Index

1. Lyn: 2019: Python XML, JSON, and the Web
2. 8

**Lyn: Python XML, JSON, and the Web [ Joe Marini ]**

**// 104 Internet Protocol and Support – py docs**

**Native py standard libs:**

urllib: work with request, error, parse, robotparser

http: client, server, cookies, cookiejar

json: dump stream of data

xml:

**Third party:**

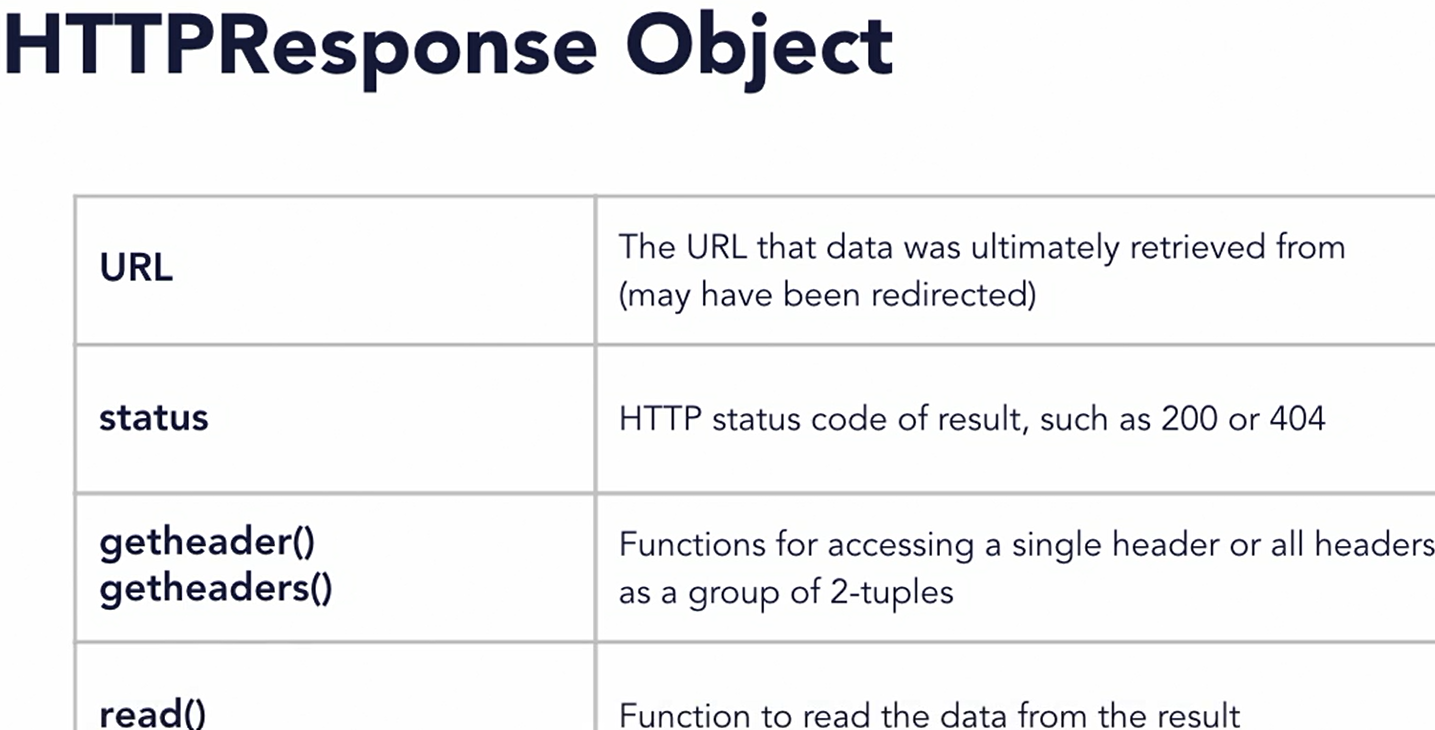
lxml: feature rich xml and html

requests: simple api request data handling

**// 105 httpbin to test api httpbin.org** # loopback

**// 201 urllib**

* Request: Opening and reading of urls
* Error raised by request
* Parse url structure
* Robot parser for robot.txt files

res = urllib.request.urlopen(url, data=None, [timeout, ... ], … )

