Intro to Computer Science

Previous

- Web servers
- HTML
- CGI

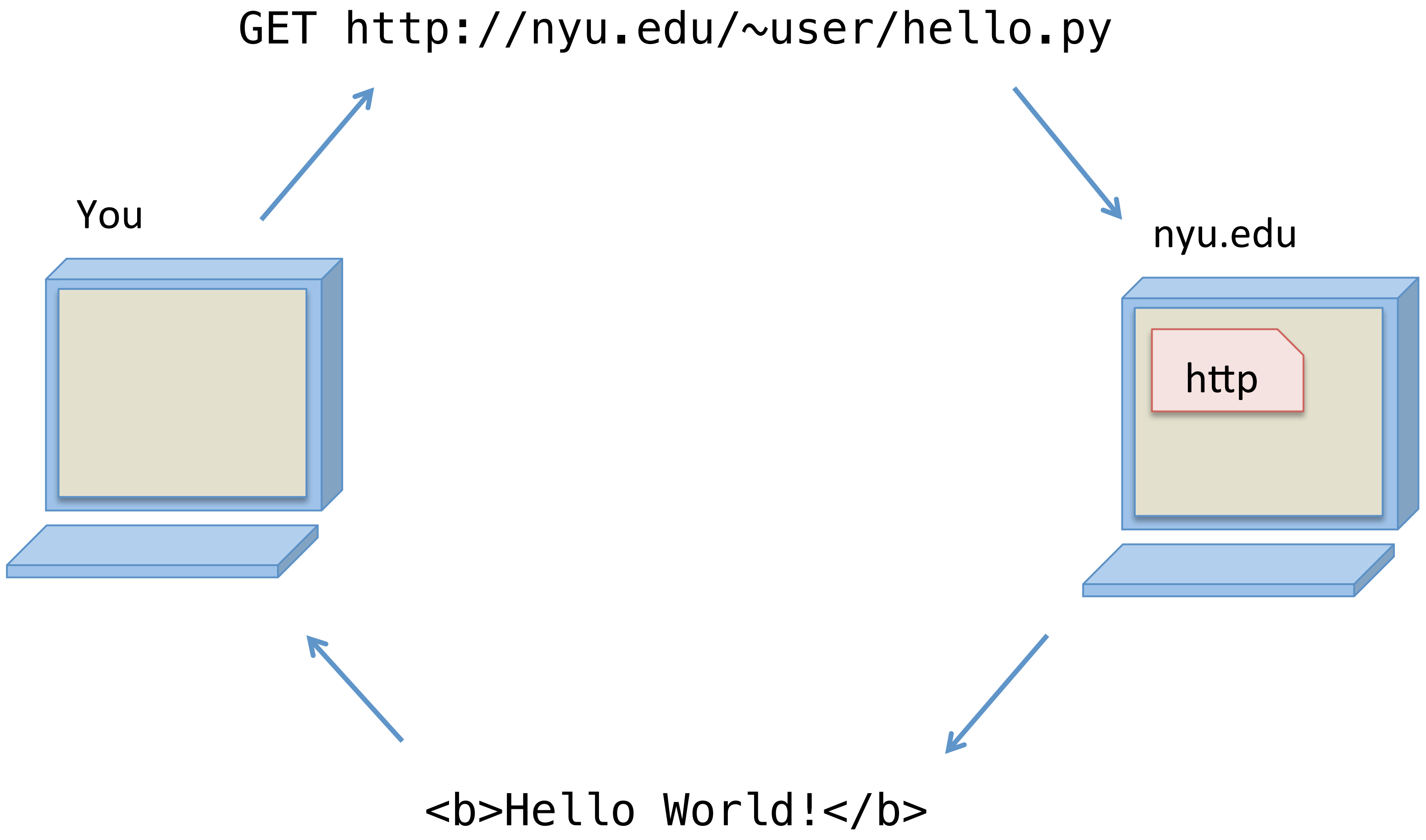
Next

- Forms

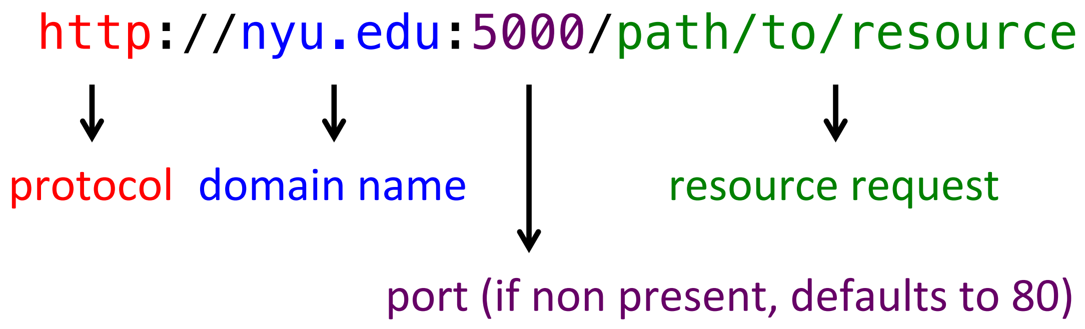
Book	Chapter
Banana	
Rollercoaster	
Rocket ship	

