DEVCON 19 austin

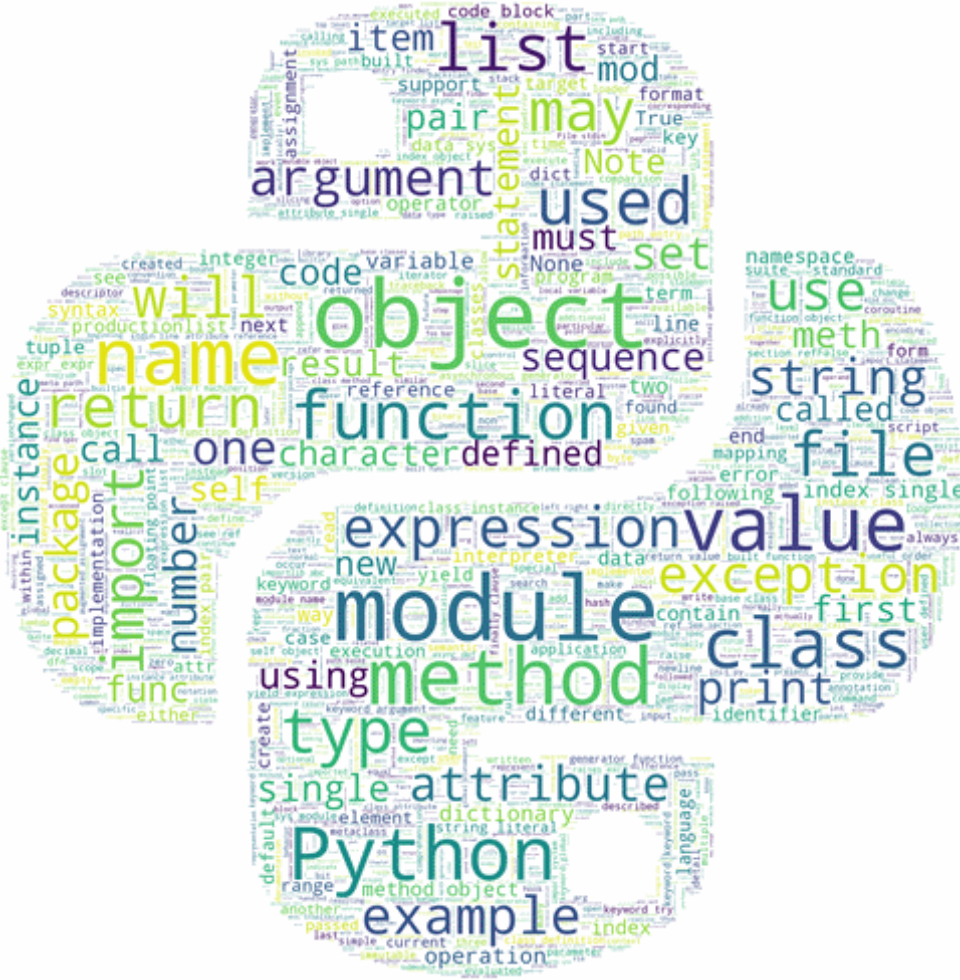
Python Programming for Blackboard Learn System Administrators

