Design & Implementierung eines Echtzeit-Q&A-Systems als Erweiterung des IAmA-Subreddits

(Python) – Dokumentation von Crawler / Analyzer - Scripts

Benedikt Hierl

Version 1.0

Sonntag, den 10.07.2016

Table of Contents

Namespace Index 2
Class Index 3
File Index 4
a__everything_Big_CSV_analyzer 5
a_author_Information 20
a_iAMA_Commenttime 22
a_question_Answered_Yes_No_Extrema 27
a_question_Answered_Yes_No_Tier_Percentage 32
a_question_Tier_Distribution 37
a_thread_Lifespan_N_Average_Commenttime 41
c_crawl_Author_Information 45
c_crawl_Differences 48
c_crawl_Random_Author_Information 51
c_crawl_Threads_N_Comments 54
d_create_Big_CSV 58
PlotlyBarChart 63
PlotlyBarChart_5_Bars 64
Class Documentation 65
PlotlyBarChart.PlotlyBarChart 65
PlotlyBarChart_5_Bars.PlotlyBarChart5Bars 69
File Documentation 75
a__everything_Big_CSV_analyzer.py 75
a_author_Information.py 79
a_iAMA_Commenttime.py 80
a_question_Answered_Yes_No_Extrema.py 81
a_question_Answered_Yes_No_Tier_Percentage.py 82
a_question_Tier_Distribution.py 83
a_thread_Lifespan_N_Average_Commenttime.py 84
c_crawl_Author_Information.py 85
c_crawl_Differences.py 86
c_crawl_Random_Author_Information.py 87
c_crawl_Threads_N_Comments.py 88
d_create_Big_CSV.py 89
PlotlyBarChart.py 90
PlotlyBarChart_5_Bars.py 91
Index 92

Namespace Index

Packages

Here are the packages with brief descriptions (if available):

a__everything_Big_CSV_analyzer 5
a_author_Information 20
a_iAMA_Commenttime 22
a_question_Answered_Yes_No_Extrema 27
a_question_Answered_Yes_No_Tier_Percentage 32
a_question_Tier_Distribution 37
a_thread_Lifespan_N_Average_Commenttime 41
c_crawl_Author_Information 45
c_crawl_Differences 48
c_crawl_Random_Author_Information 51
c_crawl_Threads_N_Comments 54
d_create_Big_CSV 58
PlotlyBarChart 63
PlotlyBarChart_5_Bars 64

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

PlotlyBarChart.PlotlyBarChart 65

PlotlyBarChart_5_Bars.PlotlyBarChart5Bars 69

File Index

File List

Here is a list of all files with brief descriptions:

a__everything_Big_CSV_analyzer.py 75
a_author_Information.py 79
a_iAMA_Commenttime.py 80
a_question_Answered_Yes_No_Extrema.py 81
a_question_Answered_Yes_No_Tier_Percentage.py 82
a_question_Tier_Distribution.py 83
a_thread_Lifespan_N_Average_Commenttime.py 84
c_crawl_Author_Information.py 85
c_crawl_Differences.py 86
c_crawl_Random_Author_Information.py 87
c_crawl_Threads_N_Comments.py 88
d_create_Big_CSV.py 89
PlotlyBarChart.py 90
PlotlyBarChart_5_Bars.py 91

Namespace Documentation

a__everything_Big_CSV_analyzer Namespace Reference

Functions

- def relation_question_upvotes_with_amount_of_questions_answered_by_iama_host ()
- def average_means_of_values_f_threads ()
- def relation_thread_upvotes_with_amount_of_comments ()
- def relation_thread_upvotes_with_amount_of_questions ()
- def relation_thread_downvotes_with_amount_of_comments ()
- def relation_thread_downvotes_with_amount_of_questions ()
- def relation_thread_upvotes_and_iama_host_response_time_comments ()
- def relation_thread_upvotes_and_iama_host_response_time_questions ()
- def relation_thread_downvotes_and_iama_host_response_time_comments ()
- def relation_thread_downvotes_and_iama_host_response_time_questions ()
- def relation_thread_lifespan_to_last_comment_and_amount_of_comments ()
- def relation_thread_lifespan_to_last_comment_and_amount_of_questions ()
- def relation_thread_lifespan_to_last_question_and_amount_of_comments ()
- def relation_thread_lifespan_to_last_question_and_amount_of_question ()
- def relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_comments ()
- def relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions ()
- def relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_comments ()
- def relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_questions ()
- def relation_thread_reaction_time_comments_and_iama_host_response_time_to_comments ()
- def relation_thread_reaction_time_comments_and_iama_host_response_time_to_questions ()
- def relation_thread_reaction_time_questions_and_iama_host_response_time_to_comments ()
- def relation_thread_reaction_time_questions_and_iama_host_response_time_to_questions ()
- def relation_thread_reaction_time_comments_and_amount_of_comments_the_iama_host_answered_to ()
- def relation_thread_reaction_time_comments_and_amount_of_questions_the_iama_host_answered_to ()
- def relation_thread_reaction_time_questions_and_amount_of_comments_the_iama_host_answered_to ()
- def relation_thread_reaction_time_questions_and_amount_of_questions_the_iama_host_answered_to ()
- def relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host ()
- def realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host ()
- def relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host ()
- def thread_overall_correlation ()
- def question_overall_correlation ()
- def average_means_of_values_f_authors ()

Variables

- author_information_iama
- author_information_random
- thread_information
- question_information
- author_amount_creation_iama_threads = author_information_iama['amount_creation_iama_threads']
- author_amount_creation_other_threads = author_information_iama['amount_creation_other_threads']
- author_amount_of_comments_except_iama = author_information_iama['amount_of_comments_except_iama']
- author_amount_of_comments_iama = author_information_iama['amount_of_comments_iama']
- author_author_birth_date = author_information_iama['author_birth_date']
- author_author_comment_karma_amount = author_information_iama['author_comment_karma_amount']
- author_author_link_karma_amount = author_information_iama['author_link_karma_amount']
- author_author_name = author_information_iama['author_name']

- author_comment_creation_every_x_sec = author_information_iama['comment_creation_every_x_sec']
- author_thread_creation_every_x_sec = author_information_iama['thread_creation_every_x_sec']
- author_time_acc_birth_first_iama_thread = author_information_iama['time_acc_birth_first_iama_thread']
- author_time_diff_acc_creation_n_first_comment = author_information_iama['time_diff_acc_creation_n_first_comment']
- author_time_diff_acc_creation_n_first_thread = author_information_iama['time_diff_acc_creation_n_first_thread']
- random_author_amount_creation_iama_threads = author_information_random['amount_creation_iama_threads']
- random_author_amount_creation_other_threads = author_information_random['amount_creation_other_threads']
- random_author_amount_of_comments_except_iama = author_information_random['amount_of_comments_except_iama']
- random_author_amount_of_comments_iama = author_information_random['amount_of_comments_iama']
- random_author_author_birth_date = author_information_random['author_birth_date']
- random_author_author_comment_karma_amount = author_information_random['author_comment_karma_amount']
- random_author_author_link_karma_amount = author_information_random['author_link_karma_amount']
- random_author_author_name = author_information_random['author_name']
- random_author_comment_creation_every_x_sec = author_information_random['comment_creation_every_x_sec']
- random_author_thread_creation_every_x_sec = author_information_random['thread_creation_every_x_sec']
- random_author_time_acc_birth_first_iama_thread = author_information_random['time_acc_birth_first_iama_thread']
- random_author_time_diff_acc_creation_n_first_comment = \
- random_author_time_diff_acc_creation_n_first_thread = author_information_random['time_diff_acc_creation_n_first_thread']
- thread_year = thread_information['Year']
- thread_id = thread_information['Thread id']
- thread_author = thread_information['Thread author']
- thread_ups = thread_information['Thread ups']
- thread_downs = thread_information['Thread downs']
- thread_creation_time_stamp = thread_information['Thread creation time stamp']
- thread_average_comment_vote_score_total
- thread_average_comment_vote_score_tier_1
- thread_average_comment_vote_score_tier_x
- thread_average_question_vote_score_total
- thread_average_question_vote_score_tier_1
- thread_average_question_vote_score_tier_x
- thread_num_comments_total_skewed
- thread_num_comments_total = thread_information['Thread num comments total']
- thread_num_comments_tier_1 = thread_information['Thread num comments tier 1']
- thread_num_comments_tier_x = thread_information['Thread num comments tier x']
- thread_num_questions_total = thread_information['Thread num questions total']
- thread_num_questions_tier_1 = thread_information['Thread num questions tier 1']
- thread_num_questions_tier_x = thread_information['Thread num questions tier x']
- thread_num_questions_answered_by_iama_host_total
- thread_num_questions_answered_by_iama_host_tier_1
- thread_num_questions_answered_by_iama_host_tier_x
- thread_num_comments_answered_by_iama_host_total
- thread_num_comments_answered_by_iama_host_tier_1
- thread_num_comments_answered_by_iama_host_tier_x
- thread_average_reaction_time_between_comments_total
- thread_average_reaction_time_between_comments_tier_1
- thread_average_reaction_time_between_comments_tier_x

- thread_average_reaction_time_between_questions_total
- thread_average_reaction_time_between_questions_tier_1
- thread_average_reaction_time_between_questions_tier_x
- thread_average_response_to_comment_time_iama_host_total
- thread_average_response_to_comment_time_iama_host_tier_1
- thread_average_response_to_comment_time_iama_host_tier_x
- thread_average_response_to_question_time_iama_host_total
- thread_average_response_to_question_time_iama_host_tier_1
- thread_average_response_to_question_time_iama_host_tier_x
- thread_amount_of_questioners_total
- thread_amount_of_questioners_tier_1
- thread_amount_of_questioners_tier_x
- thread_amount_of_commentators_total
- thread_amount_of_commentators_tier_1
- thread_amount_of_commentators_tier_x
- thread_life_span_until_last_comment
- thread_life_span_until_last_question
- question_ups = question_information['Question ups']
- question_answered_by_iAMA_host

Function Documentation

def a__everything_Big_CSV_analyzer.average_means_of_values_f_authors ()

```
Calculation of the average means of different values for author data

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.average_means_of_values_f_threads ()

```
Calculation of the average means of different values

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.question_overall_correlation ()

```
Calculation of the correlation of every column with every column for the questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host ()

```
Calculation of the correlation amount of commentators per thread <-> amount of questions answered
by iama host

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_question_upvotes_with_amount_of_questions_answered_by_iama_host ()

```
Calculation of the correlation question upvotes <-> amount of questions answered by the iama host

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host ()

```
Calculation of the correlation amount of questioners per thread <-> amount of questions answered
by iama host

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host ()

```
Calculation of the amount of questions ansked <-> amount of questions answered by iama host

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_comments ()

```
Calculation of the correlation thread downvotes <-> iama host repsonse time to comments

Args:
    -
```

```
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_questions ()

```
Calculation of the correlation thread downvotes <-> iama host repsonse time to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_comments ()

```
Calculation of the correlation thread downvotes <-> amount of comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_questions ()

```
Calculation of the correlation thread downvotes <-> amount of questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_amount_of_comments ()

```
Calculation of the correlation thread life span (until last comment) <-> amount of comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_amount_of_questions ()

```
Calculation of the correlation thread life span (until last comment) <-> amount of questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_comments ()

```
Calculation of the correlation thread life span (until last comment) <-> iama host repsonse time
to comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions ()

```
Calculation of the correlation thread life span (until last comment) <-> iama host repsonse time
to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_amount_of_comments ()

```
Calculation of the correlation thread life span (until last question) <-> amount of comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_amount_of_question ()

```
Calculation of the correlation thread life span (until last question) <-> amount of question

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_comments ()

```
Calculation of the correlation thread life span (until last question) <-> and iama host repsonse
time to comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_questions ()

```
Calculation of the correlation thread life span (until last question) <-> iama host repsonse time
to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_amount_of_comments_the_iama_host_answered_to ()

```
Calculation of the correlation thread reaction time between comments <-> amount of comments the
iama host
    reacted to

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_amount_of_questions_the_iama_host_answered_to ()

```
Calculation of the correlation thread reaction time between comments <-> amount of questions the
    iama host reacted to

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iama_host_response_time_to_comments ()

```
Calculation of the correlation thread reaction time between comments <-> iama host repsonse time
to comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iama_host_response_time_to_questions ()

```
Calculation of the correlation thread reaction time between comments <-> iama host repsonse time
to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_amount_of_comments_the_iama_host_answered_to ()

```
Calculation of the correlation thread reaction time between questions <-> amount of comments the
    iama host reacted to

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_amount_of_questions_the_iama_host_answered_to ()

```
Calculation of the correlation thread reaction time between questions <-> amount of questions the
iama
    host reacted to

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iama_host_response_time_to_comments ()

```
Calculation of the correlation thread reaction time between questions <-> iama host repsonse time
to comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iama_host_response_time_to_questions ()

```
Calculation of the correlation thread reaction time between questions <-> iama host repsonse time
to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_upvotes_and_iama_host_response_time_comments ()

```
Calculation of the correlation thread upvotes <-> iama host repsonse time to comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_upvotes_and_iama_host_response_time_questions ()

```
Calculation of the correlation thread upvotes <-> iama host repsonse time to questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_comments ()

```
Calculation of the correlation thread upvotes <-> amount of comments

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_questions ()

```
Calculation of the correlation thread upvotes <-> amount of questions

Args:
    -
Returns:
    -
```

def a__everything_Big_CSV_analyzer.thread_overall_correlation ()

```
Calculation of the correlation of every column with every column for the threads

Args:
    -
Returns:
    -
```

Variable Documentation

a__everything_Big_CSV_analyzer.author_amount_creation_iama_threads = author_information_iama['amount_creation_iama_threads']

a__everything_Big_CSV_analyzer.author_amount_creation_other_threads = author_information_iama['amount_creation_other_threads']

a__everything_Big_CSV_analyzer.author_amount_of_comments_except_iama = author_information_iama['amount_of_comments_except_iama']

a__everything_Big_CSV_analyzer.author_amount_of_comments_iama = author_information_iama['amount_of_comments_iama']

a__everything_Big_CSV_analyzer.author_author_birth_date = author_information_iama['author_birth_date']

a__everything_Big_CSV_analyzer.author_author_comment_karma_amount = author_information_iama['author_comment_karma_amount']

a__everything_Big_CSV_analyzer.author_author_link_karma_amount = author_information_iama['author_link_karma_amount']

a__everything_Big_CSV_analyzer.author_author_name = author_information_iama['author_name']

a__everything_Big_CSV_analyzer.author_comment_creation_every_x_sec = author_information_iama['comment_creation_every_x_sec']

a__everything_Big_CSV_analyzer.author_information_iama

```
Initial value:    1 = pandas.read_csv(
    2     'a_author_information_iama.csv',
    3     sep=',',
    4     na_values="None")
```

a__everything_Big_CSV_analyzer.author_information_random

```
Initial value:    1 = pandas.read_csv(
    2     'a author Information random.csv',
    3     sep=',',
    4     na_values="None")
```

a__everything_Big_CSV_analyzer.author_thread_creation_every_x_sec = author_information_iama['thread_creation_every_x_sec']

a__everything_Big_CSV_analyzer.author_time_acc_birth_first_iama_thread = author_information_iama['time_acc_birth_first_iama_thread']

a__everything_Big_CSV_analyzer.author_time_diff_acc_creation_n_first_comment = author_information_iama['time_diff_acc_creation_n_first_comment']

a__everything_Big_CSV_analyzer.author_time_diff_acc_creation_n_first_thread = author_information_iama['time_diff_acc_creation_n_first_thread']

a__everything_Big_CSV_analyzer.question_answered_by_iAMA_host

```
Initial value:    1 = question_information[
    2     'Question answered by iAMA host']
```

a__everything_Big_CSV_analyzer.question_information

```
Initial value:    1 = pandas.read_csv(
    2     'a_question_Answered_Yes_No_Tier_Percentage_2009_until_2016_ALL_tier_any.csv',
    3     sep=',',
    4     na_values="None",
    5     low_memory=False)
```

a__everything_Big_CSV_analyzer.question_ups = question_information['Question ups']

a__everything_Big_CSV_analyzer.random_author_amount_creation_iama_threads = author_information_random['amount_creation_iama_threads']

a__everything_Big_CSV_analyzer.random_author_amount_creation_other_threads = author_information_random['amount_creation_other_threads']

a__everything_Big_CSV_analyzer.random_author_amount_of_comments_except_iama = author_information_random['amount_of_comments_except_iama']

a__everything_Big_CSV_analyzer.random_author_amount_of_comments_iama = author_information_random['amount_of_comments_iama']

a__everything_Big_CSV_analyzer.random_author_author_birth_date = author_information_random['author_birth_date']

a__everything_Big_CSV_analyzer.random_author_author_comment_karma_amount = author_information_random['author_comment_karma_amount']

a__everything_Big_CSV_analyzer.random_author_author_link_karma_amount = author_information_random['author_link_karma_amount']

a__everything_Big_CSV_analyzer.random_author_author_name = author_information_random['author_name']

a__everything_Big_CSV_analyzer.random_author_comment_creation_every_x_sec = author_information_random['comment_creation_every_x_sec']

a__everything_Big_CSV_analyzer.random_author_thread_creation_every_x_sec = author_information_random['thread_creation_every_x_sec']

a__everything_Big_CSV_analyzer.random_author_time_acc_birth_first_iama_thread = author_information_random['time_acc_birth_first_iama_thread']

**a__everything_Big_CSV_analyzer.random_author_time_diff_acc_creation_n_first_comment = **

a__everything_Big_CSV_analyzer.random_author_time_diff_acc_creation_n_first_thread = author_information_random['time_diff_acc_creation_n_first_thread']

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread amount of commentators tier 1']
```

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread amount of commentators tier x']
```

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_total

```
Initial value:    1 = thread_information[
    2     'Thread amount of commentators total']
```

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread amount of questioners tier 1']
```

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread amount of questioners tier x']
```

