Python Scripting

Operating System Practice

João Vicente Ferreira Lima

Universidade Federal de Santa Maria
jvlima@inf.ufsm.br
http://www.inf.ufsm.br/~jvlima

2021/2

Outline

Outline

CSV example

A CSV file is a text file with comma-separated values (CSV).

```
10,13,11,10,"Sim, um pouco","Sim, um pouco",Feminino,39
9,14,18,16,Não,"Sim, um pouco",Masculino,30
15,18,14,18,"Sim, um pouco",Não,Feminino,20
5,15,12,8,Não,"Sim, um pouco",Feminino,20
9,20,14,14,"Sim, um pouco","Sim, um pouco",Feminino,28
```

Reading by line

```python
import csv
datafile = open('form-google-coleta.csv')
datareader = csv.reader(datafile)
for row in datareader:
    print('Row #' + str(datareader.line_num) + ' ' + str(row) )
```

```
Row #1 ['10', '13', '11', '10', 'Sim, um pouco', 'Sim, um pouco', 'Feminino', '39']
Row #2 ['9', '14', '18', '16', 'N\xc3\xa3o', 'Sim, um pouco', 'Masculino', '30']
Row #3 ['15', '18', '14', '18', 'Sim, um pouco', 'N\xc3\xa3o', 'Feminino', '20']
Row #4 ['5', '15', '12', '8', 'N\xc3\xa3o', 'Sim, um pouco', 'Feminino', '20']
Row #5 ['9', '20', '14', '14', 'Sim, um pouco', 'Sim, um pouco', 'Feminino', '28']
```

Reading a list of lines

```
import csv
datafile = open('form-google-coleta.csv')
datareader = csv.reader(datafile)

alldata = list(datareader)
print( alldata )
```

```
[['10', '13', '11', '10', 'Sim, um pouco', 'Sim, um pouco', 'Feminino', '39'], ['9',
```

Writing a CSV

```python
import csv
outputFile = open('output.csv', 'w')
outputWriter = csv.writer(outputFile)
outputWriter.writerow(['Hello ola', 'eggs', 'bacon', 'ham'])
outputWriter.writerow([1, 2, 3.14, 4])
outputFile.close()
```

```
Hello ola,eggs,bacon,ham
1,2,3.14,4
```

Outline

Reading JSON

Example from the book *Automate the Boring Stuff with Python*.

```
#!/usr/bin/env python3
import json
stringOfJsonData = '{"name": "Zophie", "isCat": true, "miceCaught": 0,
     "felineIQ": null}'
jsonData = json.loads(stringOfJsonData)
print(jsonData)
```

```
{u'miceCaught': 0, u'isCat': True, u'felineIQ': None, u'name': u'Zophie'}
```

Writing JSON

Example from the book *Automate the Boring Stuff with Python*.

```
#!/usr/bin/env python3
import json
pythonData = {'name': 'Zophie', 'isCat': True, 'miceCaught': 0, 'felineIQ': None}
jsonString = json.dumps(pythonData)
print(jsonString)
```

```
{"miceCaught": 0, "isCat": true, "felineIQ": null, "name": "Zophie"}
```

Example: Weather data

Example from the book *Automate the Boring Stuff with Python*.

```python
import json, requests, sys

if len(sys.argv) < 2:
    print('Usage: quickWeather.py location')
    sys.exit()
location = ' '.join(sys.argv[1:])

# Download the JSON data from OpenWeatherMap.org's API
url = 'http://api.openweathermap.org/data/2.5/forecast/' +
    'daily?q=%s&cnt=3' % (location)
response = requests.get(url)
response.raise_for_status()

# Load JSON data into a Python variable.
weatherData = json.loads(response.text)
```

Example: Weather data

Example from the book *Automate the Boring Stuff with Python*.

```
# Print weather descriptions.
w = weatherData['list']
print('Current weather in %s:' % (location))
print(w[0]['weather'][0]['main'], '-',
      w[0]['weather'][0]['description'])
print()
print('Tomorrow:')
print(w[1]['weather'][0]['main'], '-',
      w[1]['weather'][0]['description'])
print()
print('Day after tomorrow:')
print(w[2]['weather'][0]['main'], '-',
      w[2]['weather'][0]['description'])
```

JSON practice

In this exercicie, we are going to read weather information using the Yahoo Weather API: https://developer.yahoo.com/weather/
In the website, a simple query to the current conditions for Gramado (Brazil) is:

```
select item.condition from weather.forecast where woeid in
 (select woeid from geo.places(1) where text="Gramado")
```

JSON pratice

```
select item.condition from weather.forecast where woeid in
 (select woeid from geo.places(1) where text="Gramado")
```