**// 202 Retrieve data**

import urllib.request; url = "http://httpbin.org/xml"

res.getheaders(); res.read().decode("utf-8")

**// 203 Send Data**

# create some data to pass to the GET request

args = { "name" : "Joe Marini", "is\_author" : True }

# url-encoded data before passing as arguments

data = **urllib.parse.urlencode(args)**

# issue the request with the data params as part of the URL

# result = urllib.request.urlopen(url + "?" + data)

# issue the request with a data parameter to use POST

url = <http://httpbin.org/post>; **data = data.encode()**

result = **urllib.request.urlopen(url, data=data)**

**// 204 Error Handling**

from urllib.error import HTTPError, URLError

from http import HTTPStatus

# use exception handling to attempt the URL access

try: result = urllib.request.urlopen(url)

print("Result code: {0}".format(result.status))

if (result.getcode() == HTTPStatus.OK):

print(result.read())

# occurs when the server returns a non-success error code

except HTTPError as err:

print("Error: {0}".format(err.code))

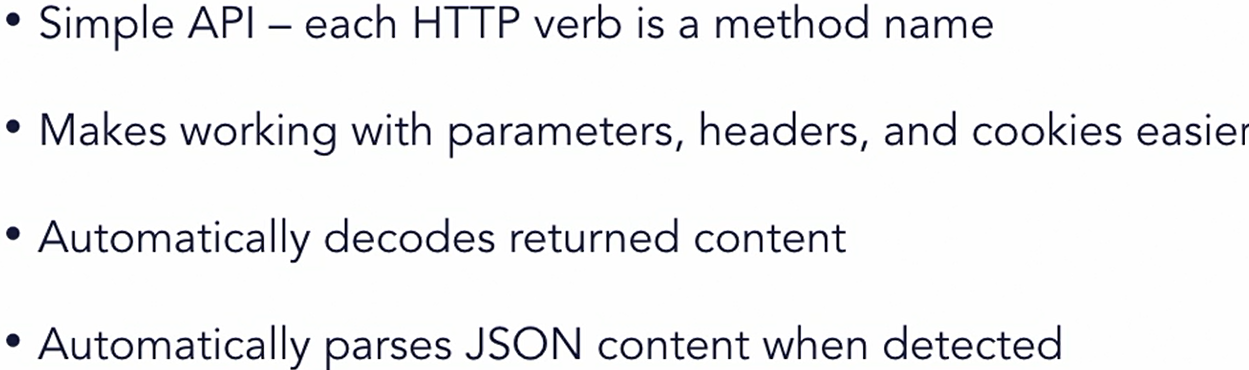
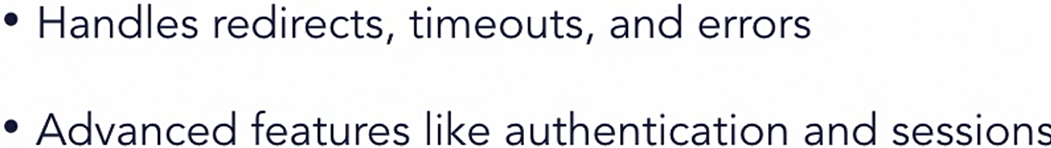
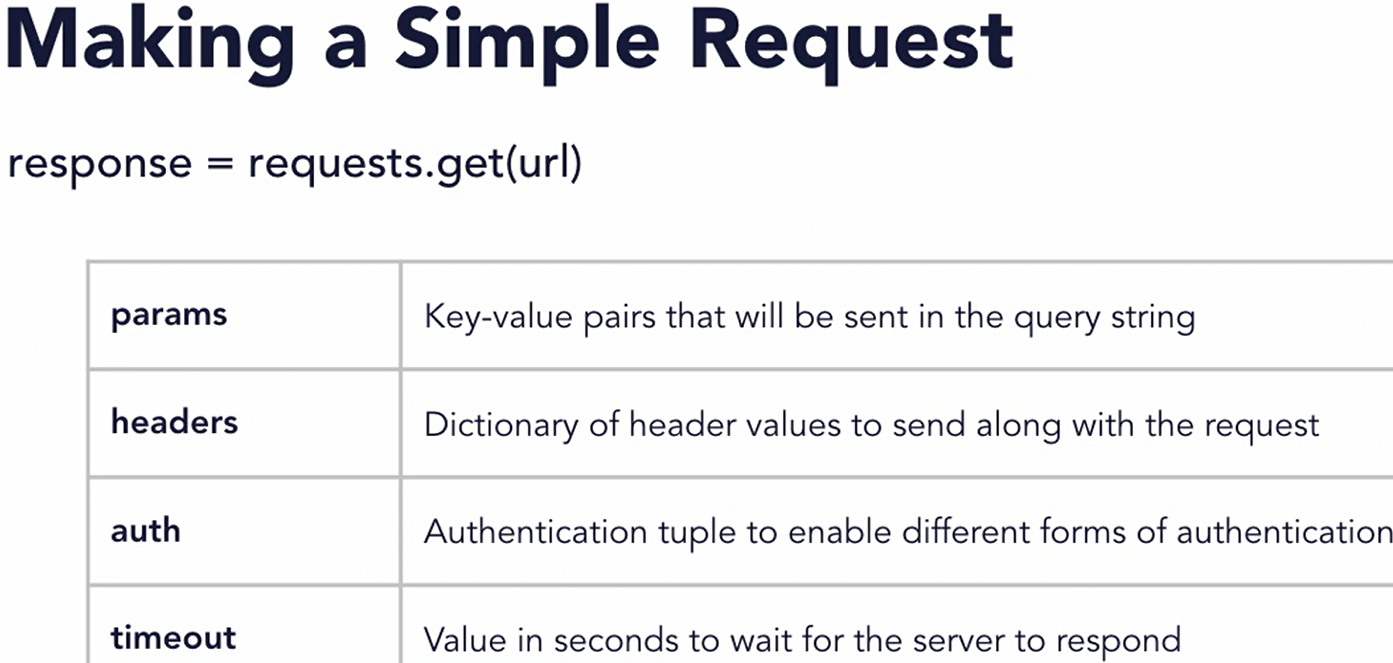
# occurs when something is wrong with the URL itself

