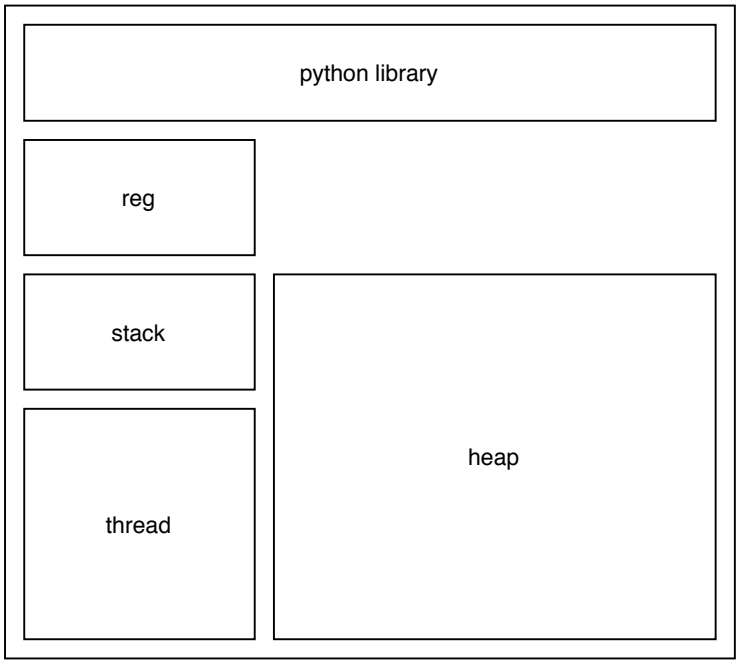
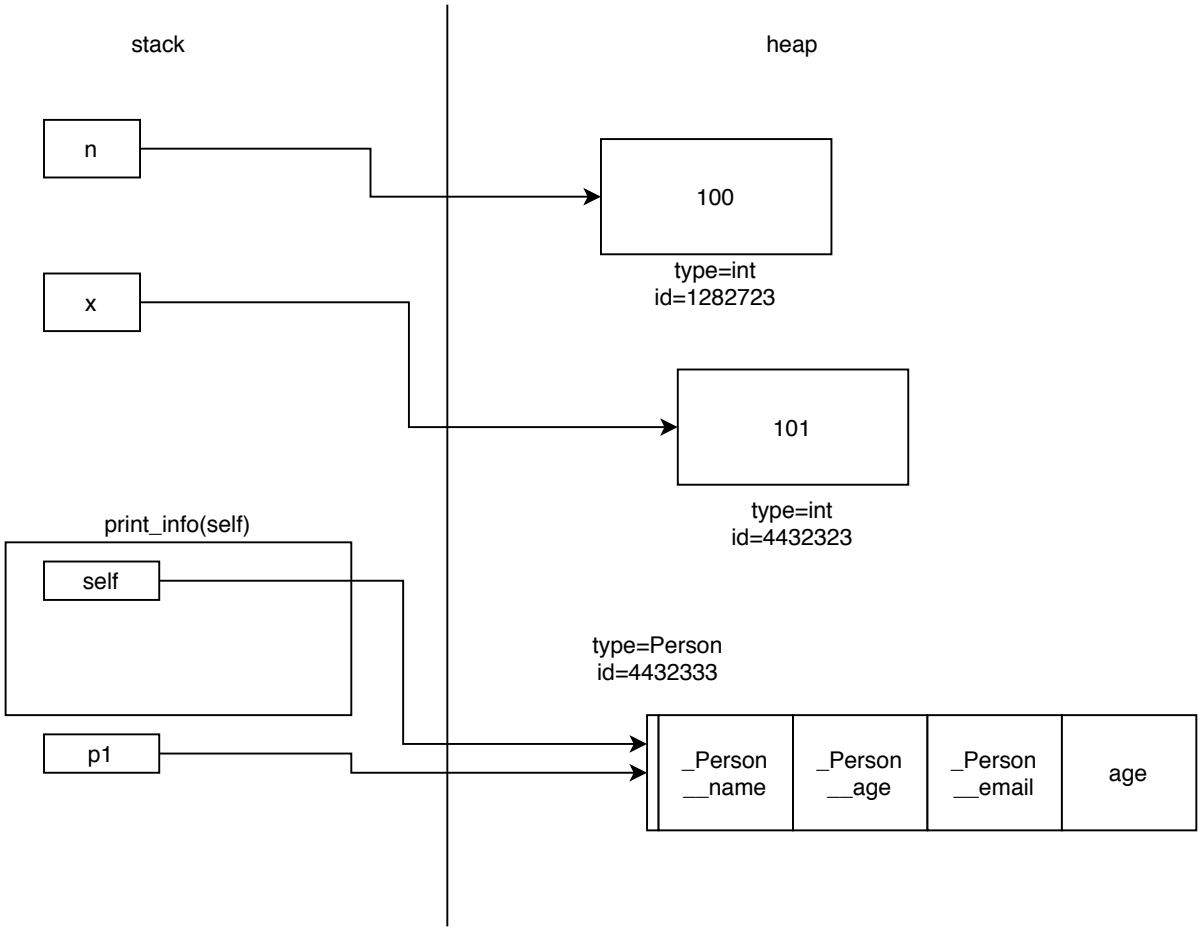