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_total

```
Initial value:    1 = thread_information[
    2     'Thread amount of questioners total']
```

a__everything_Big_CSV_analyzer.thread_author = thread_information['Thread author']

a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average comment vote score tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average comment vote score tier x']
```

a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_total

```
Initial value:    1 = thread_information[
    2     'Thread average comment vote score total']
```

a__everything_Big_CSV_analyzer.thread_average_question_vote_score_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average question vote score tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_question_vote_score_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average question vote score tier x']
```

a__everything_Big_CSV_analyzer.thread_average_question_vote_score_total

```
Initial value:    1 = thread_information[
    2     'Thread average question vote score total']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between comments tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between comments tier x']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_total

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between comments total']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between questions tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between questions tier x']
```

a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_total

```
Initial value:    1 = thread_information[
    2     'Thread average reaction time between questions total']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average response to comment time iama host tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average response to comment time iama host tier x']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_total

```
Initial value:    1 = thread_information[
    2     'Thread average response to comment time iama host total']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread average response to question time iama host tier 1']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread average response to question time iama host tier x']
```

a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_total

```
Initial value:    1 = thread_information[
    2     'Thread average response to question time iama host total']
```

a__everything_Big_CSV_analyzer.thread_creation_time_stamp = thread_information['Thread creation time stamp']

a__everything_Big_CSV_analyzer.thread_downs = thread_information['Thread downs']

a__everything_Big_CSV_analyzer.thread_id = thread_information['Thread id']

a__everything_Big_CSV_analyzer.thread_information

```
Initial value:    1 = pandas.read_csv(
    2     'd_create_Big_CSV_2009_until_2016_BIGDATA_ALL.csv',
    3     sep=',',
    4     na_values="None")
```

a__everything_Big_CSV_analyzer.thread_life_span_until_last_comment

```
Initial value:    1 = thread_information[
    2     'Thread life span until last comment']
```

a__everything_Big_CSV_analyzer.thread_life_span_until_last_question

```
Initial value:    1 = thread_information[
    2     'Thread life span until last question']
```

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread num comments answered by iama host tier 1']
```

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread num comments answered by iama host tier x']
```

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_total

```
Initial value:    1 = thread_information[
    2     'Thread num comments answered by iama host total']
```

a__everything_Big_CSV_analyzer.thread_num_comments_tier_1 = thread_information['Thread num comments tier 1']

a__everything_Big_CSV_analyzer.thread_num_comments_tier_x = thread_information['Thread num comments tier x']

a__everything_Big_CSV_analyzer.thread_num_comments_total = thread_information['Thread num comments total']

a__everything_Big_CSV_analyzer.thread_num_comments_total_skewed

```
Initial value:    1 = thread_information[
    2     'Thread num comments total skewed']
```

a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_tier_1

```
Initial value:    1 = thread_information[
    2     'Thread num questions answered by iama host tier 1']
```

a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_tier_x

```
Initial value:    1 = thread_information[
    2     'Thread num questions answered by iama host tier x']
```

a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_total

```
Initial value:    1 = thread_information[
    2     'Thread num questions answered by iama host total']
```

a__everything_Big_CSV_analyzer.thread_num_questions_tier_1 = thread_information['Thread num questions tier 1']

a__everything_Big_CSV_analyzer.thread_num_questions_tier_x = thread_information['Thread num questions tier x']

a__everything_Big_CSV_analyzer.thread_num_questions_total = thread_information['Thread num questions total']

a__everything_Big_CSV_analyzer.thread_ups = thread_information['Thread ups']

a__everything_Big_CSV_analyzer.thread_year = thread_information['Year']

a_author_Information Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters ()
- def write_csv_data ()

Variables

- mongo_db_client_instance = None
- mongo_db_author_instance = None
- mongo_db_author_collection = None
- int mongo_db_author_collection_original = 0
- string argument_db_to_choose = ""

Function Documentation

def a_author_Information.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_author_Information.initialize_mongo_db_parameters ()

```
Instantiates all necessary variables for the correct usage of the mongoDB client

Args:
    -
Returns:
    -
```

def a_author_Information.write_csv_data ()

```
Gets all information from every collection within 'iAMA_Reddit_Authors*' database and writes it
into a csv file

Args:
    -
Returns:
    -
```

Variable Documentation

string a_author_Information.argument_db_to_choose = ""

a_author_Information.mongo_db_author_collection = None

int a_author_Information.mongo_db_author_collection_original = 0

a_author_Information.mongo_db_author_instance = None

a_author_Information.mongo_db_client_instance = None

a_iAMA_Commenttime Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def prepare_data_for_graph ()
- def add_thread_list_to_global_list (list_to_append)
- def generate_data_to_be_analyzed ()
- def calculate_ar_mean_answer_time_for_questions (id_of_thread, author_of_thread)
- def check_if_comment_is_a_question (given_string)
- def check_if_comment_is_on_tier_1 (comment_parent_id)
- def check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def check_if_comment_is_answer_from_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def write_csv_data (list_with_information)
- def plot_generated_data ()

Variables

- int argument_year_beginning = 0
- int year_actually_in_progress = 0
- int argument_year_ending = 0
- string argument_tier_in_scope = ""
- string argument_plot_time_unit = ""
- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- list list_To_Be_Plotted = []
- list global_thread_list = []
- list data_to_give_plotly = []

Function Documentation

def a_iAMA_Commenttime.add_thread_list_to_global_list (*list_to_append*)

```
Adds all elements of for the current year into a global list. This global list will be written into
a csv file
later on

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    list_to_append (list) : The list which will be iterated over and which elements will be added
to the global list
Returns:
    -
```

def a_iAMA_Commenttime.calculate_ar_mean_answer_time_for_questions (*id_of_thread*, *author_of_thread*)

```
Calculates the arithmetic mean of the answer time by the iama host in minutes

In dependence of the given tier argument (second argument) the processing of tiers will be filtered

Args:
    id_of_thread (str): The id of the thread which is actually processed. (Necessary for checking
if a question
        lies on tier 1 or any other tier)
    author_of_thread (str): The name of the thread author. (Necessary for checking if a given answer
is from the
        iama host or not)
Returns:
    Whenever there was a minimum of 1 question asked and 1 answer from the iama host:
        amount of answer times (int) : The amount of the arithmetic mean time of
    Whenever there no questions have been asked for that thread / or no answers were given /
        or all values in the database were null:
        None:   Returns an empty object of the type None
```

def a_iAMA_Commenttime.calculate_time_difference (*comment_time_stamp*, *answer_time_stamp_iama_host*)

```
Calculates the time difference in seconds between the a comment and its answer from the iama host

1. The time stamps will be converted from epoch
    into float and afterwards into str again
    (necessary for correct subtraction)
2. Then the time stamps will be subtracted from each other
3. The containing time difference will be converted into seconds (int)

Args:
    comment_time_stamp (str): The time stamp of the comment
    answer time stamp iama host (str): The time stamp of the iAMA hosts answer
Returns:
    time_difference_in_seconds (int) : The time difference of the comment and its answer by the
iAMA host in seconds
```

def a_iAMA_Commenttime.check_if_comment_is_a_question (*given_string*)

```
Simply checks whether a given string is a question or not

1. This method simply checks wether a question mark exists within that string or not..
    This is just that simple because messing around with natural processing kits to determine the
semantic sense
    would blow up my bachelor work...

Args:
    given_string (int) : The string which will be checked for a question mark
Returns:
    True (bool): Whenever the given string is a question
```

False (bool): Whenever the given string is not a question

def a_iAMA_Commenttime.check_if_comment_is_answer_from_thread_author (*author_of_thread*, *comment_actual_id*, *comments_cursor*)

```
Checks whether both strings are equal or not
```

```
1. A dictionary containing flags whether that a question is answered by the host with the appropriate
timestamp will
    be created in the beginning.
2. Then the method iterates over every comment within that thread
    1.1. Whenever an answer is from the iAMA hosts and the processed comments 'parent id' matches
the iAMA hosts
        comments (answers) id, the returned dict will contain appropriate values and will be
returned
    1.2. If this is not the case, it will be returned in its default condition

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_actual_id (str) : The id of the actually processed comment
    comments_cursor (list) : The cursor which shows to the amount of comments which can be iterated
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_iAMA_Commenttime.check_if_comment_is_not_from_thread_author (*author_of_thread*, *comment_author*)

```
Checks whether both strings are equal or not

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_author (str) : The name of the comments author
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_iAMA_Commenttime.check_if_comment_is_on_tier_1 (*comment_parent_id*)

```
Checks whether a comment relies on the first tier or any other tier

Args:
    comment parent id (str) : The name id of the comments parent
Returns:
    True (bool): Whenever the comment lies on tier 1
    False (bool): Whenever the comment lies on any other tier
```

def a_iAMA_Commenttime.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_iAMA_Commenttime.generate_data_to_be_analyzed ()

```
Generates the data which will be analyzed

1. This method iterates over every thread
    1.1. It filters if that iterated thread is an iAMA-request or not
        1.1.1. If yes: this thread gets skipped and the next one will be processed
        1.1.2. If no: this thread will be processed
2. If the thread gets processed it will receive the arithmetic mean of answer time
3. This value will be added to a global list and will be plotted later on

Args:
    -
Returns:
    -
```

def a_iAMA_Commenttime.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def a_iAMA_Commenttime.plot_generated_data ()

```
Plots the data which is to be generated

1. This method plots the data which has been calculated before by using Pltoly-Framework within
a self written class

Args:
    -
Returns:
    -
```

def a_iAMA_Commenttime.prepare_data_for_graph ()

```
Sorts and prepares data for graph plotting

Args:
    -
Returns:
    -
```

def a_iAMA_Commenttime.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years

1. Triggers the data generation process and moves forward within the years
    1.1. By moving through the years a csv file will be created for every year
    1.2. Additionally an interactive chart will be plotted

Args:
    -
```

```
Returns:
    -
```

def a_iAMA_Commenttime.write_csv_data (*list_with_information*)

```
Creates a csv file containing all necessary information about the average comment time of the iama
host

Args:
    list_with_information (list) : Contains various information about thread and comment time
Returns:
    -
```

Variable Documentation

string a_iAMA_Commenttime.argument_plot_time_unit = ""

string a_iAMA_Commenttime.argument_tier_in_scope = ""

int a_iAMA_Commenttime.argument_year_beginning = 0

int a_iAMA_Commenttime.argument_year_ending = 0

list a_iAMA_Commenttime.data_to_give_plotly = []

list a_iAMA_Commenttime.global_thread_list = []

list a_iAMA_Commenttime.list_To_Be_Plotted = []

a_iAMA_Commenttime.mongo_DB_Client_Instance = None

a_iAMA_Commenttime.mongo_DB_Comments_Instance = None

a_iAMA_Commenttime.mongo_DB_Thread_Collection = None

a_iAMA_Commenttime.mongo_DB_Threads_Instance = None

int a_iAMA_Commenttime.year_actually_in_progress = 0

a_question_Answered_Yes_No_Extrema Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def generate_data_now ()
- def process_answered_questions_within_thread (id_of_thread, author_of_thread, thread_creation_date)
- def check_if_comment_is_a_question (given_string)
- def check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def check_if_comment_has_been_answered_by_thread_author (author_of_thread, comment_acutal_id, comments_cursor)
- def calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def sort_questions (list_which_is_to_be_sorted)
- def create_question_list_containing_all_years (list_with_comments_per_years)
- def write_csv_and_count_unanswered (list_with_comments)
- def plot_generated_data ()

Variables

- int argument_year_beginning = 0
- int year_actually_in_progress = 0
- int argument_year_ending = 0
- argument_sorting = bool
- int argument_amount_of_top_quotes = 0
- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- list question_information_list = []
- list data_to_give_plotly = []

Function Documentation

def a_question_Answered_Yes_No_Extrema.calculate_time_difference (*comment_time_stamp*, *answer_time_stamp_iama_host*)

```
Calculates the time difference in seconds between the a comment and its answer from the iama host

1. The time stamps will be converted from epoch
    into float and afterwards into str again
    (necessary for correct subtraction)
2. Then the time stamps will be subtracted from each other
3. The containing time difference will be converted into seconds (int)

Args:
    comment_time_stamp (str): The time stamp of the comment
    answer_time_stamp_iama_host (str): The time stamp of the iAMA hosts answer
Returns:
    time difference in seconds (int) : The time difference of the comment and its answer by the
iAMA host in seconds
```

def a_question_Answered_Yes_No_Extrema.check_if_comment_has_been_answered_by_thread_author (*author_of_thread*, *comment_acutal_id*, *comments_cursor*)

```
Checks whether both strings are equal or not

1. A dictionary containing flags whether that a question is answered by the host with the appropriate
timestamp will
    be created in the beginning.
2. Then the method iterates over every comment within that thread
    1.1. Whenever an answer is from the iAMA hosts and the processed comments 'parent id' matches
the iAMA hosts
        comments (answers) id, the returned dict will contain appropriate values and will be
returned
    1.2. If this is not the case, it will be returned in its default condition

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_acutal_id (str) : The id of the actually processed comment
    comments cursor (list) : The cursor which shows to the amount of comments which can be iterated
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_question_Answered_Yes_No_Extrema.check_if_comment_is_a_question (*given_string*)

```
Simply checks whether a given string is a question or not

1. This method simply checks wether a question mark exists within that string or not..
    This is just that simple because messing around with natural processing kits to determine the
semantic sense
    would blow up my bachelor work...

Args:
    given_string (str) : The string which will be checked for a question mark
Returns:
    True (bool): Whenever the given string is a question
```

False (bool): Whenever the given string is not a question

def a_question_Answered_Yes_No_Extrema.check_if_comment_is_not_from_thread_author (*author_of_thread*, *comment_author*)

```
Checks whether both strings are equal or not

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment author (str) : The name of the comments author
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_question_Answered_Yes_No_Extrema.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate
```

```
1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.create_question_list_containing_all_years (*list_with_comments_per_years*)

```
Creates a list, containing all questions from all years

Args:
    list_with_comments_per_years (list) : The list containing the current years questions
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.generate_data_now ()

```
Generates the data which will be written into csv and plotted later on

1. This method iterates over every thread
    1.1. It filters if that iterated thread is an iAMA-request or not
        1.1.1. If yes: this thread gets skipped and the next one will be processed
        1.1.2. If no: this thread will be processed
2. If the thread gets processed it will receive an ordered dictionary containing information about
every question
    whether it has been answered or not
3. This ordered dictionary will be appended to a global list, which will be processed afterwards
for the generation
    of plots and csv files

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.plot_generated_data ()

```
Plots the data which is to be generated

1. This method plots the data which has been calculated before by using Pltoly-Framework within
a self written class

Args:
    -
```

```
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.process_answered_questions_within_thread (*id_of_thread*, *author_of_thread*, *thread_creation_date*)

```
Checks whether an iterated question has been answered by the iama host or not

1. This method checks at first whether an iterated comment contains values (e.g. is not none)
    1.1. If not: That comment will be skipped / if no comment is remaining None will be returned
    1.2. If yes: That comment will be processed
2. Now it will be checked whether that iterated comment is a question or not
3. Afterwards it will be checked wether that comment is a comment from the iAMA Host or not
    3.1. If this is not the case the next comment will be processed
4. Whenever that processed comment is a question and not (!!) from the thread author:
    amount of tier any questions (int) will be increased by one
5. Now it will be checked whether that comment has a comment ( answer ) below it which is from the
iAMA-host
    5.1. If yes: amount_of_tier_any_questions_answered (int) will be increased by one and the
dictionary, which
        is to be returned will be filled with values
    5.2. If no: the dictionary, which is to be returned will be filled with values

Args:
    id_of_thread (str) : Contains the id of the thread which is to be iterated
    author of thread (str) : Contains the name of the thread author
    thread_creation_date (str): Contains the time
Returns:
    amount_of_questions_not_answered (int) : The amount of questions which have not been answered
```

def a_question_Answered_Yes_No_Extrema.sort_questions (*list_which_is_to_be_sorted*)

```
Sorts a list of questions for a year, depending on the upvotes

1. This method prepares the data, in kind of sorting and counting amount of questions not being
answered
2. It also returns the number of unanswered questions, necessary for chart plotting

Args:
    list_which_is_to_be_sorted (list) : The list you want to sort regarding the sorting arguments
give on execution
Returns:
    questions_sorted (list) : The amount of questions, sorted on upvotes
```

def a_question_Answered_Yes_No_Extrema.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years

1. Triggers the data generation process and moves forward within the years
    1.1. By moving through the years a csv file will be created for every year
    1.2. At the end a csv file will be generated containing all questions of all years, sorted
    1.3. Additionally an interactive chart will be plotted

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Extrema.write_csv_and_count_unanswered (*list_with_comments*)

```
Creates a csv file containing all necessary information and calculates the amount of unanswered
questions

1. This method iterates over the top / worst X comments
    1.1. By iterating: all necessary information will be written into the csv file
    1.2. By iterating: the amount of unanswered questions will be counted
2. After iterating the amount of unanswered questions will be returned, which is necessary for graph
plotting

Args:
    list_with_comments (list): Contains all comments from the year
Returns:
    amount_of_questions_not_answered (int) : The amount of questions which have not been answered
```

Variable Documentation

int a_question_Answered_Yes_No_Extrema.argument_amount_of_top_quotes = 0

a_question_Answered_Yes_No_Extrema.argument_sorting = bool

int a_question_Answered_Yes_No_Extrema.argument_year_beginning = 0

int a_question_Answered_Yes_No_Extrema.argument_year_ending = 0

list a_question_Answered_Yes_No_Extrema.data_to_give_plotly = []

a_question_Answered_Yes_No_Extrema.mongo_DB_Client_Instance = None

a_question_Answered_Yes_No_Extrema.mongo_DB_Comments_Instance = None

a_question_Answered_Yes_No_Extrema.mongo_DB_Thread_Collection = None

a_question_Answered_Yes_No_Extrema.mongo_DB_Threads_Instance = None

list a_question_Answered_Yes_No_Extrema.question_information_list = []

int a_question_Answered_Yes_No_Extrema.year_actually_in_progress = 0

a_question_Answered_Yes_No_Tier_Percentage Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def generate_data_to_be_analyzed ()
- def question_answering_distribution_tier1_tierx_tierany (id_of_thread, author_of_thread)
- def check_if_comment_is_a_question (given_string)
- def check_if_comment_is_on_tier_1 (comment_parent_id)
- def check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def check_if_comment_is_answer_from_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def write_csv (list_with_information)
- def add_local_list_to_global_list (list_to_append)
- def prepare_data_for_graph ()
- def plot_generated_data ()

Variables

- int argument_year_beginning = 0
- int year_actually_in_progress = 0
- int argument_year_ending = 0
- string argument_tier_in_scope = ""
- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- list global_question_list = []
- list year_question_list = []
- list data_to_give_plotly = []

Function Documentation

def a_question_Answered_Yes_No_Tier_Percentage.add_local_list_to_global_list (*list_to_append*)