Michael Bechtel
LMS Specialist
Indian River State College

Jean-Max Davis
Academic Educational Technologist
University of Miami

Brett Stephens
Systems Analyst
University of Miami

July 23, 2019



DEVCON 19 austin

Fortran & COBOL

~~Python~~ Programming for Blackboard Learn System Administrators

Michael Bechtel
LMS Specialist
Indian River State College

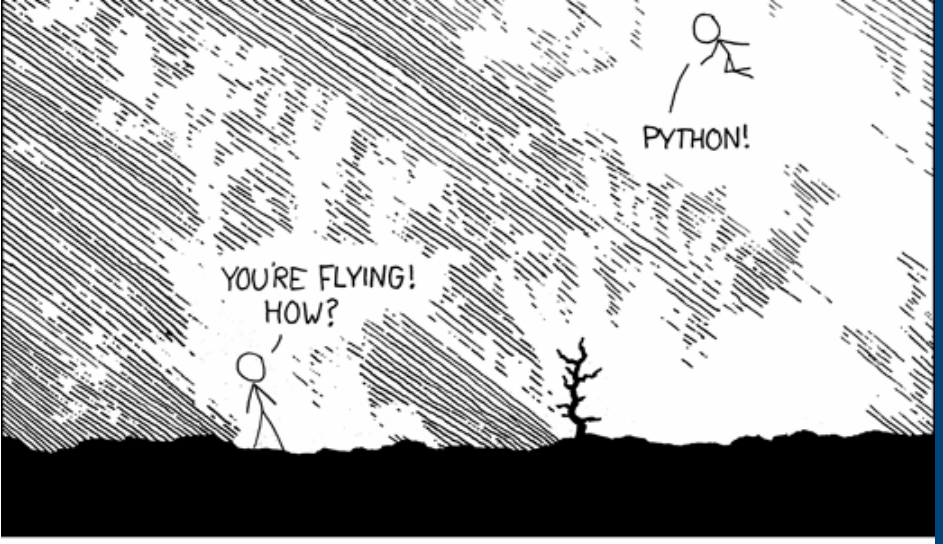
Jean-Max Davis
Academic Educational Technologist
University of Miami

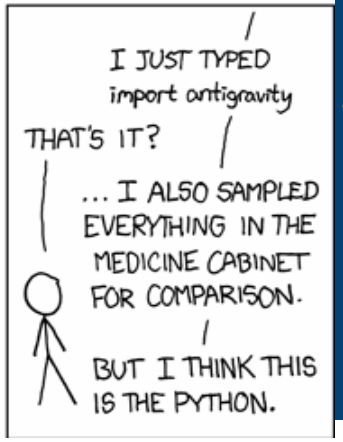
Brett Stephens
Systems Analyst
University of Miami

July 23, 2019

Slides and Sample Code available at…

https://tinyurl.com/bb19python





https://xkcd.com/353/

What? Why Python?

- Dynamically-typed
- Interpreted (no compiling)
- OOP Capable
- Well supported by community
- Python is designed to be readable

```
for n in range(10):
    print(n)
```

0

1…

9

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Readability counts.

--The Zen of Python (PEP 20)

Easter egg >>> import this

Learning Python (it takes a bit longer than 45 minutes)

Python.org
Documentation
& Beginner's Guide

Dr. Chuck's PY4E.com

Dr. Charles Severance
University of Michigan

He was here yesterday, so go ask him to teach you then...

Useful Libraries and stuff

I/O & Data

- **Requests**
- **bbRest**
- SQLAlchemy
- Selenium

I/O & Data

- Google apiclient
- json
- email, smtplib
- Beautiful Soup

Misc

- DocOpt & Schema
- Matplotlib
- **Jupyter Notebook/Lab**

DEVCON 19 austin

SQL in Python...with Alchemy!

There is a hard way and an easy way. Ain't nobody got time for the hard way.

SQL Alchemy basic example

```
1 from sqlalchemy import create_engine
2 db_string = "postgres://postgres:postgres@somehost:9879/BBDbName"
3 db = create_engine(db_string)
4 userid = 'stu_bs007’
5 results = db.execute("select u.user_id, cm.course_id, cu.role \
6                        from users u join course_users cu on \
7                        u.pk1 = cu.users_pk1 \
8                        join course_main cm on \
9                        cu.crsmain_pk1 = cm.pk1 \
10                       where u.user_id = '%s' " % userid)
```

SeeeeeQuiiiil

```
1 for row in results:
2     print(
3       f'User: {row.user_id} Course: {row.course_id} \
4       Role: {row.role}')
5
6 #output User: stu_bs007 Course: dev-course Role: S
```

DEVCON 19 austin

List and Dictionary Comprehensions

You thought you could read before...

What is a List? A Dictionary?

