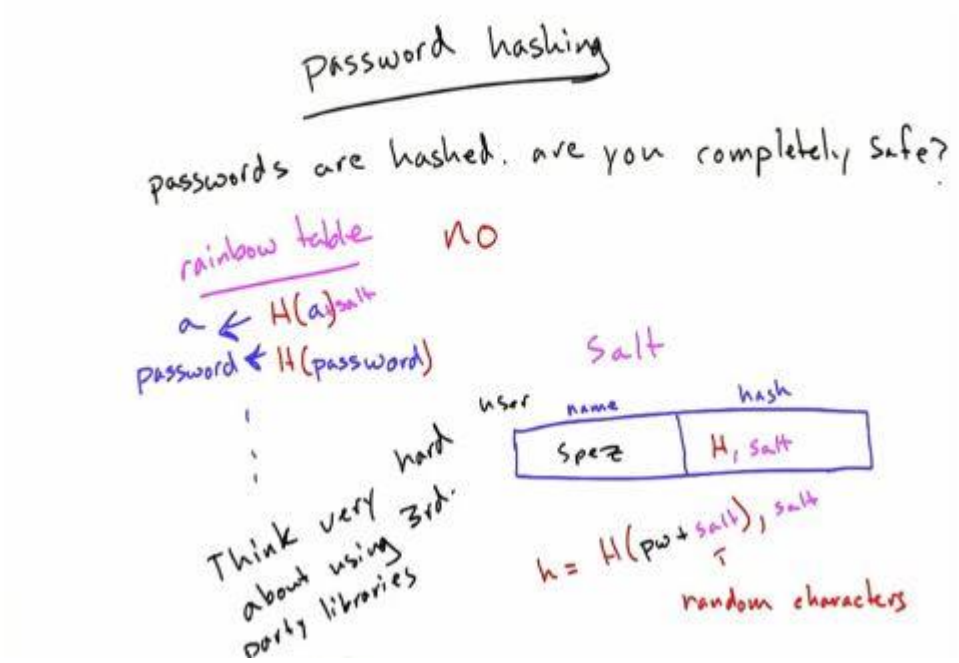
```python
import random
import string
import hashlib

def make_salt():
    return ''.join(random.choice(string.letters) for x in xrange(5))

# Implement the function valid_pw() that returns true if a user's password
# matches its hash. You mo need to modify make_pw_hash.

def make_pw_hash(name, pw, salt = None):
    if not salt:
        salt = make_salt()
    h = hashlib.sha256(name + pw + salt).hexdigest()
    return '%s,%s' % (h, salt)

def valid_pw(name, pw, h):
    salt = h.split(',')[1]
    return h == make_pw_hash(name, pw, salt)

h = make_pw_hash('spez', 'hunter2')
print valid_pw('spez', 'hunter2', h)
```

RUN

```
True
```

Rainbow Tables



Quiz

when given the choice, which is the best hashing algorithm to hash passwords?

- [ ] md5
- [ ] sha256
- [ ] HTTPs
- [x] bcrypt

Quiz

Is it really, really embarrassing to have a database stolen with plaintext passwords?

- [x] yes
- [ ] no

Quiz

why do we hash passwords?

- [x] to keep snooping sys admins from knowing everyone's passwords
- [x] because people often use the same password for many websites
- [x] if the db is compromised, the passwords are reasonably safe
- [x] if you don't you will regret it

Quiz

what can we do to get to 10,000 visits?

- [x] reload the page 10,000 times
- [ ] send the link to 10,000 friends
- [x] edit the cookie in our browser

Quiz

when does a cookie with no expires parameter get deleted from your browser?

- [ ] Jan 1, 2025
- [ ] never
- [x] when you close your browser
- [ ] in 1 day

Quiz

which of these domains could set this cookie?

Set-Cookie: user=123; Domain=ide.udacity.com

- [ ] udacity.com
- [x] ide.udacity.com
- [x] other.ide.udacity.com
- [ ] other.udacity.com

Quiz

which of these domains would receive this cookie?