```
Adds all elements of for the current year into a global list. This global list will be written into
a csv file
later on

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    list_to_append (list) : The list which will be iterated over and which elements will be added
to the global list
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_a_question (*given_string*)

```
Simply checks whether a given string is a question or not

1. This method simply checks whether a question mark exists within that string or not..
    This is just that simple because messing around with natural processing kits to determine the
semantic sense
    would blow up my bachelor work...

Args:
    given_string (str) : The string which will be checked for a question mark
Returns:
    True (bool): Whenever the given string is a question
```

False (bool): Whenever the given string is not a question

def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_answer_from_thread_author (*author_of_thread*, *comment_actual_id*, *comments_cursor*)

```
Iterates over ewery comment, while compraing the ids and the comments creation author

1. the method iterates over every comment within that thread
    1.1. Whenever an answer is from the iAMA hosts and the processed comments 'parent_id' matches
the iAMA hosts
        comments (answers) id, the returned dict will contain appropriate values and will be
returned
    1.2. If this is not the case, it will be returned in its default condition

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment actual id:  (str) : The id of the actually processed comment
    comments_cursor (Cursor) : The cursor which shows to the amount of comments which can be iterated
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
```

def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_not_from_thread_author (*author_of_thread*, *comment_author*)

```
Checks whether both strings are equal or not

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    author of thread (str) : The name of the thread author (iAMA-Host)
    comment_author (str) : The name of the comments author
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_on_tier_1 (*comment_parent_id*)

```
Simply checks whether a given string is a question posted on tier 1 or not
```

```
1. This method simply checks whether a question has been posted on tier 1 by looking whether the
given
    string contains the substring "t3_" or not

Args:
    comment_parent_id (str): The string which will be checked for "t3_" appearance in it
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.generate_data_to_be_analyzed ()

```
Generates the data which will be analyzed

1. This method iterates over every thread
    1.1. It filters if that iterated thread is an iAMA-request or not
        1.1.1. If yes: this thread gets skipped and the next one will be processed
        1.1.2. If no: this thread will be processed
2. If the thread gets processed it will receive the distribution of questions on the tiers
3. This value will be added to a global list and will be plotted later on

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.plot_generated_data ()

```
Plots the data which is to be generated

1. This method plots the data which has been calculated before by using Pltoly-Framework within
a self written class

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.prepare_data_for_graph ()

```
Sorts and prepares data for graph plotting

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.question_answering_distribution_tier1_tierx_tier any (***id_of_thread*****,** ***author_of_thread*****)**

```
Generates the data which will be analyzed

1. It iterates over every comment and
     1.1. checks if the iterated comment is a question
     1.2. checks if the iterated comment has been posted on tier 1 level
     1.3. checks if that comment is from the iAMA-Host himself or not

2. Now the posted question will be added to a global list, which will be used for csv writing and
chart generation
    later on

Args:
    id_of_thread (str) : Contains the id of the processed thread
    author_of_thread (str) : Contains the iAMA-Hosts name
Returns:
```

-

def a_question_Answered_Yes_No_Tier_Percentage.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years

1. Triggers the data generation process and moves forward within the years
    1.1. By moving through the years a csv file will be created for every year
    1.2. Additionally an interactive chart will be plotted

Args:
    -
Returns:
    -
```

def a_question_Answered_Yes_No_Tier_Percentage.write_csv (***list_with_information*****)**

```
Creates a csv file containing all necessary information about the distribution of questions on the
tiers

This method iterates over the the given list, which contains every single questions of that year
(or all years)
and writes a csv file containing misc information about those questions.

Args:
    list_with_information (list) : Contains various information about thread and comment time
Returns:
    -
```

Variable Documentation

string a_question_Answered_Yes_No_Tier_Percentage.argument_tier_in_scope = ""

int a_question_Answered_Yes_No_Tier_Percentage.argument_year_beginning = 0

int a_question_Answered_Yes_No_Tier_Percentage.argument_year_ending = 0

list a_question_Answered_Yes_No_Tier_Percentage.data_to_give_plotly = []

list a_question_Answered_Yes_No_Tier_Percentage.global_question_list = []

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Client_Instance = None

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Comments_Instance = None

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Thread_Collection = None

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Threads_Instance = None

int a_question_Answered_Yes_No_Tier_Percentage.year_actually_in_progress = 0

list a_question_Answered_Yes_No_Tier_Percentage.year_question_list = []

a_question_Tier_Distribution Namespace Reference

Functions

- def initialize_mongo_db_parameters (actually_processed_year)
- def check_script_arguments ()
- def start_data_generation_for_analysis ()
- def generate_data_to_be_analyzed ()
- def question_distribution_tier1_tierx (id_of_thread, author_of_thread)
- def check_if_comment_is_a_question (given_string)
- def check_if_comment_is_on_tier_1 (comment_parent_id)
- def check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def add_actual_year_list_to_global_list (list_to_append)
- def write_csv (list_with_information)
- def prepare_data_for_graph ()
- def plot_generated_data ()

Variables

- int argument_year_beginning = 0
- int year_actually_in_progress = 0
- int argument_year_ending = 0
- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- list current_year_question_list = []
- list global_year_question_list = []
- list data_to_give_plotly = []

Function Documentation

def a_question_Tier_Distribution.add_actual_year_list_to_global_list (*list_to_append*)

```
Iterates over a given list with thread information and adds every single element to a global list
    The global list will be printed to csv in the end

Args:
    list to append (list) : List with thread information which will be appended to a global list
Returns:
    -
```

def a_question_Tier_Distribution.check_if_comment_is_a_question (*given_string*)

```
Simply checks whether a given string is a question or not

1. This method simply checks whether a question mark exists within that string or not..
    This is just that simple because messing around with natural processing kits to determine the
semantic sense
    would blow up my bachelor work...

Args:
    given string (str) : The string which will be checked for a question mark
Returns:
```

```
    True (bool): Whenever the given string is a question
```
False (bool): Whenever the given string is not a question

def a_question_Tier_Distribution.check_if_comment_is_not_from_thread_author (*author_of_thread*, *comment_author*)

```
Checks whether both strings are equal or not

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_author (str) : The name of the comments author
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def a_question_Tier_Distribution.check_if_comment_is_on_tier_1 (*comment_parent_id*)

```
Simply checks whether a given string is a question posted on tier 1 or not

1. This method simply checks whether a question has been posted on tier 1 by looking whether the
given
    string contains the substring "t3_" or not

Args:
    comment parent id (str): The string which will be checked for "t3 " appearance in it
Returns:
    -
```

def a_question_Tier_Distribution.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_question_Tier_Distribution.generate_data_to_be_analyzed ()

```
Generates the data which will be analyzed

1. This method iterates over every thread
    1.1. It filters if that iterated thread is an iAMA-request or not
        1.1.1. If yes: this thread gets skipped and the next one will be processed
        1.1.2. If no: this thread will be processed
2. If the thread gets processed it will receive the distribution of questions on the tiers
3. This value will be added to a global list and will be plotted later on

Args:
    -
Returns:
    -
```

def a_question_Tier_Distribution.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def a_question_Tier_Distribution.plot_generated_data ()

```
Plots the data which is to be generated

1. This method plots the data which has been calculated before by using Pltoly-Framework within
a self written class

Args:
    -
Returns:
    -
```

def a_question_Tier_Distribution.prepare_data_for_graph ()

```
Sorts and prepares data for graph plotting

Args:
    -
Returns:
    -
```

def a_question_Tier_Distribution.question_distribution_tier1_tierx (*id_of_thread*, *author_of_thread*)

```
Generates the data which will be analyzed

1. It iterates over every comment and
     1.1. checks if the iterated comment is a question
     1.2. checks if the iterated comment has been posted on tier 1 level
     1.3. checks if that comment is from the iAMA-Host himself or not

2. Now the posted question will be added to a global list, which will be used for csv writing and
chart generation
    later on

Args:
    id_of_thread (str) : Contains the id of the processed thread
    author_of_thread (str) : Contains the iAMA-Hosts name
Returns:
```

    -

def a_question_Tier_Distribution.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years
```

```
1. Triggers the data generation process and moves forward within the years
    1.1. By moving through the years a csv file will be created for every year
    1.2. Additionally an interactive chart will be plotted

Args:
    -
Returns:
    -
```

def a_question_Tier_Distribution.write_csv (*list_with_information*)

```
Creates a csv file containing all necessary information about the distribution of questions on the
tiers

This method iterates over the "current_year_question_list", which contains every single questions
of that year
and writes a csv file containing misc information about those questions.

One thing is to be said: The .csv file will be written in binary mode, therefore looking at them
in a plain text
editor could be a problem - please use excel for that.
I had to use "binary" mode, otherwise the questions-text could not be written into the csv file,
because windows
has some problem by converting some special chars to utf.

Args:
    list_with_information (list) : Contains information about questions for the current year
Returns:
    -
```

Variable Documentation

int a_question_Tier_Distribution.argument_year_beginning = 0

int a_question_Tier_Distribution.argument_year_ending = 0

list a_question_Tier_Distribution.current_year_question_list = []

list a_question_Tier_Distribution.data_to_give_plotly = []

list a_question_Tier_Distribution.global_year_question_list = []

a_question_Tier_Distribution.mongo_DB_Client_Instance = None

a_question_Tier_Distribution.mongo_DB_Comments_Instance = None

a_question_Tier_Distribution.mongo_DB_Thread_Collection = None

a_question_Tier_Distribution.mongo_DB_Threads_Instance = None

int a_question_Tier_Distribution.year_actually_in_progress = 0

a_thread_Lifespan_N_Average_Commenttime Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def prepare_data_for_graph_life_span ()
- def prepare_data_for_comment_time ()
- def generate_data_to_be_analyzed ()
- def calculate_time_difference (id_of_thread, creation_date_of_thread)
- def write_csv (list_with_information)
- def add_thread_list_to_global_list (list_to_append)
- def prepare_dict_by_time_separation_for_comment_time ()
- def plot_generated_data ()

Variables

- int argument_year_beginning = 0
- string argument_calculation = ""
- int argument_year_ending = 0
- int year_actually_in_progress = 0
- string argument_plot_time_unit = ""
- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- list global_thread_list = []
- list temp_time_difference_list = []
- list list_with_currents_year_infos = []
- list data_to_give_plotly = []

Function Documentation

def a_thread_Lifespan_N_Average_Commenttime.add_thread_list_to_global_list (*list_to_append*)

```
Adds all elements of for the current year into a global list. This global list will be written into
a csv file
later on

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    list_to_append (list) : The list which will be iterated over and which elements will be added
to the global list
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.calculate_time_difference (*id_of_thread*, *creation_date_of_thread*)

```
Calculates the difference between thread creation date and the last comment found in that thread

1. The creation date of a thread gets determined
2. Then the comments will be iterated over, creating a dictionary which is structured as follows:
  {
      ('first_Comment_After_Thread_Started', int),
      ('thread_life_span', int),
      ('arithmetic_Mean_Response_Time', int),
      ('median Response Time', int),
      ('id')
  }
3. That returned dictionary will be appended to a global list
4. That List will be iterated later on and the appropriate graph will be plotted

Args:
    id_of_thread (str) : The string which contains the id of the actually processed thread
    creation date of thread (str) : The string which contains the creation date of the thread (in
epoch formatation)
Returns:
```

dict_to_be_returned (dict) : Containing information about the time difference

def a_thread_Lifespan_N_Average_Commenttime.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.generate_data_to_be_analyzed ()

```
Generates the data which will be analyzed

1. This method iterates over every thread
    1.1. It filters if that iterated thread is an iAMA-request or not
        1.1.1. If yes: this thread gets skipped and the next one will be processed
        1.1.2. If no: this thread will be processed
2. If the thread gets processed it will receive the life span and other information about the thread
as dictionary
3. This dictionary will be added to a global list and will be plotted later on

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.plot_generated_data ()

```
Plots the data which is to be generated

1. This method plots the data which has been calculated before by using Pltoly-Framework within
a self written class

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.prepare_data_for_comment_time ()

```
Prepares the average mean comment time per thread

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.prepare_data_for_graph_life_span ()

```
Calculates the distribution of single values regarding the chosen time argument

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.prepare_dict_by_time_separation_for_comment_time ()

```
Restructures the dictionary which is to be plotted for the display of the average mean comment time

1. This method processes the data in dependence of the commited time

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years

1. Triggers the data generation process and moves forward within the years
    1.1. By moving through the years a csv file will be created for every year
    1.2. Additionally an interactive chart will be plotted

Args:
    -
Returns:
    -
```

def a_thread_Lifespan_N_Average_Commenttime.write_csv (*list_with_information*)

```
Creates a csv file containing all necessary information about the life span of a thread and various
information
    about comments

Args:
    list_with_information (list) : Contains various information about thread and comment time
Returns:
    -
```

Variable Documentation

string a_thread_Lifespan_N_Average_Commenttime.argument_calculation = ""

string a_thread_Lifespan_N_Average_Commenttime.argument_plot_time_unit = ""

int a_thread_Lifespan_N_Average_Commenttime.argument_year_beginning = 0

int a_thread_Lifespan_N_Average_Commenttime.argument_year_ending = 0

list a_thread_Lifespan_N_Average_Commenttime.data_to_give_plotly = []

list a_thread_Lifespan_N_Average_Commenttime.global_thread_list = []

list a_thread_Lifespan_N_Average_Commenttime.list_with_currents_year_infos = []

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Client_Instance = None

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Comments_Instance = None

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Thread_Collection = None

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Threads_Instance = None

list a_thread_Lifespan_N_Average_Commenttime.temp_time_difference_list = []

int a_thread_Lifespan_N_Average_Commenttime.year_actually_in_progress = 0

c_crawl_Author_Information Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def generate_data_now ()
- def calculate_time_difference (time_value_1, time_value_2)
- def get_author_information (name_of_author)

Variables

- int argument_year_beginning = 0
- int year_actually_in_progress = 0
- int argument_year_ending = 0
- string argument_inverse_crawling = ""
- mongo_db_client_instance = None
- mongo_db_threads_instance = None
- mongo_db_thread_collection = None
- mongo_db_author_instance = None
- reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

Function Documentation

def c_crawl_Author_Information.calculate_time_difference (*time_value_1*, *time_value_2*)

```
Calculates the time difference between two floats in epoch style and returns seconds

Args:
    time_value_1 (float): The first time value to be used for calculation
    time_value_2 (float): The second time value to be used for calculation
Returns:
    time_diff_seconds (int): The amount of time difference in seconds
```

def c_crawl_Author_Information.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def c_crawl_Author_Information.generate_data_now ()

```
Crawls author information and writes them into the mongoDB database with the name
'iAMA Reddit Authors'
```

```
It does this by first checking the given crawling direction. The ability to crawl bidirectional
allows you to build
up you database in a much more faster way, because you can start one instance crawling forward while
the other
instance crawls backward.

This method works in the following way:

1. Checks for crawling direction
2. It checks whether an iterated collection is no "system.indexes".
3. By iterating over all collections it checks for iAMA-Requests and skips them. Because we do not
want requests
in our dataset, because we want data of actually created iama threads

4. Now it will be checked whether the author already exists within the database (collection name).
This will be
done by always re-initialising the collection.names() which is necessary to always have a
up2date-overview!

    4.1. Whenever the author does not exist yet get the necessary information and write it into
the database

    4.2. Whenever the author does already exist skip that calculation part

Args:
    -
Returns:
    -
```

def c_crawl_Author_Information.get_author_information (*name_of_author*)

```
Calculates various information about the author
    Because I have created this script shortly before my evaluation everything listed here is not
outsourced by
    written in a very sequential / procedural way, therefore I ask for you understanding.