- List is a list…
- Dictionaries contain definitions… or key/value pairs.
- Lists can contain anything as items
- Dictionaries can too, but as values. Keys can be strings, numbers, (or tuples)
- Referencing item in a list: myList[3]
- Referencing value in a dict: myDict['keyname']

List comprehension

```
1 # using a for loop
2
3 numbsq = []
4
5 for num in range(8):
6     numbsq.append(num**2)
7
8 print(numbsq)
9
10 # with list comprehension
11
12 numbsqlc = [num**2 for num in range(8)]
13
14 print(numbsqlc)
```

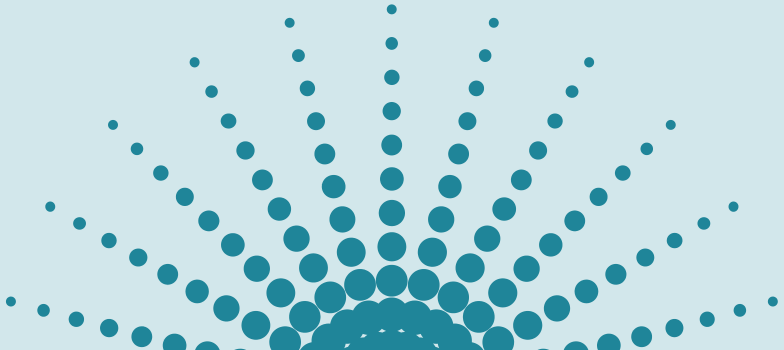
Dictionary example

```
names = ['Hal', 'Clark', 'Charles']
supernames = ['Green Lantern', 'Superman', 'Professor X']

hero_dict = {}
for name, super in zip (names, supernames):
    hero_dict[name] = super
print(hero_dict)

# {'Hal': 'Green Lantern', 'Clark': 'Superman', 'Charles': 'Professor X'}

#Dictionary Comprehension

hero_dict = dict(zip(names, supernames))

hero_dict2 = {value:key for key,value in hero_dict.items()}
print (hero_dict2)

# {'Green Lantern': 'Hal', 'Superman': 'Clark', 'Professor X': 'Charles'}
```

Don’t forget
to rate this session
in the BbWorld app.

REST

Everyone take a nap. Mike is going to talk now.



REST with the Requests library

```python
import requests
import json

key = 'KeyFromBbDevAppRegistration'
secret = 'SecretSquirrel'
target_domain = 'mybb.institution.edu'

api_base_path = '/learn/api/public/'
api_default_version = 'v1'

rest_api = {
    'token': f"{api_base_path}{api_default_version}/oauth2/token",
    'users': f"{api_base_path}{api_default_version}/users/:userId",
    'courses': f"{api_base_path}{api_default_version}/courses/:courseId",
    'memberships': f"{api_base_path}{api_default_version}/courses/:courseId/users",
    'enrollment': f"{api_base_path}{api_default_version}/courses/:courseId/users/:userId",
    'enrollments': f"{api_base_path}{api_default_version}/users/:userId/courses",
}
```

REST with the Requests library

```python
payload = {"grant_type": "client_credentials", "token": None}
# lets the auth and headers here
auth = ''
headers = ''


def set_token():
    global auth
    global headers
    oauth_url = f"https://{target_domain}{rest_api['token']}"
    print(f"[POST] OAuth2 Set Token URL: {oauth_url}")
    oauth_res = requests.post(oauth_url, data=payload, auth=(key, secret))
    print(f"=== Token Response ===\n{oauth_res.text}\n")
    payload['token'] = oauth_res.json()['access_token']
    auth = f"Bearer {payload['token']}"
    headers = {'Authorization': auth, 'Content-Type': 'application/json'}
```

REST with the Requests library

```python
def post_bb_data(version=None, **kwargs):
    bb_object = kwargs['api'].title()
    url = f"https://{target_domain}{rest_api[kwargs['api']]}"

    try:
        url = url.replace(kwargs['remove'], kwargs['replace'])
    except KeyError:
        pass

    data = kwargs['data']
    print(f"[POST] {bb_object} URL: {url}")
    rest_res = requests.post(url, json=data, headers=headers)
    print(f"=== {bb_object} Response ===\n{rest_res.text}\n")
    return rest_res.json()
```

