[320] Web 1: Flask

Tyler Caraza-Harter

Review Regular Expressions

```
Which regex will NOT match "123"
1. r"\d\d\d"
2. r"\d{3}"
3. r"\D\D\D"
4. r"..."
```

What will r"^A" match?

- 1. "A"
- 2. "^A"
- 3. "BA"
- 4. "B"
- 5. "BB"

Which one can match "HH"? 1. r"HA+H"

- 2. r"HA+?H"
- 3. r"H(A+)?H"

Which string(s) will match r"^(ha)*\$"

- 1. ""
- 2. "hahah"
- 3. "that"
- 4. "HAHA"

What is the type of the following? re.findall(r"(\d) (\w+)", some_str)[0]

- 1.list
- 2. tuple
- 3. string

Web Communication

CS 301 Web Review...

https://tyler.caraza-harter.com/cs301/fall19/schedule.html

https://mediaspace.wisc.edu/channel/CS301+-+F19/129948022



Scenario: we want to access Jupyter on our virtual machine from our laptop



Scenario: we want to access Jupyter on our virtual machine from our laptop



Issue I: firewall may be blocking some ports (we disabled this in lab)



Issue 2: there might not be any process listening on port 2020

[127.0.0.1 means "localhost", the default]

Start command: python3 -m notebook --no-browser --ip=127.0.0.1 --port=2020



Issue 3: the process may only be listening for local (not external) requests

[0.0.0.0 means all IP addresses]

Start command: python3 -m notebook --no-browser --ip=0.0.0.0 --port=2020



Success: Jupyter is listening for all 2020 requests, and the firewall isn't blocking them!

Start command: python3 -m notebook --no-browser --ip=0.0.0.0 --port=2020



Start command: python3 -m notebook --no-browser --ip=0.0.0.0 --port=2020



Your Goal: build a web application for P3

DNS (Domain Name Service)



paying to register domain name is ~\$10-15 / year

HTTPS: Hypertext Transfer Protocol Secure



(Virtual Machine)

HTTPS: Hypertext Transfer Protocol Secure



paying to register SSL certificate for encryption name is ~\$5-10 / year



Static Pages Correspond to Files

IP address: 18.216.110.65



Static Pages Correspond to Files



Static Pages Correspond to Files





Dynamic Pages Generated by Code

IP address: 18.216.110.65



Templating: Add Dynamic Content to File

IP address: 18.216.110.65



Multi-File Pages



(laptop)



IP address: 18.216.110.65



(laptop)

computer 2 (Virtual Machine)

IP address: 18.216.110.65



computer I (laptop) computer 2 (Virtual Machine)

IP address: 18.216.110.65



(laptop)

IP address: 18.216.110.65



(laptop)

computer 2 (Virtual Machine)

IP address: 18.216.110.65



(laptop)

IP address: 18.216.110.65



It's hard to scrape this kind of table: requests.get("index.html") wouldn't work...

IP address: 18.216.110.65



It's hard to scrape this kind of table: requests.get("index.html") wouldn't work...

Summary: Key Web Concepts

IP address: identifier for a computer (or network card on computer) port number: identifier used to route to specific process on computer firewall: software to block certain requests, often for certain ports listening: process is ready to receive requests from an IP/port **DNS**: service for converting domains to IP addresses HTTPS: encrypted HTTP traffic so others can't watch traffic on WIFI, etc. static pages: pages that correspond to files on the server dynamic pages: pages generated on-the-fly by some Python code templating: insert dynamic content into certain places in a file **HTTP GET:** request to download data HTTP POST: request to upload data

Web Frameworks

Python Web Frameworks (and other packages)

Python web frameworks like Flask and Django make it easy to write functions for each webpage that can return a string with the contents.



Flask Example

Example Flask application (P3 starter code):

```
import pandas as pd
from flask import Flask, request, jsonify
app = Flask(__name__)
# df = pd.read_csv("main.csv")
@app.route('/') >> decorator
def home():
    with open("index.html") as f:
        html = f.read()
    return html
if __name__ == '__main__':
    app.run(host="0.0.0.0") # don't change this line!
```

demo!

Decorators

Decorators: take a function, return a function

@name before a function "decorates" a function



Useful for (a) making lists of certain types of functions, or (b) modifying functions

Example from Course Website

```
# decorator: user must authenticate to the admin user
      def admin(fn):
          EXTRA_AUTH[fn.__name__].append(admin_check)
          return fn
      # decorator: user must authenticate and have a valid email
      def user(fn):
          EXTRA_AUTH[fn.__name__].append(user_check)
          return fn
      # decorator: user must authenticate and be a grader
      def grader(fn):
          EXTRA_AUTH[fn.__name__].append(grader_check)
          return fn
@route
@admin
def put_roster(user, event):
   s3().write_cached_json("roster.json", json.loads(event['roster']))
    return (200, 'roster uploaded')
@route
@user
def roster_entry(user, event):
    email = user['email'].lower()
    parts = email.split("@")
    if parts[1] != "wisc.edu":
        return (500, 'not a wisc.edu email')
```

Example: Test Caller

```
# if @test(...) decorator is before a function, add that function to test_funcs
def test(points):
    def add_test(fn):
        tests.append(TestFunc(fn, points))
    return add_test
```

```
@test(points=8)
def has classes():
    points = 0
    for name in ["BusDay", "Location", "Stop", "Trip"]:
        if hasattr(bus, name) and type(getattr(bus, name)) == type:
            points += 2
        else:
            print("no class named "+name)
    return points
@test(points=20)
def service_ids():
    points = 0
    for i, day in enumerate([datetime(2020, 2, 21), datetime(2020, 2, 22)]):
        bd = qet day(day)
        service ids = sorted(bd.service ids)
        err = is_expected(actual=service_ids, name="service_ids:%d"%i)
        if err != None:
            print("unexpected service_ids for {}: {}".format(day, err))
            continue
        points += 10
    return points
```

Example: Invocation Counter

```
counts = \{\}
def count(fn):
    counts[fn.__name__] = 0
    def wrapper():
        counts[fn.__name__] += 1
        fn()
    return wrapper
@count
def f():
    print("running f")
@count
def g():
    print("running g")
f()
g()
g()
print(counts)
```

JSON-chat and other examples...