Book	Chapter
Banana	
Rollercoaster	
Rocket ship	

Connection to a web server



The URL


http://nyu.edu:5000/path/to/resource
protocol
domain name
resource request
port (if non present, defaults to 80)

- “*Connect* to nyu.edu using the HTTP protocol and request the resource at path/to/page”

This is great

- Rather than HTML sitting around waiting to be read, there's a program that will run *on demand*
- A web page can reflect whatever Python can do
  - **Before**: Terminal displays results
  - **After**: Browser displays results

A one sided relationship

- Although our programs are dynamic, users don’t have much control
  - Our calendar was always the current month
- In general, lack of user input is of limited use
  - How would you log into a website?
  - How would you “like” a photo?

Giving back

There are two ways of passing information back to a web page (and in turn, your program)

1. Via the GET request
2. Via HTML forms (GET and POST)

The Query String

```
http://nyu.edu/path/to/resource?a=1
```

- Passing information via the GET request is a matter of augmenting the URL with additional information
  - ➢ It's only the web server who cares about what comes after the domain name
- Convention:
  - – A question mark denotes where the resource name stops
  - – Information after the question mark is passed to the resource

The Query String

`http://nyu.edu/path/to/resource?a=1&b=2&c=3&d=4&e=5`

- Multiple values are separated by an ampersand
- Because this convention is so common, most web frameworks takes care of
  - parsing the string to determine variables and values
  - assigning the variables to values for you

Creating query string links

- We know
  - How to build links (in HTML)
  - The structure of the query string
- We can now build our own links that pass information

```
<a href="?var1=blah">click!</a>
```

By default, HREF will refer to itself (the same page) if a URL is not present

The addition of a question mark, and a series of variable/variable values

Giving back

There are two ways of passing information back to a web page (and in turn, your program)

1. Via the GET request
2. Via HTML forms

Forms

- You've seen these before
- There are a series of elements (tags) that make up a form:
  - text
  - password
  - checkbox
  - radio button
  - drop down menu
  - submit button
  - (and a few more)

Form basics

More on this later…

There are various *types* of input

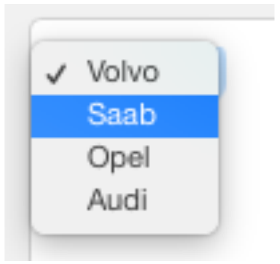
```
<form method="GET">

    First name: <input type="text" name="fname">
    Last name: <input type="text" name="lname">
    <input type="submit" value="myform">

</form>
```

The name will be a key in a Flask dictionary

Selection

- Selection elements produce drop-down menus

```
<form>
<select name="car">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="opel">Opel</option>
  <option value="audi">Audi</option>
</select>
</form>
```



Options
- Fall within a select block
- select name is the key; option value is the value
- *Can be numerous!*

Form basics

- Method refers to the HTTP command that is used to transfer the form data
- It’s usually either GET or POST

```
<form method=“GET”>

    First name: <input type=“text” name=“fname”>
    Last name: <input type=“text” name=“lname”>
    <input type=“submit” value=“myform”>

</form>
```

Form basics

```
<form method=“POST”>

    First name: <input type=“text” name=“fname”>
    Last name: <input type=“text” name=“lname”>
    <input type=“submit” value=“myform”>

</form>
```

HTTP POST

- GET passes information as part of the URL
  - Query string
- POST passes it as part of the message body
  - Along with the URL, send a message containing the form information

Comparison

GET	POST
Information sent via the query string	Information sent within the request
Page creator can manipulate links	Information is exchanged through forms
Limited by URL length (~2000 characters)	Effectively no limit on how large the POST can be
Information that is sent is visible to users	Information is “hidden” from view (good for sensitive information)
Page reloading seems normal	Page reload may prompt “confirmation” notification from browser

- Today we will use both just for practice
- In reality, choice will depend on conditions
- *Understanding both is advantageous*