except URLError as err:

print("Yeah, that server is bunk. {0}".format(err.reason))

**// 205 Drawback of urllib**

* Only support a subset of HTTP by default
* Not support patch, delete methods
* Doesn’t auto decode returned data
* No build in features for cookies, session, auth
* Require extra more modules
* Difficult to implement advanced features such as timeouts
* Processing json is cumbersome

**// 301 Request library**

**// 302 Send data - retrieve**

import requests

def main():

result = requests.get(xml\_url); printResults(result)

# Send some parameters to the URL via a GET request

dataValues = { 'key1' : 'value1', 'key2' : 'value2' }

result = **requests.get(get**\_url, **params=dataValues)**

# result = **requests.post(post\_url, data=dataValues)**

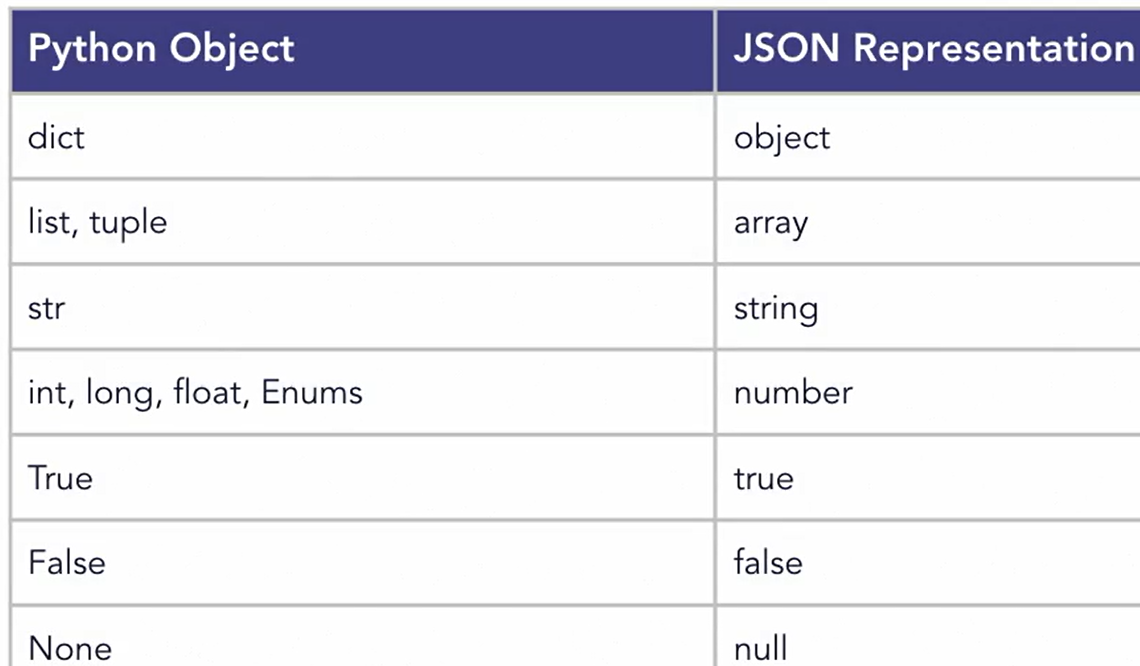
printResults(result)

# Pass a custom header to the server

headerValues = { 'User-Agent' : 'Joe Marini App / 1.0.0' }

result = requests.get(get\_url, headers=headerValues)

printResults(result)

xml\_url = "http://httpbin.org/xml"

get\_url = "http://httpbin.org/get"

post\_url = “<http://httpbin.org/post>”

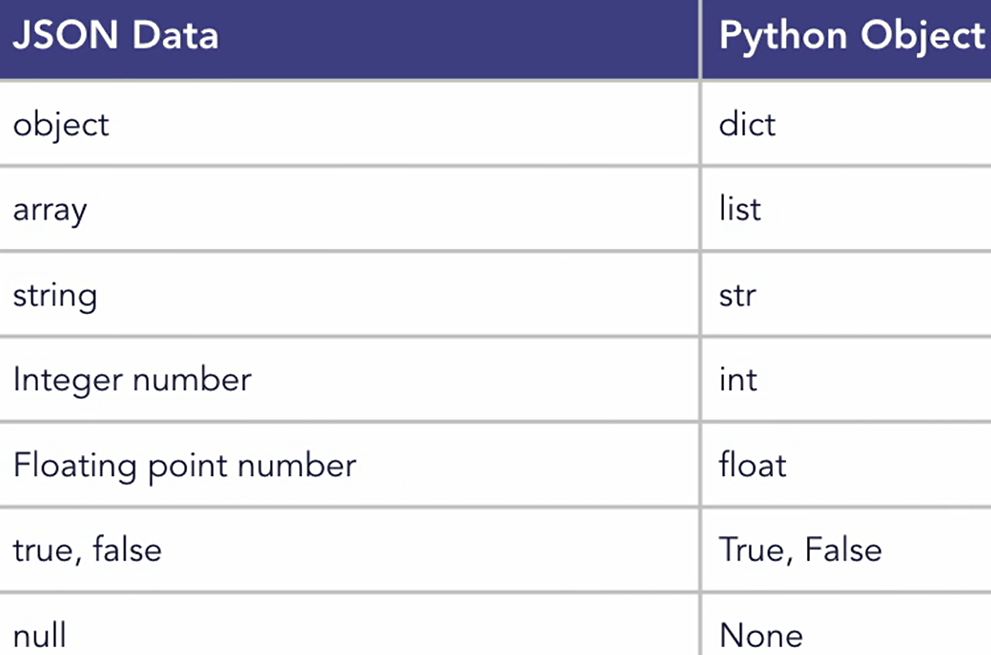
print("Result code: {0}".format(resData.status\_code))

print(resData.headers)

print("Returned data: ----------------------", resData.text)

**// 303 Request error handling**

**import requests**

**from requests.exceptions import HTTPError, Timeout**

try:

url = "http://httpbin.org/status/404"

