# Modules

## Objectives

- Define what a module is
- Import code from built-in modules
- Import code from other files
- Import code from external modules using pip
- Describe common module patterns
- Describe the request/response cycle in HTTP
- Use the requests module to make requests to web apps

## Why Use Modules?

- Keep Python files small
- Reuse code across multiple files by importing
- A module is just a Python file!

# Built-in Modules Example

```
import random

random.choice(["apple", "banana", "cherry", "durian"])
random.shuffle(["apple", "banana", "cherry", "durian"])
```

```
import random as r

r.choice(["apple", "banana", "cherry", "durian"])
r.shuffle(["apple", "banana", "cherry", "durian"])
```

## Importing Parts of a Module

- The ***from*** keyword lets you import parts of a module
- Handy rule of thumb: only import what you need!
- If you still want to import everything, you can also use the `from MODULE import *` pattern

## Different Ways to Import

All of these work!

- `import random`
- `import random as omg_so_random`
- `from random import *`
- `from random import choice, shuffle`
- `from random import choice as gimme_one, shuffle as mix_up_fruits`

## Custom Modules

## Custom Modules

- You can **import** from your own code too
- The syntax is the same as before
- import from the name of the Python file

## Custom Modules Example

file1.py

```
def fn():
    return "do some stuff"

def other_fn():
    return "do some other stuff"
```

file2.py

```
import file1

file1.fn() # 'do some stuff'

file2.fn() # 'do some other stuff'
```

YOUR
TURN

## External Modules

## External Modules

- Built-in modules come with Python
- External modules are downloaded from the internet
- You can download external modules using **pip**

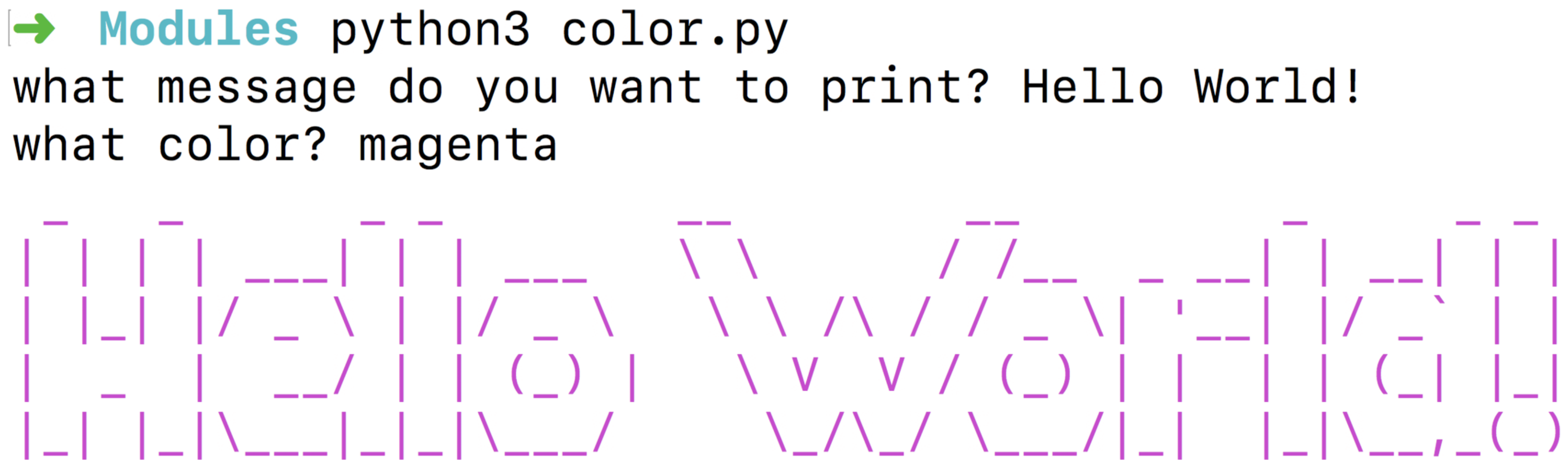
## pip

- Package management system for Python
- As of 3.4, comes with Python by default
- `python3 -m pip install NAME_OF_PACKAGE`

## External Modules Example

- termcolor - Adds colors to output in a Python shell
- pyfiglet - Ascii art creator!

# ASCII ART EXERCISE



```
➜  Modules python3 color.py
what message do you want to print? Hello World!
what color? magenta
 _   _      _ _        __        __         _     _ _ 
| | | | ___| | | ___   \ \      / /__  _ __| | __| | |
| |_| |/ _ \ | |/ _ \   \ \ /\ / / _ \| '__| |/ _` | |
|  _  |  __/ | | (_) |   \ V  V / (_) | |  | | (_| |_|
|_| |_|\___|_|_|\___/     \_/\_/ \___/|_|  |_|\__,_(_)
```

Use the pyfiglet package!

