

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

Benedikt Hierl
Version 1.0
28 April 2016

Table of Contents

Namespace Index	2
Class Index	3
File Index	4
a__everything_Big_CSV_analyzer	5
a_iAMA_Commenttime	19
a_question_Answered_Yes_No_Extrema	25
a_question_Answered_Yes_No_Tier_Percentage	31
a_question_Tier_Distribution	37
a_thread_Lifespan_N_Average_Commenttime	42
c_crawl_Differences	47
c_crawl_Threads_N_Comments	51
d_create_Big_CSV	55
PlotlyBarChart	61
PlotlyBarChart_5_Bars	62
Class Documentation	63
PlotlyBarChart.PlotlyBarChart	63
PlotlyBarChart_5_Bars.PlotlyBarChart5Bars	67
File Documentation	74
a__everything_Big_CSV_analyzer.py	74
a_iAMA_Commenttime.py	77
a_question_Answered_Yes_No_Extrema.py	78
a_question_Answered_Yes_No_Tier_Percentage.py	79
a_question_Tier_Distribution.py	80
a_thread_Lifespan_N_Average_Commenttime.py	81
c_crawl_Differences.py	82
c_crawl_Threads_N_Comments.py	83
d_create_Big_CSV.py	84
PlotlyBarChart.py	85
PlotlyBarChart_5_Bars.py	86
Index	87

Namespace Index

Packages

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

<u>a everything Big CSV analyzer</u>	5
<u>a iAMA Commenttime</u>	19
<u>a question Answered Yes No Extrema</u>	25
<u>a question Answered Yes No Tier Percentage</u>	31
<u>a question Tier Distribution</u>	37
<u>a thread Lifespan N Average Commenttime</u>	42
<u>c crawl Differences</u>	47
<u>c crawl Threads N Comments</u>	51
<u>d create Big CSV</u>	55
<u>PlotlyBarChart</u>	61
<u>PlotlyBarChart 5 Bars</u>	62

Class Index

Class List

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

<u>PlotlyBarChart.PlotlyBarChart</u>	63
<u>PlotlyBarChart_5_Bars.PlotlyBarChart5Bars</u>	67

File Index

File List

Here is a list of all files with brief descriptions:

<u>a everything Big CSV analyzer.py</u>	74
<u>a iAMA Commenttime.py</u>	77
<u>a question Answered Yes No Extrema.py</u>	78
<u>a question Answered Yes No Tier Percentage.py</u>	79
<u>a question Tier Distribution.py</u>	80
<u>a thread Lifespan N Average Commenttime.py</u>	81
<u>c crawl Differences.py</u>	82
<u>c crawl Threads N Comments.py</u>	83
<u>d create Big CSV.py</u>	84
<u>PlotlyBarChart.py</u>	85
<u>PlotlyBarChart 5 Bars.py</u>	86

Namespace Documentation

a__everything_Big_CSV_analyzer Namespace Reference

Functions

- def [relation_question_upvotes_with_amount_of_questions_answered_by_iamahost\(\)](#)
- def [average_means_of_values\(\)](#)
- 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_iamahost_response_time_comments\(\)](#)
- def [relation_thread_upvotes_and_iamahost_response_time_questions\(\)](#)
- def [relation_thread_downvotes_and_iamahost_response_time_comments\(\)](#)
- def [relation_thread_downvotes_and_iamahost_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_iamahost_response_time_to_comments\(\)](#)
- def [relation_thread_lifespan_to_last_comment_and_iamahost_response_time_to_questions\(\)](#)
- def [relation_thread_lifespan_to_last_question_and_iamahost_response_time_to_comments\(\)](#)
- def [relation_thread_lifespan_to_last_question_and_iamahost_response_time_to_questions\(\)](#)
- def [relation_thread_reaction_time_comments_and_iamahost_response_time_to_comments\(\)](#)
- def [relation_thread_reaction_time_comments_and_iamahost_response_time_to_questions\(\)](#)
- def [relation_thread_reaction_time_questions_and_iamahost_response_time_to_comments\(\)](#)
- def [relation_thread_reaction_time_questions_and_iamahost_response_time_to_questions\(\)](#)
- def [relation_thread_reaction_time_comments_and_amount_of_comments_the_iamahost_answered_to\(\)](#)
- def [relation_thread_reaction_time_comments_and_amount_of_questions_the_iamahost_answered_to\(\)](#)
- def [relation_thread_reaction_time_questions_and_amount_of_comments_the_iamahost_answered_to\(\)](#)
- def [relation_thread_reaction_time_questions_and_amount_of_questions_the_iamahost_answered_to\(\)](#)
- def [relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iamahost\(\)](#)
- def [relation_thread_amount_of_commentators_total_and_num_comments_answered_by_iamahost\(\)](#)
- def [relation_thread_amount_of_questions_and_amount_questions_answered_by_iamahost\(\)](#)
- def [thread_overall_correlation\(\)](#)
- def [question_overall_correlation\(\)](#)

Variables

- [thread_information](#)
- [question_information](#)
- [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 ()

Calculation of the average means of different values

Args:

-

Returns:

-

Definition at line 235 of file a__everything_Big_CSV_analyzer.py.

def a__everything_Big_CSV_analyzer.question_overall_correlation ()

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

Args:
-
Returns:
-

Definition at line 3132 of file a__everything_Big_CSV_analyzer.py.

```
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:
-

Definition at line 2895 of file a__everything_Big_CSV_analyzer.py.

```
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:
-

Definition at line 201 of file a__everything_Big_CSV_analyzer.py.

```
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:
-

Definition at line 2767 of file a__everything_Big_CSV_analyzer.py.

```
def  
a__everything_Big_CSV_analyzer.relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host ()
```

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

Args:

```
-  
Returns:  
-
```

Definition at line 3023 of file a__everything_Big_CSV_analyzer.py.

```
def  
a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_co  
mments ()
```

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

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

Definition at line 780 of file a__everything_Big_CSV_analyzer.py.

```
def  
a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iama_host_response_time_qu  
estions ()
```

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

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

Definition at line 864 of file a__everything_Big_CSV_analyzer.py.

```
def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_comments ()
```

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

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

Definition at line 480 of file a__everything_Big_CSV_analyzer.py.

```
def a__everything_Big_CSV_analyzer.relation_thread_downvotes_with_amount_of_questions ()
```

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

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

Definition at line 547 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 952 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1019 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1227 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1355 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1086 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1153 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1483 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 1611 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 2251 of file a__everything_Big_CSV_analyzer.py.

```
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:  
-
```

Definition at line 2380 of file a__everything_Big_CSV_analyzer.py.

```
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 response time  
to comments  
  
Args:  
-  
Returns:  
-
```

Definition at line 1739 of file a__everything_Big_CSV_analyzer.py.

```
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 response time  
to questions  
  
Args:  
-  
Returns:  
-
```

Definition at line 1867 of file a__everything_Big_CSV_analyzer.py.

```
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:
-
```

Definition at line 2509 of file a__everything_Big_CSV_analyzer.py.

```
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:
-
```

Definition at line 2638 of file a__everything_Big_CSV_analyzer.py.

```
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 response time
to comments
```

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

Definition at line 1995 of file a__everything_Big_CSV_analyzer.py.

```
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 response time
to questions
```

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

Definition at line 2123 of file a__everything_Big_CSV_analyzer.py.

