Examples

??????

- Custom Class
- Dictionary/Set/List Data
- Set Methods

```
def get_event_date(event):
  return event.date

def current_users(events):
  events.sort(key=get_event_date)
  machines = {}
  for event in events:
    if event.machine not in machines:
      machines[event.machine] = set()
    if event.type == "login":
      machines[event.machine].add(event.user)
    elif event.type == "logout":
      machines[event.machine].remove(event.user)
  return machines

def generate_report(machines):
  for machine, users in machines.items():
    if len(users) > 0:
      user_list = ", ".join(users)
      print("{}: {}".format(machine, user_list))

class Event:
  def __init__(self, event_date, event_type, machine_name, user):
    self.date = event_date
    self.type = event_type
    self.machine = machine_name
    self.user = user
```

```
events = [
  Event('2020-01-21 12:45:46', 'login', 'myworkstation.local', 'jordan'),
  Event('2020-01-22 15:53:42', 'logout', 'webserver.local', 'jordan'),
  Event('2020-01-21 18:53:21', 'login', 'webserver.local', 'lane'),
  Event('2020-01-22 10:25:34', 'logout', 'myworkstation.local', 'jordan'),
  Event('2020-01-21 08:20:01', 'login', 'webserver.local', 'jordan'),
  Event('2020-01-23 11:24:35', 'login', 'mailserver.local', 'chris'),
]

users = current_users(events)
print(users)
# Output: {'webserver.local': {'lane'}, 'myworkstation.local': set(), 'mailserver.local': {'chris'}}

generate_report(users)
# Output:
# webserver.local: lane
# mailserver.local: chris
```

?? Syslog

- dictionary.get()
- re.search()
- with open() as f

```
import re
import sys

logfile = sys.argv[1]
usernames = {}
with open(logfile) as f:
  for line in f:
    if "CRON" not in line:
      continue
    pattern = r"USER \((\w+)\)$"
    result = re.search(pattern, line)

    if result is None:
      continue
    name = result[1]
    usernames[name] = usernames.get(name, 0) + 1
```

```
print(usernames)
```

???

fishy.log:

```
July 31 02:25:52 mycomputername system[41921]: WARN Failed to start CPU thread[39016]
July 31 02:34:37 mycomputername kernel[32280]: INFO Loading...
July 31 02:36:44 mycomputername NetworkManager[90289]: WARN Failed to start CPU thread[39016]
July 31 02:39:01 mycomputername CRON[89330]: ERROR Unable to perform package upgrade
July 31 02:45:39 mycomputername utility[57387]: INFO Access permitted
July 31 02:58:44 mycomputername process[44707]: WARN Computer needs to be turned off and on again
July 31 02:59:35 mycomputername system[55024]: WARN Packet loss
July 31 03:09:30 mycomputername kernel[40705]: ERROR The cake is a lie!
July 31 03:23:16 mycomputername cacheclient[57185]: INFO Checking process [16121]
July 31 03:26:56 mycomputername cacheclient[90154]: INFO Healthy resource usage
July 31 03:28:52 mycomputername CRON[55441]: INFO Loading...
July 31 03:29:34 mycomputername dhcpclient[69232]: ERROR Unable to download more RAM
July 31 03:34:41 mycomputername NetworkManager[14120]: ERROR 404 error not found
July 31 03:36:26 mycomputername dhcpclient[79731]: ERROR The cake is a lie!
July 31 03:38:24 mycomputername CRON[92141]: INFO Access permitted
July 31 03:40:00 mycomputername dhcpclient[40114]: INFO Starting sync
July 31 03:42:45 mycomputername utility[53726]: INFO I'm sorry Dave. I'm afraid I can't do that
July 31 03:47:07 mycomputername NetworkManager[63805]: WARN Please reboot user
July 31 04:09:16 mycomputername CRON[52593]: WARN PC Load Letter
July 31 04:11:32 mycomputername CRON[51253]: ERROR: Failed to start CRON job due to script syntax error. Inform the CRON job owner!
July 31 04:11:32 mycomputername jam_tag=psim[84082]: ERROR ID: 10t
July 31 04:12:05 mycomputername utility[63418]: INFO Successfully connected
July 31 04:14:22 mycomputername utility[53225]: ERROR I am error
July 31 04:31:00 mycomputername NetworkManager[23060]: ERROR Out of yellow ink, specifically, even though you want grayscale
```