## The `__name__` Variable

## `__name__`

- When run, every Python file has a __name__ variable
- If the file is the main file being run, its value is "__main__"
- Otherwise, its value is the file name

## `import` Revisited

When you use **import**, Python...

1. Tries to find the module (if it fails, it throws an error),
2. Runs the code inside of the module being imported,
3. Creates variables in the namespace of the file with the import statement.

## Ignoring Code on Import

```
if __name__ == "__main__":
    # this code will only run
    # if the file is the main file!
```

YOUR
TURN

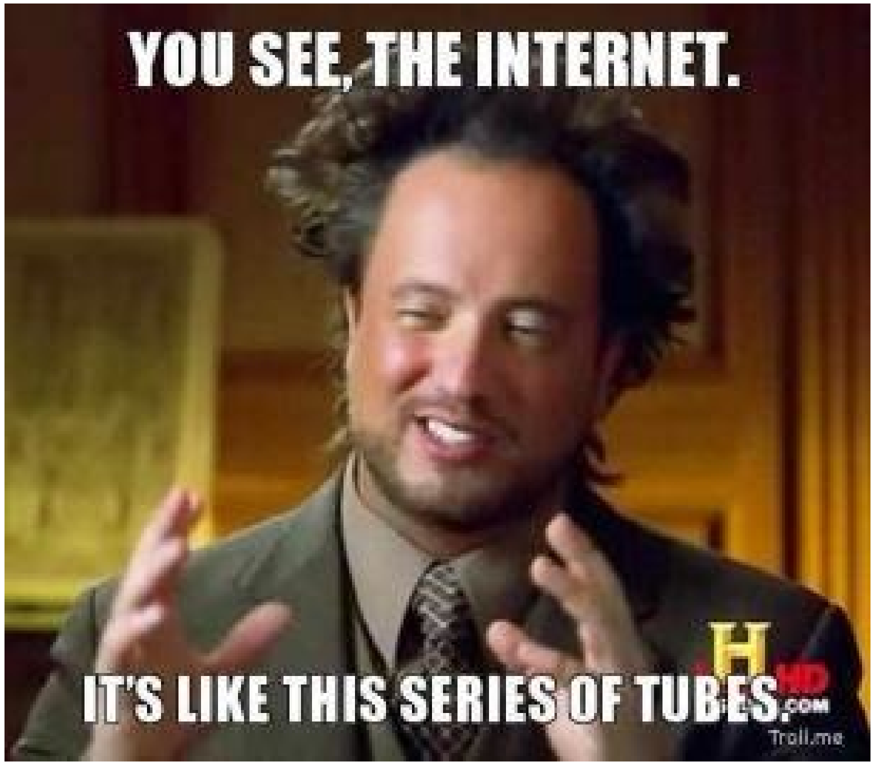
## THIS SECTION IS OPTIONAL

## HTTP Introduction

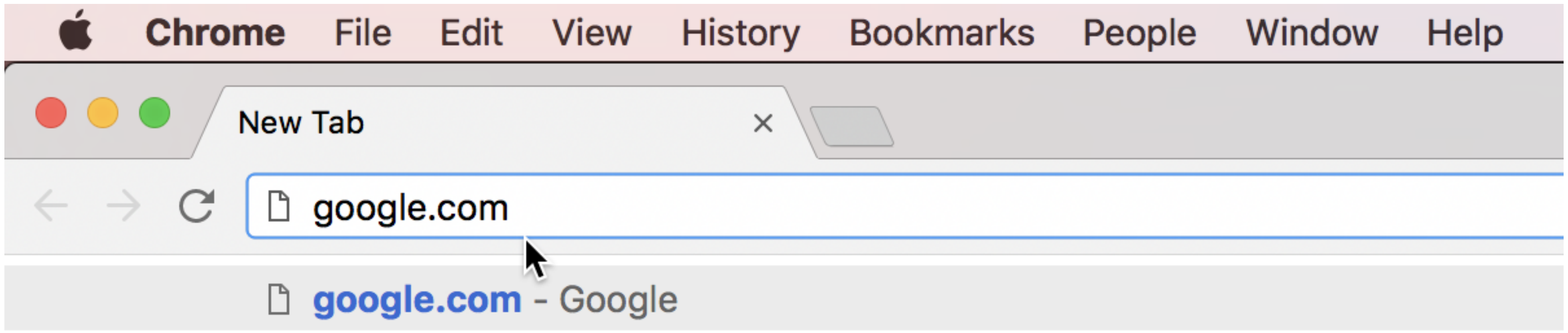
## HTTP Introduction

- Describe what happens when you type a URL in the URL bar
- Describe the request/response cycle
- Explain what a request or response header is, and give examples
- Explain the different categories of response codes
- Compare GET and POST requests