The URL:

```
https://query.yahooapis.com/v1/public/yql?q=select%20
item.condition%20from%20weather.forecast%20where%20%20
woeid%20in%20(select%20woeid%20from%20geo.places(1)%20
where%20text%3D%22Gramado%22)&format=json&env=store%3A%2F%2F
datatables.org%2Falltableswithkeys
```

JSON practice

The response:

```json
{
 "query": {
  "count": 1,
  "created": "2016-09-26T18:03:52Z",
  "lang": "en-US",
  "results": {
   "channel": {
    "item": {
     "condition": {
      "code": "32",
      "date": "Mon, 26 Sep 2016 02:00 PM BRT",
      "temp": "64",
      "text": "Sunny"
     }
    }
   }
  }
 }
}
```

Outline

Command-line arguments

Command-line arguments in Python are avaible through `sys.argv`.

```
#!/usr/bin/env python3
import sys

print('Command-line arguments are: ' + str(sys.argv))
print('Total: ' + str(len(sys.argv)))
```

```
Command-line arguments are: ['']
Total: 1
```

The argparse module makes easy to parse command-line arguments. Refer to the manual for all options.

Simple example

```python
#!/usr/bin/env python3
import argparse

import argparse
parser = argparse.ArgumentParser()
parser.add_argument('--foo', help='foo help')
args = parser.parse_args()

parser.print_help()
```

```
usage: [-h] [--foo FOO]

optional arguments:
  -h, --help   show this help message and exit
  --foo FOO    foo help
```

Adding numbers

```
#!/usr/bin/env python3
import argparse

parser = argparse.ArgumentParser(
    description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int,
                    nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate',
                    action='store_const',
                    const=sum, default=max,
               help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

Adding numbers

```
$ python prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
 N             an integer for the accumulator

optional arguments:
 -h, --help   show this help message and exit
 --sum        sum the integers (default: find the max)
```

Outline

HTTP request

```
import http.client

def check_webserver(address, port, resource):
  if not resource.startswith('/'):
    resource = '/' + resource
  try:
    conn = http.client.HTTPConnection(address, port)
    req = conn.request('GET', resource)
    response = conn.getresponse()
    print("Response status: {0}".format(response.status))
  except sock.error as e:
    print("HTTP connection failed: {0}".format(e))
  finally:
    conn.close()

  if response.status in [200, 301]:
    return True
  else:
    return False
```

HTTP request

```python
if __name__ == '__main__':
    address = 'www.inf.ufsm.br'
    port = 80
    resource = 'index'
    check = check_webserver(address, port, resource)
    print("Check result: " + str(check))
```

```
HTTP connection successfull
Response status: 200
Check result: True
```

Outline

Opening a browser

```
#!/usr/bin/env python3
import webbrowser

webbrowser.open('http://www.inf.ufsm.br')
```

Downloading

```
pip3 install requests
```

```
#!/usr/bin/env python3
import requests

res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')
print( len(res.text) )
print( res.text[:100] )
```

```
178981
The Project Gutenberg EBook of Romeo and Juliet, by William Shakespeare

This eBook is for the us
```

Errors

```python
#!/usr/bin/env python3
import requests

res = requests.get('http://inventingfoo.com')
try:
    res.raise_for_status()
except Exception as exc:
    print('There was a problem: %s' % (exc))
```

```
There was a problem: 404 Client Error: Not Found for url: ..
```

Saving contents

```
#!/usr/bin/env python3
import requests

res = requests.get('http://www.gutenberg.org/cache/epub/1112/pg1112.txt')
output = open('RomeoAndJuliet.txt', 'wb')
for chunk in res.iter_content(100000):
    output.write(chunk)
output.close()
```

HTML

`pip3 install beautifulsoup4` https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
#!/usr/bin/env python3
import requests, bs4

res = requests.get('http://www.inf.ufsm.br')
res.raise_for_status()
soup = bs4.BeautifulSoup(res.text)
print( type(soup) )
```

HTML

```python
#!/usr/bin/env python3
import requests, bs4

res = requests.get('http://www.inf.ufsm.br')
res.raise_for_status()
soup = bs4.BeautifulSoup(res.text)
for tag in soup.find_all(True):
    print(tag.name)
```

```
html
head
meta
script
meta
```

HTML

```
#!/usr/bin/env python3
import requests, bs4

res = requests.get('http://www.inf.ufsm.br')
res.raise_for_status()
soup = bs4.BeautifulSoup(res.text)
for tag in soup.find_all('img'):
    print(tag.attrs['src'])
```

```
images/phd_logo.png
images/h_comics.png
images/h_store.png
images/h_events.png
images/h_phdtv.png
mages/h_about.png
```