find_error.py

Usage: `./find_error.py fishy.log`

```
import sys
import os
```

```
import re

def error_search(log_file):
    error = input("What is the error? ")
    returned_errors = []

    with open(log_file, mode='r', encoding='UTF-8') as file:
        for log in file.readlines():
            error_patterns = ["error"]
            for i in range(len(error.split(' '))):
                error_patterns.append(r"{}".format(error.split(' ')[i].lower()))

            if all(re.search(error_pattern, log.lower()) for error_pattern in error_patterns):
                returned_errors.append(log)

        file.close()
    return returned_errors

def file_output(returned_errors):
    with open(os.path.expanduser('~') + '/data/errors_found.log', 'w') as file:
        for error in returned_errors:
            file.write(error)

    file.close()

if __name__ == "__main__":
    log_file = sys.argv[1]
    returned_errors = error_search(log_file)
    file_output(returned_errors)
    sys.exit(0)
```

?? Syslog 2

syslog.log :

```
Jan 31 00:09:39 ubuntu.local ticky: INFO Created ticket [#4217] (mdouglas)
Jan 31 00:16:25 ubuntu.local ticky: INFO Closed ticket [#1754] (noel)
Jan 31 00:21:30 ubuntu.local ticky: ERROR The ticket was modified while updating (breee)
Jan 31 00:44:34 ubuntu.local ticky: ERROR Permission denied while closing ticket (ac)
Jan 31 01:00:50 ubuntu.local ticky: INFO Commented on ticket [#4709] (blossom)
```

Jan 31 01:29:16 ubuntu.local ticky: INFO Commented on ticket [#6518] (rr.robinson)
Jan 31 01:33:12 ubuntu.local ticky: ERROR Tried to add information to closed ticket (mcintosh)
Jan 31 01:43:10 ubuntu.local ticky: ERROR Tried to add information to closed ticket (jackowens)
Jan 31 01:49:29 ubuntu.local ticky: ERROR Tried to add information to closed ticket (mdouglas)
Jan 31 02:30:04 ubuntu.local ticky: ERROR Timeout while retrieving information (oren)
Jan 31 02:55:31 ubuntu.local ticky: ERROR Ticket doesn't exist (xlg)
Jan 31 03:05:35 ubuntu.local ticky: ERROR Timeout while retrieving information (ahmed.miller)
Jan 31 03:08:55 ubuntu.local ticky: ERROR Ticket doesn't exist (blossom)
Jan 31 03:39:27 ubuntu.local ticky: ERROR The ticket was modified while updating (bpacheco)
Jan 31 03:47:24 ubuntu.local ticky: ERROR Ticket doesn't exist (enim.non)
Jan 31 04:30:04 ubuntu.local ticky: ERROR Permission denied while closing ticket (rr.robinson)
Jan 31 04:31:49 ubuntu.local ticky: ERROR Tried to add information to closed ticket (oren)
Jan 31 04:32:49 ubuntu.local ticky: ERROR Timeout while retrieving information (mcintosh)
Jan 31 04:44:23 ubuntu.local ticky: ERROR Timeout while retrieving information (ahmed.miller)
Jan 31 04:44:46 ubuntu.local ticky: ERROR Connection to DB failed (jackowens)
Jan 31 04:49:28 ubuntu.local ticky: ERROR Permission denied while closing ticket (flavia)
Jan 31 05:12:39 ubuntu.local ticky: ERROR Tried to add information to closed ticket (oren)
Jan 31 05:18:45 ubuntu.local ticky: ERROR Tried to add information to closed ticket (sri)
Jan 31 05:23:14 ubuntu.local ticky: INFO Commented on ticket [#1097] (breee)
Jan 31 05:35:00 ubuntu.local ticky: ERROR Connection to DB failed (nonummy)
Jan 31 05:45:30 ubuntu.local ticky: INFO Created ticket [#7115] (noel)
Jan 31 05:51:30 ubuntu.local ticky: ERROR The ticket was modified while updating (flavia)
Jan 31 05:57:46 ubuntu.local ticky: INFO Commented on ticket [#2253] (nonummy)
Jan 31 06:12:02 ubuntu.local ticky: ERROR Connection to DB failed (oren)
Jan 31 06:26:38 ubuntu.local ticky: ERROR Timeout while retrieving information (xlg)
Jan 31 06:32:26 ubuntu.local ticky: INFO Created ticket [#7298] (ahmed.miller)
Jan 31 06:36:25 ubuntu.local ticky: ERROR Timeout while retrieving information (flavia)
Jan 31 06:57:00 ubuntu.local ticky: ERROR Connection to DB failed (jackowens)
Jan 31 06:59:57 ubuntu.local ticky: INFO Commented on ticket [#7255] (oren)
Jan 31 07:59:56 ubuntu.local ticky: ERROR Ticket doesn't exist (flavia)
Jan 31 08:01:40 ubuntu.local ticky: ERROR Tried to add information to closed ticket (jackowens)
Jan 31 08:03:19 ubuntu.local ticky: INFO Closed ticket [#1712] (britanni)
Jan 31 08:22:37 ubuntu.local ticky: INFO Created ticket [#2860] (mcintosh)
Jan 31 08:28:07 ubuntu.local ticky: ERROR Timeout while retrieving information (montanap)
Jan 31 08:49:15 ubuntu.local ticky: ERROR Permission denied while closing ticket (britanni)
Jan 31 08:50:50 ubuntu.local ticky: ERROR Permission denied while closing ticket (montanap)
Jan 31 09:04:27 ubuntu.local ticky: ERROR Tried to add information to closed ticket (noel)
Jan 31 09:15:41 ubuntu.local ticky: ERROR Timeout while retrieving information (oren)
Jan 31 09:18:47 ubuntu.local ticky: INFO Commented on ticket [#8385] (mdouglas)
Jan 31 09:28:18 ubuntu.local ticky: INFO Closed ticket [#2452] (jackowens)