```
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:
-
```

Definition at line 614 of file a__everything_Big_CSV_analyzer.py.

```
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:
-
```

Definition at line 694 of file a__everything_Big_CSV_analyzer.py.

```
def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_comments ()
```

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

Args:
-
Returns:
-
```

Definition at line 345 of file a__everything_Big_CSV_analyzer.py.

```
def a__everything_Big_CSV_analyzer.relation_thread_upvotes_with_amount_of_questions ()
```

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

Args:
-
Returns:
-
```

Definition at line 413 of file a__everything_Big_CSV_analyzer.py.

```
def a__everything_Big_CSV_analyzer.thread_overall_correlation ()
```

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

Args:
-
Returns:
-
```


Definition at line 3120 of file a__everything_Big_CSV_analyzer.py.

Variable Documentation

a__everything_Big_CSV_analyzer.question_answered_by_iAMA_host

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

Definition at line 195 of file a__everything_Big_CSV_analyzer.py.

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)
```

Definition at line 79 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.question_ups = [question_information](#)['Question ups']

Definition at line 194 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_1

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

Definition at line 183 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_tier_x

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

Definition at line 185 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_commentators_total

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

Definition at line 181 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_1

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

Definition at line 176 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_tier_x

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

Definition at line 178 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_amount_of_questioners_total

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

Definition at line 174 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_author = [thread_information](#)['Thread author']

Definition at line 102 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 110 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 112 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_average_comment_vote_score_total

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

Definition at line 107 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 117 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 119 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_average_question_vote_score_total

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

Definition at line 115 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 148 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 150 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 146 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 155 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 157 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 153 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 162 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 164 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 160 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 169 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 171 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 167 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 105 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_downs = [thread_information](#)['Thread downs']

Definition at line 104 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_id = [thread_information](#)['Thread id']

Definition at line 101 of file a__everything_Big_CSV_analyzer.py.

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")
```

Definition at line 75 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_life_span_until_last_comment

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

Definition at line 189 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_life_span_until_last_question

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

Definition at line 191 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iam_a_host_tier_1

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

Definition at line 141 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iam_a_host_tier_x

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

Definition at line 143 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_num_comments_answered_by_iam_a_host_total

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

Definition at line 139 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 125 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 126 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 124 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_num_comments_total_skewed

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

Definition at line 122 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 134 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 136 of file a__everything_Big_CSV_analyzer.py.

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']
```

Definition at line 132 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 129 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 130 of file a__everything_Big_CSV_analyzer.py.

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

Definition at line 128 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_ups = [thread_information](#)['Thread ups']

Definition at line 103 of file a__everything_Big_CSV_analyzer.py.

a__everything_Big_CSV_analyzer.thread_year = [thread_information](#)['Year']

Definition at line 100 of file a__everything_Big_CSV_analyzer.py.

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 whether 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:
    -
```

Definition at line 248 of file a_iAMA_Commenttime.py.

```
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
```

Definition at line 315 of file a_iAMA_Commenttime.py.

```
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
```

Definition at line 592 of file a_iAMA_Commenttime.py.

```
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 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 (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
```

Definition at line 490 of file a_iAMA_Commenttime.py.

```
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)
```

Definition at line 547 of file a_iAMA_Commenttime.py.

```
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 whether 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)
```

Definition at line 527 of file a_iAMA_Commenttime.py.

```
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
```

Definition at line 511 of file a_iAMA_Commenttime.py.

```
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:
-
```

Definition at line 20 of file a_iAMA_Commenttime.py.

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:
-
```

Definition at line 267 of file a_iAMA_Commenttime.py.

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:
-
```

Definition at line 47 of file a_iAMA_Commenttime.py.

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:
-
```

Definition at line 688 of file a_iAMA_Commenttime.py.

def a_iAMA_Commenttime.prepare_data_for_graph ()

```
Sorts and prepares data for graph plotting
```

```
Args:
-
```

```
Returns:  
-
```

Definition at line 147 of file a_iAMA_Commenttime.py.

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:  
-
```

Definition at line 67 of file a_iAMA_Commenttime.py.

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:  
-
```

Definition at line 631 of file a_iAMA_Commenttime.py.

Variable Documentation

string a_iAMA_Commenttime.argument_plot_time_unit = ""

Definition at line 717 of file a_iAMA_Commenttime.py.

string a_iAMA_Commenttime.argument_tier_in_scope = ""

Definition at line 714 of file a_iAMA_Commenttime.py.

int a_iAMA_Commenttime.argument_year_beginning = 0

Definition at line 705 of file a_iAMA_Commenttime.py.

int a_iAMA_Commenttime.argument_year_ending = 0

Definition at line 711 of file a_iAMA_Commenttime.py.

list a_iAMA_Commenttime.data_to_give_plotly = []

Definition at line 749 of file a_iAMA_Commenttime.py.

list a_iAMA_Commenttime.global_thread_list = []

Definition at line 735 of file a_iAMA_Commenttime.py.

list a_iAMA_Commenttime.list_To_Be_Plotted = []

Definition at line 732 of file a_iAMA_Commenttime.py.

a_iAMA_Commenttime.mongo_DB_Client_Instance = None

Definition at line 720 of file a_iAMA_Commenttime.py.

a_iAMA_Commenttime.mongo_DB_Comments_Instance = None

Definition at line 729 of file a_iAMA_Commenttime.py.

a_iAMA_Commenttime.mongo_DB_Thread_Collection = None

Definition at line 726 of file a_iAMA_Commenttime.py.

a_iAMA_Commenttime.mongo_DB_Threads_Instance = None

Definition at line 723 of file a_iAMA_Commenttime.py.

int a_iAMA_Commenttime.year_actually_in_progress = 0

Definition at line 708 of file a_iAMA_Commenttime.py.

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
```

Definition at line 425 of file a_question_Answered_Yes_No_Extrema.py.

```
def
a_question_Answered_Yes_No_Extrema.check_if_comment_has_been_answered_by_thread_auth
or ( 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)
```

Definition at line 381 of file a_question_Answered_Yes_No_Extrema.py.

```
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 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 (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
```

Definition at line 340 of file a_question_Answered_Yes_No_Extrema.py.

```
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 whether 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)
```

Definition at line 361 of file a_question_Answered_Yes_No_Extrema.py.

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:
-
```

Definition at line 22 of file a_question_Answered_Yes_No_Extrema.py.

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:
-
```

Definition at line 494 of file a_question_Answered_Yes_No_Extrema.py.

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:
-
```

Definition at line 165 of file a_question_Answered_Yes_No_Extrema.py.

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:
-
```

Definition at line 59 of file a_question_Answered_Yes_No_Extrema.py.

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:
-
```

Definition at line 583 of file a_question_Answered_Yes_No_Extrema.py.

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 whether 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
```

Definition at line 217 of file a_question_Answered_Yes_No_Extrema.py.

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
```

Definition at line 462 of file a_question_Answered_Yes_No_Extrema.py.

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:
-

```

Definition at line 79 of file `a_question_Answered_Yes_No_Extrema.py`.