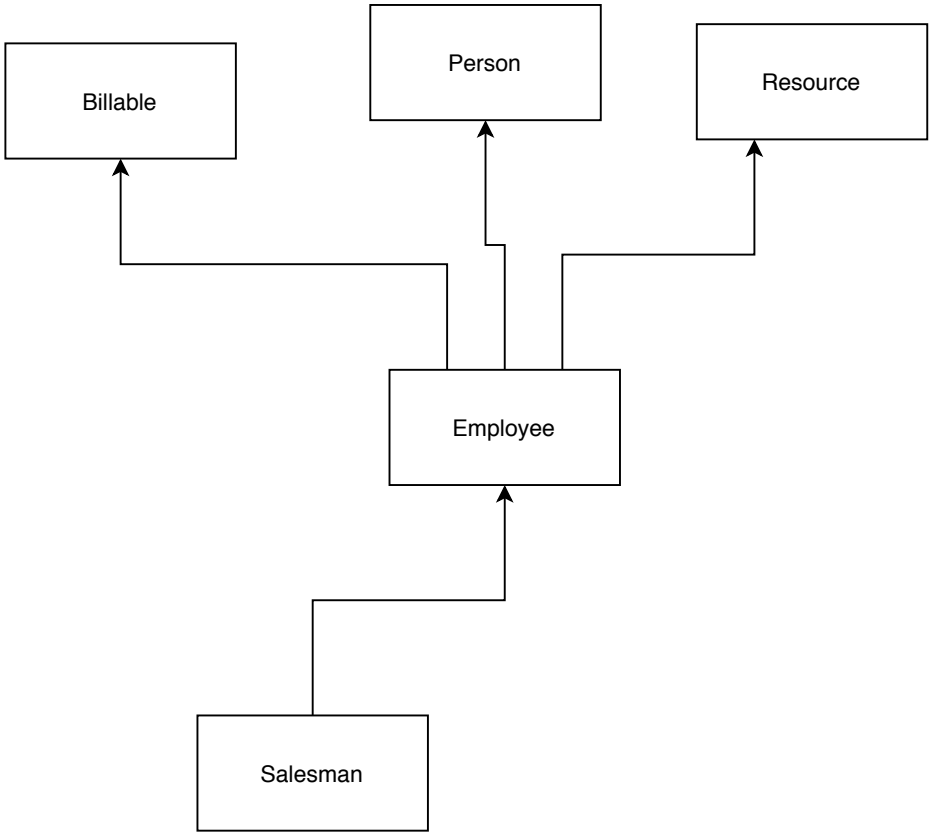