Jan 31 09:41:16 ubuntu.local ticky: ERROR Connection to DB failed (ac)
Jan 31 10:11:35 ubuntu.local ticky: ERROR Timeout while retrieving information (blossom)
Jan 31 10:21:36 ubuntu.local ticky: ERROR Permission denied while closing ticket (montanap)
Jan 31 11:04:02 ubuntu.local ticky: ERROR Tried to add information to closed ticket (breee)
Jan 31 11:19:37 ubuntu.local ticky: ERROR Connection to DB failed (sri)
Jan 31 11:22:06 ubuntu.local ticky: ERROR Timeout while retrieving information (montanap)
Jan 31 11:31:34 ubuntu.local ticky: ERROR Permission denied while closing ticket (ahmed.miller)
Jan 31 11:40:25 ubuntu.local ticky: ERROR Connection to DB failed (mai.hendrix)
Jan 31 11:47:07 ubuntu.local ticky: INFO Commented on ticket [#4562] (ac)
Jan 31 11:58:33 ubuntu.local ticky: ERROR Tried to add information to closed ticket (ahmed.miller)
Jan 31 12:00:17 ubuntu.local ticky: INFO Created ticket [#7897] (kirknixon)
Jan 31 12:02:49 ubuntu.local ticky: ERROR Permission denied while closing ticket (mai.hendrix)
Jan 31 12:20:23 ubuntu.local ticky: ERROR Connection to DB failed (kirknixon)
Jan 31 12:20:40 ubuntu.local ticky: ERROR Ticket doesn't exist (flavia)
Jan 31 12:24:32 ubuntu.local ticky: INFO Created ticket [#5784] (sri)
Jan 31 12:50:10 ubuntu.local ticky: ERROR Permission denied while closing ticket (blossom)
Jan 31 12:58:16 ubuntu.local ticky: ERROR Tried to add information to closed ticket (nonummy)
Jan 31 13:08:10 ubuntu.local ticky: INFO Closed ticket [#8685] (rr.robinson)
Jan 31 13:48:45 ubuntu.local ticky: ERROR The ticket was modified while updating (breee)
Jan 31 14:13:00 ubuntu.local ticky: INFO Commented on ticket [#4225] (noel)
Jan 31 14:38:50 ubuntu.local ticky: ERROR The ticket was modified while updating (enim.non)
Jan 31 14:41:18 ubuntu.local ticky: ERROR Timeout while retrieving information (xlg)
Jan 31 14:45:55 ubuntu.local ticky: INFO Closed ticket [#7948] (noel)
Jan 31 14:50:41 ubuntu.local ticky: INFO Commented on ticket [#8628] (noel)
Jan 31 14:56:35 ubuntu.local ticky: ERROR Tried to add information to closed ticket (noel)
Jan 31 15:27:53 ubuntu.local ticky: ERROR Ticket doesn't exist (blossom)
Jan 31 15:28:15 ubuntu.local ticky: ERROR Permission denied while closing ticket (enim.non)