`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

```

Definition at line 516 of file `a_question_Answered_Yes_No_Extrema.py`.

Variable Documentation

`int a_question_Answered_Yes_No_Extrema.argument_amount_of_top_quotes = 0`

Definition at line 613 of file `a_question_Answered_Yes_No_Extrema.py`.

`a_question_Answered_Yes_No_Extrema.argument_sorting = bool`

Definition at line 610 of file `a_question_Answered_Yes_No_Extrema.py`.

`int a_question_Answered_Yes_No_Extrema.argument_year_beginning = 0`

Definition at line 600 of file `a_question_Answered_Yes_No_Extrema.py`.

`int a_question_Answered_Yes_No_Extrema.argument_year_ending = 0`

Definition at line 606 of file `a_question_Answered_Yes_No_Extrema.py`.

`list a_question_Answered_Yes_No_Extrema.data_to_give_plotly = []`

Definition at line 642 of file a_question_Answered_Yes_No_Extrema.py.

a_question_Answered_Yes_No_Extrema.mongo_DB_Client_Instance = None

Definition at line 617 of file a_question_Answered_Yes_No_Extrema.py.

a_question_Answered_Yes_No_Extrema.mongo_DB_Comments_Instance = None

Definition at line 626 of file a_question_Answered_Yes_No_Extrema.py.

a_question_Answered_Yes_No_Extrema.mongo_DB_Thread_Collection = None

Definition at line 623 of file a_question_Answered_Yes_No_Extrema.py.

a_question_Answered_Yes_No_Extrema.mongo_DB_Threads_Instance = None

Definition at line 620 of file a_question_Answered_Yes_No_Extrema.py.

list a_question_Answered_Yes_No_Extrema.question_information_list = []

Definition at line 630 of file a_question_Answered_Yes_No_Extrema.py.

int a_question_Answered_Yes_No_Extrema.year_actually_in_progress = 0

Definition at line 603 of file a_question_Answered_Yes_No_Extrema.py.

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 whether 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:
    -
```

Definition at line 485 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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 (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
```

Definition at line 326 of file a_question_Answered_Yes_No_Tier_Percentage.py.

def a_question_Answered_Yes_No_Tier_Percentage.check_if_comment_is_answer_from_thread_auth or (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 (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
```

Definition at line 386 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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 whether 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)
```

Definition at line 365 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
    -
```

Definition at line 347 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
    -
```

Definition at line 20 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
    -
```

Definition at line 141 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
    -
```

Definition at line 45 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
-
```

Definition at line 530 of file a_question_Answered_Yes_No_Tier_Percentage.py.

def a_question_Answered_Yes_No_Tier_Percentage.prepare_data_for_graph ()

```
Sorts and prepares data for graph plotting

Args:
-
Returns:
-
```

Definition at line 504 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
-
```

Definition at line 184 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
-
```

Definition at line 65 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
    -
```

Definition at line 419 of file a_question_Answered_Yes_No_Tier_Percentage.py.

Variable Documentation

string a_question_Answered_Yes_No_Tier_Percentage.argument_tier_in_scope = ""

Definition at line 558 of file a_question_Answered_Yes_No_Tier_Percentage.py.

int a_question_Answered_Yes_No_Tier_Percentage.argument_year_beginning = 0

Definition at line 549 of file a_question_Answered_Yes_No_Tier_Percentage.py.

int a_question_Answered_Yes_No_Tier_Percentage.argument_year_ending = 0

Definition at line 555 of file a_question_Answered_Yes_No_Tier_Percentage.py.

list a_question_Answered_Yes_No_Tier_Percentage.data_to_give_plotly = []

Definition at line 589 of file a_question_Answered_Yes_No_Tier_Percentage.py.

list a_question_Answered_Yes_No_Tier_Percentage.global_question_list = []

Definition at line 574 of file a_question_Answered_Yes_No_Tier_Percentage.py.

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Client_Instance = None

Definition at line 562 of file a_question_Answered_Yes_No_Tier_Percentage.py.

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Comments_Instance = None

Definition at line 571 of file a_question_Answered_Yes_No_Tier_Percentage.py.

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Thread_Collection = None

Definition at line 568 of file a_question_Answered_Yes_No_Tier_Percentage.py.

a_question_Answered_Yes_No_Tier_Percentage.mongo_DB_Threads_Instance = None

Definition at line 565 of file a_question_Answered_Yes_No_Tier_Percentage.py.

int a_question_Answered_Yes_No_Tier_Percentage.year_actually_in_progress = 0

Definition at line 552 of file a_question_Answered_Yes_No_Tier_Percentage.py.

list a_question_Answered_Yes_No_Tier_Percentage.year_question_list = []

Definition at line 577 of file a_question_Answered_Yes_No_Tier_Percentage.py.

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:
```

```
-
```

Definition at line 329 of file a_question_Tier_Distribution.py.

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 (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
```

Definition at line 269 of file a_question_Tier_Distribution.py.

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 whether 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)
```

Definition at line 308 of file a_question_Tier_Distribution.py.

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:
    -
```

Definition at line 290 of file a_question_Tier_Distribution.py.

def a_question_Tier_Distribution.check_script_arguments ()

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

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:
    -
```

Definition at line 40 of file a_question_Tier_Distribution.py.

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:
-

```

Definition at line 143 of file a_question_Tier_Distribution.py.

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:
-

```

Definition at line 20 of file a_question_Tier_Distribution.py.

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:
-

```

Definition at line 437 of file a_question_Tier_Distribution.py.

def a_question_Tier_Distribution.prepare_data_for_graph ()

```

Sorts and prepares data for graph plotting

Args:
-
Returns:
-

```

Definition at line 412 of file a_question_Tier_Distribution.py.

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:
```

-

Definition at line 186 of file a_question_Tier_Distribution.py.

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:
-
```

Definition at line 67 of file a_question_Tier_Distribution.py.

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:
-
```

Definition at line 345 of file a_question_Tier_Distribution.py.

Variable Documentation

int a_question_Tier_Distribution.argument_year_beginning = 0

Definition at line 454 of file a_question_Tier_Distribution.py.

int a_question_Tier_Distribution.argument_year_ending = 0

Definition at line 460 of file a_question_Tier_Distribution.py.

list a_question_Tier_Distribution.current_year_question_list = []

Definition at line 476 of file a_question_Tier_Distribution.py.

list a_question_Tier_Distribution.data_to_give_plotly = []

Definition at line 491 of file a_question_Tier_Distribution.py.

list a_question_Tier_Distribution.global_year_question_list = []

Definition at line 479 of file a_question_Tier_Distribution.py.

a_question_Tier_Distribution.mongo_DB_Client_Instance = None

Definition at line 464 of file a_question_Tier_Distribution.py.

a_question_Tier_Distribution.mongo_DB_Comments_Instance = None

Definition at line 473 of file a_question_Tier_Distribution.py.

a_question_Tier_Distribution.mongo_DB_Thread_Collection = None

Definition at line 470 of file a_question_Tier_Distribution.py.

a_question_Tier_Distribution.mongo_DB_Threads_Instance = None

Definition at line 467 of file a_question_Tier_Distribution.py.

int a_question_Tier_Distribution.year_actually_in_progress = 0

Definition at line 457 of file a_question_Tier_Distribution.py.

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 whether 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:
    -
```

Definition at line 740 of file a_thread_Lifespan_N_Average_Commenttime.py.

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

Definition at line 480 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:

-

Definition at line 21 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:

-

Definition at line 422 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
    -
```

Definition at line 48 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
    -
```

Definition at line 879 of file a_thread_Lifespan_N_Average_Commenttime.py.

def a_thread_Lifespan_N_Average_Commenttime.prepare_data_for_comment_time ()

```
Prepares the average mean comment time per thread

Args:
    -
Returns:
    -
```

Definition at line 316 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
    -
```

Definition at line 215 of file a_thread_Lifespan_N_Average_Commenttime.py.