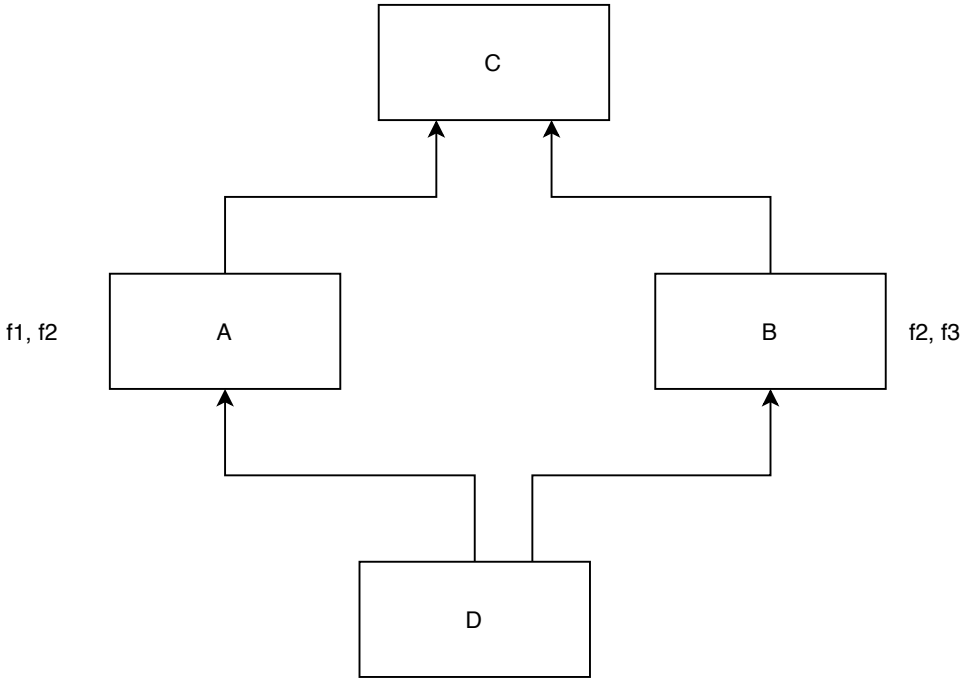
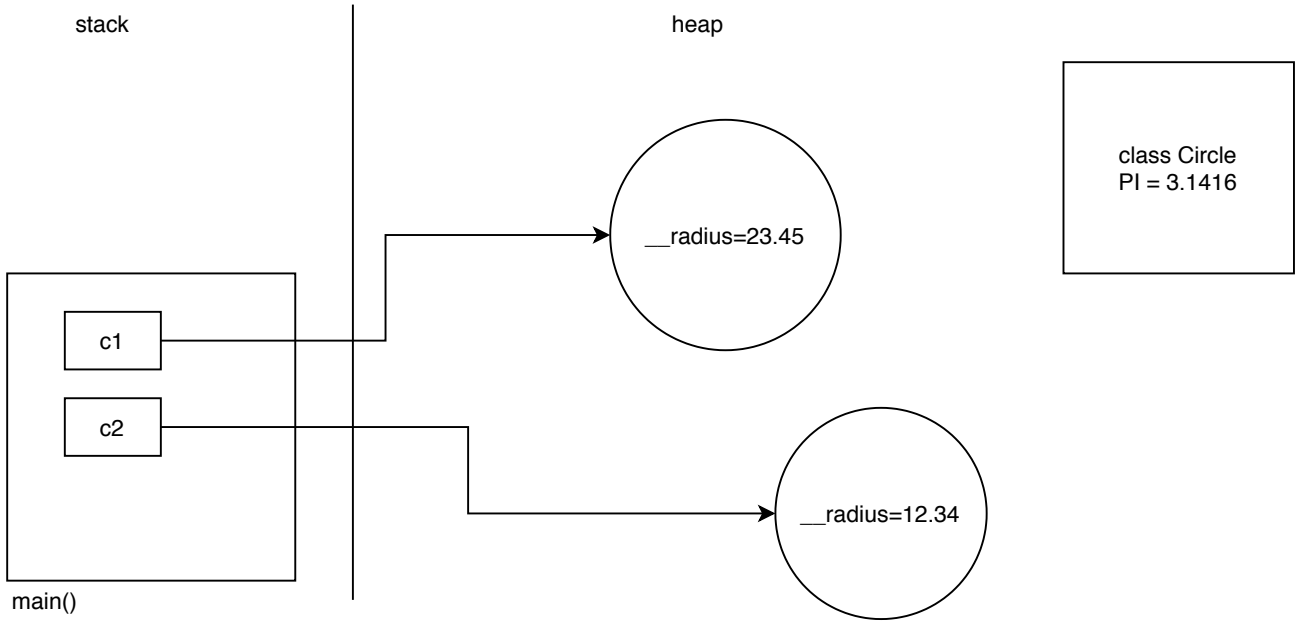