Jan 31 15:44:25 ubuntu.local ticky: INFO Closed ticket [#7333] (enim.non)
Jan 31 16:17:20 ubuntu.local ticky: INFO Commented on ticket [#1653] (noel)
Jan 31 16:19:40 ubuntu.local ticky: ERROR The ticket was modified while updating (mdouglas)
Jan 31 16:24:31 ubuntu.local ticky: INFO Created ticket [#5455] (ac)
Jan 31 16:35:46 ubuntu.local ticky: ERROR Timeout while retrieving information (oren)
Jan 31 16:53:54 ubuntu.local ticky: INFO Commented on ticket [#3813] (mcintosh)
Jan 31 16:54:18 ubuntu.local ticky: ERROR Connection to DB failed (bpacheco)
Jan 31 17:15:47 ubuntu.local ticky: ERROR The ticket was modified while updating (mcintosh)
Jan 31 17:29:11 ubuntu.local ticky: ERROR Connection to DB failed (oren)
Jan 31 17:51:52 ubuntu.local ticky: INFO Closed ticket [#8604] (mcintosh)
Jan 31 18:09:17 ubuntu.local ticky: ERROR The ticket was modified while updating (noel)
Jan 31 18:43:01 ubuntu.local ticky: ERROR Ticket doesn't exist (nonummy)
Jan 31 19:00:23 ubuntu.local ticky: ERROR Timeout while retrieving information (blossom)

```
Jan 31 19:20:22 ubuntu.local ticky: ERROR Timeout while retrieving information (mai.hendrix)
Jan 31 19:59:06 ubuntu.local ticky: INFO Created ticket [#6361] (enim.non)
Jan 31 20:02:41 ubuntu.local ticky: ERROR Timeout while retrieving information (xlg)
Jan 31 20:21:55 ubuntu.local ticky: INFO Commented on ticket [#7159] (ahmed.miller)
Jan 31 20:28:26 ubuntu.local ticky: ERROR Connection to DB failed (breee)
Jan 31 20:35:17 ubuntu.local ticky: INFO Created ticket [#7737] (nonummy)
Jan 31 20:48:02 ubuntu.local ticky: ERROR Connection to DB failed (mdouglas)
Jan 31 20:56:58 ubuntu.local ticky: INFO Closed ticket [#4372] (oren)
Jan 31 21:00:23 ubuntu.local ticky: INFO Commented on ticket [#2389] (sri)
Jan 31 21:02:06 ubuntu.local ticky: ERROR Connection to DB failed (breee)
Jan 31 21:20:33 ubuntu.local ticky: INFO Closed ticket [#3297] (kirknixon)
Jan 31 21:29:24 ubuntu.local ticky: ERROR The ticket was modified while updating (blossom)
Jan 31 22:58:55 ubuntu.local ticky: INFO Created ticket [#2461] (jackowens)
Jan 31 23:25:18 ubuntu.local ticky: INFO Closed ticket [#9876] (blossom)
Jan 31 23:35:40 ubuntu.local ticky: INFO Created ticket [#5896] (mcintosh)
```