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 committed time

Args:
    -
Returns:
    -
```

Definition at line 759 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
-
```

Definition at line 68 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
-
```

Definition at line 685 of file a_thread_Lifespan_N_Average_Commenttime.py.

Variable Documentation

string a_thread_Lifespan_N_Average_Commenttime.argument_calculation = ""

Definition at line 898 of file a_thread_Lifespan_N_Average_Commenttime.py.

string a_thread_Lifespan_N_Average_Commenttime.argument_plot_time_unit = ""

Definition at line 907 of file a_thread_Lifespan_N_Average_Commenttime.py.

int a_thread_Lifespan_N_Average_Commenttime.argument_year_beginning = 0

Definition at line 895 of file a_thread_Lifespan_N_Average_Commenttime.py.

int a_thread_Lifespan_N_Average_Commenttime.argument_year_ending = 0

Definition at line 901 of file a_thread_Lifespan_N_Average_Commenttime.py.

list a_thread_Lifespan_N_Average_Commenttime.data_to_give_plotly = []

Definition at line 943 of file a_thread_Lifespan_N_Average_Commenttime.py.

list a_thread_Lifespan_N_Average_Commenttime.global_thread_list = []

Definition at line 922 of file a_thread_Lifespan_N_Average_Commenttime.py.

list a_thread_Lifespan_N_Average_Commenttime.list_with_currents_year_infos = []

Definition at line 928 of file a_thread_Lifespan_N_Average_Commenttime.py.

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Client_Instance = None

Definition at line 910 of file a_thread_Lifespan_N_Average_Commenttime.py.

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Comments_Instance = None

Definition at line 919 of file a_thread_Lifespan_N_Average_Commenttime.py.

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Thread_Collection = None

Definition at line 916 of file a_thread_Lifespan_N_Average_Commenttime.py.

a_thread_Lifespan_N_Average_Commenttime.mongo_DB_Threads_Instance = None

Definition at line 913 of file a_thread_Lifespan_N_Average_Commenttime.py.

list a_thread_Lifespan_N_Average_Commenttime.temp_time_difference_list = []

Definition at line 925 of file a_thread_Lifespan_N_Average_Commenttime.py.

int a_thread_Lifespan_N_Average_Commenttime.year_actually_in_progress = 0

Definition at line 904 of file a_thread_Lifespan_N_Average_Commenttime.py.

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:
    -
```

Definition at line 213 of file c_crawl_Differences.py.

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:
    -
```

Definition at line 353 of file c_crawl_Differences.py.

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:
-
```

Definition at line 14 of file c_crawl_Differences.py.

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:
-
```

Definition at line 76 of file c_crawl_Differences.py.

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:
```

-

Definition at line 269 of file c_crawl_Differences.py.

def c_crawl_Differences.initialize_mongo_db_parameters ()

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

Definition at line 49 of file c_crawl_Differences.py.

def c_crawl_Differences.start_crawling_for_diffs ()

```
This method starts the crawling, with the method you have defined in your arguments  
Args:  
-  
Returns:  
-
```

Definition at line 406 of file c_crawl_Differences.py.

Variable Documentation

string c_crawl_Differences.argument_inverse_crawling = ""

Definition at line 481 of file c_crawl_Differences.py.

string c_crawl_Differences.argument_year_beginning = ""

Definition at line 475 of file c_crawl_Differences.py.

string c_crawl_Differences.argument_year_ending = ""

Definition at line 478 of file c_crawl_Differences.py.

c_crawl_Differences.mongo_DB_Client_Instance = None

Definition at line 460 of file c_crawl_Differences.py.

c_crawl_Differences.mongo_DB_Comments_Collection = None

Definition at line 472 of file c_crawl_Differences.py.

c_crawl_Differences.mongo_DB_Comments_Instance = None

Definition at line 469 of file c_crawl_Differences.py.

c_crawl_Differences.mongo_DB_Thread_Collection = None

Definition at line 466 of file c_crawl_Differences.py.

c_crawl_Differences.mongo_DB_Threads_Instance = None

Definition at line 463 of file c_crawl_Differences.py.

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)

Definition at line 373 of file c_crawl_Threads_N_Comments.py.

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:
-
```

Definition at line 36 of file c_crawl_Threads_N_Comments.py.

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
```

Definition at line 71 of file c_crawl_Threads_N_Comments.py.

def c_crawl_Threads_N_Comments.crawl_comments ()

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

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:
-
```

Definition at line 258 of file c_crawl_Threads_N_Comments.py.

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:
-
```

Definition at line 121 of file c_crawl_Threads_N_Comments.py.

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:
-
```

Definition at line 140 of file c_crawl_Threads_N_Comments.py.

def c_crawl_Threads_N_Comments.initialize_mongo_db_parameters ()

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

Args:
-
Returns:
-
```

Definition at line 21 of file c_crawl_Threads_N_Comments.py.

Variable Documentation

c_crawl_Threads_N_Comments.argument_crawl_type = None

Definition at line 480 of file c_crawl_Threads_N_Comments.py.

c_crawl_Threads_N_Comments.argument_hours_to_shift = None

Definition at line 496 of file c_crawl_Threads_N_Comments.py.

c_crawl_Threads_N_Comments.argument_year_beginning = None

Definition at line 483 of file c_crawl_Threads_N_Comments.py.

c_crawl_Threads_N_Comments.argument_year_end = None

Definition at line 486 of file c_crawl_Threads_N_Comments.py.

c_crawl_Threads_N_Comments.mongo_DB_Client_Instance = None

Definition at line 474 of file c_crawl_Threads_N_Comments.py.

c_crawl_Threads_N_Comments.reddit_Instance = None

Definition at line 477 of file c_crawl_Threads_N_Comments.py.

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 )
```

Definition at line 516 of file c_crawl_Threads_N_Comments.py.

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:

```
-
```

Definition at line 1028 of file d_create_Big_CSV.py.

**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

Definition at line 973 of file d_create_Big_CSV.py.

**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

Definition at line 889 of file d_create_Big_CSV.py.

**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

```

Definition at line 850 of file d_create_Big_CSV.py.

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)

Definition at line 802 of file d_create_Big_CSV.py.

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 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 (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

Definition at line 745 of file d_create_Big_CSV.py.

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 whether 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)

Definition at line 782 of file d_create_Big_CSV.py.

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
```

Definition at line 766 of file d_create_Big_CSV.py.

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:
-
```

Definition at line 20 of file d_create_Big_CSV.py.

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:
-
```

Definition at line 105 of file d_create_Big_CSV.py.

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:
-
```

Definition at line 45 of file d_create_Big_CSV.py.

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:
    -

```

Definition at line 260 of file d_create_Big_CSV.py.

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:
    -

```

Definition at line 65 of file d_create_Big_CSV.py.

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:
    -

```

Definition at line 1044 of file d_create_Big_CSV.py.

Variable Documentation

int d_create_Big_CSV.argument_year_beginning = 0

Definition at line 1203 of file d_create_Big_CSV.py.

int d_create_Big_CSV.argument_year_ending = 0

Definition at line 1206 of file d_create_Big_CSV.py.

list d_create_Big_CSV.list_current_year = []

Definition at line 1212 of file d_create_Big_CSV.py.