REST with the Requests library

```python
def get_bb_data(version=None, **kwargs):
    bb_object = kwargs['api'].title()
    url = f"https://{target_domain}{rest_api[kwargs['api']]}"

    if version:
        url = url.replace(api_default_version, version)

    if kwargs['api'] == 'users' or kwargs['api'] == 'enrollments':
        url = url.replace(':userId', kwargs['userId'])
    elif kwargs['api'] == 'courses' or kwargs['api'] == 'memberships':
        url = url.replace(':courseId', kwargs['courseId'])
    elif kwargs['api'] == 'enrollment':
        url = url.replace(':courseId', kwargs['courseId'])
        url = url.replace(':userId', kwargs['userId'])

    print(f"[GET] {bb_object} URL: {url}")

    res = requests.get(url, headers=headers)

    print(f"=== {bb_object} Response ===\n{res.text}\n")
    return res.json()
```

REST with the Requests library

```
def main():
    set_token()
    print(auth)
    print(headers)
    userId = 'userName:mbechtel'
    courseId = 'externalId:LOR-CC-mbechtel'
    user = get_bb_data(api='users', userId=userId)
    print(f"user ID: {user['userName']} email: {user['contact']['email']}\n")
    course = get_bb_data(api='courses', courseId=courseId)
    print(f"course ID: {course['id']} external ID: {course['externalId']}\n")
    memberships = get_bb_data(api='memberships', courseId=courseId)
    print('memberships:', memberships, '\n')
    enrollments = get_bb_data(api='enrollments', userId=userId)
    print('user enrollments:', enrollments, '\n')
    enrollment = get_bb_data(
        api='enrollment', courseId=courseId, userId=userId)
    print('course user enrollment:', enrollment, '\n')
```

REST with the Requests library

```
    new_user_data = {
        'externalId': 'devcon19_user',
        'userName': 'devcon19user',
        'name': {
            'given': 'DevCon19',
            'family': 'User'
        },
        'password': '1234',
        'contact': {
            'email': 'no-reply@no.email.here'
        }
    }

    new_user = post_bb_data(api='users', data=new_user_data,
                            remove='/:userId', replace='')
    print('Created new user:', new_user)


if __name__ == '__main__':
    main()
```

DEVCON 19 austin

bbRest

You've seen the hard way, now check out the eas…less hard way!

REST, the less difficult way with bbRest

```python
from bbrest import BbRest

bbURL = 'https://bb.yourinst.edu'
restKey = 'RESTapiKeyFromDeveloper.blackboard.com'
restSecret = 'SetecAstronomy'


def main():
    bb = BbRest(restKey, restSecret, bbURL)
    userName = 'bstephens'
    courseId = 'dev-bs'
    user = bb.GetUser(userId=userName).json()
    print(f"user ID: {user['userName']} \
          email: {user['contact']['email']}")
    course = bb.GetCourse(courseId=courseId).json()
    print(f"course ID: {course['id']} \
          external ID: {course['externalId']}")
    memberships = bb.GetCourseMemberships(courseId=courseId).json()
    print('memberships:', memberships)
    enrollments = bb.GetUserMemberships(userId=userName).json()
    print('user enrollments:', enrollments)
    enrollment =bb.GetMembership(courseId=courseId,
                                 userId=userName).json()
    print('course user enrollment:', enrollment)
```