Set-Cookie: user=123; Domain = ide.udacity.com

- [ ] udacity.com
- [x] ide.udacity.com
- [x] other.ide.udacity.com
- [ ] other.udacity.com

Quiz

which header does a server use to set a cookie?

Set-Cookie

Quiz

which header does a browser use to send a cookie to a server?

Cookie

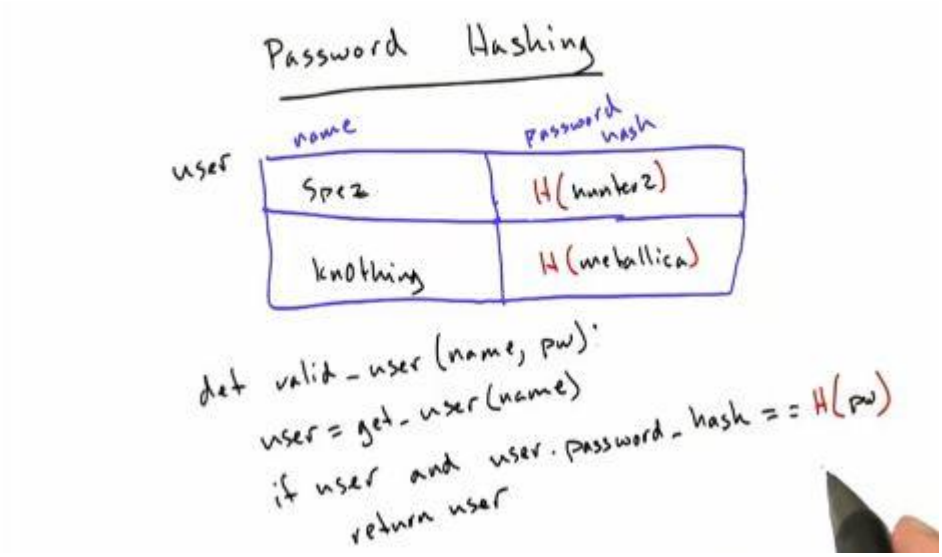
"shouldn't"

Quiz

what are ~~good~~ appropriate uses of cookies?

- [x] Storing login information
- [x] Storing small amounts of data to avoid hitting a db
- [x] Storing user preference info ← want data to survive
- [x] tracking you for ads

Password Hashing

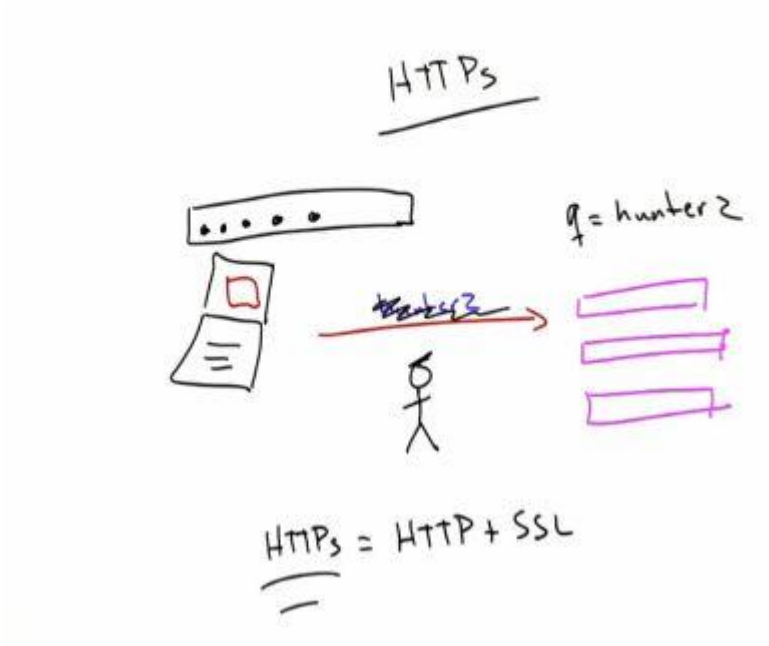



```
1  import random
2  import string
3
4  # implement the function make_salt() that returns a string of 5 random
5  # characters use python's random module
6
7  def make_salt():
8      return ''.join(random.choice(string.letters) for x in xrange(5))
9
10 print make_salt()
```