python library
reg
stack
thread
heap

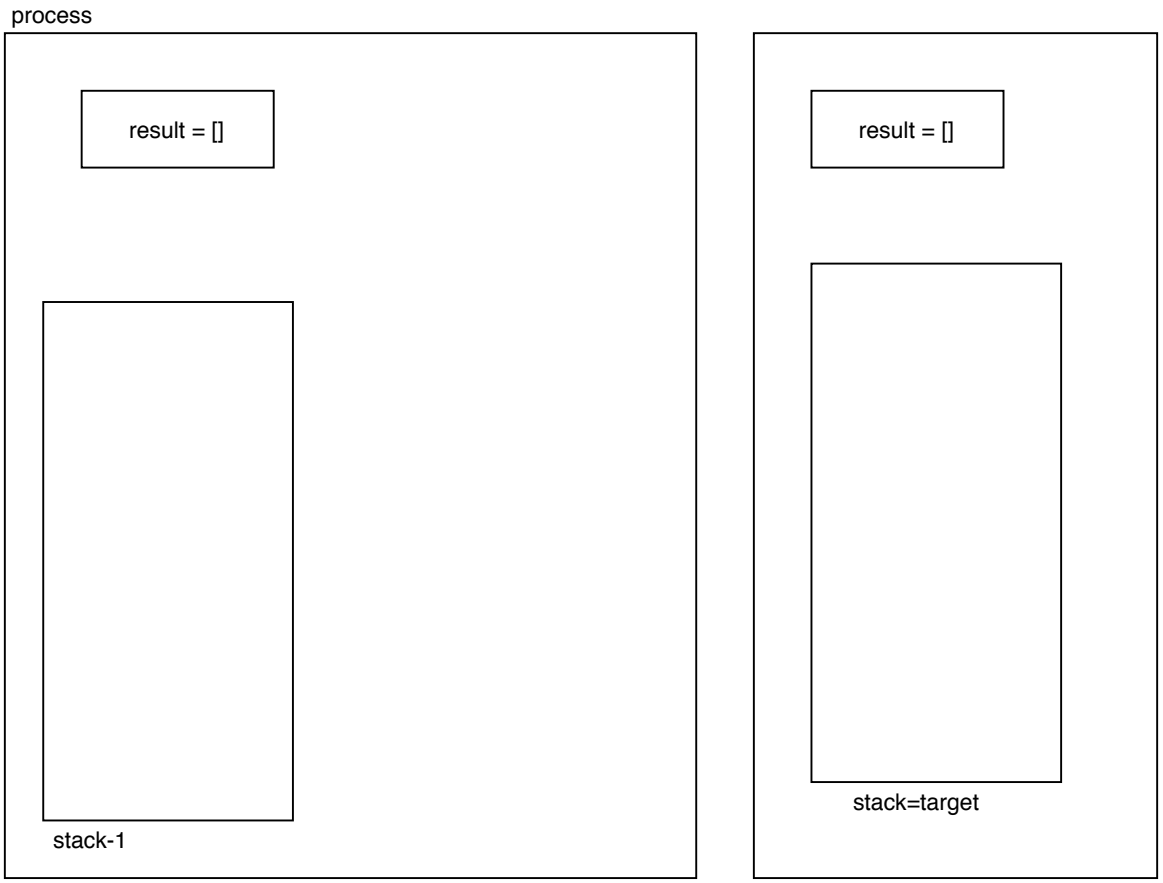

stack
heap
n
100
type=int
id=1282723
x
101
type=int
id=4432323
print_info(self)
self
type=Person
id=4432333
p1
_Person
__name
_Person
__age
_Person
__email
age