    The method does the following:

    1. Referencing a reddit author object, which is necessary to get all that necessary data
    2. Declaration of necessary variables for later assignment
    3. Trial of receiving the authors birthday
        We have to try this here, because, if the account has already been deleted a http error
will be thrown
        and we would have to recrawl all that data.
    4. Receival authors comment / link karma - amount
    5. Trial of receiving of all links / comments the author every made
        Because it could happen, that there is an internal error in reddit ongoing (Error 500) which
will reset
        the connection and therefore we would have to recrawl all of our data
    6. Iteration of all comments / links and therefore saving the time difference (in seconds)
between each created
        comment / link
    7. Calculation of time difference between acc birth & first iama in seconds
    8. Patching up a big dictionary which will be sorted (alphabetically correct)
    9. Return that dictionary

Args:
    name_of_author (str): The name of the author which information need to be calculated
Returns:
    dict_to_be_returned (dict): Dictionary containing various information about the author. It will
be written
```

def c_crawl_Author_Information.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB client
```

```
Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def c_crawl_Author_Information.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years
    After every year cycle the mongo db parameters will be reinitialized

Args:
    -
Returns:
    -
```

Variable Documentation

string c_crawl_Author_Information.argument_inverse_crawling = ""

int c_crawl_Author_Information.argument_year_beginning = 0

int c_crawl_Author_Information.argument_year_ending = 0

c_crawl_Author_Information.mongo_db_author_instance = None

c_crawl_Author_Information.mongo_db_client_instance = None

c_crawl_Author_Information.mongo_db_thread_collection = None

c_crawl_Author_Information.mongo_db_threads_instance = None

c_crawl_Author_Information.reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

int c_crawl_Author_Information.year_actually_in_progress = 0

c_crawl_Differences Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters ()
- def crawl_missing_collection_into_comments_database (name_of_missing_collection)
- def check_if_collection_is_missing_in_comments_database ()
- def crawl_missing_collection_into_threads_database (name_of_missing_collection)
- def check_if_collection_is_missing_in_threads_database ()
- def start_crawling_for_diffs ()

Variables

- mongo_DB_Client_Instance = None
- mongo_DB_Threads_Instance = None
- mongo_DB_Thread_Collection = None
- mongo_DB_Comments_Instance = None
- mongo_DB_Comments_Collection = None
- string argument_year_beginning = ""
- string argument_year_ending = ""
- string argument_inverse_crawling = ""

Function Documentation

def c_crawl_Differences.check_if_collection_is_missing_in_comments_database ()

```
Checks if a specific collection (thread) is missing in the appropriate comments database

    The method starts the diff checking for all collections within the threads database.
    Whenever a thread exists in the comment database but not in the threads database it will be
crawled from the
    reddit servers and written into the database.

Args:
    -
Returns:
    -
```

def c_crawl_Differences.check_if_collection_is_missing_in_threads_database ()

```
Checks if a specific collection (thread) is missing in the appropriate threads database

    The method starts the diff checking for all collections within the threads database.
    Whenever a thread exists in the comment database but not in the threads database it will be
crawled from the
    reddit servers and written into the database.

Args:
    -
Returns:
    -
```

def c_crawl_Differences.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def c_crawl_Differences.crawl_missing_collection_into_comments_database (*name_of_missing_collection*)

```
Crawls a specific thread, which is missing in the comments database and writes the appropriate entry
in the db

    The method works as follows:
    1. It checks whether that thread / collection is really missing (even when that has been done
before, we check
        it again here, just to make sure that collection has not been created in the meanwhile by
another crawling
        process.
    2. Now the comments will be crawled from the reddit servers with flattened hierarchy
    3. Yet the comments will be written into the appropriate comments database. The correct database
will be
        deviated from the threads creation timestamp.


Args:
    name of missing collection (str) : The id of the collection which is actually missing in the
comments database
Returns:
    -
```

def c_crawl_Differences.crawl_missing_collection_into_threads_database (*name_of_missing_collection*)

```
Crawls a specific thread, which is missing in the thread database and writes the appropriate entry
in the db

    The method works as follows:
    1. It checks whether that thread / collection is really missing (even when that has been done
before, we check
        it again here, just to make sure that collection has not been created in the meanwhile by
another crawling
        process.
    2. Now the the thread will be crawled from the reddit servers
    3. Yet the thread will be written into the appropriate threads database. The correct database
will be
        deviated from the threads creation timestamp.

Args:
    name_of_missing_collection (str) : The id of the collection which is actually missing in the
comments database
Returns:
    -
```

def c_crawl_Differences.initialize_mongo_db_parameters ()

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    -
Returns:
    -
```

def c_crawl_Differences.start_crawling_for_diffs ()

```
This method starts the crawling, with the method you have defined in your arguments

Args:
    -
Returns:
    -
```

Variable Documentation

string c_crawl_Differences.argument_inverse_crawling = ""

string c_crawl_Differences.argument_year_beginning = ""

string c_crawl_Differences.argument_year_ending = ""

c_crawl_Differences.mongo_DB_Client_Instance = None

c_crawl_Differences.mongo_DB_Comments_Collection = None

c_crawl_Differences.mongo_DB_Comments_Instance = None

c_crawl_Differences.mongo_DB_Thread_Collection = None

c_crawl_Differences.mongo_DB_Threads_Instance = None

c_crawl_Random_Author_Information Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters ()
- def start_data_generation_for_analysis ()
- def generate_data_now (randomized_author_name)
- def calculate_time_difference (time_value_1, time_value_2)
- def get_author_information (name_of_author)

Variables

- argument_limit_crawling_amount = None
- mongo_db_client_instance = None
- mongo_db_random_author_instance = None
- mongo_db_random_author_collection = None
- mongo_db_iama_author_instance = None
- mongo_db_iama_author_collection = None
- int mongo_db_iama_author_collection_amount = 0
- reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

Function Documentation

def c_crawl_Random_Author_Information.calculate_time_difference (*time_value_1*, *time_value_2*)

```
Calculates the time difference between two floats in epoch style and returns seconds

Args:
    time_value_1 (float): The first time value to be used for calculation
    time value 2 (float): The second time value to be used for calculation
Returns:
    time_diff_seconds (int): The amount of time difference in seconds
```

def c_crawl_Random_Author_Information.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
Returns:
    -
```

def c_crawl_Random_Author_Information.generate_data_now (*randomized_author_name*)

```
Crawls author information and writes them into the mongoDB database with the name
'iAMA_Reddit_Authors_Random'
```

```
It does this by first checking the given crawling direction. The ability to crawl bidirectional
allows you to build
up you database in a much more faster way, because you can start one instance crawling forward while
the other
instance crawls backward.

This method works in the following way:

1. Checks for crawling direction
2. It checks whether an iterated collection is no "system.indexes".
3. By iterating over all collections it checks for iAMA-Requests and skips them. Because we do not
want requests
in our dataset, because we want data of actually created iama threads

4. Now it will be checked whether the author already exists within the database (collection name).
This will be
done by always re-initialising the collection.names() which is necessary to always have a
up2date-overview!

    4.1. Whenever the author does not exist yet get the necessary information and write it into
the database

    4.2. Whenever the author does already exist skip that calculation part

Args:
    -
Returns:
    -
```

def c_crawl_Random_Author_Information.get_author_information (*name_of_author*)

```
Calculates various information about the author
    Because I have created this script shortly before my evaluation everything listed here is not
outsourced by
    written in a very sequential / procedural way, therefore I ask for you understanding.

    The method does the following:

    1. Referencing a reddit author object, which is necessary to get all that necessary data
    2. Declaration of necessary variables for later assignment
    3. Trial of receiving the authors birthday
        We have to try this here, because, if the account has already been deleted a http error
will be thrown
        and we would have to recrawl all that data.
    4. Receival authors comment / link karma - amount
    5. Trial of receiving of all links / comments the author every made
        Because it could happen, that there is an internal error in reddit ongoing (Error 500) which
will reset
        the connection and therefore we would have to recrawl all of our data
    6. Iteration of all comments / links and therefore saving the time difference (in seconds)
between each created
        comment / link
    7. Calculation of time difference between acc birth & first iama in seconds
    8. Patching up a big dictionary which will be sorted (alphabetically correct)
    9. Return that dictionary

Args:
    name_of_author (str): The name of the author which information need to be calculated
Returns:
    dict_to_be_returned (dict): Dictionary containing various information about the author. It will
be written
```

def c_crawl_Random_Author_Information.initialize_mongo_db_parameters ()

```
Instantiates all necessary variables for the correct usage of the mongoDB client
```

```
Args:
    -
Returns:
    -
```

def c_crawl_Random_Author_Information.start_data_generation_for_analysis ()

```
Starts the data processing by swichting through the years
    After every year cycle the mongo db parameters will be reinitialized

Args:
    -
Returns:
    -
```

Variable Documentation

c_crawl_Random_Author_Information.argument_limit_crawling_amount = None

c_crawl_Random_Author_Information.mongo_db_client_instance = None

c_crawl_Random_Author_Information.mongo_db_iama_author_collection = None

int c_crawl_Random_Author_Information.mongo_db_iama_author_collection_amount = 0

c_crawl_Random_Author_Information.mongo_db_iama_author_instance = None

c_crawl_Random_Author_Information.mongo_db_random_author_collection = None

c_crawl_Random_Author_Information.mongo_db_random_author_instance = None

c_crawl_Random_Author_Information.reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

c_crawl_Threads_N_Comments Namespace Reference

Functions

- def initialize_mongo_db_parameters ()
- def check_script_arguments ()
- def convert_argument_year_to_epoch (year)
- def crawl_data ()
- def crawl_threads ()
- def crawl_comments ()
- def check_if_coll_in_db_already_exists_up2date (submission)

Variables

- mongo_DB_Client_Instance = None
- reddit_Instance = None
- argument_crawl_type = None
- argument_year_beginning = None
- argument_year_end = None
- argument_hours_to_shift = None
- time_shift_difference

Function Documentation

def c_crawl_Threads_N_Comments.check_if_coll_in_db_already_exists_up2date (*submission*)

```
Checks if a collection already exists in the database or not

This is necessary, otherwise thread information would be written into the database twice.
It works the following way:

1. Define a tolerance factor (necessary because reddit skews information about the amount of
    "upvotes"). Without defining that tolerance factor every thread would be created anew.
    After messing around a few days I found this one to be the best value to work with

2. Create values for temporary values for checking

3. Check and recreate collection if necessary

4. Return appropriate boolean value if collection already existed within the database or not


Args:
    submission (Submission) : The thread which will be processed / iterated over at the moment
Returns:
    True / False (bool) : Whenever the collection already exists within the database (True) or not
(False)
```

def c_crawl_Threads_N_Comments.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values

Args:
    -
```

```
Returns:
    -
```

def c_crawl_Threads_N_Comments.convert_argument_year_to_epoch (*year*)

```
"Converts" a given string into the appropriate epoch string format (int)

Args:
    year (str) : The year which will be "converted" into epoch format (necessary for correct PRAW
API behaviour)
Returns:
    year (int) : The year "converted" into epoch format as integer
```

def c_crawl_Threads_N_Comments.crawl_comments ()

```
Crawls thread information and writes them into the mongoDB storage
It works as follwoing:

1. At first an attempt to the amazon cloud search will be made, with necessary parameters which
returns an object,
    of the class "Generator" which contains all comments for the given / crawled time windows

2. After that the "Generator"s elements will be iterated over

    2.1. It will be checked if that iterated collection already exists within the database or not

        2.2.1. If it already exists, it will be checked whether if it is up to date or not
            2.2.1.1. If up2date: do nothing
            2.2.1.2. If not up2date: drop that collection within the database and crawl the
collection anew

        2.2.2. If it does not yet exist: create that collection in the database with the necessary
information

3. Whenever there are no elements left to iterate over the time crawling window will be shifted
into the future by
    using the given amount in hours (fourth argument), whenever the ending year (third argument)
is not reached yet

Args:
    -
Returns:
    -
```

def c_crawl_Threads_N_Comments.crawl_data ()

```
Crawls data from reddit, depending on the first argument (threads / comments) you give the script

Args:
    -
Returns:
    -
```

def c_crawl_Threads_N_Comments.crawl_threads ()

```
Crawls thread information and writes them into the mongoDB storage
It works as follwoing:
```

```
1. At first an attempt to the amazon cloud search will be made, with necessary parameters which
returns an object,
    of the class "Generator" which contains all threads for the given / crawled time windows

2. After that the "Generator"s elements will be iterated over

    2.1. It will be checked if that iterated collection already exists within the database or not

        2.2.1. If it already exists, it will be checked whether if it is up to date or not
            2.2.1.1. If up2date: do nothing
            2.2.1.2. If not up2date: drop that collection within the database and crawl the
collection anew

        2.2.2. If it does not yet exist: create that collection in the database with the necessary
information

3. Whenever there are no elements left to iterate over the time crawling window will be shifted
into the future by
    using the given amount in hours (third argument), whenever the ending year (second argument)
is not reached yet

Args:
    -
Returns:
    -
```

def c_crawl_Threads_N_Comments.initialize_mongo_db_parameters ()

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    -
Returns:
    -
```

Variable Documentation

c_crawl_Threads_N_Comments.argument_crawl_type = None

c_crawl_Threads_N_Comments.argument_hours_to_shift = None

c_crawl_Threads_N_Comments.argument_year_beginning = None

c_crawl_Threads_N_Comments.argument_year_end = None

c_crawl_Threads_N_Comments.mongo_DB_Client_Instance = None

c_crawl_Threads_N_Comments.reddit_Instance = None

c_crawl_Threads_N_Comments.time_shift_difference

```
Initial value:    1 = int(
    2     round(time.mktime(
    3         (datetime.fromtimestamp(argument_year_beginning) +
    4          timedelta(
    5              hours=argument_hours_to_shift)
    6          ).timetuple()
    7     )
    8     )
```

```
    9 )
```

d_create_Big_CSV Namespace Reference

Functions

- def check_script_arguments ()
- def initialize_mongo_db_parameters (actually_processed_year)
- def start_data_generation_for_analysis ()
- def generate_data ()
- def process_specific_thread (thread_id, thread_creation_time_stamp, thread_author)
- def check_if_comment_is_a_question (given_string)
- def check_if_comment_is_on_tier_1 (comment_parent_id)
- def check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def check_if_comment_has_been_answered_by_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def calculate_reaction_time_average (list_to_be_processed, thread_creation_time_stamp)
- def calculate_life_span (thread_creation_time_stamp, time_value_of_last_comment, time_value_of_last_question)
- def add_actual_year_list_to_global_list (list_to_append)
- def write_csv_data (list_with_information)

Variables

- int argument_year_beginning = 0
- int argument_year_ending = 0
- int year_actually_in_progress = 0
- list list_current_year = []
- list list_global_year = []

Function Documentation

def d_create_Big_CSV.add_actual_year_list_to_global_list (*list_to_append*)

```
Iterates over a given list with thread information and adds every single element to a global list
    The global list will be printed to csv in the end

Args:
    list_to_append (list) : List with thread information which will be appended to a global list
Returns:
    -
```

def d_create_Big_CSV.calculate_life_span (*thread_creation_time_stamp*, *time_value_of_last_comment*, *time_value_of_last_question*)

```
Calculates the life span between to time stamps

1. The creation date of a thread gets determined
2. Then the comments will be iterated over, creating a dictionary which is structured as follows:
  {
      ('first_Comment_After_Thread_Started', int),
      ('thread life span', int),
      ('arithmetic_Mean_Response_Time', int),
      ('median Response Time', int),
      ('id')
```

```
  }
3. That returned dictionary will be appended to a global list
4. That List will be iterated later on and the appropriate graph will be plotted

Args:
    thread_creation_time_stamp (float) : The time stamp (utc epoch) of the thread creation
    time_value_of_last_comment (float) : The time stamp (utc epoch) of the threads last comment
    time_value_of_last_question (float) : The time stamp (utc epoch) of the threads last question
Returns:
    dict_to_be_returned (dict) : Containing information about the time differences:
        Thread creation timestamp <-> Last question time stamp
```

Thread creation timestamp <-> Last comment time stamp

def d_create_Big_CSV.calculate_reaction_time_average (*list_to_be_processed*, *thread_creation_time_stamp*)

```
Calculates the reaction time of a list with time values in it

Args:
    list_to_be_processed (list) : The list which contains time values (utc epoch)
    thread creation time stamp (str) : The string which contains the creation date of the thread
(utc epoch)
Returns:
    None : Whenever there were no time values given
```

np.mean(time_difference) (float) : Time arithmetic mean of the reaction time in seconds

def d_create_Big_CSV.calculate_time_difference (*comment_time_stamp*, *answer_time_stamp_iama_host*)

```
Calculates the time difference in seconds between the a comment and its answer from the iama host

1. The time stamps will be converted from epoch
    into float and afterwards into str again
    (necessary for correct subtraction)
2. Then the time stamps will be subtracted from each other
3. The containing time difference will be converted into seconds (int)

Args:
    comment time stamp (str): The time stamp of the comment
    answer_time_stamp_iama_host (str): The time stamp of the iAMA hosts answer
Returns:
    time_difference_in_seconds (int) : The time difference of the comment and its answer by the
iAMA host in seconds
```

def d_create_Big_CSV.check_if_comment_has_been_answered_by_thread_author (*author_of_thread*, *comment_actual_id*, *comments_cursor*)

```
Checks whether both strings are equal or not

1. A dictionary containing flags whether that a question is answered by the host with the appropriate
timestamp will
    be created in the beginning.
2. Then the method iterates over every comment within that thread
    1.1. Whenever an answer is from the iAMA hosts and the processed comments 'parent_id' matches
the iAMA hosts
        comments (answers) id, the returned dict will contain appropriate values and will be
returned
    1.2. If this is not the case, it will be returned in its default condition

Note: We take a list as 'comments_cursor' and not a real cursor, because real cursors can be
exhausted, which
        could lead to, that not all comments will be iterated.. This is especially critical when
you have to do
```

```
        many iterations with only one cursor... [took me 8 hours to figure this "bug" out...]

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_actual_id (str) : The id of the actually processed comment
    comments_cursor (list) : The list containing all comments
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def d_create_Big_CSV.check_if_comment_is_a_question (*given_string*)

```
Simply checks whether a given string is a question or not

This method simply checks wether a question mark exists within that string or not..
    This is just that simple because messing around with natural processing kits to determine the
semantic sense
    would blow up my bachelor work...

Args:
    given_string (int) : The string which will be checked for a question mark
Returns:
    True (bool): Whenever the given string is a question
```

False (bool): Whenever the given string is not a question

def d_create_Big_CSV.check_if_comment_is_not_from_thread_author (*author_of_thread*, *comment_author*)

```
Checks whether both strings are equal or not

1. This method simply checks wether both strings match each other or not.
    I have built this extra method to have a better overview in the main code..