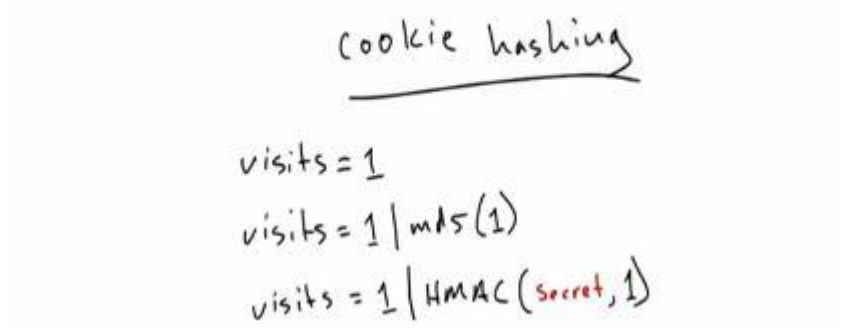
RUN

CnnEF

HTTPS



Incorporating HMAC




```
# Implement the hash_str function to use HMAC and our SECRET instead of md5
SECRET = 'imsosecret'
def hash_str(s):
    return hmac.new(SECRET, s).hexdigest()

def make_secure_val(s):
    return "%s|%s" % (s, hash_str(s))

def check_secure_val(h):
    val = h.split('|')[0]
    if h == make_secure_val(val):
        return val

print make_secure_val("test!!")
```

UN

t!!|96fc783b50f770c1a648f068c0fd5fee

```python
import random
import string
import hashlib

def make_salt():
    return ''.join(random.choice(string.letters) for x in xrange(5))

# implement the function make_pw_hash(name, pw) that returns a hashed password
# of the format:
# HASH(name + pw + salt),salt
# use sha256

def make_pw_hash(name, pw):
    salt = make_salt()
    h = hashlib.sha256(name + pw + salt).hexdigest()
    return '%s,%s' % (h, salt)

print make_pw_hash('spez', 'hunter2')
```

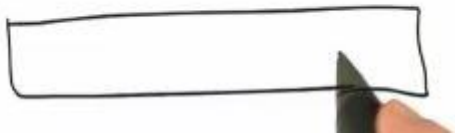
RUN

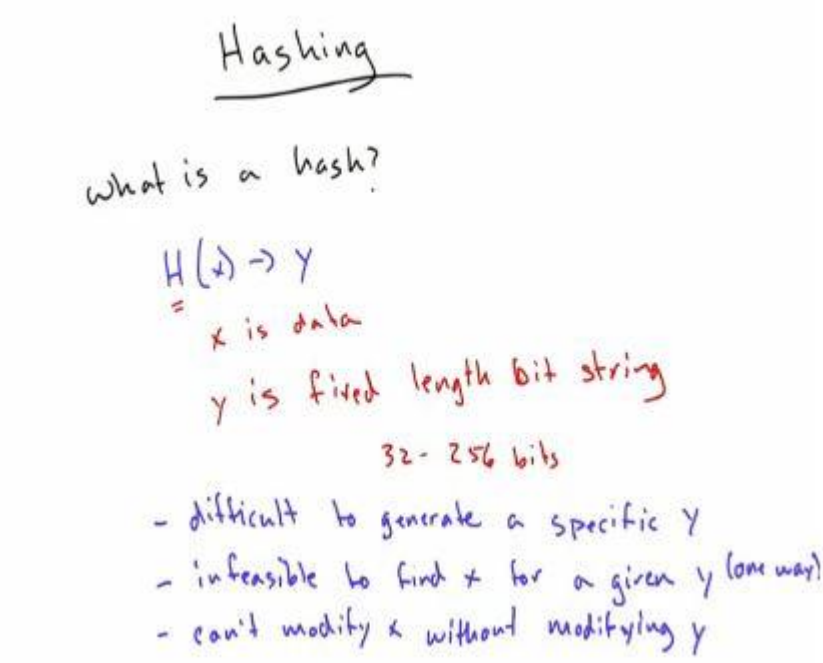
```
c7bc319229a56c0a1a694ece2740e35fcf82f808ca2caaeecc6f2e6938e81b0c,LIWyO
```

```
In [2]: hashlib.sha256("udacity").hexdigest()
Out[2]: '016d473857f1029884ec80ede8ae486f33d2fdad9411d63cd2aab11097ee997c'
```