#url = "http://httpbin.org/delay/5"

result = requests.get(url, timeout=2)

# raise\_for\_status will throw an exception if an HTTP error

# code was returned as part of the response

**result.raise\_for\_status()**

printResults(result)

**except HTTPError as err**:

print("Error: {0}".format(err))

**except Timeout as err:**

print("Request timed out: {0}".format(err))

**// 304 Using Authentication**

from requests.auth import HTTPBasicAuth

# Access a URL that requires authentication - the format of this URL is that you provide the username/password to auth against

url = "http://httpbin.org/basic-auth/JoeMarini/MySecretWord"

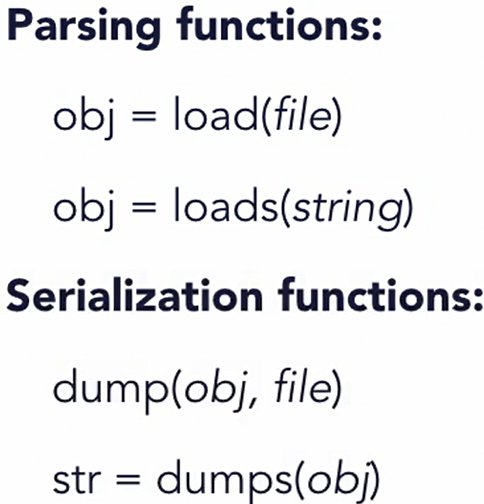
# Create a credentials object using HTTPBasicAuth

**myCreds = HTTPBasicAuth('JoeMarini', 'MySecretWord')**

# Issue the request with the authentication credentials

result = requests.get(url, **auth=myCreds)**

printResults(result)

**// 401 JSON Module**

json.load # read json from file

json.dump # write obj to file as json

**# practice what will get convert to what on console**

**// 402 Parsing \_ Serialize JSON**

# define a string of JSON code

jsonStr = '''{ "sandwich" : "Reuben", "toasted" : true,

"toppings" : [ "Thousand Island Dressing", "Sauerkraut" ],

"price" : 8.99 }'''

# parse the JSON data using loads

data = json.loads(jsonStr)

# print information from the data structure

print("Sandwich: " + data['sandwich'])

if (**data['toasted']**): print("And it's toasted!")

for topping in data['toppings']: print("Topping: " + topping)

------ ------ ------

# define a python dictionary to serialize to JSON using dump

pythonData = {

"sandwich": "Reuben",

"toasted": True,

"toppings": ["Thousand Island Dressing", "Sauerkraut" ],

"price": 8.99 }

# serialize to JSON using dumps

jsonStr = json.dumps(pythonData, indent=4)

# print the resulting JSON string

print("JSON Data: --------"); print(jsonStr)

**// 403 JSON Exception Handling**

from json **import JSONDecodeError**

try: # parse the JSON data using loads

data = json.loads(jsonStr)

# print information from the data structure

print("Sandwich: " + data['sandwich'])

if (data['toasted']): print("And it's toasted!")

for topping in data['toppings']: print("Topping: " + topping)

except **JSONDecodeError as err:**

print("Whoops, JSON Decoding error:")

**print(err.msg); print(err.lineno, err.colno)**

**// 404 Request JSON**

url = "http://httpbin.org/json"

result = **requests.get(url)**

# Use the built-in JSON function to return parsed data

dataobj = **result.json()**

print(**json.dumps(dataobj,indent=4))**

# Access data in the python object

print(**list(dataobj.keys()))**

print(dataobj['slideshow']['title'])

print("There are {0} slides"**.format( len(dataobj['slideshow']['slides']))** )

**// 501 XML parsing models // 502 SAX API**

**// 503 XML SAX Module** *# skipped*

**// 601 XML Dom // 602 XML DOM minidom**

**// 603 Element Tree API // 604 Using LXML**

*// skipped xml as it has limited use for us*

**// 701 Next**

Learning Python GUI Programming

Python Data Analysis

**Lyn: TypeScrg**

https://catalog.workshops.aws/stepfunctions/en-US

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**

**// 201 A**

**// 202 A**

**// 203 A**

**// 204 A**

**// 05 A**

**// 06 A**