HTML

```python
#!/usr/bin/env python3
import requests, bs4

res = requests.get('http://www.inf.ufsm.br')
res.raise_for_status()
soup = bs4.BeautifulSoup(res.text)
elems = soup.select('img')

print(elems)
print( elems[0].get('src') )
print( elems[0].attrs )
```

```
[<img alt="Send this page to somebody" id="icon-sendto" src="http://www.inf.ufsm.br/i
http://www.inf.ufsm.br/index/mail_icon.gif
{'src': 'http://www.inf.ufsm.br/index/mail_icon.gif', 'alt': 'Send this page to someb
```

XKCD comic

```
import os, requests, bs4

url = 'http://xkcd.com'
os.makedirs('xkcd', exist_ok=True)
print('Downling page %s...' % url)
res = requests.get(url)
res.raise_for_status()

soup = bs4.BeautifulSoup(res.text)
```

XKCD comic

```python
for fig in soup.select('#comic img'):
  comicUrl = 'http:' + fig.get('src')
  print('Downloading image %s...' % (comicUrl))
  res = requests.get(comicUrl)
  res.raise_for_status()

  imgFile = open(os.path.join('xkcd',
                 os.path.basename(comicUrl)), 'wb')

  for chunk in res.iter_content(100000):
    imgFile.write(chunk)
  imgFile.close()

print('Done!')
```

Outline

Regex objects

```python
#!/usr/bin/env python3
import re

p = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
m = p.search('The number is 415-555-3232.')

print('Number found: ' + m.group())
```

```
Number found: 415-555-3232
```

Regular expression matching steps

1. Import the regex module `import re`
2. Create a `Regex` object with the `re.compile()` function (raw string).
3. Pass the string into the `Regex` object's `search()` method, which retursn a `Match` object.
4. Call `group()` method to return a string of actual matched text.

Special sequences

- `\d` matches any decimal digit (`[0-9]`).
- `\D` matches any non-digit (`[^0-9]`).
- `\s` matches any whitespace (`[ \t\n\r\f\v]`).
- `\S` matches any non-whitespace (`[^ \t\n\r\f\v]`).
- `\w` matches any alphanumeric (`[a-zA-Z0-9_]`).
- `\W` matches any non-alphanumeric (`[^a-zA-Z0-9_]`).

Metacharacters:

```
. ^ $ * + ? { } [ ] \ | ( )
```

Matches and attributes

Matches:

`match()`	Determine if the RE matches at the beginning of the string.
`search()`	Scan through a string, looking for any location where this RE matches.
`findall()`	Find all substrings where the RE matches, and returns them as a list.

Attributes:

`group()`	Return the string matched by the RE.
`start()`	Return the starting position of the match.
`end()`	Return the ending position of the match.
`span()`	Return a tuple containing the (start, end) positions of the match.

Grouping

```python
#!/usr/bin/env python3
import re

p = re.compile(r'(\d\d)-(\d\d\d\d)')
m = p.search('My number is 11-3344.')

print(m.group(1))
print(m.group(2))
print(m.group())
```

```
11
3344
11-3344
```

Optional Matching

```python
#!/usr/bin/env python3
import re

p = re.compile(r'Bat(wo)?man')
m1 = p.search('The adventures of Batman')
print(m1.group())

m2 = p.search('The adventures of Batwoman')
print(m2.group())
```

```
Batman
Batwoman
```

Splitting

```
#!/usr/bin/env python3
import re

p = re.compile(r'\W+')
m = p.split('This is a simple split test.')
print(m)
```

```
['This', 'is', 'a', 'simple', 'split', 'test', '']
```

Search and replace

```python
#!/usr/bin/env python3
import re

p = re.compile(r'(blue|white|red)')
m = p.sub('colour', 'blue socks, red shoes, and white shirt')
print(m)
```

```
colour socks, colour shoes, and colour shirt
```

Outline

SSH basics

```
ssh localhost
```

If the command above asked for a password, try to generate a pair of SSH keys (public and private).

```
ssh-keygen -t rsa
```

The command generate the public key `~/.ssh/id_rsa.pub`, then execute this command to login by SSH without password (locally):

```
cat ~/.ssh/id_rsa.pub > ~/.ssh/authorized_keys
```

If your want to copy files to a remote server:

```
scp file.txt ssh.inf.ufsm.br:
```

Or a directory from the remote server:

```
scp -r ssh.inf.ufsm.br:public_html .
```

Running a SSH server

If you do not have a SSH server:

```
sudo apt-get install openssh-server
sudo service ssh start
```

SSH client

In Python, you may need to install the `paramiko` package:

```
sudo apt-get install python3-paramiko
```

```
#!/usr/bin/env python3
import paramiko

hostname = 'localhost'
port = 22
username = 'ncc'
password = 'ncc'

if __name__ == "__main__":
  paramiko.util.log_to_file('paramiko.log')
  s = paramiko.SSHClient()
  s.load_system_host_keys()
  s.connect(hostname, port, username, password)
  stdin, stdout, stderr = s.exec_command('ls')
  print( stdout.read() )
  s.close()
```