```
list d_create_Big_CSV.list_global_year = []
```

Definition at line 1215 of file d_create_Big_CSV.py.

```
int d_create_Big_CSV.year_actually_in_progress = 0
```

Definition at line 1209 of file d_create_Big_CSV.py.

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:  
-
```

Definition at line 13 of file PlotlyBarChart.py.

Constructor & Destructor Documentation

def PlotlyBarChart.PlotlyBarChart.__init__ (self)

```
Instantiates the class  
Args:  
-
```

```
Returns:  
-
```

Definition at line 44 of file PlotlyBarChart.py.

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:  
-
```

Definition at line 208 of file PlotlyBarChart.py.

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:  
-
```

Definition at line 129 of file PlotlyBarChart.py.

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:  
-
```

Definition at line 160 of file PlotlyBarChart.py.

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:  
-
```

Definition at line 81 of file PlotlyBarChart.py.

```
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:  
    -
```

Definition at line 97 of file PlotlyBarChart.py.

```
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:  
    -
```

Definition at line 113 of file PlotlyBarChart.py.

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

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

Definition at line 235 of file PlotlyBarChart.py.

```
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:  
    -
```

Definition at line 55 of file PlotlyBarChart.py.

Member Data Documentation

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

Definition at line 42 of file PlotlyBarChart.py.

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

Definition at line 32 of file PlotlyBarChart.py.

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

Definition at line 26 of file PlotlyBarChart.py.

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

Definition at line 33 of file PlotlyBarChart.py.

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

Definition at line 36 of file PlotlyBarChart.py.

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

Definition at line 39 of file PlotlyBarChart.py.

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

Definition at line 29 of file PlotlyBarChart.py.

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

Definition at line 23 of file PlotlyBarChart.py.

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

Definition at line 24 of file PlotlyBarChart.py.

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:  
    -
```

Definition at line 13 of file PlotlyBarChart_5_Bars.py.

Constructor & Destructor Documentation

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.__init__ (self)

```
Instantiates the class  
  
Args:  
    -  
Returns:  
    -
```

Definition at line 71 of file PlotlyBarChart_5_Bars.py.

Member Function Documentation

def PlotlyBarChart_5_Bars.PlotlyBarChart5Bars.fill_bar_annotations () [static]

Definition at line 291 of file PlotlyBarChart_5_Bars.py.

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:  
    -
```

Definition at line 246 of file PlotlyBarChart_5_Bars.py.

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:  
-
```

Definition at line 144 of file PlotlyBarChart_5_Bars.py.

```
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:  
-
```

Definition at line 189 of file PlotlyBarChart_5_Bars.py.

```
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:  
-
```

Definition at line 108 of file PlotlyBarChart_5_Bars.py.

```
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:  
-
```

Definition at line 124 of file PlotlyBarChart_5_Bars.py.

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

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

Definition at line 393 of file PlotlyBarChart_5_Bars.py.

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