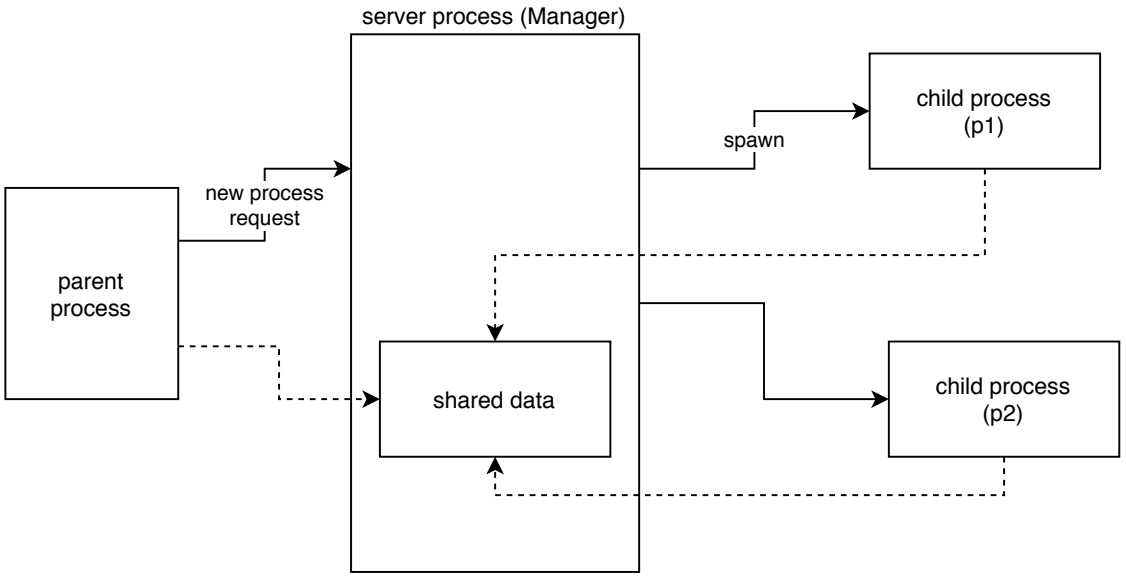


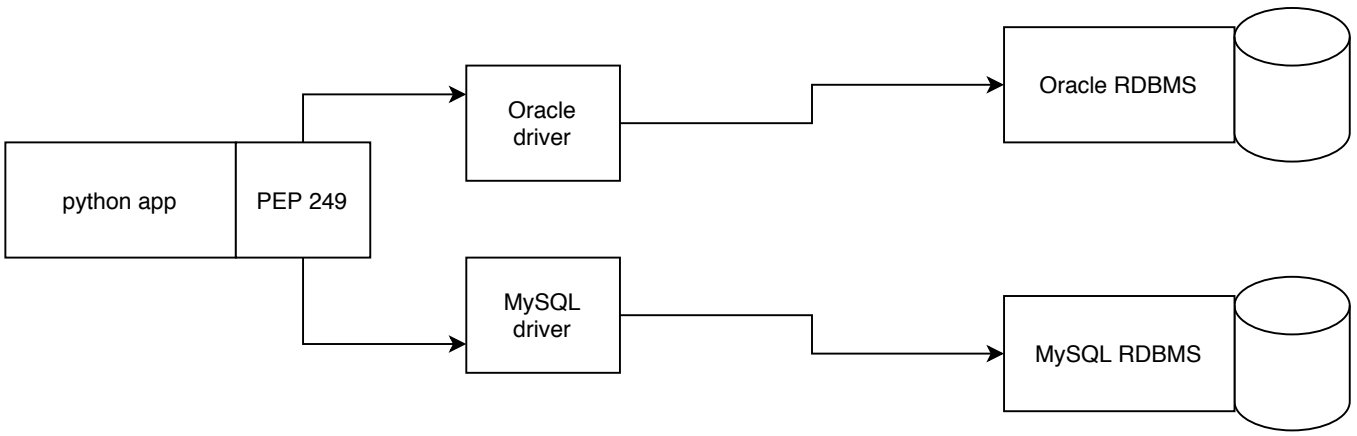