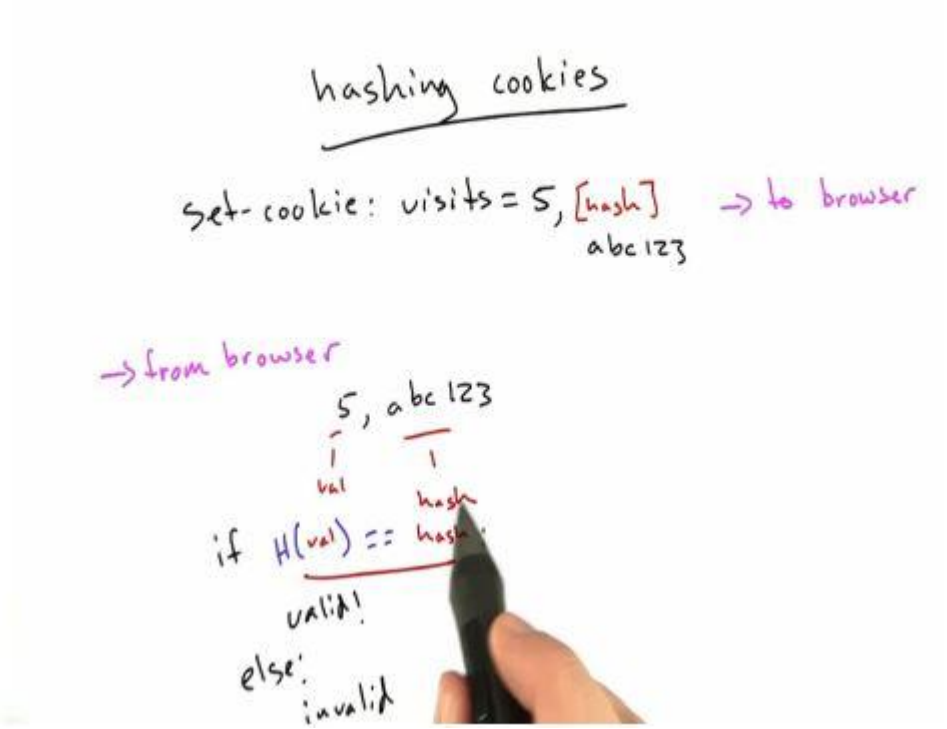
Quiz

use the hashlib library in python to find the sha256 hash of the string udacity (lowercase)





Hashing Cookies



Hash Algorithms

don't write your own!
(unless you're making a hashtable)

collision - two thing hash to the same value
(hard to find)

fast ↓ slow

crc32 - checksums, fast
md5 - fast, ~~secure~~
H(x) -> y
sha1 - secure-ish
sha256 - pretty good