Args:
    author_of_thread (str) : The name of the thread author (iAMA-Host)
    comment_author (str) : The name of the comments author
Returns:
    True (bool): Whenever the strings do not match
    False (bool): Whenever the strings do match
     answered that given question)
```

def d_create_Big_CSV.check_if_comment_is_on_tier_1 (*comment_parent_id*)

```
Checks whether a comment relies on the first tier or any other tier

Args:
    comment_parent_id (str) : The name id of the comments parent
Returns:
    True (bool): Whenever the comment lies on tier 1
    False (bool): Whenever the comment lies on any other tier
```

def d_create_Big_CSV.check_script_arguments ()

```
Checks if enough and correct arguments have been given to run this script adequate

1. It checks in the first instance if enough arguments have been given
2. Then necessary variables will be filled with appropriate values
```

```
Args:
    -
Returns:
    -
```

def d_create_Big_CSV.generate_data ()

```
Starts calculating various information about thread and iama behaviour related to the year which
is currently
    being processed

After the caluclations have every iteration the results will ber appended to a list, which will
contain all that
    information for the current year... That list will be writtend to csv and appended to a global
list in other
    methods

Args:
    -
Returns:
    -
```

def d_create_Big_CSV.initialize_mongo_db_parameters (*actually_processed_year*)

```
Instantiates all necessary variables for the correct usage of the mongoDB-Client

Args:
    actually_processed_year (int) : The year with which parameters the database should be accessed
Returns:
    -
```

def d_create_Big_CSV.process_specific_thread (*thread_id*, *thread_creation_time_stamp*, *thread_author*)

```
Does the needed operations, for gaining information / knowledge about threads on the given thread
id

After the caluclations have every iteration the results will ber appended to a list, which will
contain all that
    information for the current year... That list will be writtend to csv and appended to a global
list in other
    methods

Args:
    thread_id (str) : The id, needed for operating (i.E. comparison of parent - child relation)
    thread_creation_time_stamp (int) : Creation time stamp of thread, needed for time difference
calculation
    thread_author (str): The name of the threads author, needed for answer checking of a post
Returns:
    -
```

def d_create_Big_CSV.start_data_generation_for_analysis ()

```
Starts the whole combination of generating data, checking data and writing them into csv files

1. Triggers the data generation process and moves forward within the years -
    by moving through the years a csv file will be created for every year
```

```
Args:
    -
Returns:
    -
```

def d_create_Big_CSV.write_csv_data (*list_with_information*)

```
Creates a csv file containing all necessary information about the thread and its mannerism to do
research on

Args:
    list_with_information (list) : Contains various information about threads mannerism
Returns:
    -
```

Variable Documentation

int d_create_Big_CSV.argument_year_beginning = 0

int d_create_Big_CSV.argument_year_ending = 0

list d_create_Big_CSV.list_current_year = []

list d_create_Big_CSV.list_global_year = []

int d_create_Big_CSV.year_actually_in_progress = 0

PlotlyBarChart Namespace Reference

Classes

- class PlotlyBarChart

PlotlyBarChart_5_Bars Namespace Reference

Classes

- class PlotlyBarChart5Bars

Class Documentation

PlotlyBarChart.PlotlyBarChart Class Reference

Public Member Functions

- def __init__ (self)
- def main_method (self, list_of_calculated_data)

Static Public Member Functions

- def fill_x_axis_list (list_of_calculated_data)
- def fill_y_axis_answered_list (list_of_calculated_data)
- def fill_y_axis_unanswered_list (list_of_calculated_data)
- def fill_bar_percentages_values (list_of_calculated_data)
- def fill_chart_title_description (list_of_calculated_data)
- def fill_bar_description (list_of_calculated_data)
- def generate_chart ()

Static Public Attributes

- time_now_date = time.strftime("%d.%m.%Y")
- time_now_time = time.strftime("%H:%M:%S")
- string bar_x_axis_text = 'Chart creation date: '
- string chart_title = ""
- list bar_value_description = []
- list bar_x_axis_values = []
- list bar_y_axis_first_values = []
- list bar_y_axis_second_values = []
- list bar_first_n_second_values_percentage = []

Detailed Description

```
The class to create a stacked bar chart.
    This class is heavily modified because it pyplot normally is not designed to run offline this way..

Args:
    -
Returns:
    -
```

Constructor & Destructor Documentation

def PlotlyBarChart.PlotlyBarChart.__init__ (*self*)

```
Instanciates the class

Args:
    -
Returns:
    -
```

Member Function Documentation

def PlotlyBarChart.PlotlyBarChart.fill_bar_description (*list_of_calculated_data*) [static]

```
Defines the bar description in dependence to given parameters list_of_calculated_data[0][0]

Args:
    list_of_calculated_data (list) : Will be accessed to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.fill_bar_percentages_values (*list_of_calculated_data*) [static]

```
Calculates percentages to be shown within the graph..
    This is not supported within pyplot under normal circumstances.. so we're tricking the HTML
settings

Args:
    list of calculated data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.fill_chart_title_description (*list_of_calculated_data*) [static]

```
Defines the chart title in dependence to sorting method and processed years

Args:
    list_of_calculated_data (list) : Will be accessed to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.fill_x_axis_list (*list_of_calculated_data*) [static]

```
Fills the "x axis" with the values of the years

Args:
    list of calculated data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.fill_y_axis_answered_list (*list_of_calculated_data*) [static]

```
Fills an bar within the chart with values of the amount of unanswered questions

Args:
    list_of_calculated_data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.fill_y_axis_unanswered_list (*list_of_calculated_data*) `[static]`

```
Fills an bar within the chart with values of the amount of unanswered questions

Args:
    list_of_calculated_data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.generate_chart () `[static]`

```
Generates the chart "temp-plot.html" which will be automatically opened within the browser

Args:
    -
Returns:
    -
```

def PlotlyBarChart.PlotlyBarChart.main_method (*self*, *list_of_calculated_data*)

```
Sequential fills the necessary varibales for the graph
Structure of list of calculated data:

[ "sorting", [year, answered, unanswered], [year, answered, unanswered], ... ]
i.e. ["top",
      [2009, 900, 1536],
      [2010, 500, 500],
      [2011, 300, 700]
      ]

Args:
    list_of_calculated_data (list): Contains sorting method, and the years data
Returns:
    -
```

Member Data Documentation

list PlotlyBarChart.PlotlyBarChart.bar_first_n_second_values_percentage = [] `[static]`

list PlotlyBarChart.PlotlyBarChart.bar_value_description = [] `[static]`

string PlotlyBarChart.PlotlyBarChart.bar_x_axis_text = 'Chart creation date: ' `[static]`

list PlotlyBarChart.PlotlyBarChart.bar_x_axis_values = [] `[static]`

list PlotlyBarChart.PlotlyBarChart.bar_y_axis_first_values = [] `[static]`

list PlotlyBarChart.PlotlyBarChart.bar_y_axis_second_values = [] `[static]`

string PlotlyBarChart.PlotlyBarChart.chart_title = "" `[static]`

PlotlyBarChart.PlotlyBarChart.time_now_date = time.strftime("%d.%m.%Y") `[static]`

PlotlyBarChart.PlotlyBarChart.time_now_time = time.strftime("%H:%M:%S") `[static]`

The documentation for this class was generated from the following file:

- PlotlyBarChart.py

PlotlyBarChart_5_Bars.PlotlyBarChart5Bars Class Reference

Public Member Functions

- def __init__ (self)
- def main_method (self, list_of_calculated_data)

Static Public Member Functions

- def fill_x_axis_list (list_of_calculated_data)
- def fill_y_axis_values (list_of_calculated_data)
- def fill_bar_percentages_values (list_of_calculated_data)
- def fill_chart_title_description (list_of_calculated_data)
- def fill_bar_description (list_of_calculated_data)
- def fill_bar_annotations ()
- def generate_chart ()

Static Public Attributes

- string color_1 = 'rgba(255, 114, 86, 1.0)'
- string color_1_border = 'rgba(238, 106, 80, 1.0)'
- string color_2 = 'rgba(238, 118, 0, 1.0)'
- string color_2_border = 'rgba(205, 102, 0, 1.0)'
- string color_3 = 'rgba(0, 201, 87, 1.0)'
- string color_3_border = 'rgba(0, 139, 0, 1.0)'
- string color_4 = 'rgba(0, 205, 205, 1.0)'
- string color_4_border = 'rgba(0, 139, 139, 1.0)'
- string color_5 = 'rgba(137, 104, 205, 1.0)'
- string color_5_border = 'rgba(39, 71, 139, 1.0)'
- time_now_date = time.strftime("%d.%m.%Y")
- time_now_time = time.strftime("%H:%M:%S")
- string bar_x_axis_text = 'Chart creation date: '
- string chart_title = ""
- list bar_value_description = []
- list bar_x_axis_values = []
- list bar_y_axis_first_values = []
- list bar_y_axis_second_values = []
- list bar_y_axis_third_values = []
- list bar_y_axis_fourth_values = []
- list bar_y_axis_fifth_values = []
- list bar_percentages_values_1 = []
- list bar_percentages_values_2 = []
- list bar_percentages_values_3 = []
- list bar_percentages_values_4 = []
- list bar_percentages_values_5 = []
- list annotations_1 = []
- list annotations_2 = []
- list annotations_3 = []
- list annotations_4 = []
- list annotations_5 = []
- list annotations_all = []

Detailed Description

```
The class to create a stacked bar chart.
    This class is heavily modified because it pyplot normally is not designed to run offline this way..

Args:
    -
Returns:
    -
```

Constructor & Destructor Documentation

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.__init__ (*self*)

```
Instanciates the class

Args:
    -
Returns:
    -
```

Member Function Documentation

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_bar_annotations ()`[static]`

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_bar_description (*list_of_calculated_data*)`[static]`

```
Defines the bar description in dependence to given parameters list_of_calculated_data[0][0]

Args:
    list of calculated data (list) : Will be accessed to gain necessary values
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_bar_percentages_values (*list_of_calculated_data*)`[static]`

```
Calculates percentages to be shown within the graph..
    This is not supported within pyplot under normal circumstances.. so we're tricking the HTML
settings

Args:
    list_of_calculated_data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_chart_title_description (*list_of_calculated_data*) `[static]`

```
Defines the chart title in dependence to sorting method and processed years

Args:
    list_of_calculated_data (list) : Will be accessed to gain necessary values
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_x_axis_list (*list_of_calculated_data*) `[static]`

```
Fills the "x axis" with the values of the years

Args:
    list_of_calculated_data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_y_axis_values (*list_of_calculated_data*) `[static]`

```
Fills an bar within the chart with values of the amount of unanswered questions

Args:
    list of calculated data (list) : Will be iterated to gain necessary values
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.generate_chart () `[static]`

```
Generates the chart "temp-plot.html" which will be automatically opened within the browser

Args:
    -
Returns:
    -
```

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.main_method (*self*, *list_of_calculated_data*)

```
Sequential fills the necessary varibales for the graph
Structure of list of calculated data:

[ "sorting", [year, answered, unanswered], [year, answered, unanswered], ... ]
i.e. ["top",
      [2009, 900, 1536],
      [2010, 500, 500],
      [2011, 300, 700]
      ]

Args:
    list of calculated data (list): Contains sorting method, and the years data
Returns:
    -
```

Member Data Documentation

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_1 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_2 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_3 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_4 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_5 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.annotations_all = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_percentages_values_1 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_percentages_values_2 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_percentages_values_3 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_percentages_values_4 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_percentages_values_5 = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_value_description = [] [static]

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_x_axis_text = 'Chart creation date: ' [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_x_axis_values = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_y_axis_fifth_values = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_y_axis_first_values = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_y_axis_fourth_values = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_y_axis_second_values = [] [static]

list PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.bar_y_axis_third_values = [] [static]

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.chart_title = "" [static]

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_1 = 'rgba(255, 114, 86, 1.0)' [static]

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_1_border = 'rgba(238, 106, 80, 1.0)' [static]

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_2 = 'rgba(238, 118, 0, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_2_border = 'rgba(205, 102, 0, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_3 = 'rgba(0, 201, 87, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_3_border = 'rgba(0, 139, 0, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_4 = 'rgba(0, 205, 205, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_4_border = 'rgba(0, 139, 139, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_5 = 'rgba(137, 104, 205, 1.0)' `[static]`

string PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.color_5_border = 'rgba(39, 71, 139, 1.0)' `[static]`

PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.time_now_date = time.strftime("%d.%m.%Y") `[static]`

PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.time_now_time = time.strftime("%H:%M:%S") `[static]`

The documentation for this class was generated from the following file:

- PlotlyBarChart_5_Bars.py

File Documentation

a__everything_Big_CSV_analyzer.py File Reference

Namespaces

- a__everything_Big_CSV_analyzer

Functions

- def a__everything_Big_CSV_analyzer.relation_question_upvotes_with_amount_of_questions_answered_by_iama_host ()
- def a__everything_Big_CSV_analyzer.average_means_of_values_f_threads ()
- def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_upvotes_and_iama_host_response_time_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_upvotes_and_iama_host_response_time_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_amount_of_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_amount_of_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_amount_of_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_amount_of_question ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iama_host_response_time_to_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iama_host_response_time_to_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iama_host_response_time_to_comments ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iama_host_response_time_to_questions ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_amount_of_comments_the_iama_host_answered_to ()

- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_amount_of_questions_the_iama_host_answered_to ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_amount_of_comments_the_iama_host_answered_to ()
- def a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_amount_of_questions_the_iama_host_answered_to ()
- def a__everything_Big_CSV_analyzer.relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host ()
- def a__everything_Big_CSV_analyzer.realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host ()
- def a__everything_Big_CSV_analyzer.relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host ()
- def a__everything_Big_CSV_analyzer.thread_overall_correlation ()
- def a__everything_Big_CSV_analyzer.question_overall_correlation ()
- def a__everything_Big_CSV_analyzer.average_means_of_values_f_authors ()

Variables

- a__everything_Big_CSV_analyzer.author_information_iama
- a__everything_Big_CSV_analyzer.author_information_random
- a__everything_Big_CSV_analyzer.thread_information
- a__everything_Big_CSV_analyzer.question_information
- a__everything_Big_CSV_analyzer.author_amount_creation_iama_threads = author_information_iama['amount_creation_iama_threads']
- a__everything_Big_CSV_analyzer.author_amount_creation_other_threads = author_information_iama['amount_creation_other_threads']
- a__everything_Big_CSV_analyzer.author_amount_of_comments_except_iama = author_information_iama['amount_of_comments_except_iama']
- a__everything_Big_CSV_analyzer.author_amount_of_comments_iama = author_information_iama['amount_of_comments_iama']
- a__everything_Big_CSV_analyzer.author_author_birth_date = author_information_iama['author_birth_date']
- a__everything_Big_CSV_analyzer.author_author_comment_karma_amount = author_information_iama['author_comment_karma_amount']
- a__everything_Big_CSV_analyzer.author_author_link_karma_amount = author_information_iama['author_link_karma_amount']
- a__everything_Big_CSV_analyzer.author_author_name = author_information_iama['author_name']
- a__everything_Big_CSV_analyzer.author_comment_creation_every_x_sec = author_information_iama['comment_creation_every_x_sec']
- a__everything_Big_CSV_analyzer.author_thread_creation_every_x_sec = author_information_iama['thread_creation_every_x_sec']
- a__everything_Big_CSV_analyzer.author_time_acc_birth_first_iama_thread = author_information_iama['time_acc_birth_first_iama_thread']
- a__everything_Big_CSV_analyzer.author_time_diff_acc_creation_n_first_comment = author_information_iama['time_diff_acc_creation_n_first_comment']
- a__everything_Big_CSV_analyzer.author_time_diff_acc_creation_n_first_thread = author_information_iama['time_diff_acc_creation_n_first_thread']
- a__everything_Big_CSV_analyzer.random_author_amount_creation_iama_threads = author_information_random['amount_creation_iama_threads']
- a__everything_Big_CSV_analyzer.random_author_amount_creation_other_threads = author_information_random['amount_creation_other_threads']

- a__everything_Big_CSV_analyzer.random_author_amount_of_comments_except_iama = author_information_random['amount_of_comments_except_iama']
- a__everything_Big_CSV_analyzer.random_author_amount_of_comments_iama = author_information_random['amount_of_comments_iama']
- a__everything_Big_CSV_analyzer.random_author_author_birth_date = author_information_random['author_birth_date']
- a__everything_Big_CSV_analyzer.random_author_author_comment_karma_amount = author_information_random['author_comment_karma_amount']
- a__everything_Big_CSV_analyzer.random_author_author_link_karma_amount = author_information_random['author_link_karma_amount']
- a__everything_Big_CSV_analyzer.random_author_author_name = author_information_random['author_name']
- a__everything_Big_CSV_analyzer.random_author_comment_creation_every_x_sec = author_information_random['comment_creation_every_x_sec']
- a__everything_Big_CSV_analyzer.random_author_thread_creation_every_x_sec = author_information_random['thread_creation_every_x_sec']
- a__everything_Big_CSV_analyzer.random_author_time_acc_birth_first_iama_thread = author_information_random['time_acc_birth_first_iama_thread']
- a__everything_Big_CSV_analyzer.random_author_time_diff_acc_creation_n_first_comment = \
- a__everything_Big_CSV_analyzer.random_author_time_diff_acc_creation_n_first_thread = author_information_random['time_diff_acc_creation_n_first_thread']
- a__everything_Big_CSV_analyzer.thread_year = thread_information['Year']
- a__everything_Big_CSV_analyzer.thread_id = thread_information['Thread id']
- a__everything_Big_CSV_analyzer.thread_author = thread_information['Thread author']
- a__everything_Big_CSV_analyzer.thread_ups = thread_information['Thread ups']
- a__everything_Big_CSV_analyzer.thread_downs = thread_information['Thread downs']
- a__everything_Big_CSV_analyzer.thread_creation_time_stamp = thread_information['Thread creation time stamp']
- a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_total
- a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_tier_1
- a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_tier_x
- a__everything_Big_CSV_analyzer.thread_average_question_vote_score_total
- a__everything_Big_CSV_analyzer.thread_average_question_vote_score_tier_1
- a__everything_Big_CSV_analyzer.thread_average_question_vote_score_tier_x
- a__everything_Big_CSV_analyzer.thread_num_comments_total_skewed
- a__everything_Big_CSV_analyzer.thread_num_comments_total = thread_information['Thread num comments total']
- a__everything_Big_CSV_analyzer.thread_num_comments_tier_1 = thread_information['Thread num comments tier 1']
- a__everything_Big_CSV_analyzer.thread_num_comments_tier_x = thread_information['Thread num comments tier x']
- a__everything_Big_CSV_analyzer.thread_num_questions_total = thread_information['Thread num questions total']
- a__everything_Big_CSV_analyzer.thread_num_questions_tier_1 = thread_information['Thread num questions tier 1']
- a__everything_Big_CSV_analyzer.thread_num_questions_tier_x = thread_information['Thread num questions tier x']
- a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_total
- a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_tier_1
- a__everything_Big_CSV_analyzer.thread_num_questions_answered_by_iama_host_tier_x
- a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_total
- a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_tier_1
- a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iama_host_tier_x
- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_total
- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_tier_1
- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_comments_tier_x

- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_total
- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_tier_1
- a__everything_Big_CSV_analyzer.thread_average_reaction_time_between_questions_tier_x
- a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_total
- a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_tier_1
- a__everything_Big_CSV_analyzer.thread_average_response_to_comment_time_iama_host_tier_x
- a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_total
- a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_tier_1
- a__everything_Big_CSV_analyzer.thread_average_response_to_question_time_iama_host_tier_x
- a__everything_Big_CSV_analyzer.thread_amount_of_questioners_total
- a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_1
- a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_x
- a__everything_Big_CSV_analyzer.thread_amount_of_commentators_total
- a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_1
- a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_x
- a__everything_Big_CSV_analyzer.thread_life_span_until_last_comment
- a__everything_Big_CSV_analyzer.thread_life_span_until_last_question
- a__everything_Big_CSV_analyzer.question_ups = question_information['Question ups']
- a__everything_Big_CSV_analyzer.question_answered_by_iAMA_host

a_author_Information.py File Reference

Namespaces

- a_author_Information

Functions

- def a_author_Information.check_script_arguments ()
- def a_author_Information.initialize_mongo_db_parameters ()
- def a_author_Information.write_csv_data ()

Variables

- a_author_Information.mongo_db_client_instance = None
- a_author_Information.mongo_db_author_instance = None
- a_author_Information.mongo_db_author_collection = None
- int a_author_Information.mongo_db_author_collection_original = 0
- string a_author_Information.argument_db_to_choose = ""

a_iAMA_Commenttime.py File Reference

Namespaces

- a_iAMA_Commenttime

Functions

- def a_iAMA_Commenttime.check_script_arguments ()
- def a_iAMA_Commenttime.initialize_mongo_db_parameters (actually_processed_year)
- def a_iAMA_Commenttime.start_data_generation_for_analysis ()
- def a_iAMA_Commenttime.prepare_data_for_graph ()
- def a_iAMA_Commenttime.add_thread_list_to_global_list (list_to_append)
- def a_iAMA_Commenttime.generate_data_to_be_analyzed ()
- def a_iAMA_Commenttime.calculate_ar_mean_answer_time_for_questions (id_of_thread, author_of_thread)
- def a_iAMA_Commenttime.check_if_comment_is_a_question (given_string)
- def a_iAMA_Commenttime.check_if_comment_is_on_tier_1 (comment_parent_id)
- def a_iAMA_Commenttime.check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def a_iAMA_Commenttime.check_if_comment_is_answer_from_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def a_iAMA_Commenttime.calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def a_iAMA_Commenttime.write_csv_data (list_with_information)
- def a_iAMA_Commenttime.plot_generated_data ()

Variables

- int a_iAMA_Commenttime.argument_year_beginning = 0
- int a_iAMA_Commenttime.year_actually_in_progress = 0
- int a_iAMA_Commenttime.argument_year_ending = 0
- string a_iAMA_Commenttime.argument_tier_in_scope = ""
- string a_iAMA_Commenttime.argument_plot_time_unit = ""
- a_iAMA_Commenttime.mongo_DB_Client_Instance = None
- a_iAMA_Commenttime.mongo_DB_Threads_Instance = None
- a_iAMA_Commenttime.mongo_DB_Thread_Collection = None
- a_iAMA_Commenttime.mongo_DB_Comments_Instance = None
- list a_iAMA_Commenttime.list_To_Be_Plotted = []
- list a_iAMA_Commenttime.global_thread_list = []
- list a_iAMA_Commenttime.data_to_give_plotly = []

a_question_Answered_Yes_No_Extrema.py File Reference

Namespaces

- a_question_Answered_Yes_No_Extrema

Functions

- def a_question_Answered_Yes_No_Extrema.check_script_arguments ()
- def a_question_Answered_Yes_No_Extrema.initialize_mongo_db_parameters (actually_processed_year)
- def a_question_Answered_Yes_No_Extrema.start_data_generation_for_analysis ()
- def a_question_Answered_Yes_No_Extrema.generate_data_now ()
- def a_question_Answered_Yes_No_Extrema.process_answered_questions_within_thread (id_of_thread, author_of_thread, thread_creation_date)
- def a_question_Answered_Yes_No_Extrema.check_if_comment_is_a_question (given_string)
- def a_question_Answered_Yes_No_Extrema.check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def a_question_Answered_Yes_No_Extrema.check_if_comment_has_been_answered_by_thread_author (author_of_thread, comment_acutal_id, comments_cursor)
- def a_question_Answered_Yes_No_Extrema.calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def a_question_Answered_Yes_No_Extrema.sort_questions (list_which_is_to_be_sorted)
- def a_question_Answered_Yes_No_Extrema.create_question_list_containing_all_years (list_with_comments_per_years)
- def a_question_Answered_Yes_No_Extrema.write_csv_and_count_unanswered (list_with_comments)
- def a_question_Answered_Yes_No_Extrema.plot_generated_data ()

Variables

- int a_question_Answered_Yes_No_Extrema.argument_year_beginning = 0
- int a_question_Answered_Yes_No_Extrema.year_actually_in_progress = 0
- int a_question_Answered_Yes_No_Extrema.argument_year_ending = 0
- a_question_Answered_Yes_No_Extrema.argument_sorting = bool
- int a_question_Answered_Yes_No_Extrema.argument_amount_of_top_quotes = 0
- a_question_Answered_Yes_No_Extrema.mongo_DB_Client_Instance = None
- a_question_Answered_Yes_No_Extrema.mongo_DB_Threads_Instance = None
- a_question_Answered_Yes_No_Extrema.mongo_DB_Thread_Collection = None
- a_question_Answered_Yes_No_Extrema.mongo_DB_Comments_Instance = None
- list a_question_Answered_Yes_No_Extrema.question_information_list = []
- list a_question_Answered_Yes_No_Extrema.data_to_give_plotly = []

a_question_Answered_Yes_No_Tier_Percentage.py File Reference

Namespaces

- a_question_Answered_Yes_No_Tier_Percentage

Functions

- def a_question_Answered_Yes_No_Tier_Percentage.check_script_arguments ()
- def a_question_Answered_Yes_No_Tier_Percentage.initialize_mongo_db_parameters (actually_processed_year)
- def a_question_Answered_Yes_No_Tier_Percentage.start_data_generation_for_analysis ()
- def a_question_Answered_Yes_No_Tier_Percentage.generate_data_to_be_analyzed ()
- def a_question_Answered_Yes_No_Tier_Percentage.question_answering_distribution_tier1_tierx_tierany (id_of_thread, author_of_thread)
- def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_a_question (given_string)
- def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_on_tier_1 (comment_parent_id)
- def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_answer_from_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def a_question_Answered_Yes_No_Tier_Percentage.write_csv (list_with_information)
- def a_question_Answered_Yes_No_Tier_Percentage.add_local_list_to_global_list (list_to_append)
- def a_question_Answered_Yes_No_Tier_Percentage.prepare_data_for_graph ()
- def a_question_Answered_Yes_No_Tier_Percentage.plot_generated_data ()

Variables

- int a_question_Answered_Yes_No_Tier_Percentage.argument_year_beginning = 0
- int a_question_Answered_Yes_No_Tier_Percentage.year_actually_in_progress = 0
- int a_question_Answered_Yes_No_Tier_Percentage.argument_year_ending = 0
- string a_question_Answered_Yes_No_Tier_Percentage.argument_tier_in_scope = ""
- a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Client_Instance = None
- a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Threads_Instance = None
- a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Thread_Collection = None
- a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Comments_Instance = None
- list a_question_Answered_Yes_No_Tier_Percentage.global_question_list = []
- list a_question_Answered_Yes_No_Tier_Percentage.year_question_list = []
- list a_question_Answered_Yes_No_Tier_Percentage.data_to_give_plotly = []

a_question_Tier_Distribution.py File Reference

Namespaces

- a_question_Tier_Distribution

Functions

- def a_question_Tier_Distribution.initialize_mongo_db_parameters (actually_processed_year)
- def a_question_Tier_Distribution.check_script_arguments ()
- def a_question_Tier_Distribution.start_data_generation_for_analysis ()
- def a_question_Tier_Distribution.generate_data_to_be_analyzed ()
- def a_question_Tier_Distribution.question_distribution_tier1_tierx (id_of_thread, author_of_thread)
- def a_question_Tier_Distribution.check_if_comment_is_a_question (given_string)
- def a_question_Tier_Distribution.check_if_comment_is_on_tier_1 (comment_parent_id)
- def a_question_Tier_Distribution.check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def a_question_Tier_Distribution.add_actual_year_list_to_global_list (list_to_append)
- def a_question_Tier_Distribution.write_csv (list_with_information)
- def a_question_Tier_Distribution.prepare_data_for_graph ()
- def a_question_Tier_Distribution.plot_generated_data ()

Variables

- int a_question_Tier_Distribution.argument_year_beginning = 0
- int a_question_Tier_Distribution.year_actually_in_progress = 0
- int a_question_Tier_Distribution.argument_year_ending = 0
- a_question_Tier_Distribution.mongo_DB_Client_Instance = None
- a_question_Tier_Distribution.mongo_DB_Threads_Instance = None
- a_question_Tier_Distribution.mongo_DB_Thread_Collection = None
- a_question_Tier_Distribution.mongo_DB_Comments_Instance = None
- list a_question_Tier_Distribution.current_year_question_list = []
- list a_question_Tier_Distribution.global_year_question_list = []
- list a_question_Tier_Distribution.data_to_give_plotly = []

a_thread_Lifespan_N_Average_Commenttime.py File Reference

Namespaces

- a_thread_Lifespan_N_Average_Commenttime

Functions

- def a_thread_Lifespan_N_Average_Commenttime.check_script_arguments ()
- def a_thread_Lifespan_N_Average_Commenttime.initialize_mongo_db_parameters (actually_processed_year)
- def a_thread_Lifespan_N_Average_Commenttime.start_data_generation_for_analysis ()
- def a_thread_Lifespan_N_Average_Commenttime.prepare_data_for_graph_life_span ()
- def a_thread_Lifespan_N_Average_Commenttime.prepare_data_for_comment_time ()
- def a_thread_Lifespan_N_Average_Commenttime.generate_data_to_be_analyzed ()
- def a_thread_Lifespan_N_Average_Commenttime.calculate_time_difference (id_of_thread, creation_date_of_thread)
- def a_thread_Lifespan_N_Average_Commenttime.write_csv (list_with_information)
- def a_thread_Lifespan_N_Average_Commenttime.add_thread_list_to_global_list (list_to_append)
- def a_thread_Lifespan_N_Average_Commenttime.prepare_dict_by_time_separation_for_comment_time ()
- def a_thread_Lifespan_N_Average_Commenttime.plot_generated_data ()

Variables

- int a_thread_Lifespan_N_Average_Commenttime.argument_year_beginning = 0
- string a_thread_Lifespan_N_Average_Commenttime.argument_calculation = ""
- int a_thread_Lifespan_N_Average_Commenttime.argument_year_ending = 0
- int a_thread_Lifespan_N_Average_Commenttime.year_actually_in_progress = 0
- string a_thread_Lifespan_N_Average_Commenttime.argument_plot_time_unit = ""
- a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Client_Instance = None
- a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Threads_Instance = None
- a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Thread_Collection = None
- a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Comments_Instance = None
- list a_thread_Lifespan_N_Average_Commenttime.global_thread_list = []
- list a_thread_Lifespan_N_Average_Commenttime.temp_time_difference_list = []
- list a_thread_Lifespan_N_Average_Commenttime.list_with_currents_year_infos = []
- list a_thread_Lifespan_N_Average_Commenttime.data_to_give_plotly = []

c_crawl_Author_Information.py File Reference

Namespaces

- c_crawl_Author_Information

Functions

- def c_crawl_Author_Information.check_script_arguments ()
- def c_crawl_Author_Information.initialize_mongo_db_parameters (actually_processed_year)
- def c_crawl_Author_Information.start_data_generation_for_analysis ()
- def c_crawl_Author_Information.generate_data_now ()
- def c_crawl_Author_Information.calculate_time_difference (time_value_1, time_value_2)
- def c_crawl_Author_Information.get_author_information (name_of_author)

Variables

- int c_crawl_Author_Information.argument_year_beginning = 0
- int c_crawl_Author_Information.year_actually_in_progress = 0
- int c_crawl_Author_Information.argument_year_ending = 0
- string c_crawl_Author_Information.argument_inverse_crawling = ""
- c_crawl_Author_Information.mongo_db_client_instance = None
- c_crawl_Author_Information.mongo_db_threads_instance = None
- c_crawl_Author_Information.mongo_db_thread_collection = None
- c_crawl_Author_Information.mongo_db_author_instance = None
- c_crawl_Author_Information.reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

c_crawl_Differences.py File Reference

Namespaces

- c_crawl_Differences

Functions

- def c_crawl_Differences.check_script_arguments ()
- def c_crawl_Differences.initialize_mongo_db_parameters ()
- def c_crawl_Differences.crawl_missing_collection_into_comments_database (name_of_missing_collection)
- def c_crawl_Differences.check_if_collection_is_missing_in_comments_database ()
- def c_crawl_Differences.crawl_missing_collection_into_threads_database (name_of_missing_collection)
- def c_crawl_Differences.check_if_collection_is_missing_in_threads_database ()
- def c_crawl_Differences.start_crawling_for_diffs ()

Variables

- c_crawl_Differences.mongo_DB_Client_Instance = None
- c_crawl_Differences.mongo_DB_Threads_Instance = None
- c_crawl_Differences.mongo_DB_Thread_Collection = None
- c_crawl_Differences.mongo_DB_Comments_Instance = None
- c_crawl_Differences.mongo_DB_Comments_Collection = None
- string c_crawl_Differences.argument_year_beginning = ""
- string c_crawl_Differences.argument_year_ending = ""
- string c_crawl_Differences.argument_inverse_crawling = ""

c_crawl_Random_Author_Information.py File Reference

Namespaces

- c_crawl_Random_Author_Information

Functions

- def c_crawl_Random_Author_Information.check_script_arguments ()
- def c_crawl_Random_Author_Information.initialize_mongo_db_parameters ()
- def c_crawl_Random_Author_Information.start_data_generation_for_analysis ()
- def c_crawl_Random_Author_Information.generate_data_now (randomized_author_name)
- def c_crawl_Random_Author_Information.calculate_time_difference (time_value_1, time_value_2)
- def c_crawl_Random_Author_Information.get_author_information (name_of_author)

Variables

- c_crawl_Random_Author_Information.argument_limit_crawling_amount = None
- c_crawl_Random_Author_Information.mongo_db_client_instance = None
- c_crawl_Random_Author_Information.mongo_db_random_author_instance = None
- c_crawl_Random_Author_Information.mongo_db_random_author_collection = None
- c_crawl_Random_Author_Information.mongo_db_iama_author_instance = None
- c_crawl_Random_Author_Information.mongo_db_iama_author_collection = None
- int c_crawl_Random_Author_Information.mongo_db_iama_author_collection_amount = 0
- c_crawl_Random_Author_Information.reddit_instance = praw.Reddit(user_agent="University_Regensburg_iAMA_Crawler_0.001")

c_crawl_Threads_N_Comments.py File Reference

Namespaces

- c_crawl_Threads_N_Comments

Functions

- def c_crawl_Threads_N_Comments.initialize_mongo_db_parameters ()
- def c_crawl_Threads_N_Comments.check_script_arguments ()
- def c_crawl_Threads_N_Comments.convert_argument_year_to_epoch (year)
- def c_crawl_Threads_N_Comments.crawl_data ()
- def c_crawl_Threads_N_Comments.crawl_threads ()
- def c_crawl_Threads_N_Comments.crawl_comments ()
- def c_crawl_Threads_N_Comments.check_if_coll_in_db_already_exists_up2date (submission)

Variables

- c_crawl_Threads_N_Comments.mongo_DB_Client_Instance = None
- c_crawl_Threads_N_Comments.reddit_Instance = None
- c_crawl_Threads_N_Comments.argument_crawl_type = None
- c_crawl_Threads_N_Comments.argument_year_beginning = None
- c_crawl_Threads_N_Comments.argument_year_end = None
- c_crawl_Threads_N_Comments.argument_hours_to_shift = None
- c_crawl_Threads_N_Comments.time_shift_difference

d_create_Big_CSV.py File Reference

Namespaces

- d_create_Big_CSV

Functions

- def d_create_Big_CSV.check_script_arguments ()
- def d_create_Big_CSV.initialize_mongo_db_parameters (actually_processed_year)
- def d_create_Big_CSV.start_data_generation_for_analysis ()
- def d_create_Big_CSV.generate_data ()
- def d_create_Big_CSV.process_specific_thread (thread_id, thread_creation_time_stamp, thread_author)
- def d_create_Big_CSV.check_if_comment_is_a_question (given_string)
- def d_create_Big_CSV.check_if_comment_is_on_tier_1 (comment_parent_id)
- def d_create_Big_CSV.check_if_comment_is_not_from_thread_author (author_of_thread, comment_author)
- def d_create_Big_CSV.check_if_comment_has_been_answered_by_thread_author (author_of_thread, comment_actual_id, comments_cursor)
- def d_create_Big_CSV.calculate_time_difference (comment_time_stamp, answer_time_stamp_iama_host)
- def d_create_Big_CSV.calculate_reaction_time_average (list_to_be_processed, thread_creation_time_stamp)
- def d_create_Big_CSV.calculate_life_span (thread_creation_time_stamp, time_value_of_last_comment, time_value_of_last_question)
- def d_create_Big_CSV.add_actual_year_list_to_global_list (list_to_append)
- def d_create_Big_CSV.write_csv_data (list_with_information)

