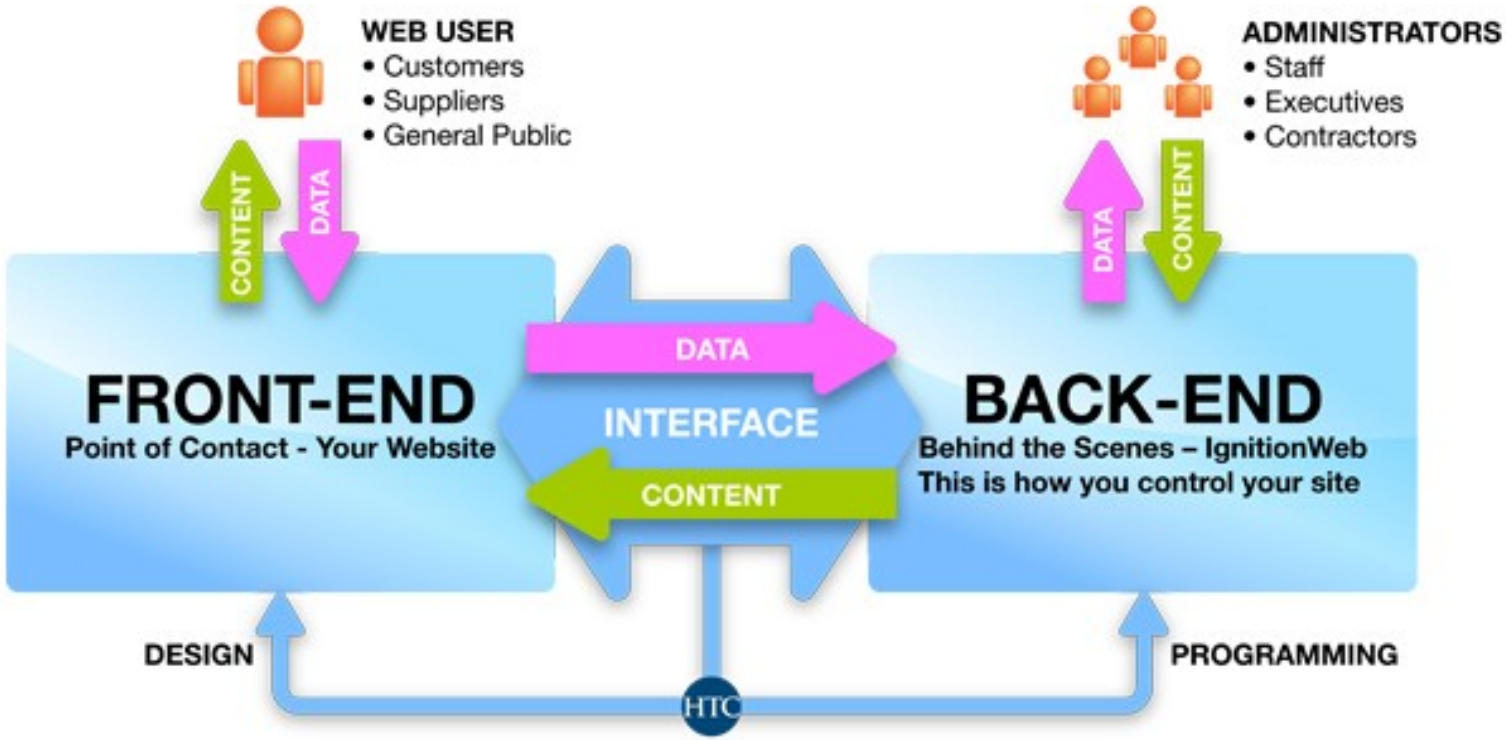
- It’s easy to set up
- Instant Results
- Online Training – Availability of resource

- You can start your buisness – by giving your idea to an visualization , an User Perspective – User Interface

  Example : Facebook – web coding

Divisions Web Development

- Front-end development - client-side development .
- Back-end development - server-side development .



Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

Why Python web framework

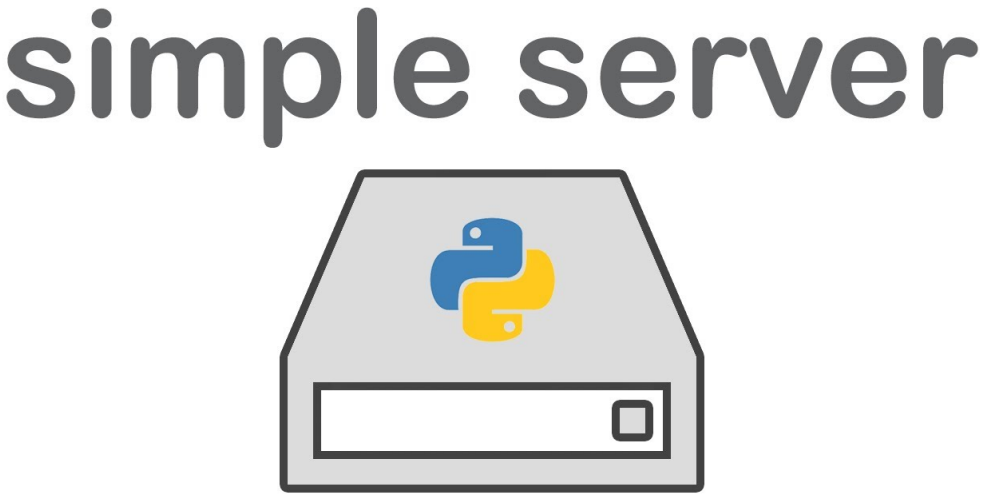
- Easy to learn - Hides complicated low-level details and requires lesser coding
- Acts as foundation to learn other languages
- Perfect to build prototypes
- Interoperability with Other Programming Languages
  eg . Cpython - C & IronPython - .NET
- Used and trusted by many large companies :
  Pyramid and Django are used by companies like Bitbucket, Pinterest, Instagram and Dropbox in their application

Creating a Simple Server

$ python -m SimpleHTTPServer

-m : run library module as a script

Create a simple server at default 8080 port



Python Framework

Choosing Framework

Flask	Django	Pyramid
Flask	Larger application	Larger application, targetting flexibility tools developer choice
In built routing and authenication	In built templating, forms, routing, authentication, basic database administration, and many more	In built routing and authentication, but templating and database administration require external libraries.
Mid 2010	First release 2005, got name on 2010	First release 2006
5,000+ StackOverflow questions 10,900+ github stars	Most active community 80,000+ StackOverflow questions 11,300+ github stars	Smaller community

Installation of framework

- Flask:
  pip install flask
- Django:
  pip install Django
- Pyramid:

  pip install pyramid pyramid_ipython

What is this pip

- Pip is a package manager for Python packages, or modules if you like.
- Packages are usually installed from the Python Package Index (PyPI) repository of software for the Python programming
- Replacement for easy_install

```
pip install <package-name>
```

Django – Getting Started

$ django-admin startproject hello_django
$ django-admin startapp App

```
hello_django
├── hello_django
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── App
│   ├── admin.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
└── manage.py
```

Note : pip will install django v1.11 by default

Pyramid – Getting Started

$ pcreate -s starter hello_pyramid

```
hello_pyramid
├── CHANGES.txt
├── development.ini
├── MANIFEST.in
├── production.ini
├── hello_pyramid
│   ├── __init__.py
│   ├── static
│   │   ├── pyramid-16x16.png
│   │   ├── pyramid.png
│   │   ├── theme.css
│   │   └── theme.min.css
│   ├── templates
│   │   └── mytemplate.pt
│   ├── tests.py
│   └── views.py
├── README.txt
└── setup.py
```