Cookie Headers

HTTP Response

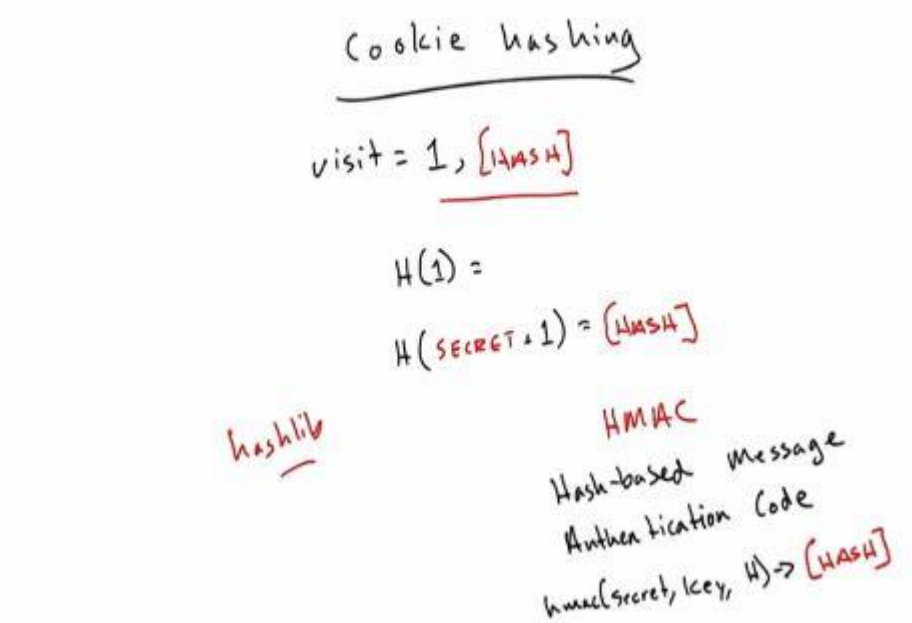
Set-Cookie: user_id=12345
name value (pk)
~20ℓ
Set-Cookie: last-seen = Dec 25 1985