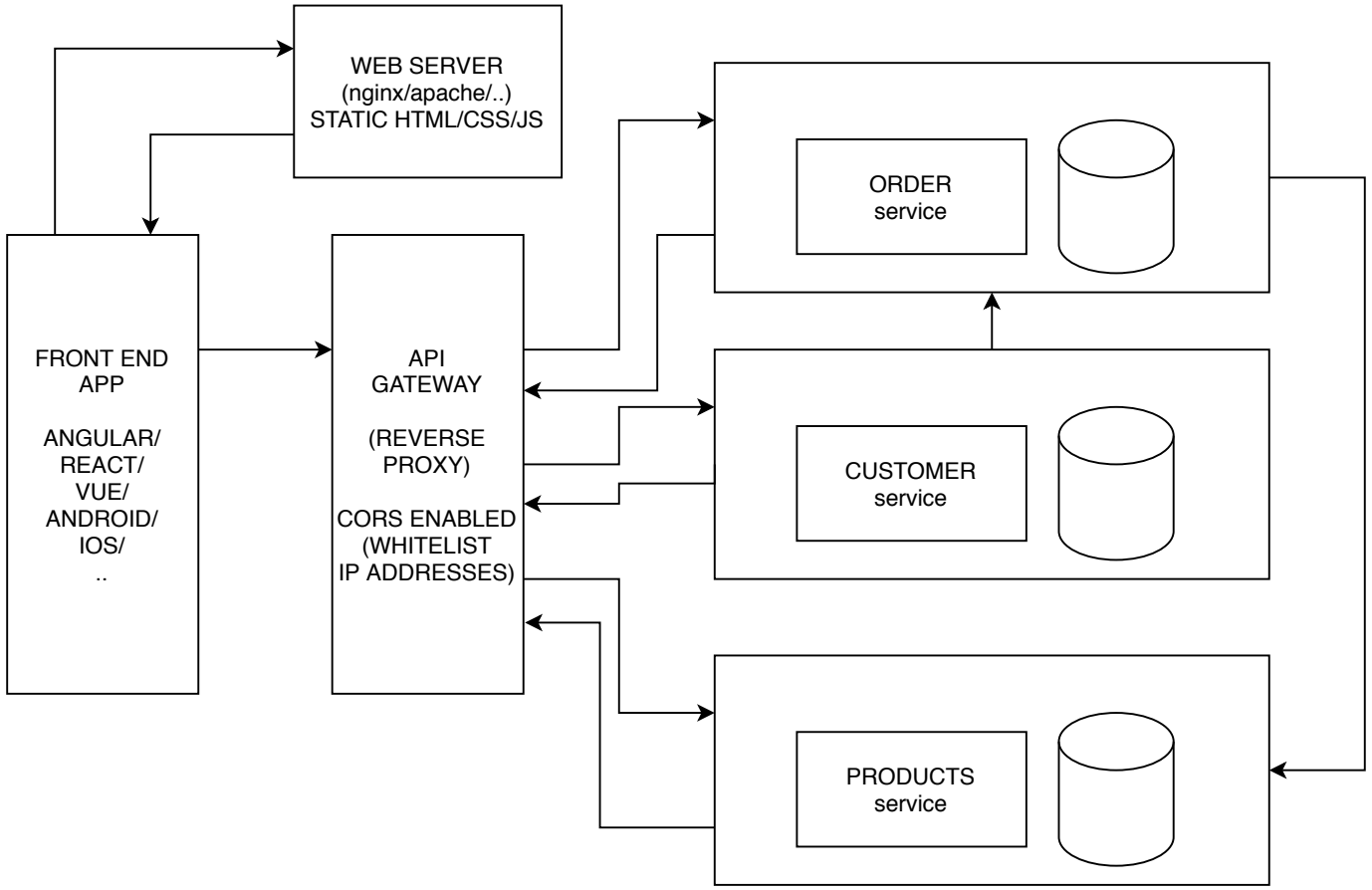
d1=D()
d1.f2() --> A.f2, B.f2 ??