REST, the less difficult way with bbRest

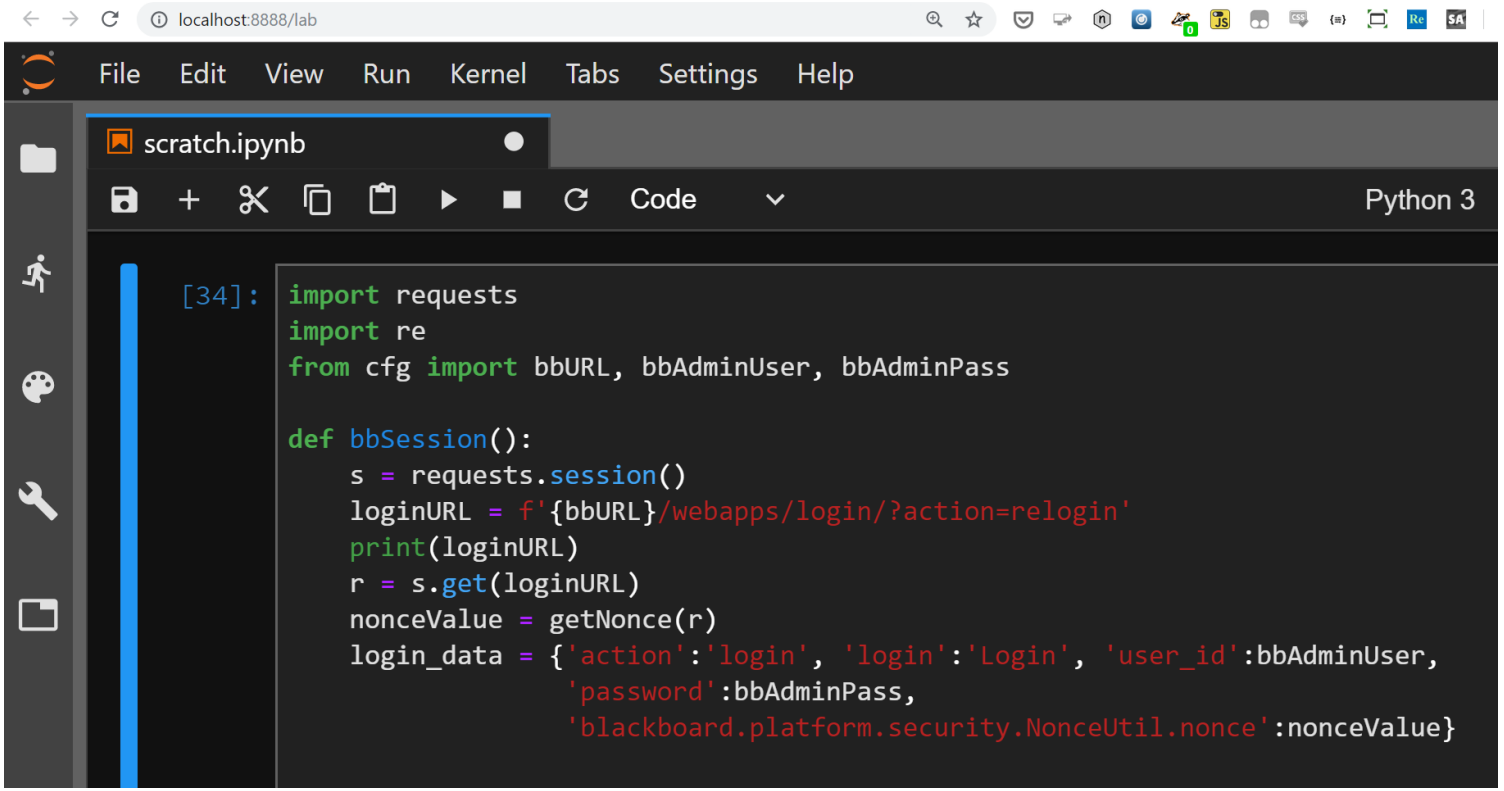
```
new_user_data = {
        'externalId': 'devcon19_user',
        'userName': 'devcon19user',
        'name': {
            'given': 'DevCon19',
            'family': 'User'
        },
        'password': 'p455w0rd',
        'contact': {
            'email': 'email@example.edu'
        }
    }

    new_user = bb.CreateUser(payload=new_user_data)
    print('Created new user:', new_user.json())


if __name__ == '__main__':
    main()
```