```

Sequential fills the necessary variables 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:
    -

```

Definition at line 82 of file PlotlyBarChart_5_Bars.py.

Member Data Documentation

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

Definition at line 64 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 65 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 66 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 67 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 68 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 69 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 58 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 59 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 60 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 61 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 62 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 47 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 41 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 49 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 55 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 51 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 54 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 52 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 53 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 44 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 23 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 24 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 26 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 27 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 29 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 30 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 32 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 33 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 35 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 36 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 38 of file PlotlyBarChart_5_Bars.py.

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

Definition at line 39 of file PlotlyBarChart_5_Bars.py.

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_iamahost\(\)](#)
- def [a__everything_Big_CSV_analyzer.average_means_of_values\(\)](#)
- 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_iamahost_response_time_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_upvotes_and_iamahost_response_time_questions\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iamahost_response_time_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_downvotes_and_iamahost_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_iamahost_response_time_to_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_comment_and_iamahost_response_time_to_questions\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iamahost_response_time_to_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_lifespan_to_last_question_and_iamahost_response_time_to_questions\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iamahost_response_time_to_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_iamahost_response_time_to_questions\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iamahost_response_time_to_comments\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_reaction_time_questions_and_iamahost_response_time_to_questions\(\)](#)
- def [a__everything_Big_CSV_analyzer.relation_thread_reaction_time_comments_and_amount_of_comments_the_iamahost_answered_to\(\)](#)

- [def a everything Big CSV analyzer.relation thread reaction time comments and amount of questions the ia ma host answered to \(\)](#)
- [def a everything Big CSV analyzer.relation thread reaction time questions and amount of comments the ia ma host answered to \(\)](#)
- [def a everything Big CSV analyzer.relation thread reaction time questions and amount of questions the ia ma 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.relation 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 \(\)](#)

Variables

- [a everything Big CSV analyzer.thread information](#)
- [a everything Big CSV analyzer.question information](#)
- [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_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_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_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 63
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 68
a__everything_Big_CSV_analyzer 5
average_means_of_values 6
question_answered_by_iAMA_host 14
question_information 14
question_overall_correlation 6
question_ups 14
realation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host 7
relation_question_upvotes_with_amount_of_questions_answered_by_iama_host 7
relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host 7
relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host 7
relation_thread_downvotes_and_iama_host_response_time_comments 8
relation_thread_downvotes_and_iama_host_response_time_questions 8
relation_thread_downvotes_with_amount_of_comments 8
relation_thread_downvotes_with_amount_of_questions 8
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 9
relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions 9
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 10
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 11
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 12
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 14
thread_amount_of_commentators_tier_x 14
thread_amount_of_commentators_total 14
thread_amount_of_questioners_tier_1 14
thread_amount_of_questioners_tier_x 14
thread_amount_of_questioners_total 14
thread_author 15
thread_average_comment_vote_score_tier_1 15
thread_average_comment_vote_score_tier_x 15
thread_average_comment_vote_score_total 15
thread_average_question_vote_score_tier_1 15
thread_average_question_vote_score_tier_x 15
thread_average_question_vote_score_total 15
thread_average_reaction_time_between_comments_tier_1 15
thread_average_reaction_time_between_comments_tier_x 15
thread_average_reaction_time_between_comments_total 15
thread_average_reaction_time_between_questions_tier_1 15
thread_average_reaction_time_between_questions_tier_x 16
thread_average_reaction_time_between_questions_total 16
thread_average_response_to_comment_time_iama_host_tier_1 16
thread_average_response_to_comment_time_iama_host_tier_x 16
thread_average_response_to_comment_time_iama_host_total 16
thread_average_response_to_question_time_iama_host_tier_1 16
thread_average_response_to_question_time_iama_host_tier_x 16

thread_average_response_to_question_time_iama_16
 host_total 16
 thread_creation_time_stamp 16
 thread_downs 16
 thread_id 16
 thread_information 17
 thread_life_span_until_last_comment 17
 thread_life_span_until_last_question 17
 thread_num_comments_answered_by_iama_host_tier_1 17
 thread_num_comments_answered_by_iama_host_tier_x 17
 thread_num_comments_answered_by_iama_host_total 17
 thread_num_comments_tier_1 17
 thread_num_comments_tier_x 17
 thread_num_comments_total 17
 thread_num_comments_total_skewed 17
 thread_num_questions_answered_by_iama_host_tier_1 18
 thread_num_questions_answered_by_iama_host_tier_x 18
 thread_num_questions_answered_by_iama_host_total 18
 thread_num_questions_tier_1 18
 thread_num_questions_tier_x 18
 thread_num_questions_total 18
 thread_overall_correlation 13
 thread_ups 18
 thread_year 18
 a_everything_Big_CSV_analyzer.py 74
 a_iAMA_Commenttime 19
 add_thread_list_to_global_list 19
 argument_plot_time_unit 23
 argument_tier_in_scope 23
 argument_year_beginning 23
 argument_year_ending 23
 calculate_ar_mean_answer_time_for_questions 20
 calculate_time_difference 20
 check_if_comment_is_a_question 20
 check_if_comment_is_answer_from_thread_author 21
 check_if_comment_is_not_from_thread_author 21
 check_if_comment_is_on_tier_1 21
 check_script_arguments 21
 data_to_give_plotly 24
 generate_data_to_be_analyzed 22
 global_thread_list 24
 initialize_mongo_db_parameters 22
 list_To_Be_Plotted 24
 mongo_DB_Client_Instance 24
 mongo_DB_Comments_Instance 24
 mongo_DB_Thread_Collection 24
 mongo_DB_Threads_Instance 24
 plot_generated_data 22
 prepare_data_for_graph 22
 start_data_generation_for_analysis 23
 write_csv_data 23
 year_actually_in_progress 24
 a_iAMA_Commenttime.py 77
 a_question_Answered_Yes_No_Extrema 25
 argument_amount_of_top_quotes 29
 argument_sorting 29
 argument_year_beginning 29
 argument_year_ending 29
 calculate_time_difference 25
 check_if_comment_has_been_answered_by_thread_author 26
 check_if_comment_is_a_question 26
 check_if_comment_is_not_from_thread_author 26
 check_script_arguments 27
 create_question_list_containing_all_years 27
 data_to_give_plotly 29
 generate_data_now 27
 initialize_mongo_db_parameters 27
 mongo_DB_Client_Instance 30
 mongo_DB_Comments_Instance 30
 mongo_DB_Thread_Collection 30
 mongo_DB_Threads_Instance 30
 plot_generated_data 28
 process_answered_questions_within_thread 28
 question_information_list 30
 sort_questions 28
 start_data_generation_for_analysis 28
 write_csv_and_count_unanswered 29
 year_actually_in_progress 30
 a_question_Answered_Yes_No_Extrema.py 78
 a_question_Answered_Yes_No_Tier_Percentage 31
 add_local_list_to_global_list 31
 argument_tier_in_scope 35
 argument_year_beginning 35
 argument_year_ending 35
 check_if_comment_is_a_question 32
 check_if_comment_is_answer_from_thread_author 32
 check_if_comment_is_not_from_thread_author 32
 check_if_comment_is_on_tier_1 33
 check_script_arguments 33
 data_to_give_plotly 35
 generate_data_to_be_analyzed 33
 global_question_list 35
 initialize_mongo_db_parameters 33
 mongo_DB_Client_Instance 35
 mongo_DB_Comments_Instance 35
 mongo_DB_Thread_Collection 36
 mongo_DB_Threads_Instance 36
 plot_generated_data 34
 prepare_data_for_graph 34
 question_answering_distribution_tier1_tierx_tieran_y 34

start_data_generation_for_analysis 34
 write_csv 35
 year_actually_in_progress 36
 year_question_list 36