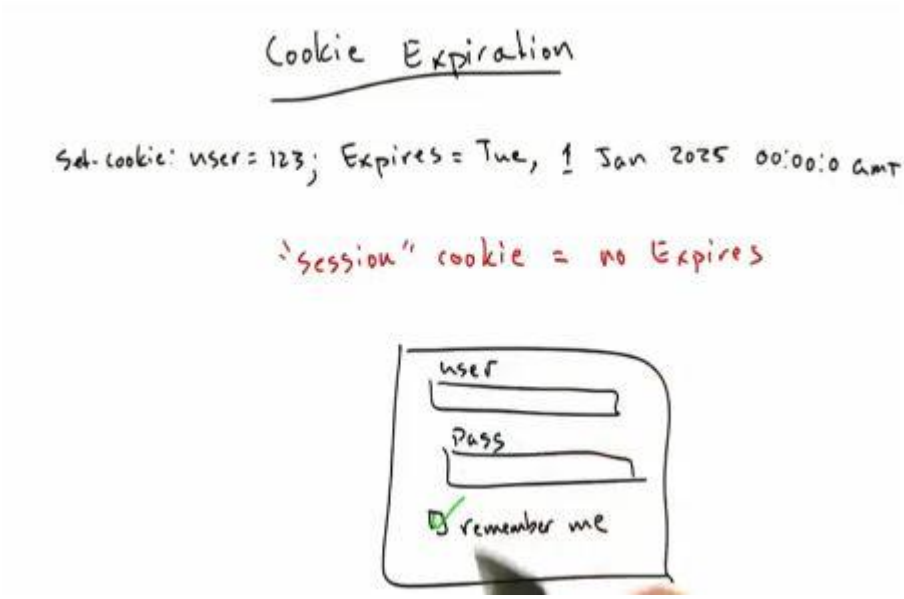
HTTP Request

Cookie: user_id=12345; last-seen=Dec 2

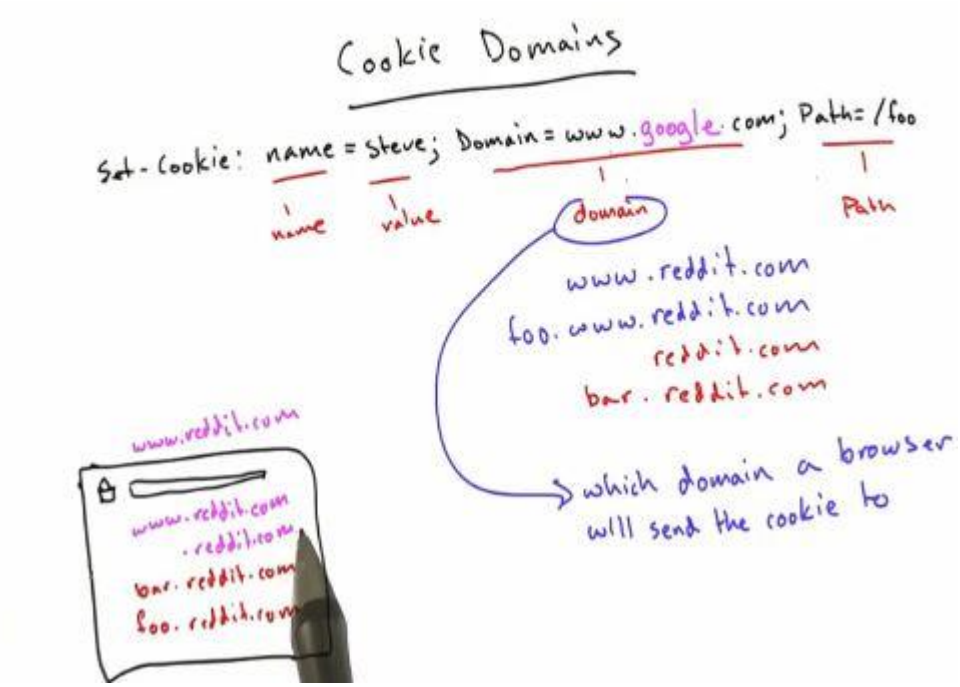
don't

Cookie Hashing

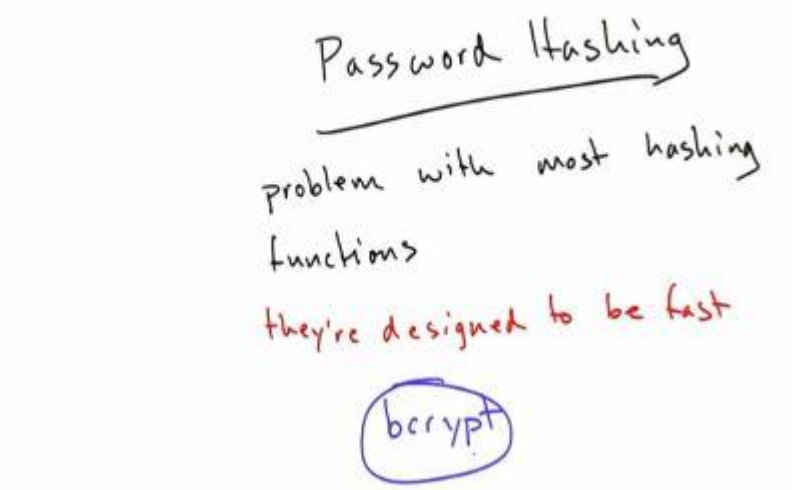




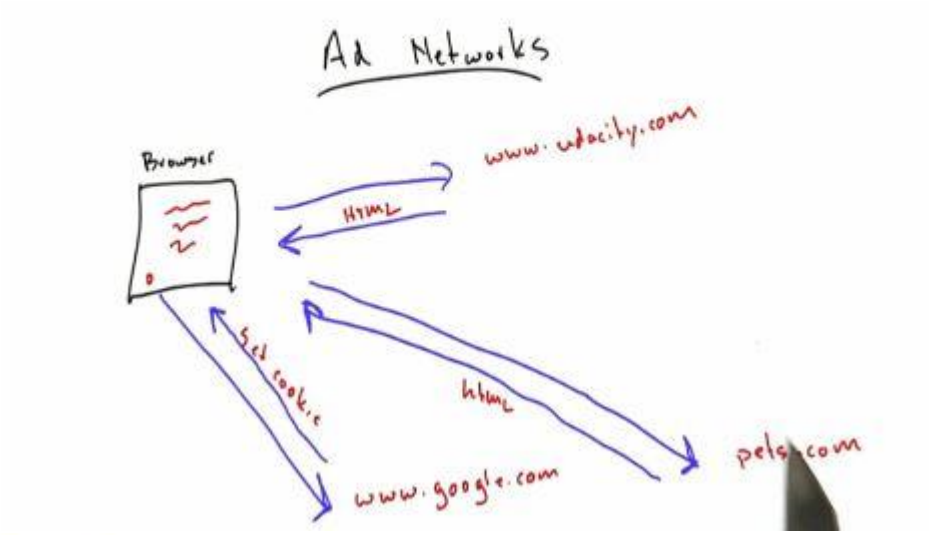
Cookie domains



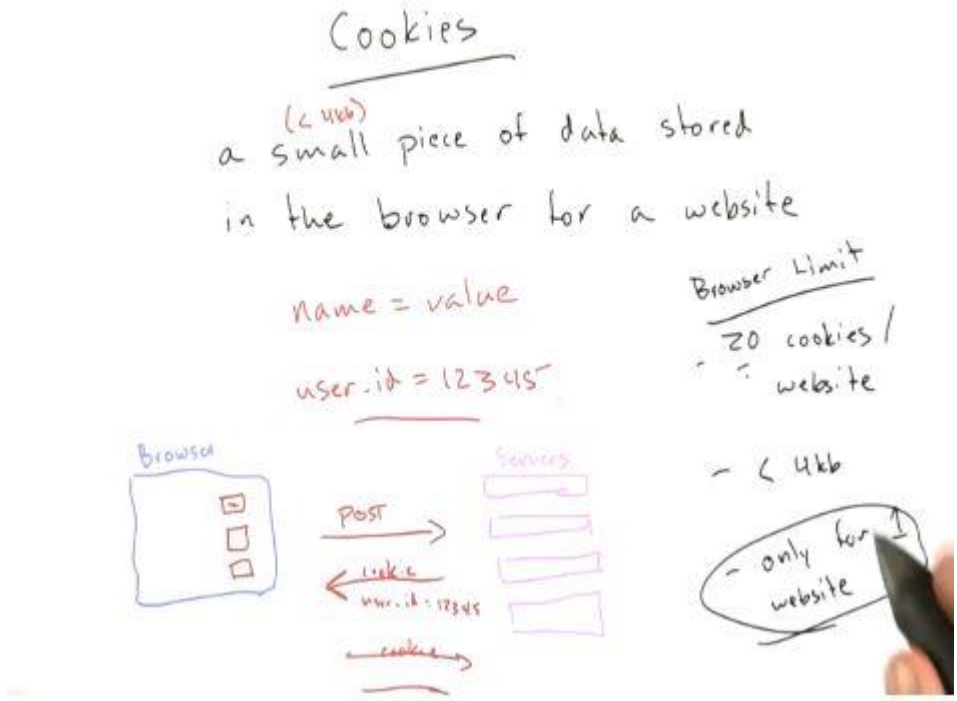
Bcrypt



Ad Networks



Cookies




6 # Implement the function make_secure_val, which takes a string and returns a
7 # string of the format:
8 # s,HASH
9
10 def make_secure_val(s):
11 return "%s,%s" % (s, hash_str(s))
12
13 print make_secure_val("cool")
RUN
cool,b1f4f9a523e36fd969f4573e25af4540