Connecting with SSH keys

```python
#!/usr/bin/env python3
import paramiko

hostname = 'ssh.inf.ufsm.br'
port = 22
username = 'jvlima'
key_file = '/Users/jvlima/.ssh/id_rsa'

if __name__ == "__main__":
  paramiko.util.log_to_file('paramiko.log')
  key = paramiko.RSAKey.from_private_key_file(key_file)
  s = paramiko.SSHClient()
  s.load_system_host_keys()
  s.connect(hostname, port, pkey=key)
  stdin, stdout, stderr = s.exec_command('ls public_html')
  print( stdout.read() )
  s.close()
```

```
b'algo2016a\nelc1066\nindex.html\nl22016a\npg1112.txt\npso2016b\n'
```

Retrieving files

```
import paramiko, os
hostname = 'ssh.inf.ufsm.br'
port = 22
username = 'jvlima'
key_file = '/Users/jvlima/.ssh/id_rsa'
dir_path = '/home/profs/jvlima/public_html/pso2016b'
if __name__ == "__main__":
  key = paramiko.RSAKey.from_private_key_file(key_file)
  t = paramiko.Transport((hostname, port))
  t.connect(username=username, pkey=key)

  sftp = paramiko.SFTPClient.from_transport(t)
  files = sftp.listdir(dir_path)
  for f in files:
    print("Retrieving " + f)
    sftp.get(os.path.join(dir_path, f), f)
  t.close()
```

```
Retrieving pso2016b.html
```

Outline

Sockets

Reading 1024 bytes from Google (HTTP request).

```
import socket, sys

try:
  s = socket.socket()
except socket.error as err:
  print("Socket error: {0}".format(err))
try:
  ip = socket.gethostbyname('www.google.com.br')
except socket.gaierror:
  print("Error: DNS error")
  sys.exit()

print("Google IP is: " + str(ip))
s.connect((ip, 80))
s.send(b"GET / HTTP/1.0\n\n")
print(s.recv(1024))
s.close()
```

Sockets

```
Google IP is: 172.217.29.35
HTTP/1.0 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.com.br/?gfe_rd=cr&ei=7zXrV5aPLoqF8Qfw-oNw
Content-Length: 260
Date: Wed, 28 Sep 2016 03:15:59 GMT

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.com.br/?gfe_rd=cr&amp;ei=7zXrV5aPLoqF8Qfw-oNw">here</A>.
</BODY></HTML>
```

Port checker

```
#!/usr/bin/env python3
import socket, sys

def check_server(address, port):
  s = socket.socket()
  try:
    s.connect((address, port))
    print("Connected to {0} on port {1}".format(address, port))
    return True
  except socket.error as e:
    print("Connection to {0} port {1} failed: {2}".format(
      address, port, e))
    return False
```

Port checker

```python
if __name__ == '__main__':
    address = 'www.inf.ufsm.br'
    port = 80
    print("Checking host " + address + " port " + str(port))
    check = check_server(address, port)
    print("Check result: " + str(check))
```

```
Checking host www.inf.ufsm.br port 80
Connected to www.inf.ufsm.br on port 80
Check result: True
```

Outline

filecmp

The `filecmp` module allows fast comparisons of files and directories.
For files, it returns `True` and `False`:

```
#!/usr/bin/env python3
import sys
import filecmp

if len(sys.argv) < 3:
  print('Error: need 2 files')
  sys.exit()

if filecmp.cmp(sys.argv[1], sys.argv[2]):
  print('Equal files')
else:
  print('Different files')
```

filecmp

In directories, `filecmp` has a number of attributes:

```
#!/usr/bin/env python3
import sys
import filecmp

if len(sys.argv) < 3:
  print('Error: need 2 directories')
  sys.exit()

print(filecmp.dircmp(sys.argv[1], sys.argv[2]).report())
```

```
$ python 13_dircmp.py ../tmp ../tmp1
diff ../tmp ../tmp1
Only in ../tmp : ['b.txt.gz', 'foo', 'spam.txt.gz']
Identical files : ['spam.txt']
None
```

MD5 checksum

Using a MD5 Checksum we are able to compare files byte-by-byte and be 100 percent accurate.

```
#!/usr/bin/env python3
import sys
import hashlib

def create_checksum(path):
    fp = open(path)
    checksum = hashlib.md5()
    while True:
        buffer = fp.read(8192)
        if not buffer:
            break
        checksum.update(buffer.encode('utf8'))
    fp.close()
    checksum = checksum.digest()
    return checksum
```

MD5 checksum

```
if __name__ == '__main__':
    if len(sys.argv) < 3:
        print('ERROR: need 2 files as parameters')
        sys.exit()
    file1 = sys.argv[1]
    file2 = sys.argv[2]
    if create_checksum(file1) == create_checksum(file2):
        print('Equal files.')
    else:
        print('Different files.')
```