ticky_check.py

Usage: `./ticky_check.py`

```
#!/usr/bin/env python3
import sys
import re
import operator
import csv

# Dict: Count number of entries for each user
per_user = {}  # Splitting between INFO and ERROR
# Dict: Number of different error messages
errors = {}

# * Read file and create dictionaries
with open('syslog.log') as file:
    # read each line
    for line in file.readlines():
        # regex search
        # * Sample Line of log file
        # "May 27 11:45:40 ubuntu.local ticky: INFO: Created ticket [#1234] (username)"
        match = re.search(
            r"ticky: ([\w+]*):? ([\w' ]*)[\[[#0-9]*\]?]? ?\((.*)\)$", line)
```

```
        code, error_msg, user = match.group(1), match.group(2), match.group(3)

        # Populates error dict with ERROR messages from log file
        if error_msg not in errors.keys():
            errors[error_msg] = 1
        else:
            errors[error_msg] += 1
        # Populates per_user dict with users and default values
        if user not in per_user.keys():
            per_user[user] = {}
            per_user[user]['INFO'] = 0
            per_user[user]['ERROR'] = 0
        # Populates per_user dict with users logs entry
        if code == 'INFO':
            if user not in per_user.keys():
                per_user[user] = {}
                per_user[user]['INFO'] = 0
            else:
                per_user[user]["INFO"] += 1
        elif code == 'ERROR':
            if user not in per_user.keys():
                per_user[user] = {}
                per_user[user]['INFO'] = 0
            else:
                per_user[user]['ERROR'] += 1



# Sorted by VALUE (Most common to least common)
errors_list = sorted(errors.items(), key=operator.itemgetter(1), reverse=True)

# Sorted by USERNAME
per_user_list = sorted(per_user.items(), key=operator.itemgetter(0))

file.close()
# Insert at the beginning of the list
errors_list.insert(0, ('Error', 'Count'))
per_user_list.insert(0, ('Username', {'INFO': 'INFO', 'ERROR': 'ERROR'}))

# * Create CSV file user_statistics
with open('user_statistics.csv', 'w', newline='') as user_csv:
```

```
    for key, value in per_user_list:
        user_csv.write(str(key) + ',' +
                       str(value['INFO']) + ',' + str(value['ERROR'])+'\n')

# * Create CSV error_message
with open('error_message.csv', 'w', newline='') as error_csv:
    for key, value in errors_list:
        error_csv.write(str(key) + ',' + str(value) + '\n')
```

csv_to_html.py

Usage: `./csv_to_html.py user_statistics.csv /var/www/html/<html-filename>.html`

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



import sys
import csv
import os

def process_csv(csv_file):
    """Turn the contents of the CSV file into a list of lists"""
    print("Processing {}".format(csv_file))
    with open(csv_file,"r") as datafile:
        data = list(csv.reader(datafile))
    return data

def data_to_html(title, data):
    """Turns a list of lists into an HTML table"""

    # HTML Headers
    html_content = """
<html>
<head>
<style>
table {
    width: 25%;
    font-family: arial, sans-serif;
    border-collapse: collapse;
}
```

```
tr:nth-child(odd) {
    background-color: #dddddd;
}

td, th {
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}
</style>
</head>
<body>
"""


    # Add the header part with the given title
    html_content += "<h2>{}</h2><table>".format(title)

    # Add each row in data as a row in the table
    # The first line is special and gets treated separately
    for i, row in enumerate(data):
        html_content += "<tr>"
        for column in row:
            if i == 0:
                html_content += "<th>{}</th>".format(column)
            else:
                html_content += "<td>{}</td>".format(column)
        html_content += "</tr>"

    html_content += """</tr></table></body></html>"""
    return html_content


def write_html_file(html_string, html_file):

    # Making a note of whether the html file we're writing exists or not
    if os.path.exists(html_file):
        print("{} already exists. Overwriting...".format(html_file))
```

```
    with open(html_file,'w') as htmlfile:
        htmlfile.write(html_string)
    print("Table succesfully written to {}".format(html_file))

def main():
    """Verifies the arguments and then calls the processing function"""
    # Check that command-line arguments are included
    if len(sys.argv) < 3:
        print("ERROR: Missing command-line argument!")
        print("Exiting program...")
        sys.exit(1)

    # Open the files
    csv_file = sys.argv[1]
    html_file = sys.argv[2]

    # Check that file extensions are included
    if ".csv" not in csv_file:
        print('Missing ".csv" file extension from first command-line argument!')
        print("Exiting program...")
        sys.exit(1)

    if ".html" not in html_file:
        print('Missing ".html" file extension from second command-line argument!')
        print("Exiting program...")
        sys.exit(1)

    # Check that the csv file exists
    if not os.path.exists(csv_file):
        print("{} does not exist".format(csv_file))
        print("Exiting program...")
        sys.exit(1)

    # Process the data and turn it into an HTML
    data = process_csv(csv_file)
    title = os.path.splitext(os.path.basename(csv_file))[0].replace("_", " ").title()
    html_string = data_to_html(title, data)
    write_html_file(html_string, html_file)

if __name__ == "__main__":
```

```
main()
```

Revision #6
Created 21 November 2024 17:32:58 by Admin
Updated 9 December 2024 20:37:03 by Admin