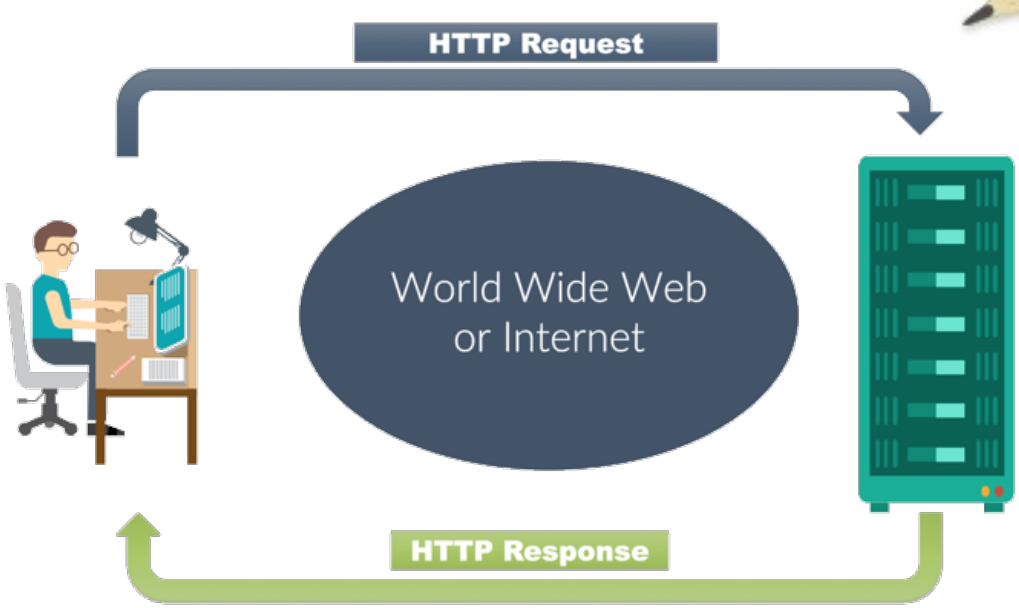
Flask - Getting Started

```python
# from http://flask.pocoo.org/ tutorial
from flask import Flask
app = Flask(__name__)

@app.route("/") # take note of this decorator syntax, it's a common pattern
def hello():
    return "Hello World!"

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

Jinga2

- Flask uses jinga2 template engine. A web template contains HTML syntax interspersed placeholders for variables and expressions (in these case Python expressions) which are replaced values when the template is rendered.
- The Jinga2 template engine uses the following delimiters for escaping from HTML.
    - {% ... %} for Statements
    - {{ ... }} for Expressions to print to the template output
    - {# ... #} for Comments not included in the template output
    - # ... ## for Line Statements

HTTP (HyperText Transfer Protocol) Model



Method	Meaning
GET	Read data
POST	Insert data
PUT or PATCH	Update data, or insert if a new id
DELETE	Delete data

Flask – simple AJAX request

Step 1 : Creating Flask instance with file location

```
app = Flask(__name__,static_url_path='files',static_folder='web/static',template_folder='web/templates')
```

- static_url_path=‘‘ removes any preceding path from the URL (i.e. the default /static).
- static_folder='web/static' will tell Flask serve the files found at web/static.
- template_folder='web/templates', similarly, this changes the templates folder

Step2 : Creating Server and Routing