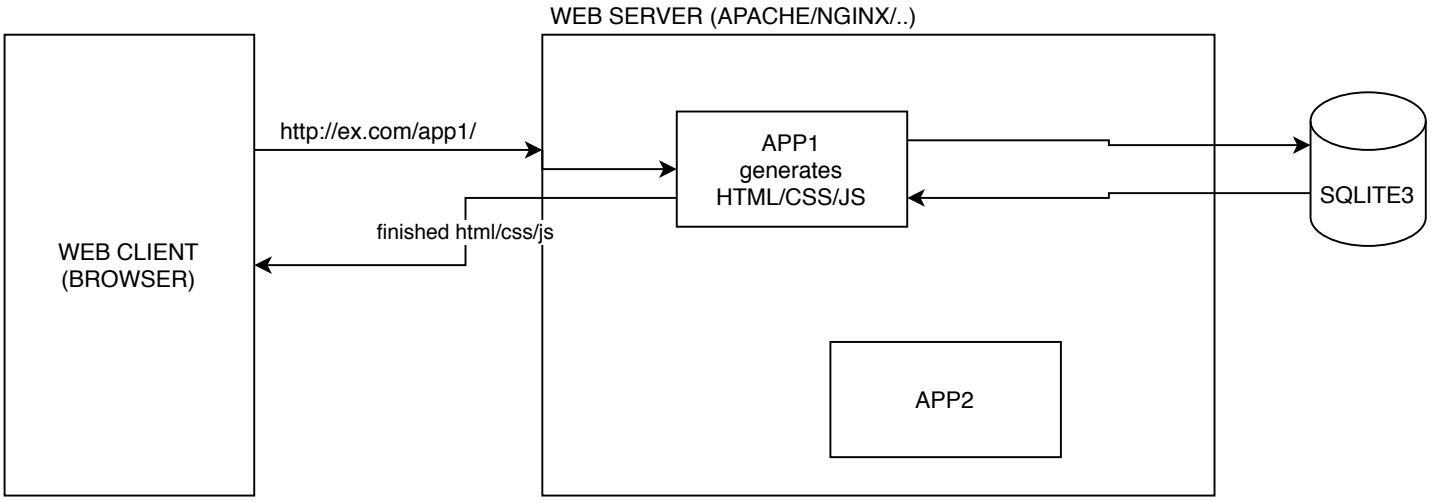

stack
heap
class Circle
PI = 3.1416
__radius=23.45
c1
c2
__radius=12.34
main()

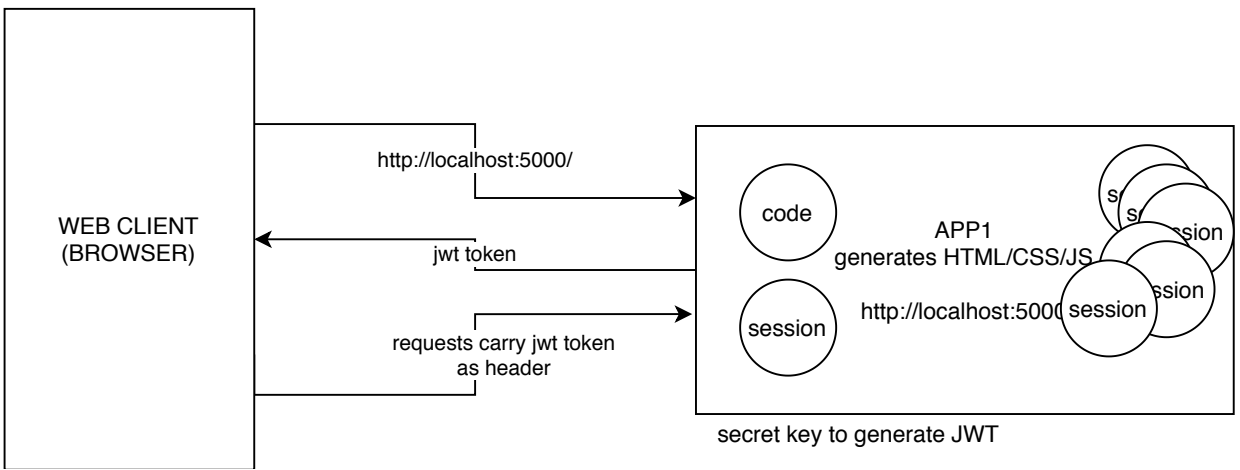

process
result = []
result = []
stack-1
stack=target


server process (Manager)
child process
(p1)
spawn
new process
request
parent
process
shared data
child process
(p2)