 a_question_Answered_Yes_No_Tier_Percentage.py 79
 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 41
 data_to_give_plotly 41
 generate_data_to_be_analyzed 38
 global_year_question_list 41
 initialize_mongo_db_parameters 39
 mongo_DB_Client_Instance 41
 mongo_DB_Comments_Instance 41
 mongo_DB_Thread_Collection 41
 mongo_DB_Threads_Instance 41
 plot_generated_data 39
 prepare_data_for_graph 39
 question_distribution_tier1_tierx 39
 start_data_generation_for_analysis 40
 write_csv 40
 year_actually_in_progress 41
 a_question_Tier_Distribution.py 80
 a_thread_Lifespan_N_Average_Commenttime 42
 add_thread_list_to_global_list 42
 argument_calculation 45
 argument_plot_time_unit 45
 argument_year_beginning 45
 argument_year_ending 45
 calculate_time_difference 43
 check_script_arguments 43
 data_to_give_plotly 45
 generate_data_to_be_analyzed 43
 global_thread_list 46
 initialize_mongo_db_parameters 43
 list_with_currents_year_infos 46
 mongo_DB_Client_Instance 46
 mongo_DB_Comments_Instance 46
 mongo_DB_Thread_Collection 46
 mongo_DB_Threads_Instance 46
 plot_generated_data 44
 prepare_data_for_comment_time 44
 prepare_data_for_graph_life_span 44
 prepare_dict_by_time_separation_for_comment_time 44
 start_data_generation_for_analysis 45
 temp_time_difference_list 46
 write_csv 45
 year_actually_in_progress 46
 a_thread_Lifespan_N_Average_Commenttime.py 81
 add_actual_year_list_to_global_list
 a_question_Tier_Distribution 37
 d_create_Big_CSV 55
 add_local_list_to_global_list
 a_question_Answered_Yes_No_Tier_Percentage 31
 add_thread_list_to_global_list
 a_iAMA_Commenttime 19
 a_thread_Lifespan_N_Average_Commenttime 42
 annotations_1
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 annotations_2
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 annotations_3
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 annotations_4
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 annotations_5
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 annotations_all
 PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 70
 argument_amount_of_top_quotes
 a_question_Answered_Yes_No_Extrema 29
 argument_calculation
 a_thread_Lifespan_N_Average_Commenttime 45
 argument_crawl_type
 c_crawl_Threads_N_Comments 53
 argument_hours_to_shift
 c_crawl_Threads_N_Comments 53
 argument_inverse_crawling
 c_crawl_Differences 49
 argument_plot_time_unit
 a_iAMA_Commenttime 23
 a_thread_Lifespan_N_Average_Commenttime 45
 argument_sorting
 a_question_Answered_Yes_No_Extrema 29
 argument_tier_in_scope
 a_iAMA_Commenttime 23
 a_question_Answered_Yes_No_Tier_Percentage 35
 argument_year_beginning
 a_iAMA_Commenttime 23
 a_question_Answered_Yes_No_Extrema 29
 a_question_Answered_Yes_No_Tier_Percentage 35
 a_question_Tier_Distribution 40
 a_thread_Lifespan_N_Average_Commenttime 45
 c_crawl_Differences 49
 c_crawl_Threads_N_Comments 54
 d_create_Big_CSV 59
 argument_year_end
 c_crawl_Threads_N_Comments 54
 argument_year_ending
 a_iAMA_Commenttime 23
 a_question_Answered_Yes_No_Extrema 29

a_question_Answered_Yes_No_Tier_Percentage	35	mongo_DB_Comments_Instance	50
a_question_Tier_Distribution	40	mongo_DB_Thread_Collection	50
a_thread_Lifespan_N_Average_Commenttime	45	mongo_DB_Threads_Instance	50
c_crawl_Differences	49	start_crawling_for_diffs	49
d_create_Big_CSV	59	c_crawl_Differences.py	82
average_means_of_values		c_crawl_Threads_N_Comments	51
a_everything_Big_CSV_analyzer	6	argument_crawl_type	53
bar_first_n_second_values_percentage		argument_hours_to_shift	53
PlotlyBarChart::PlotlyBarChart	66	argument_year_beginning	54
bar_percentages_values_1		argument_year_end	54
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	70	check_if_coll_in_db_already_exists_up2date	51
bar_percentages_values_2		check_script_arguments	51
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	70	convert_argument_year_to_epoch	52
bar_percentages_values_3		crawl_comments	52
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	crawl_data	52
bar_percentages_values_4		crawl_threads	53
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	initialize_mongo_db_parameters	53
bar_percentages_values_5		mongo_DB_Client_Instance	54
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	reddit_Instance	54
bar_value_description		time_shift_difference	54
PlotlyBarChart::PlotlyBarChart	66	c_crawl_Threads_N_Comments.py	83
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	calculate_ar_mean_answer_time_for_questions	
bar_x_axis_text		a_iAMA_Commenttime	20
PlotlyBarChart::PlotlyBarChart	66	calculate_life_span	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	d_create_Big_CSV	55
bar_x_axis_values		calculate_reaction_time_average	
PlotlyBarChart::PlotlyBarChart	66	d_create_Big_CSV	56
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	calculate_time_difference	
bar_y_axis_fifth_values		a_iAMA_Commenttime	20
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	a_question_Answered_Yes_No_Extrema	25
bar_y_axis_first_values		a_thread_Lifespan_N_Average_Commenttime	43
PlotlyBarChart::PlotlyBarChart	66	d_create_Big_CSV	56
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	chart_title	
bar_y_axis_fourth_values		PlotlyBarChart::PlotlyBarChart	66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
bar_y_axis_second_values		check_if_coll_in_db_already_exists_up2date	
PlotlyBarChart::PlotlyBarChart	66	c_crawl_Threads_N_Comments	51
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	check_if_collection_is_missing_in_comments_database	
bar_y_axis_third_values		c_crawl_Differences	47
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	71	check_if_collection_is_missing_in_threads_database	
c_crawl_Differences	47	c_crawl_Differences	47
argument_inverse_crawling	49	check_if_comment_has_been_answered_by_thread_author	
argument_year_beginning	49	a_question_Answered_Yes_No_Extrema	26
argument_year_ending	49	d_create_Big_CSV	56
check_if_collection_is_missing_in_comments_database	47	check_if_comment_is_a_question	
check_if_collection_is_missing_in_threads_database	47	a_iAMA_Commenttime	20
check_script_arguments	48	a_question_Answered_Yes_No_Extrema	26
crawl_missing_collection_into_comments_database	48	a_question_Answered_Yes_No_Tier_Percentage	32
crawl_missing_collection_into_threads_database	48	a_question_Tier_Distribution	37
initialize_mongo_db_parameters	49	d_create_Big_CSV	57
mongo_DB_Client_Instance	49	check_if_comment_is_answer_from_thread_author	
mongo_DB_Comments_Collection	49	a_iAMA_Commenttime	21
		a_question_Answered_Yes_No_Tier_Percentage	32

check_if_comment_is_not_from_thread_author	
a_iAMA_Commenttime	21
a_question_Answered_Yes_No_Extrema	26
a_question_Answered_Yes_No_Tier_Percentage	32
a_question_Tier_Distribution	38
d_create_Big_CSV	57
check_if_comment_is_on_tier_1	
a_iAMA_Commenttime	21
a_question_Answered_Yes_No_Tier_Percentage	33
a_question_Tier_Distribution	38
d_create_Big_CSV	57
check_script_arguments	
a_iAMA_Commenttime	21
a_question_Answered_Yes_No_Extrema	27
a_question_Answered_Yes_No_Tier_Percentage	33
a_question_Tier_Distribution	38
a_thread_Lifespan_N_Average_Commenttime	43
c_crawl_Differences	48
c_crawl_Threads_N_Comments	51
d_create_Big_CSV	58
color_1	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_1_border	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_2	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_2_border	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_3	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_3_border	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_4	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_4_border	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_5	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	72
color_5_border	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	73
convert_argument_year_to_epoch	
c_crawl_Threads_N_Comments	52
crawl_comments	
c_crawl_Threads_N_Comments	52
crawl_data	
c_crawl_Threads_N_Comments	52
crawl_missing_collection_into_comments_database	
c_crawl_Differences	48
crawl_missing_collection_into_threads_database	
c_crawl_Differences	48
crawl_threads	
c_crawl_Threads_N_Comments	53
create_question_list_containing_all_years	
a_question_Answered_Yes_No_Extrema	27
current_year_question_list	
a_question_Tier_Distribution	41
d_create_Big_CSV	55
add_actual_year_list_to_global_list	55
argument_year_beginning	59
argument_year_ending	59
calculate_life_span	55
calculate_reaction_time_average	56
calculate_time_difference	56
check_if_comment_has_been_answered_by_thread	
author	56
check_if_comment_is_a_question	57
check_if_comment_is_not_from_thread_author	
author	57
check_if_comment_is_on_tier_1	57
check_script_arguments	58
generate_data	58
initialize_mongo_db_parameters	58
list_current_year	59
list_global_year	60
process_specific_thread	58
start_data_generation_for_analysis	59
write_csv_data	59
year_actually_in_progress	60
d_create_Big_CSV.py	84
data_to_give_plotly	
a_iAMA_Commenttime	24
a_question_Answered_Yes_No_Extrema	29
a_question_Answered_Yes_No_Tier_Percentage	35
a_question_Tier_Distribution	41
a_thread_Lifespan_N_Average_Commenttime	45
fill_bar_annotations	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	68
fill_bar_description	
PlotlyBarChart::PlotlyBarChart	64
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	68
fill_bar_percentages_values	
PlotlyBarChart::PlotlyBarChart	64
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	68
fill_chart_title_description	
PlotlyBarChart::PlotlyBarChart	64
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	69
fill_x_axis_list	
PlotlyBarChart::PlotlyBarChart	64
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	69
fill_y_axis_answered_list	
PlotlyBarChart::PlotlyBarChart	65
fill_y_axis_unanswered_list	
PlotlyBarChart::PlotlyBarChart	65
fill_y_axis_values	
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	69
generate_chart	
PlotlyBarChart::PlotlyBarChart	65
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	69
generate_data	
d_create_Big_CSV	58