```
app.run(host='0.0.0.0',port=3000,debug = True) # create the server and host
@app.route("/",methods=['GET'] )   # re route the links
        def root():
```

Step3 : Creating Frontend and rendering

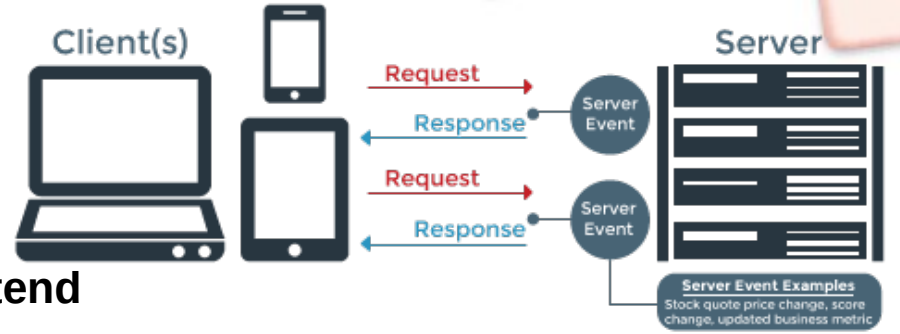
- render_template('index.html', data=data) #render the files with parameter
- app.send_static_file('index.html') #send file without parameter or compiling

Step 4 : Loading external files

<script src="{{ url_for('static', filename='index.js') }}"></script>

<script src="/static/index.js"> </script>

both pull the files from static folder



Step 5 : Manipulate python Object in frontend

- {{ item }} -> use the python object variable
- {% if .. %} -> operating on the python object

Step 6: HTTP request handling from client to Server

Client :

- Using HTML form
- Using HTTP request in javascript – AJAX,XML ,fetch

Step 7 : Data Handling in Server

Server :

from flask import request # handles the request

- request.data Contains the incoming request data as string in case it came with a mimetype Flask does not handle.
- request.args: the key/value pairs in the URL query string
- request.form: the key/value pairs in the body, from a HTML post form, or JavaScript request that isn't JSON encoded
- request.files: the files in the body, which Flask keeps separate from form. HTML forms must use enctype=multipart/form-data or files will not be uploaded.
- request.values: combined args and form, preferring args if keys overlap

For URL Query parameter, use request.args

- search = request.args.get("search")

For Form input, use request.form

- email = request.form.get('email')

For data type application/json, use request.data

data in string format and you have to parse into dictionary

- data = request.data
- dataDict = json.loads(data)

Step 8: Sending the response to Client.

Jsonify -> send the data in json format

or you can render a new html

Step 9 : Process the Data in the Client

- Processing in the Success Event

Try the Same Example with jQuery, JSON and perform AJAX request

Intercepting the data

- **Postman :**

  Postman makes API development faster, easier, and better. The free app is used by more than 3.5 million developers and 30,000 companies worldwide. Postman is designed with the developer in mind, and packed with features and options.