Oracle RDBMS
Oracle
driver
python app
PEP 249
MySQL
driver
MySQL RDBMS

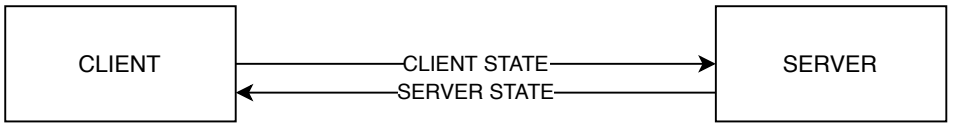

WEB SERVER
(nginx/apache/..)
STATIC HTML/CSS/JS
FRONT END
APP
ANGULAR/
REACT/
VUE/
ANDROID/
IOS/
..
API
GATEWAY
(REVERSE
PROXY)
CORS ENABLED
(WHITELIST
IP ADDRESSES)
ORDER
service
CUSTOMER
service
PRODUCTS
service


WEB SERVER (APACHE/NGINX/..)
WEB CLIENT
(BROWSER)
http://ex.com/app1/
finished html/css/js
APP1
generates
HTML/CSS/JS
SQLITE3
APP2


WEB CLIENT
(BROWSER)
http://localhost:5000/
jwt token
requests carry jwt token
as header
code
session
APP1
generates HTML/CSS/JS
http://localhost:5000
session
secret key to generate JWT

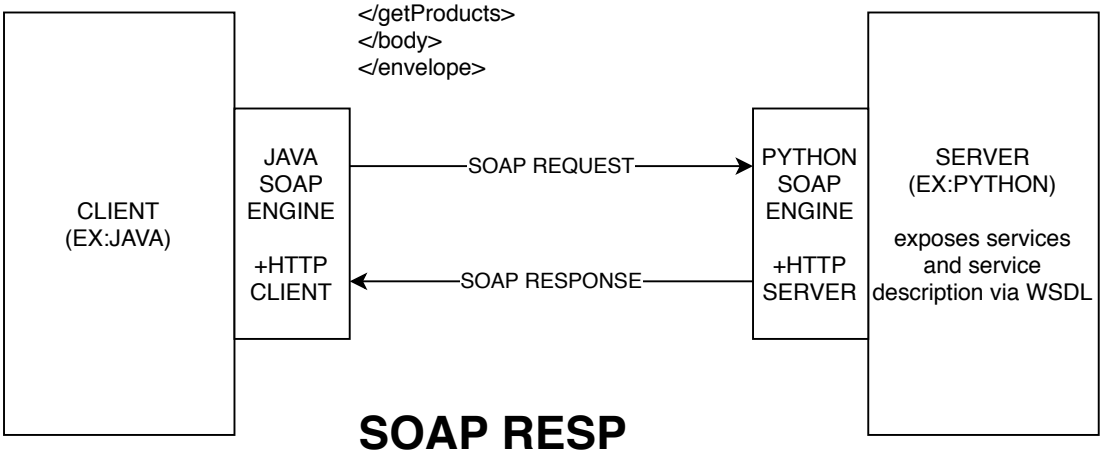
ReST

Representational (XML/JSON/..)
State (INFORMATION)
Transfer (HTTP/HTTPS --> GET/POST/PUT/DELETE/PATCH)



SOAP REQ

```
<envelope>
<header></header>
<body>
<getProducts>
<minPrice>10.0</minPrice>
<maxPrice>100.0</maxPrice>
</getProducts>
</body>
</envelope>
```



SOAP RESP

```
<envelope>
<header></header>
<body>
<getProductsResp>
<products>
<product id="1">
<name>xyz</name>
<price>15.0</price>
</product>
<product id="2">
<name>asdf</name>
<price>56.0</price>
</product>
</products>
</getProductsResp >
</body>
</envelope>
```

REST constraints:

1. Client/Server
2. Uniform interfaces (HTTP URI)
3. Stateless
4. Cacheable
5. Layered system (controller/service/dao/eis)
6. Code on demand (optional)

A resource is identifiable using URI (ex: http://localhost:8080/api/contacts)
An action is performed on a resource using HTTP verbs (ex: GET, POST, PUT, DELETE, PATCH)
Client and server can negotiate for the type of content (representation) using HTTP headers:
Accept --> client wants a specific representation (ex: application/json or text/html)
Content-Type --> client is sending (via POST/PATCH/PUT) a specific representation

CORS

Cross Origin Resource Sharing

URL --> scheme (protocol) + host + port + URI-segments

any of scheme/host/port is different ?? --> cors