## The Internet



# What Happens When...

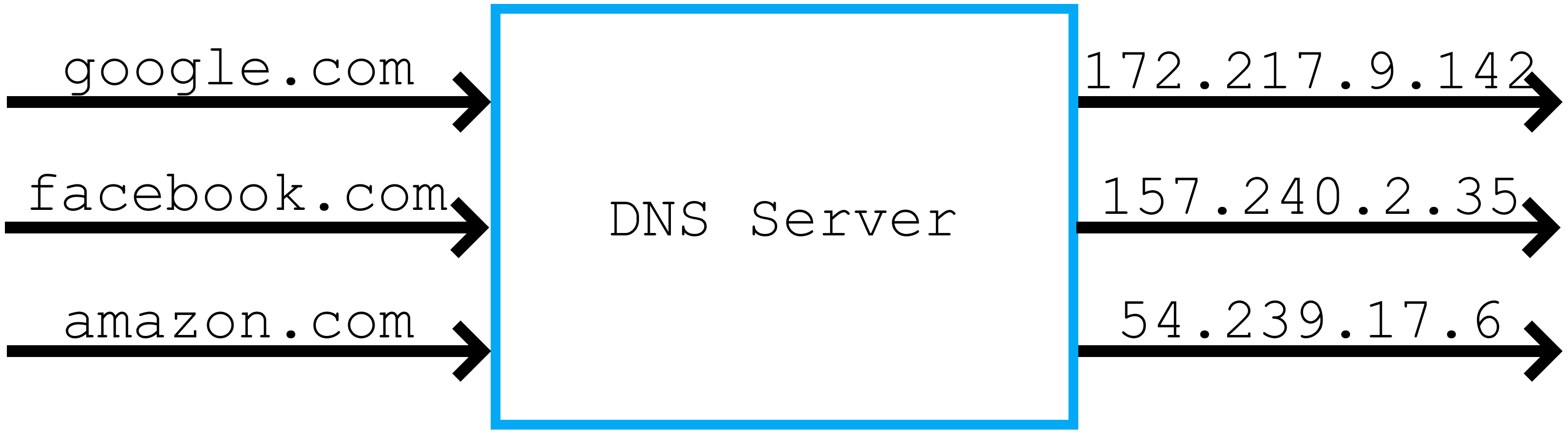


1. DNS Lookup
2. Computer makes a REQUEST to a server
3. Server processes the REQUEST
4. Server issues a RESPONSE

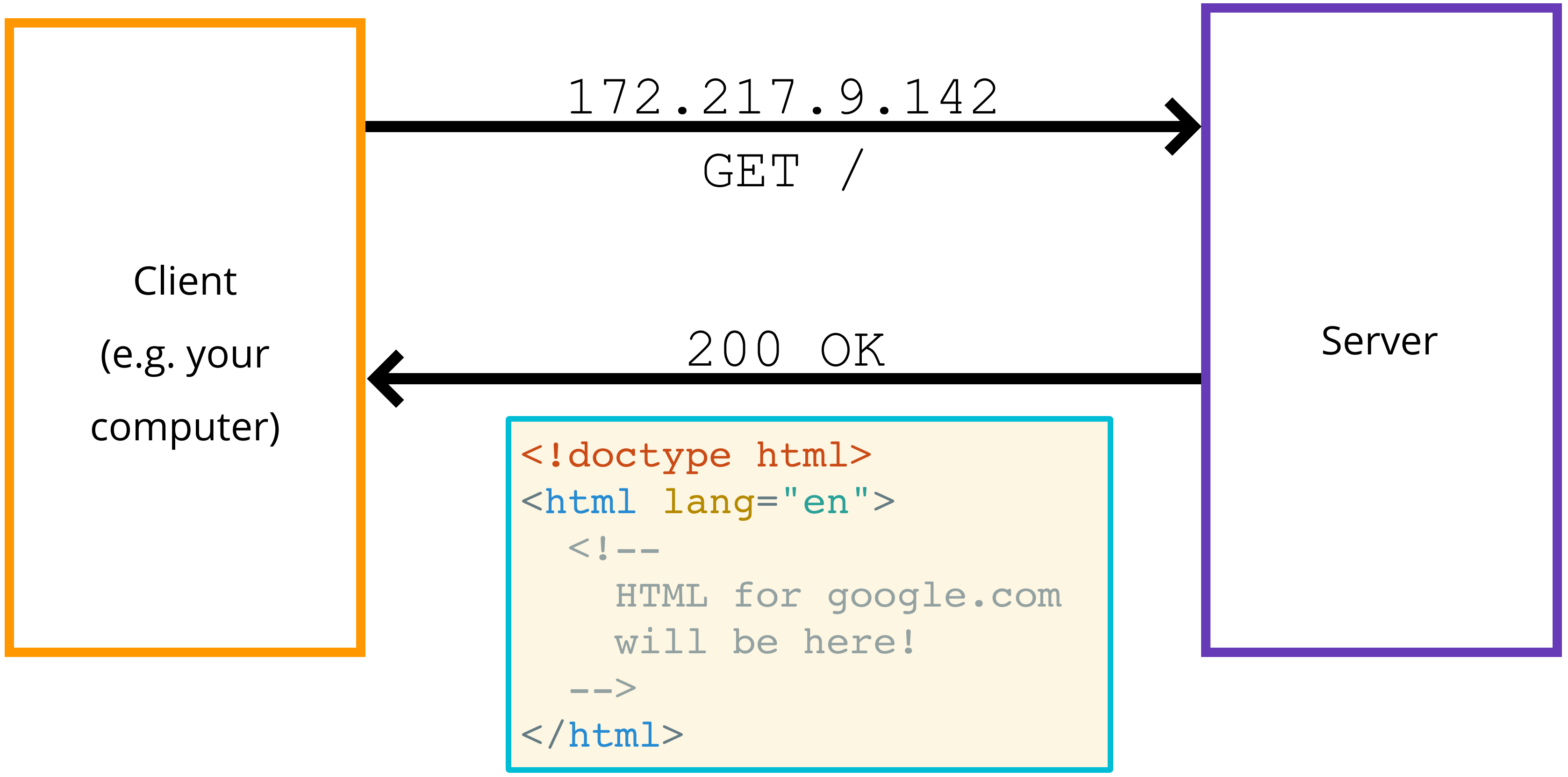
Request/Response cycle

# DNS Lookup

Like a Phonebook for the Internet!



## Requests and Responses



```
<!doctype html>
<html lang="en">
  <!--
    HTML for google.com
    will be here!
  -->
</html>
```

## HTTP Headers

- Sent with both requests and responses
- Provide additional information about the request or response

## Header Examples

**Request Headers**

- **Accept** - Acceptable content-types for response (e.g. html, json, xml)
- **Cache-Control** - Specify caching behavior
- **User-Agent** - Information about the software used to make the request

**Response Headers**

- **Access-Control-Allow-Origin** - specify domains that can make requests
- **Allowed -** HTTP verbs that are allowed in requests

# Response Status Codes

- 2xx - Success
- 3xx - Redirect
- 4xx - Client Error (your fault!)
- 5xx - Server Error (not your fault!)

## HTTP Verbs

### GET

- Useful for retrieving data
- Data passed in query string
- Should have no "side-effects"
- Can be cached
- Can be bookmarked

### POST

- Useful for writing data
- Data passed in request body
- Can have "side-effects"
- Not cached
- Can't be bookmarked

## APIs

- API - Application Programming Interface
- Allows you to get data from another application without needing to understand how the application works
- Can often send data back in different formats
- Examples of companies with APIs: GitHub, Spotify, Google

## Using the `requests` Module

## `requests` Module

- Lets us make HTTP requests from our Python code!
- Installed using pip