- **Advanced REST client :**
  A better API testing tool!
  Save your time with the easiest API testing tool out there. No complicated forms and scripts. Easy to use yet very powerful.

HTTP Status Code

To know more :https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

1xx: Informational - Request received, continuing process
2xx: Success - The action was successfully received, understood, and accepted
3xx: Redirection - Further action must be taken in order to complete the request
4xx: Client Error - The request contains bad syntax or cannot be fulfilled
5xx: Server Error - The server failed to fulfill an apparently valid request

200	Success
400	Bad Request
500	Internal Server Error
404	Not Found

Connecting Database

pip install flask_sqlalchemy
pip install sqlalchemy,flask-wtf

Step 1: Configure the database

- app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test1.db'
- app.config['SECRET_KEY'] = "random"
- from flask_sqlalchemy import SQLAlchemy
- db = SQLAlchemy(app)

Step 2: Create a model class to access ORM

- class Items(db.Model):
  .....

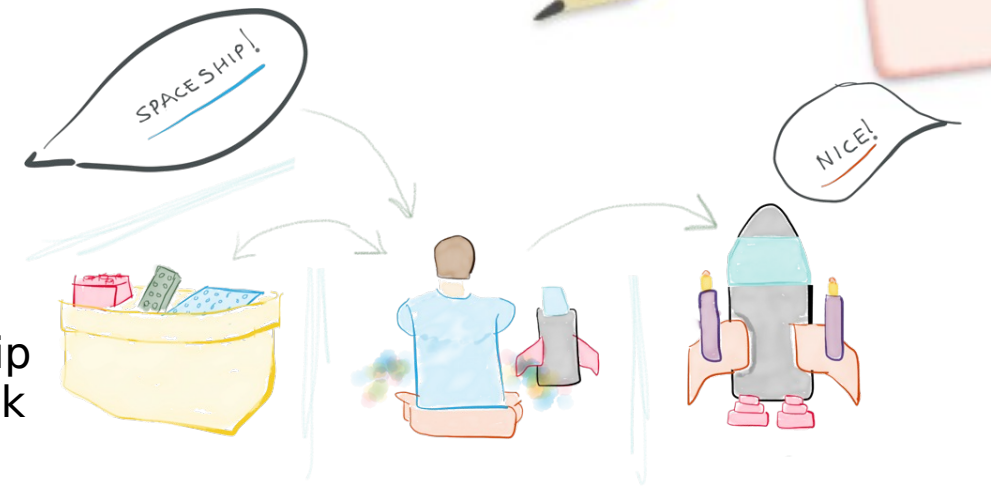
Step 3 : Do the CRUD operation in the database

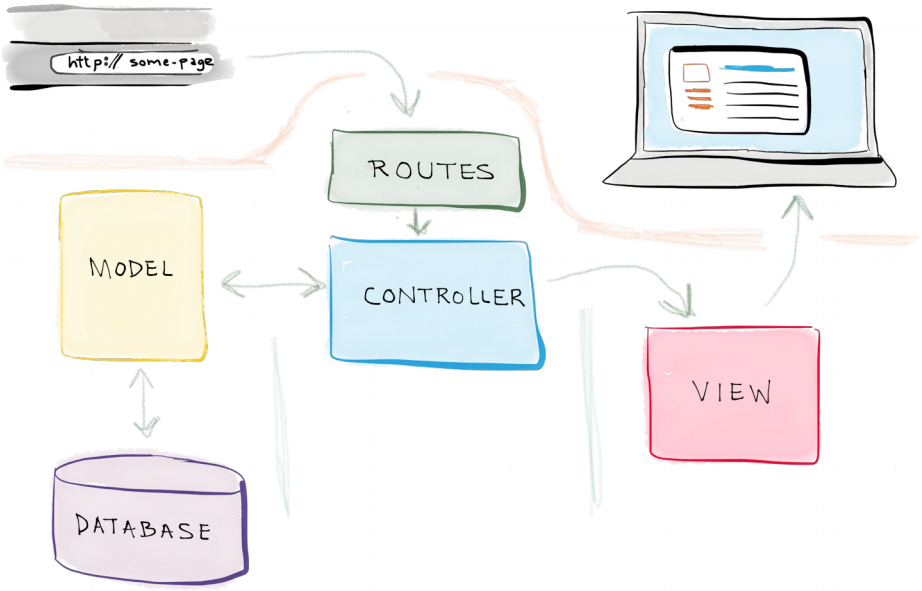
- db.session.add()
- db.commit()
- Items.query.all()

MVC - Architecture

Example :

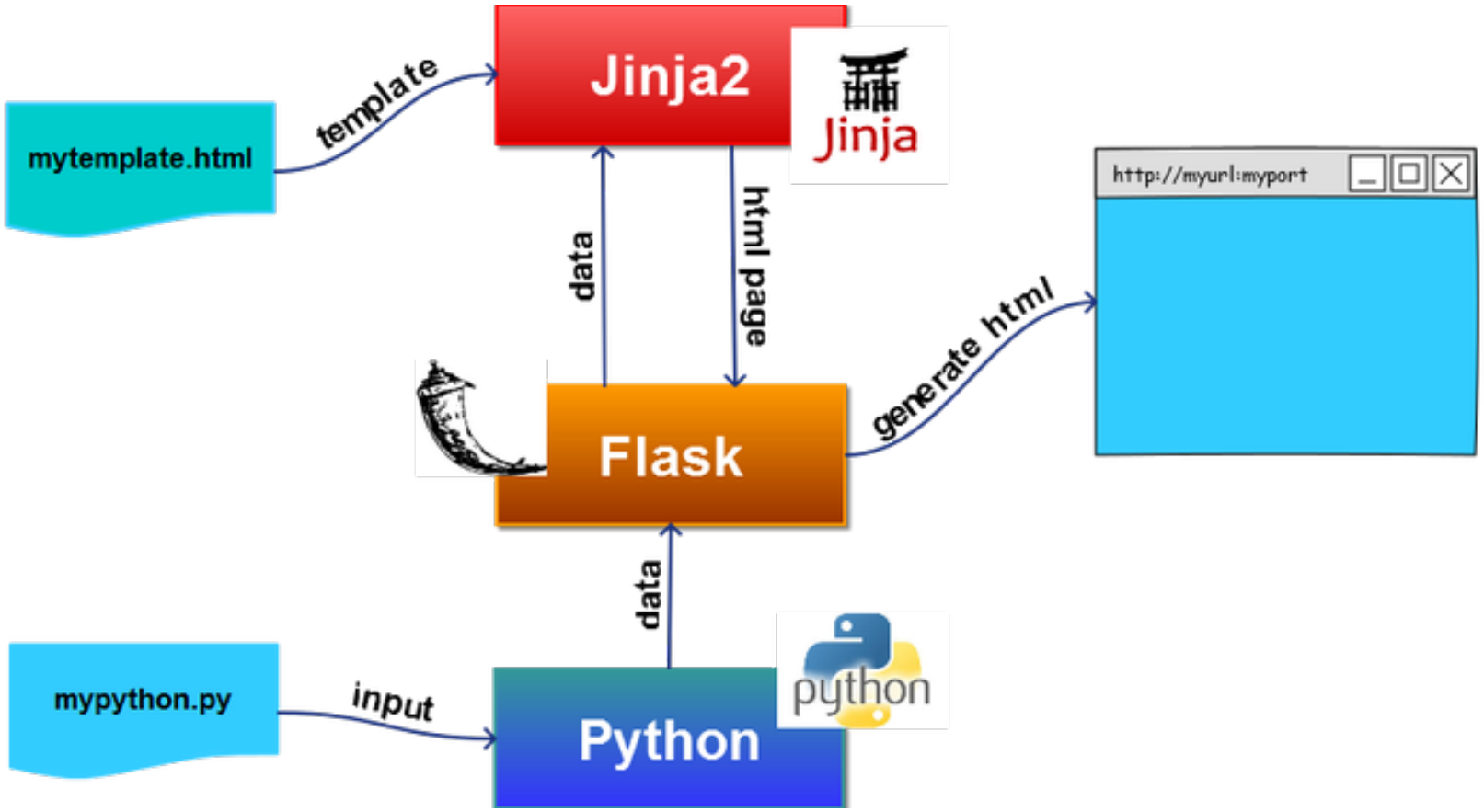
- Your brother makes a request that you build a spaceship.
- You receive the request.
- You retrieve and organize all the Legos you need to construct the spaceship.
- You use the Legos to build the spaceship and present the finished spaceship back to your brother.





- A user requests to view a page by entering a URL.
- The Controller receives that request.
- It uses the Models to retrieve all of the necessary data, organizes it, and sends it.
- View, which then uses that data to render the final webpage presented to the the user in their browser.

Flask Architecture - MTV



Session Management

Step 1 : Login Page

- Collect User & Password from the user and give to server

- Step 2 : Validate & Set Cookies
  - resp.set_cookie('userID', request.form['username'],max_age=10,expires=expire_date)

    It sets the cookie with expire timing or with Max seconds
- Step 3 : Redirect to Success Page