DEVCON 19 austin

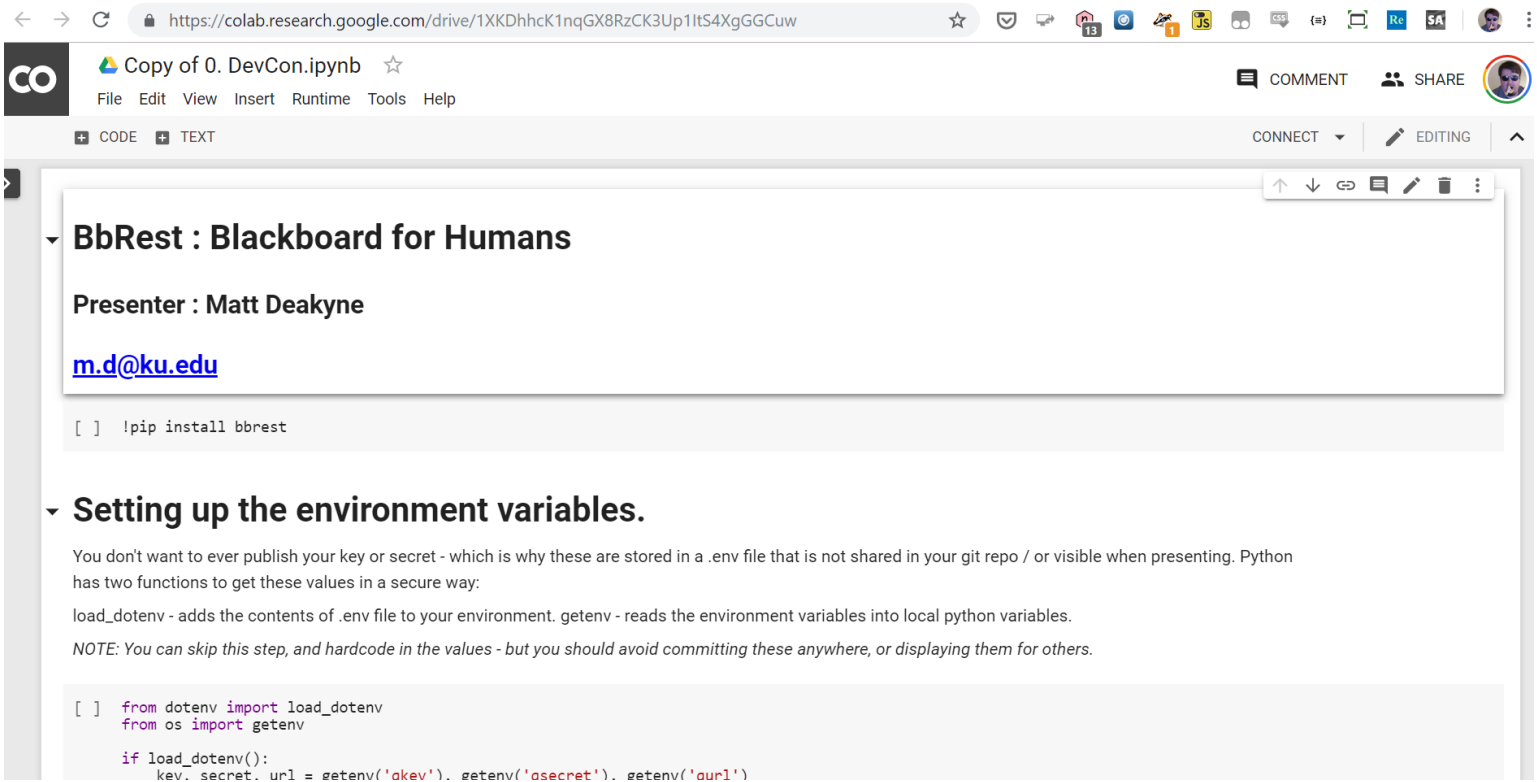
Jupyter Lab & Google Colaboratory

Working on the command line just isn't as fun as it used to be…

Jupyter Lab


localhost:8888/lab
File Edit View Run Kernel Tabs Settings Help
scratch.ipynb
Code
Python 3
[34]:
import requests
import re
from cfg import bbURL, bbAdminUser, bbAdminPass
def bbSession():
s = requests.session()
loginURL = f'{bbURL}/webapps/login/?action=relogin'
print(loginURL)
r = s.get(loginURL)
nonceValue = getNonce(r)
login_data = {'action':'login', 'login':'Login', 'user_id':bbAdminUser,
'password':bbAdminPass,
'blackboard.platform.security.NonceUtil.nonce':nonceValue}

Google Colaboratory



DEVCON 19 austin

QUESTIONS?

We can keep talking until they kick us out.

Don’t forget
to rate this session
in the BbWorld app.

DEVCON 19 austin