- Useful for web scraping/crawling, grabbing data from other APIs, etc

# Making a Request

```python
import requests

response = requests.get("http://www.example.com")
```

## Request Headers

```python
import requests

response = requests.get(
    "http://www.example.com",
    headers={
        "header1": "value1",
        "header2": "value2"
    }
)
```

## What's a Query String?

- A way to pass data to the server as part of a GET request
- http://www.example.com/*?key1=value1&key2=value2*
- Browsers enforce a maximum size on length of the query string

## Query String

```
# option 1

import requests

response = requests.get(
    "http://www.example.com?key1=value1&key2=value2
)
```

```
# option 2 - preferable!

import requests

response = requests.get(
    "http://www.example.com",
    params={
        "key1": "value1",
        "key2": "value2"
    }
)
```

## POST Request

```python
import requests
import json

response = requests.post(
    "http://www.example.com",
    data=json.dumps({
        "key1": "value1",
        "key2": "value2"
    })
)
```

## A Note on APIs

- Some APIs require a key in order for you to use them
- Especially true of APIs that allow you to send data, rather than just getting data
- Typically sent as part of URL
- API keys allow for greater control of how users interact with the API
- Instructions for obtaining a key vary by API

## Recap

- Python modules let you import code from other files
- There are three types of modules: built-in, custom, and external
- pip is the package management system for Python
- To ignore code during an import, use `if __name__ == "__main__"`
- Fundamental Internet vocabulary: DNS, Request/Response, Headers, Status Codes, HTTP Verbs, etc.
- Requests is a module for making HTTP requests in Python

YOUR
TURN