  redirect(url_for('user'))
- Step 4 : Display Error
- Step 5 : Providing Logout

  It deletes the cookies
  - request.cookies.pop('userID', None)

Web Storage

LocalStorage	5MB/10MB storage It's not session based, need to be deleted via JS or manually Client side reading only Less older browsers support
SessionStorage	5MB storage It's session based and working per window or tab Client side reading only Less older browsers support
Cookie	4KB storage Expiry depends on the setting and working per window or tab Server and client side reading More older browsers support

Web Authentication Methods Explained

- **HTTP Basic authentication :**
  - the client to provide a username and a password when making a request.
  - client has to send the Authorization header along with every request

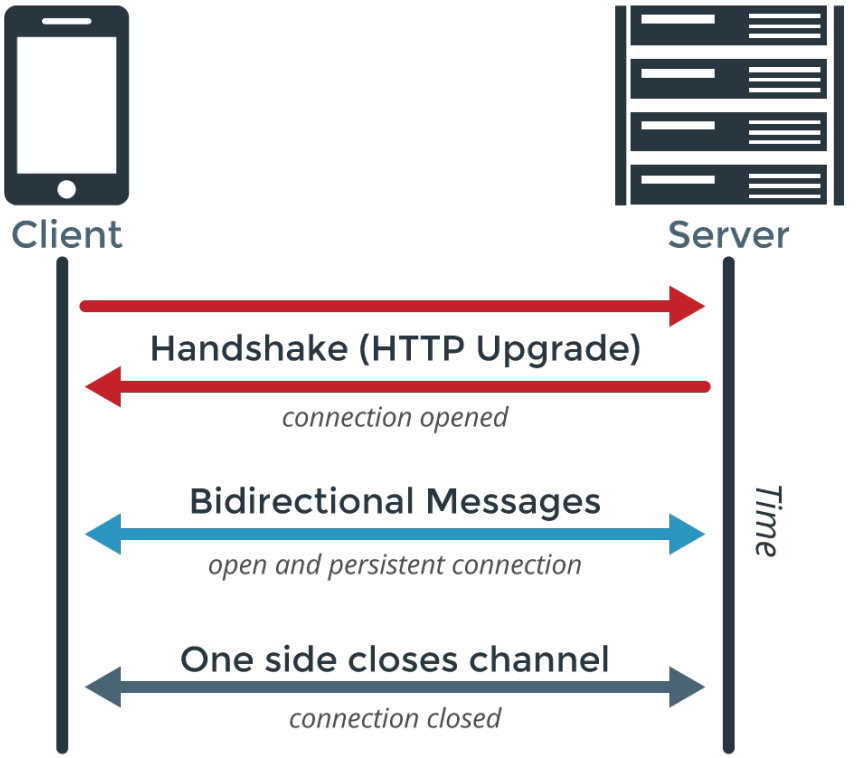
    eg. Authorization: Basic am9objpzZWNyZXQ=
  - Username:password is encoded with Base64

- **Cookies**
  - When a server receives an HTTP request in the response, it can send a Set-Cookie header. The browser puts it into a cookie jar, and the cookie will be sent along with every request made to the same origin in the Cookie HTTP header.
  - Always use HttpOnly /signed cookies

- **Tokens (JWT - JSON Web Token)**
  - JWT consists of three parts:
    - Header, containing the type of the token and the hashing algorithm
    - Payload, containing the claims
    - Signature, which can be calculated by HMAC SHA256
  - eg. Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
- **Signatures :**
  - When a consumer of an API makes a request it has to sign it, meaning it has to create a hash from the entire request using a private key
  - even if the transport layer gets compromised, an attacker can only read your traffic, won't be able to act as a user
  - eg. AWS
- **OTP (One Time Password) :**
  - One-Time passwords algorithms generate a one-time password with a shared secret and either the current time or a counter:
  - Time-based One-time Password Algorithm, based on the current time,
  - HMAC-based One-time Password Algorithm, based on a counter.

WebSockets

- WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection
- WebSocket is a different protocol from HTTP.



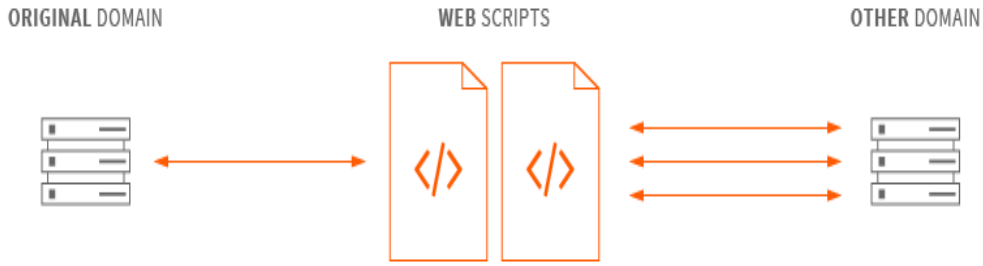