Variables

- int d_create_Big_CSV.argument_year_beginning = 0
- int d_create_Big_CSV.argument_year_ending = 0
- int d_create_Big_CSV.year_actually_in_progress = 0
- list d_create_Big_CSV.list_current_year = []
- list d_create_Big_CSV.list_global_year = []

PlotlyBarChart.py File Reference

Classes

- class PlotlyBarChart.PlotlyBarChart

Namespaces

- PlotlyBarChart

PlotlyBarChart_5_Bars.py File Reference

Classes

- class PlotlyBarChart_5_Bars.PlotlyBarChart5Bars

Namespaces

- PlotlyBarChart_5_Bars

Index

__init__
PlotlyBarChart::PlotlyBarChart 65
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
a__everything_Big_CSV_analyzer 5
author_amount_creation_iama_threads 14
author_amount_creation_other_threads 14
author_amount_of_comments_except_iama 14
author_amount_of_comments_iama 14
author_author_birth_date 14
author_author_comment_karma_amount 14
author_author_link_karma_amount 14
author_author_name 14
author_comment_creation_every_x_sec 14
author_information_iama 14
author_information_random 14
author_thread_creation_every_x_sec 15
author_time_acc_birth_first_iama_thread 15
author_time_diff_acc_creation_n_first_comment 15
author_time_diff_acc_creation_n_first_thread 15
average_means_of_values_f_authors 7
average_means_of_values_f_threads 7
question_answered_by_iAMA_host 15
question_information 15
question_overall_correlation 7
question_ups 16
random_author_amount_creation_iama_threads 16
random_author_amount_creation_other_threads 16
random_author_amount_of_comments_except_iama 16
random_author_amount_of_comments_iama 16
random_author_author_birth_date 16
random_author_author_comment_karma_amount 16
random_author_author_link_karma_amount 16
random_author_author_name 16
random_author_comment_creation_every_x_sec 16
random_author_thread_creation_every_x_sec 16
random_author_time_acc_birth_first_iama_thread 16
random_author_time_diff_acc_creation_n_first_comment 16
random_author_time_diff_acc_creation_n_first_thread 16
realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host 8
relation_question_upvotes_with_amount_of_questions_answered_by_iama_host 8
relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host 8
relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host 8
relation_thread_downvotes_and_iama_host_response_time_comments 8
relation_thread_downvotes_and_iama_host_response_time_questions 9
relation_thread_downvotes_with_amount_of_comments 9
relation_thread_downvotes_with_amount_of_questions 9
relation_thread_lifespan_to_last_comment_and_amount_of_comments 9
relation_thread_lifespan_to_last_comment_and_amount_of_questions 9
relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_comments 10
relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions 10
relation_thread_lifespan_to_last_question_and_amount_of_comments 10
relation_thread_lifespan_to_last_question_and_amount_of_question 10
relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_comments 10
relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_questions 11
relation_thread_reaction_time_comments_and_amount_of_comments_the_iama_host_answered_to 11
relation_thread_reaction_time_comments_and_amount_of_questions_the_iama_host_answered_to 11
relation_thread_reaction_time_comments_and_iama_host_response_time_to_comments 11
relation_thread_reaction_time_comments_and_iama_host_response_time_to_questions 12
relation_thread_reaction_time_questions_and_amount_of_comments_the_iama_host_answered_to 12
relation_thread_reaction_time_questions_and_amount_of_questions_the_iama_host_answered_to 12
relation_thread_reaction_time_questions_and_iama_host_response_time_to_comments 12
relation_thread_reaction_time_questions_and_iama_host_response_time_to_questions 13
relation_thread_upvotes_and_iama_host_response_time_comments 13
relation_thread_upvotes_and_iama_host_response_time_questions 13

relation_thread_upvotes_with_amount_of_comments 13
relation_thread_upvotes_with_amount_of_questions 13
thread_amount_of_commentators_tier_1 16
thread_amount_of_commentators_tier_x 16
thread_amount_of_commentators_total 16
thread_amount_of_questioners_tier_1 17
thread_amount_of_questioners_tier_x 17
thread_amount_of_questioners_total 17
thread_author 17
thread_average_comment_vote_score_tier_1 17
thread_average_comment_vote_score_tier_x 17
thread_average_comment_vote_score_total 17
thread_average_question_vote_score_tier_1 17
thread_average_question_vote_score_tier_x 17
thread_average_question_vote_score_total 17
thread_average_reaction_time_between_comments_tier_1 17
thread_average_reaction_time_between_comments_tier_x 17
thread_average_reaction_time_between_comments_total 17
thread_average_reaction_time_between_questions_tier_1 17
thread_average_reaction_time_between_questions_tier_x 17
thread_average_reaction_time_between_questions_total 18
thread_average_response_to_comment_time_iama_host_tier_1 18
thread_average_response_to_comment_time_iama_host_tier_x 18
thread_average_response_to_comment_time_iama_host_total 18
thread_average_response_to_question_time_iama_host_tier_1 18
thread_average_response_to_question_time_iama_host_tier_x 18
thread_average_response_to_question_time_iama_host_total 18
thread_creation_time_stamp 18
thread_downs 18
thread_id 18
thread_information 18
thread_life_span_until_last_comment 18
thread_life_span_until_last_question 18
thread_num_comments_answered_by_iama_host_tier_1 18
thread_num_comments_answered_by_iama_host_tier_x 18
thread_num_comments_answered_by_iama_host_total 19
thread_num_comments_tier_1 19
thread_num_comments_tier_x 19
thread_num_comments_total 19
thread_num_comments_total_skewed 19
thread_num_questions_answered_by_iama_host_tier_1 19
thread_num_questions_answered_by_iama_host_tier_x 19
thread_num_questions_answered_by_iama_host_total 19
thread_num_questions_tier_1 19
thread_num_questions_tier_x 19
thread_num_questions_total 19
thread_overall_correlation 14
thread_ups 19
thread_year 19
a__everything_Big_CSV_analyzer.py 75
a_author_Information 20
argument_db_to_choose 21
check_script_arguments 20
initialize_mongo_db_parameters 20
mongo_db_author_collection 21
mongo_db_author_collection_original 21
mongo_db_author_instance 21
mongo_db_client_instance 21
write_csv_data 20
a_author_Information.py 79
a_iAMA_Commenttime 22
add_thread_list_to_global_list 22
argument_plot_time_unit 26
argument_tier_in_scope 26
argument_year_beginning 26
argument_year_ending 26
calculate_ar_mean_answer_time_for_questions 23
calculate_time_difference 23
check_if_comment_is_a_question 23
check_if_comment_is_answer_from_thread_author 23
check_if_comment_is_not_from_thread_author 24
check_if_comment_is_on_tier_1 24
check_script_arguments 24
data_to_give_plotly 26
generate_data_to_be_analyzed 25
global_thread_list 26
initialize_mongo_db_parameters 25
list_To_Be_Plotted 26
mongo_DB_Client_Instance 26
mongo_DB_Comments_Instance 26
mongo_DB_Thread_Collection 26
mongo_DB_Threads_Instance 26
plot_generated_data 25
prepare_data_for_graph 25
start_data_generation_for_analysis 25
write_csv_data 26
year_actually_in_progress 26
a_iAMA_Commenttime.py 80
a_question_Answered_Yes_No_Extrema 27
argument_amount_of_top_quotes 31
argument_sorting 31

argument_year_beginning 31
argument_year_ending 31
calculate_time_difference 27
check_if_comment_has_been_answered_by_thread_author 28
check_if_comment_is_a_question 28
check_if_comment_is_not_from_thread_author 28
check_script_arguments 28
create_question_list_containing_all_years 29
data_to_give_plotly 31
generate_data_now 29
initialize_mongo_db_parameters 29
mongo_DB_Client_Instance 31
mongo_DB_Comments_Instance 31
mongo_DB_Thread_Collection 31
mongo_DB_Threads_Instance 31
plot_generated_data 29
process_answered_questions_within_thread 30
question_information_list 31
sort_questions 30
start_data_generation_for_analysis 30
write_csv_and_count_unanswered 31
year_actually_in_progress 31
a_question_Answered_Yes_No_Extrema.py 81
a_question_Answered_Yes_No_Tier_Percentage 32
add_local_list_to_global_list 32
argument_tier_in_scope 36
argument_year_beginning 36
argument_year_ending 36
check_if_comment_is_a_question 33
check_if_comment_is_answer_from_thread_author 33
check_if_comment_is_not_from_thread_author 33
check_if_comment_is_on_tier_1 33
check_script_arguments 34
data_to_give_plotly 36
generate_data_to_be_analyzed 34
global_question_list 36
initialize_mongo_db_parameters 34
mongo_DB_Client_Instance 36
mongo_DB_Comments_Instance 36
mongo_DB_Thread_Collection 36
mongo_DB_Threads_Instance 36
plot_generated_data 34
prepare_data_for_graph 35
question_answering_distribution_tier1_tierx_tieran y 35
start_data_generation_for_analysis 35
write_csv 35
year_actually_in_progress 36
year_question_list 36
a_question_Answered_Yes_No_Tier_Percentage.py 82
a_question_Tier_Distribution 37
add_actual_year_list_to_global_list 37
argument_year_beginning 40
argument_year_ending 40
check_if_comment_is_a_question 37
check_if_comment_is_not_from_thread_author 38
check_if_comment_is_on_tier_1 38
check_script_arguments 38
current_year_question_list 40
data_to_give_plotly 40
generate_data_to_be_analyzed 38
global_year_question_list 40
initialize_mongo_db_parameters 39
mongo_DB_Client_Instance 40
mongo_DB_Comments_Instance 40
mongo_DB_Thread_Collection 40
mongo_DB_Threads_Instance 40
plot_generated_data 39
prepare_data_for_graph 39
question_distribution_tier1_tierx 39
start_data_generation_for_analysis 39
write_csv 40
year_actually_in_progress 40
a_question_Tier_Distribution.py 83
a_thread_Lifespan_N_Average_Commenttime 41
add_thread_list_to_global_list 41
argument_calculation 44
argument_plot_time_unit 44
argument_year_beginning 44
argument_year_ending 44
calculate_time_difference 41
check_script_arguments 42
data_to_give_plotly 44
generate_data_to_be_analyzed 42
global_thread_list 44
initialize_mongo_db_parameters 42
list_with_currents_year_infos 44
mongo_DB_Client_Instance 44
mongo_DB_Comments_Instance 44
mongo_DB_Thread_Collection 44
mongo_DB_Threads_Instance 44
plot_generated_data 43
prepare_data_for_comment_time 43
prepare_data_for_graph_life_span 43
prepare_dict_by_time_separation_for_comment_time 43
start_data_generation_for_analysis 43
temp_time_difference_list 44
write_csv 44
year_actually_in_progress 44
a_thread_Lifespan_N_Average_Commenttime.py 84
add_actual_year_list_to_global_list
a_question_Tier_Distribution 37
d_create_Big_CSV 58
add_local_list_to_global_list
a_question_Answered_Yes_No_Tier_Percentage 32

add_thread_list_to_global_list
a_iAMA_Commenttime 22
a_thread_Lifespan_N_Average_Commenttime 41
annotations_1
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
annotations_2
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
annotations_3
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
annotations_4
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
annotations_5
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
annotations_all
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
argument_amount_of_top_quotes
a_question_Answered_Yes_No_Extrema 31
argument_calculation
a_thread_Lifespan_N_Average_Commenttime 44
argument_crawl_type
c_crawl_Threads_N_Comments 56
argument_db_to_choose
a_author_Information 21
argument_hours_to_shift
c_crawl_Threads_N_Comments 56
argument_inverse_crawling
c_crawl_Author_Information 47
c_crawl_Differences 50
argument_limit_crawling_amount
c_crawl_Random_Author_Information 53
argument_plot_time_unit
a_iAMA_Commenttime 26
a_thread_Lifespan_N_Average_Commenttime 44
argument_sorting
a_question_Answered_Yes_No_Extrema 31
argument_tier_in_scope
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Tier_Percentage 36
argument_year_beginning
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Author_Information 47
c_crawl_Differences 50
c_crawl_Threads_N_Comments 56
d_create_Big_CSV 62
argument_year_end
c_crawl_Threads_N_Comments 56
argument_year_ending
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Author_Information 47
c_crawl_Differences 50
d_create_Big_CSV 62
author_amount_creation_iama_threads
a__everything_Big_CSV_analyzer 14
author_amount_creation_other_threads
a__everything_Big_CSV_analyzer 14
author_amount_of_comments_except_iama
a__everything_Big_CSV_analyzer 14
author_amount_of_comments_iama
a__everything_Big_CSV_analyzer 14
author_author_birth_date
a__everything_Big_CSV_analyzer 14
author_author_comment_karma_amount
a__everything_Big_CSV_analyzer 14
author_author_link_karma_amount
a__everything_Big_CSV_analyzer 14
author_author_name
a__everything_Big_CSV_analyzer 14
author_comment_creation_every_x_sec
a__everything_Big_CSV_analyzer 14
author_information_iama
a__everything_Big_CSV_analyzer 14
author_information_random
a__everything_Big_CSV_analyzer 14
author_thread_creation_every_x_sec
a__everything_Big_CSV_analyzer 15
author_time_acc_birth_first_iama_thread
a__everything_Big_CSV_analyzer 15
author_time_diff_acc_creation_n_first_comment
a__everything_Big_CSV_analyzer 15
author_time_diff_acc_creation_n_first_thread
a__everything_Big_CSV_analyzer 15
average_means_of_values_f_authors
a__everything_Big_CSV_analyzer 7
average_means_of_values_f_threads
a__everything_Big_CSV_analyzer 7
bar_first_n_second_values_percentage
PlotlyBarChart::PlotlyBarChart 68
bar_percentages_values_1
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_percentages_values_2
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_percentages_values_3
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_percentages_values_4
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_percentages_values_5
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_value_description
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_x_axis_text
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73

bar_x_axis_values
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_y_axis_fifth_values
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_y_axis_first_values
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_y_axis_fourth_values
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_y_axis_second_values
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
bar_y_axis_third_values
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
c_crawl_Author_Information 45
argument_inverse_crawling 47
argument_year_beginning 47
argument_year_ending 47
calculate_time_difference 45
check_script_arguments 45
generate_data_now 45
get_author_information 46
initialize_mongo_db_parameters 46
mongo_db_author_instance 47
mongo_db_client_instance 47
mongo_db_thread_collection 47
mongo_db_threads_instance 47
reddit_instance 47
start_data_generation_for_analysis 47
year_actually_in_progress 47
c_crawl_Author_Information.py 85
c_crawl_Differences 48
argument_inverse_crawling 50
argument_year_beginning 50
argument_year_ending 50
check_if_collection_is_missing_in_comments_database 48
check_if_collection_is_missing_in_threads_database 48
check_script_arguments 49
crawl_missing_collection_into_comments_database 49
crawl_missing_collection_into_threads_database 49
initialize_mongo_db_parameters 50
mongo_DB_Client_Instance 50
mongo_DB_Comments_Collection 50
mongo_DB_Comments_Instance 50
mongo_DB_Thread_Collection 50
mongo_DB_Threads_Instance 50
start_crawling_for_diffs 50
c_crawl_Differences.py 86
c_crawl_Random_Author_Information 51
argument_limit_crawling_amount 53
calculate_time_difference 51
check_script_arguments 51
generate_data_now 51
get_author_information 52
initialize_mongo_db_parameters 52
mongo_db_client_instance 53
mongo_db_iama_author_collection 53
mongo_db_iama_author_collection_amount 53
mongo_db_iama_author_instance 53
mongo_db_random_author_collection 53
mongo_db_random_author_instance 53
reddit_instance 53
start_data_generation_for_analysis 53
c_crawl_Random_Author_Information.py 87
c_crawl_Threads_N_Comments 54
argument_crawl_type 56
argument_hours_to_shift 56
argument_year_beginning 56
argument_year_end 56
check_if_coll_in_db_already_exists_up2date 54
check_script_arguments 54
convert_argument_year_to_epoch 55
crawl_comments 55
crawl_data 55
crawl_threads 55
initialize_mongo_db_parameters 56
mongo_DB_Client_Instance 56
reddit_Instance 56
time_shift_difference 56
c_crawl_Threads_N_Comments.py 88
calculate_ar_mean_answer_time_for_questions
a_iAMA_Commenttime 23
calculate_life_span
d_create_Big_CSV 58
calculate_reaction_time_average
d_create_Big_CSV 59
calculate_time_difference
a_iAMA_Commenttime 23
a_question_Answered_Yes_No_Extrema 27
a_thread_Lifespan_N_Average_Commenttime 41
c_crawl_Author_Information 45
c_crawl_Random_Author_Information 51
d_create_Big_CSV 59
chart_title
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
check_if_coll_in_db_already_exists_up2date
c_crawl_Threads_N_Comments 54
check_if_collection_is_missing_in_comments_database
c_crawl_Differences 48
check_if_collection_is_missing_in_threads_database
c_crawl_Differences 48
check_if_comment_has_been_answered_by_thread_author
a_question_Answered_Yes_No_Extrema 28
d_create_Big_CSV 59
check_if_comment_is_a_question
a_iAMA_Commenttime 23

a_question_Answered_Yes_No_Extrema 28
a_question_Answered_Yes_No_Tier_Percentage 33
a_question_Tier_Distribution 37
d_create_Big_CSV 60
check_if_comment_is_answer_from_thread_author
a_iAMA_Commenttime 23
a_question_Answered_Yes_No_Tier_Percentage 33
check_if_comment_is_not_from_thread_author
a_iAMA_Commenttime 24
a_question_Answered_Yes_No_Extrema 28
a_question_Answered_Yes_No_Tier_Percentage 33
a_question_Tier_Distribution 38
d_create_Big_CSV 60
check_if_comment_is_on_tier_1
a_iAMA_Commenttime 24
a_question_Answered_Yes_No_Tier_Percentage 33
a_question_Tier_Distribution 38
d_create_Big_CSV 60
check_script_arguments
a_author_Information 20
a_iAMA_Commenttime 24
a_question_Answered_Yes_No_Extrema 28
a_question_Answered_Yes_No_Tier_Percentage 34
a_question_Tier_Distribution 38
a_thread_Lifespan_N_Average_Commenttime 42
c_crawl_Author_Information 45
c_crawl_Differences 49
c_crawl_Random_Author_Information 51
c_crawl_Threads_N_Comments 54
d_create_Big_CSV 60
color_1
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
color_1_border
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 73
color_2
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_2_border
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_3
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_3_border
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_4
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_4_border
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_5
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
color_5_border
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
convert_argument_year_to_epoch
c_crawl_Threads_N_Comments 55
crawl_comments
c_crawl_Threads_N_Comments 55
crawl_data
c_crawl_Threads_N_Comments 55
crawl_missing_collection_into_comments_database
c_crawl_Differences 49
crawl_missing_collection_into_threads_database
c_crawl_Differences 49
crawl_threads
c_crawl_Threads_N_Comments 55
create_question_list_containing_all_years
a_question_Answered_Yes_No_Extrema 29
current_year_question_list
a_question_Tier_Distribution 40
d_create_Big_CSV 58
add_actual_year_list_to_global_list 58
argument_year_beginning 62
argument_year_ending 62
calculate_life_span 58
calculate_reaction_time_average 59
calculate_time_difference 59
check_if_comment_has_been_answered_by_thread_author 59
check_if_comment_is_a_question 60
check_if_comment_is_not_from_thread_author 60
check_if_comment_is_on_tier_1 60
check_script_arguments 60
generate_data 61
initialize_mongo_db_parameters 61
list_current_year 62
list_global_year 62
process_specific_thread 61
start_data_generation_for_analysis 61
write_csv_data 62
year_actually_in_progress 62
d_create_Big_CSV.py 89
data_to_give_plotly
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
fill_bar_annotations
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
fill_bar_description
PlotlyBarChart::PlotlyBarChart 66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
fill_bar_percentages_values
PlotlyBarChart::PlotlyBarChart 66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
fill_chart_title_description
PlotlyBarChart::PlotlyBarChart 66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 71
fill_x_axis_list
PlotlyBarChart::PlotlyBarChart 66

PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 71
fill_y_axis_answered_list
PlotlyBarChart::PlotlyBarChart 66
fill_y_axis_unanswered_list
PlotlyBarChart::PlotlyBarChart 67
fill_y_axis_values
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 71
generate_chart
PlotlyBarChart::PlotlyBarChart 67
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 71
generate_data
d_create_Big_CSV 61
generate_data_now
a_question_Answered_Yes_No_Extrema 29
c_crawl_Author_Information 45
c_crawl_Random_Author_Information 51
generate_data_to_be_analyzed
a_iAMA_Commenttime 25
a_question_Answered_Yes_No_Tier_Percentage 34
a_question_Tier_Distribution 38
a_thread_Lifespan_N_Average_Commenttime 42
get_author_information
c_crawl_Author_Information 46
c_crawl_Random_Author_Information 52
global_question_list
a_question_Answered_Yes_No_Tier_Percentage 36
global_thread_list
a_iAMA_Commenttime 26
a_thread_Lifespan_N_Average_Commenttime 44
global_year_question_list
a_question_Tier_Distribution 40
initialize_mongo_db_parameters
a_author_Information 20
a_iAMA_Commenttime 25
a_question_Answered_Yes_No_Extrema 29
a_question_Answered_Yes_No_Tier_Percentage 34
a_question_Tier_Distribution 39
a_thread_Lifespan_N_Average_Commenttime 42
c_crawl_Author_Information 46
c_crawl_Differences 50
c_crawl_Random_Author_Information 52
c_crawl_Threads_N_Comments 56
d_create_Big_CSV 61
list_current_year
d_create_Big_CSV 62
list_global_year
d_create_Big_CSV 62
list_To_Be_Plotted
a_iAMA_Commenttime 26
list_with_currents_year_infos
a_thread_Lifespan_N_Average_Commenttime 44
main_method
PlotlyBarChart::PlotlyBarChart 67
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 71
mongo_db_author_collection
a_author_Information 21
mongo_db_author_collection_original
a_author_Information 21
mongo_db_author_instance
a_author_Information 21
c_crawl_Author_Information 47
mongo_db_client_instance
a_author_Information 21
c_crawl_Author_Information 47
c_crawl_Random_Author_Information 53
mongo_DB_Client_Instance
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Differences 50
c_crawl_Threads_N_Comments 56
mongo_DB_Comments_Collection
c_crawl_Differences 50
mongo_DB_Comments_Instance
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Differences 50
mongo_db_iama_author_collection
c_crawl_Random_Author_Information 53
mongo_db_iama_author_collection_amount
c_crawl_Random_Author_Information 53
mongo_db_iama_author_instance
c_crawl_Random_Author_Information 53
mongo_db_random_author_collection
c_crawl_Random_Author_Information 53
mongo_db_random_author_instance
c_crawl_Random_Author_Information 53
mongo_db_thread_collection
c_crawl_Author_Information 47
mongo_DB_Thread_Collection
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Differences 50
mongo_db_threads_instance
c_crawl_Author_Information 47
mongo_DB_Threads_Instance
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36

a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Differences 50
plot_generated_data
a_iAMA_Commenttime 25
a_question_Answered_Yes_No_Extrema 29
a_question_Answered_Yes_No_Tier_Percentage 34
a_question_Tier_Distribution 39
a_thread_Lifespan_N_Average_Commenttime 43
PlotlyBarChart 63
PlotlyBarChart.PlotlyBarChart 65
PlotlyBarChart.py 90
PlotlyBarChart::PlotlyBarChart
__init__ 65
bar_first_n_second_values_percentage 68
bar_value_description 68
bar_x_axis_text 68
bar_x_axis_values 68
bar_y_axis_first_values 68
bar_y_axis_second_values 68
chart_title 68
fill_bar_description 66
fill_bar_percentages_values 66
fill_chart_title_description 66
fill_x_axis_list 66
fill_y_axis_answered_list 66
fill_y_axis_unanswered_list 67
generate_chart 67
main_method 67
time_now_date 68
time_now_time 68
PlotlyBarChart_5_Bars 64
PlotlyBarChart_5_Bars.PlotlyBarChart5Bars 69
PlotlyBarChart_5_Bars.py 91
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars
__init__ 70
annotations_1 73
annotations_2 73
annotations_3 73
annotations_4 73
annotations_5 73
annotations_all 73
bar_percentages_values_1 73
bar_percentages_values_2 73
bar_percentages_values_3 73
bar_percentages_values_4 73
bar_percentages_values_5 73
bar_value_description 73
bar_x_axis_text 73
bar_x_axis_values 73
bar_y_axis_fifth_values 73
bar_y_axis_first_values 73
bar_y_axis_fourth_values 73
bar_y_axis_second_values 73
bar_y_axis_third_values 73
chart_title 73
color_1 73
color_1_border 73
color_2 74
color_2_border 74
color_3 74
color_3_border 74
color_4 74
color_4_border 74
color_5 74
color_5_border 74
fill_bar_annotations 70
fill_bar_description 70
fill_bar_percentages_values 70
fill_chart_title_description 71
fill_x_axis_list 71
fill_y_axis_values 71
generate_chart 71
main_method 71
time_now_date 74
time_now_time 74
prepare_data_for_comment_time
a_thread_Lifespan_N_Average_Commenttime 43
prepare_data_for_graph
a_iAMA_Commenttime 25
a_question_Answered_Yes_No_Tier_Percentage 35
a_question_Tier_Distribution 39
prepare_data_for_graph_life_span
a_thread_Lifespan_N_Average_Commenttime 43
prepare_dict_by_time_separation_for_comment_time
a_thread_Lifespan_N_Average_Commenttime 43
process_answered_questions_within_thread
a_question_Answered_Yes_No_Extrema 30
process_specific_thread
d_create_Big_CSV 61
question_answered_by_iAMA_host
a__everything_Big_CSV_analyzer 15
question_answering_distribution_tier1_tierx_tierany
a_question_Answered_Yes_No_Tier_Percentage 35
question_distribution_tier1_tierx
a_question_Tier_Distribution 39
question_information
a__everything_Big_CSV_analyzer 15
question_information_list
a_question_Answered_Yes_No_Extrema 31
question_overall_correlation
a__everything_Big_CSV_analyzer 7
question_ups
a__everything_Big_CSV_analyzer 16
random_author_amount_creation_iama_threads
a__everything_Big_CSV_analyzer 16
random_author_amount_creation_other_threads
a__everything_Big_CSV_analyzer 16
random_author_amount_of_comments_except_iama
a__everything_Big_CSV_analyzer 16
random_author_amount_of_comments_iama

a__everything_Big_CSV_analyzer 16
random_author_author_birth_date
a__everything_Big_CSV_analyzer 16
random_author_author_comment_karma_amount
a__everything_Big_CSV_analyzer 16
random_author_author_link_karma_amount
a__everything_Big_CSV_analyzer 16
random_author_author_name
a__everything_Big_CSV_analyzer 16
random_author_comment_creation_every_x_sec
a__everything_Big_CSV_analyzer 16
random_author_thread_creation_every_x_sec
a__everything_Big_CSV_analyzer 16
random_author_time_acc_birth_first_iama_thread
a__everything_Big_CSV_analyzer 16
random_author_time_diff_acc_creation_n_first_comment
a__everything_Big_CSV_analyzer 16
random_author_time_diff_acc_creation_n_first_thread
a__everything_Big_CSV_analyzer 16
realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host
a__everything_Big_CSV_analyzer 8
reddit_instance
c_crawl_Author_Information 47
c_crawl_Random_Author_Information 53
reddit_Instance
c_crawl_Threads_N_Comments 56
relation_question_upvotes_with_amount_of_questions_answered_by_iama_host
a__everything_Big_CSV_analyzer 8
relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host
a__everything_Big_CSV_analyzer 8
relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host
a__everything_Big_CSV_analyzer 8
relation_thread_downvotes_and_iama_host_response_time_comments
a__everything_Big_CSV_analyzer 8
relation_thread_downvotes_and_iama_host_response_time_questions
a__everything_Big_CSV_analyzer 9
relation_thread_downvotes_with_amount_of_comments
a__everything_Big_CSV_analyzer 9
relation_thread_downvotes_with_amount_of_questions
a__everything_Big_CSV_analyzer 9
relation_thread_lifespan_to_last_comment_and_amount_of_comments
a__everything_Big_CSV_analyzer 9
relation_thread_lifespan_to_last_comment_and_amount_of_questions
a__everything_Big_CSV_analyzer 9
relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_comments
a__everything_Big_CSV_analyzer 10
relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions
a__everything_Big_CSV_analyzer 10
relation_thread_lifespan_to_last_question_and_amount_of_comments
a__everything_Big_CSV_analyzer 10
relation_thread_lifespan_to_last_question_and_amount_of_question
a__everything_Big_CSV_analyzer 10
relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_comments
a__everything_Big_CSV_analyzer 10
relation_thread_lifespan_to_last_question_and_iama_host_response_time_to_questions
a__everything_Big_CSV_analyzer 11
relation_thread_reaction_time_comments_and_amount_of_comments_the_iama_host_answered_to
a__everything_Big_CSV_analyzer 11
relation_thread_reaction_time_comments_and_amount_of_questions_the_iama_host_answered_to
a__everything_Big_CSV_analyzer 11
relation_thread_reaction_time_comments_and_iama_host_response_time_to_comments
a__everything_Big_CSV_analyzer 11
relation_thread_reaction_time_comments_and_iama_host_response_time_to_questions
a__everything_Big_CSV_analyzer 12
relation_thread_reaction_time_questions_and_amount_of_comments_the_iama_host_answered_to
a__everything_Big_CSV_analyzer 12
relation_thread_reaction_time_questions_and_amount_of_questions_the_iama_host_answered_to
a__everything_Big_CSV_analyzer 12
relation_thread_reaction_time_questions_and_iama_host_response_time_to_comments
a__everything_Big_CSV_analyzer 12
relation_thread_reaction_time_questions_and_iama_host_response_time_to_questions
a__everything_Big_CSV_analyzer 13
relation_thread_upvotes_and_iama_host_response_time_comments
a__everything_Big_CSV_analyzer 13
relation_thread_upvotes_and_iama_host_response_time_questions
a__everything_Big_CSV_analyzer 13
relation_thread_upvotes_with_amount_of_comments
a__everything_Big_CSV_analyzer 13
relation_thread_upvotes_with_amount_of_questions
a__everything_Big_CSV_analyzer 13
sort_questions
a_question_Answered_Yes_No_Extrema 30
start_crawling_for_diffs
c_crawl_Differences 50
start_data_generation_for_analysis

a_iAMA_Commenttime 25
a_question_Answered_Yes_No_Extrema 30
a_question_Answered_Yes_No_Tier_Percentage 35
a_question_Tier_Distribution 39
a_thread_Lifespan_N_Average_Commenttime 43
c_crawl_Author_Information 47
c_crawl_Random_Author_Information 53
d_create_Big_CSV 61
temp_time_difference_list
a_thread_Lifespan_N_Average_Commenttime 44
thread_amount_of_commentators_tier_1
a__everything_Big_CSV_analyzer 16
thread_amount_of_commentators_tier_x
a__everything_Big_CSV_analyzer 16
thread_amount_of_commentators_total
a__everything_Big_CSV_analyzer 16
thread_amount_of_questioners_tier_1
a__everything_Big_CSV_analyzer 17
thread_amount_of_questioners_tier_x
a__everything_Big_CSV_analyzer 17
thread_amount_of_questioners_total
a__everything_Big_CSV_analyzer 17
thread_author
a__everything_Big_CSV_analyzer 17
thread_average_comment_vote_score_tier_1
a__everything_Big_CSV_analyzer 17
thread_average_comment_vote_score_tier_x
a__everything_Big_CSV_analyzer 17
thread_average_comment_vote_score_total
a__everything_Big_CSV_analyzer 17
thread_average_question_vote_score_tier_1
a__everything_Big_CSV_analyzer 17
thread_average_question_vote_score_tier_x
a__everything_Big_CSV_analyzer 17
thread_average_question_vote_score_total
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_comments_tier_1
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_comments_tier_x
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_comments_total
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_questions_tier_1
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_questions_tier_x
a__everything_Big_CSV_analyzer 17
thread_average_reaction_time_between_questions_total
a__everything_Big_CSV_analyzer 18
thread_average_response_to_comment_time_iama_host_tier_1
a__everything_Big_CSV_analyzer 18
thread_average_response_to_comment_time_iama_host_tier_x
a__everything_Big_CSV_analyzer 18
thread_average_response_to_comment_time_iama_host_total
a__everything_Big_CSV_analyzer 18
thread_average_response_to_question_time_iama_host_tier_1
a__everything_Big_CSV_analyzer 18
thread_average_response_to_question_time_iama_host_tier_x
a__everything_Big_CSV_analyzer 18
thread_average_response_to_question_time_iama_host_total
a__everything_Big_CSV_analyzer 18
thread_creation_time_stamp
a__everything_Big_CSV_analyzer 18
thread_downs
a__everything_Big_CSV_analyzer 18
thread_id
a__everything_Big_CSV_analyzer 18
thread_information
a__everything_Big_CSV_analyzer 18
thread_life_span_until_last_comment
a__everything_Big_CSV_analyzer 18
thread_life_span_until_last_question
a__everything_Big_CSV_analyzer 18
thread_num_comments_answered_by_iama_host_tier_1
a__everything_Big_CSV_analyzer 18
thread_num_comments_answered_by_iama_host_tier_x
a__everything_Big_CSV_analyzer 18
thread_num_comments_answered_by_iama_host_total
a__everything_Big_CSV_analyzer 19
thread_num_comments_tier_1
a__everything_Big_CSV_analyzer 19
thread_num_comments_tier_x
a__everything_Big_CSV_analyzer 19
thread_num_comments_total
a__everything_Big_CSV_analyzer 19
thread_num_comments_total_skewed
a__everything_Big_CSV_analyzer 19
thread_num_questions_answered_by_iama_host_tier_1
a__everything_Big_CSV_analyzer 19
thread_num_questions_answered_by_iama_host_tier_x
a__everything_Big_CSV_analyzer 19
thread_num_questions_answered_by_iama_host_total
a__everything_Big_CSV_analyzer 19
thread_num_questions_tier_1
a__everything_Big_CSV_analyzer 19
thread_num_questions_tier_x

a__everything_Big_CSV_analyzer 19
thread_num_questions_total
a__everything_Big_CSV_analyzer 19
thread_overall_correlation
a__everything_Big_CSV_analyzer 14
thread_ups
a__everything_Big_CSV_analyzer 19
thread_year
a__everything_Big_CSV_analyzer 19
time_now_date
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
time_now_time
PlotlyBarChart::PlotlyBarChart 68
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 74
time_shift_difference
c_crawl_Threads_N_Comments 56
write_csv
a_question_Answered_Yes_No_Tier_Percentage 35
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
write_csv_and_count_unanswered
a_question_Answered_Yes_No_Extrema 31
write_csv_data
a_author_Information 20
a_iAMA_Commenttime 26
d_create_Big_CSV 62
year_actually_in_progress
a_iAMA_Commenttime 26
a_question_Answered_Yes_No_Extrema 31
a_question_Answered_Yes_No_Tier_Percentage 36
a_question_Tier_Distribution 40
a_thread_Lifespan_N_Average_Commenttime 44
c_crawl_Author_Information 47
d_create_Big_CSV 62
year_question_list
a_question_Answered_Yes_No_Tier_Percentage 36