```
$ pip install flask_socketio
```

CORS - Cross-origin resource sharing

- Cross-origin resource sharing (CORS) is a standard for accessing web resources on different domains. CORS allows web scripts to interact more openly with content outside of the original domain, leading to better integration between web services

The following are the new HTTP headers added by the CORS standard:

- Access-Control-Allow-Origin
- Access-Control-Allow-Credentials
- Access-Control-Allow-Headers
- Access-Control-Allow-Methods
- Access-Control-Expose-Headers
- Access-Control-Max-Age
- Access-Control-Request-Headers
- Access-Control-Request-Method
- Origin



$ pip install -U flask-cors

Example :

```
from flask_cors import CORS
cors = CORS(app)
cors = CORS(app, resources={r"/api/*": {"origins": "*"}})
Or
@cross_origin()
```

Limitations of Python Web framework

- Slow speed – because python is an interpreted language eg. Cpython is slower than C
- Weak in Mobile Computing
- Designs issues – more testing
- Less developed database access layers

Web Development Trends in 2018

- Accelerated Mobile Pages (AMP) – better conversion rate
- Progressive Web Applications (PWA)
- Single Page Applications (SPA)
- Chatbots and proper online support
- Push notifications
- RAIL (Response, Animation, Idle, and Load)
- Static Site Generator
- Motion UI
- Real-time web apps

References

- https://pythonspot.com/web-dev/
- https://codeburst.io/the-2018-web-developer-roadmap-826b1b806e8d
- https://levelup.gitconnected.com/the-non-developers-guide-to-development-in-2018-7f023a2ff5e1
- https://realpython.com/tutorials/web-dev/
- https://scotch.io/bar-talk/processing-incoming-request-data-in-flask

Thank You