```

generate_data_now
  a_question_Answered_Yes_No_Extrema 27
generate_data_to_be_analyzed
  a_iAMA_Commenttime 22
  a_question_Answered_Yes_No_Tier_Percentage 33
  a_question_Tier_Distribution 38
  a_thread_Lifespan_N_Average_Commenttime 43
global_question_list
  a_question_Answered_Yes_No_Tier_Percentage 35
global_thread_list
  a_iAMA_Commenttime 24
  a_thread_Lifespan_N_Average_Commenttime 46
global_year_question_list
  a_question_Tier_Distribution 41
initialize_mongo_db_parameters
  a_iAMA_Commenttime 22
  a_question_Answered_Yes_No_Extrema 27
  a_question_Answered_Yes_No_Tier_Percentage 33
  a_question_Tier_Distribution 39
  a_thread_Lifespan_N_Average_Commenttime 43
  c_crawl_Differences 49
  c_crawl_Threads_N_Comments 53
  d_create_Big_CSV 58
list_current_year
  d_create_Big_CSV 59
list_global_year
  d_create_Big_CSV 60
list_To_Be_Plotted
  a_iAMA_Commenttime 24
list_with_currents_year_infos
  a_thread_Lifespan_N_Average_Commenttime 46
main_method
  PlotlyBarChart::PlotlyBarChart 65
  PlotlyBarChart_5_Bars::PlotlyBarChart5Bars 69
mongo_DB_Client_Instance
  a_iAMA_Commenttime 24
  a_question_Answered_Yes_No_Extrema 30
  a_question_Answered_Yes_No_Tier_Percentage 35
  a_question_Tier_Distribution 41
  a_thread_Lifespan_N_Average_Commenttime 46
  c_crawl_Differences 49
  c_crawl_Threads_N_Comments 54
mongo_DB_Comments_Collection
  c_crawl_Differences 49
mongo_DB_Comments_Instance
  a_iAMA_Commenttime 24
  a_question_Answered_Yes_No_Extrema 30
  a_question_Answered_Yes_No_Tier_Percentage 35
  a_question_Tier_Distribution 41
  a_thread_Lifespan_N_Average_Commenttime 46
  c_crawl_Differences 50
mongo_DB_Thread_Collection
  a_iAMA_Commenttime 24
  a_question_Answered_Yes_No_Extrema 30
  a_question_Answered_Yes_No_Tier_Percentage 36
  a_question_Tier_Distribution 41
  a_thread_Lifespan_N_Average_Commenttime 46
  c_crawl_Differences 50
mongo_DB_Threads_Instance
  a_iAMA_Commenttime 24
  a_question_Answered_Yes_No_Extrema 30
  a_question_Answered_Yes_No_Tier_Percentage 36
  a_question_Tier_Distribution 41
  a_thread_Lifespan_N_Average_Commenttime 46
  c_crawl_Differences 50
plot_generated_data
  a_iAMA_Commenttime 22
  a_question_Answered_Yes_No_Extrema 28
  a_question_Answered_Yes_No_Tier_Percentage 34
  a_question_Tier_Distribution 39
  a_thread_Lifespan_N_Average_Commenttime 44
PlotlyBarChart 61
PlotlyBarChart.PlotlyBarChart 63
PlotlyBarChart.py 85
PlotlyBarChart::PlotlyBarChart
  __init__ 63
  bar_first_n_second_values_percentage 66
  bar_value_description 66
  bar_x_axis_text 66
  bar_x_axis_values 66
  bar_y_axis_first_values 66
  bar_y_axis_second_values 66
  chart_title 66
  fill_bar_description 64
  fill_bar_percentages_values 64
  fill_chart_title_description 64
  fill_x_axis_list 64
  fill_y_axis_answered_list 65
  fill_y_axis_unanswered_list 65
  generate_chart 65
  main_method 65
  time_now_date 66
  time_now_time 66
PlotlyBarChart_5_Bars 62
PlotlyBarChart_5_Bars.PlotlyBarChart5Bars 67
PlotlyBarChart_5_Bars.py 86
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars
  __init__ 68
  annotations_1 70
  annotations_2 70
  annotations_3 70
  annotations_4 70
  annotations_5 70
  annotations_all 70
  bar_percentages_values_1 70
  bar_percentages_values_2 70

```

bar_percentages_values_3 71
 bar_percentages_values_4 71
 bar_percentages_values_5 71
 bar_value_description 71
 bar_x_axis_text 71
 bar_x_axis_values 71
 bar_y_axis_fifth_values 71
 bar_y_axis_first_values 71
 bar_y_axis_fourth_values 71
 bar_y_axis_second_values 71
 bar_y_axis_third_values 71
 chart_title 72
 color_1 72
 color_1_border 72
 color_2 72
 color_2_border 72
 color_3 72
 color_3_border 72
 color_4 72
 color_4_border 72
 color_5 72
 color_5_border 73
 fill_bar_annotations 68
 fill_bar_description 68
 fill_bar_percentages_values 68
 fill_chart_title_description 69
 fill_x_axis_list 69
 fill_y_axis_values 69
 generate_chart 69
 main_method 69
 time_now_date 73
 time_now_time 73
 prepare_data_for_comment_time
 a_thread_Lifespan_N_Average_Commenttime 44
 prepare_data_for_graph
 a_iAMA_Commenttime 22
 a_question_Answered_Yes_No_Tier_Percentage 34
 a_question_Tier_Distribution 39
 prepare_data_for_graph_life_span
 a_thread_Lifespan_N_Average_Commenttime 44
 prepare_dict_by_time_separation_for_comment_time
 a_thread_Lifespan_N_Average_Commenttime 44
 process_answered_questions_within_thread
 a_question_Answered_Yes_No_Extrema 28
 process_specific_thread
 d_create_Big_CSV 58
 question_answered_by_iAMA_host
 a__everything_Big_CSV_analyzer 14
 question_answering_distribution_tier1_tierx_tierany
 a_question_Answered_Yes_No_Tier_Percentage 34
 question_distribution_tier1_tierx
 a_question_Tier_Distribution 39
 question_information
 a__everything_Big_CSV_analyzer 14
 question_information_list
 a_question_Answered_Yes_No_Extrema 30
 question_overall_correlation
 a__everything_Big_CSV_analyzer 6
 question_ups
 a__everything_Big_CSV_analyzer 14
 relation_thread_amount_of_commentators_total_and_num_comments_answered_by_iama_host
 a__everything_Big_CSV_analyzer 7
 reddit_Instance
 c_crawl_Threads_N_Comments 54
 relation_question_upvotes_with_amount_of_question_s_answered_by_iama_host
 a__everything_Big_CSV_analyzer 7
 relation_thread_amount_of_questioners_total_and_num_questions_answered_by_iama_host
 a__everything_Big_CSV_analyzer 7
 relation_thread_amount_of_questions_and_amount_questions_answered_by_iama_host
 a__everything_Big_CSV_analyzer 7
 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 8
 relation_thread_downvotes_with_amount_of_comments
 a__everything_Big_CSV_analyzer 8
 relation_thread_downvotes_with_amount_of_questions
 a__everything_Big_CSV_analyzer 8
 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 9
 relation_thread_lifespan_to_last_comment_and_iama_host_response_time_to_questions
 a__everything_Big_CSV_analyzer 9
 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 10
 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_amou
 nt_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 11
 relation_thread_reaction_time_questions_and_amoun
 t_of_comments_the_iama_host_answered_to
 a__everything_Big_CSV_analyzer 12
 relation_thread_reaction_time_questions_and_amoun
 t_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 12
 relation_thread_upvotes_and_iama_host_response_ti
 me_comments
 a__everything_Big_CSV_analyzer 13
 relation_thread_upvotes_and_iama_host_response_ti
 me_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 28
 start_crawling_for_diffs
 c_crawl_Differences 49
 start_data_generation_for_analysis
 a_iAMA_Commenttime 23
 a_question_Answered_Yes_No_Extrema 28
 a_question_Answered_Yes_No_Tier_Percentage
 34
 a_question_Tier_Distribution 40
 a_thread_Lifespan_N_Average_Commenttime 45
 d_create_Big_CSV 59
 temp_time_difference_list
 a_thread_Lifespan_N_Average_Commenttime 46
 thread_amount_of_commentators_tier_1
 a__everything_Big_CSV_analyzer 14
 thread_amount_of_commentators_tier_x
 a__everything_Big_CSV_analyzer 14
 thread_amount_of_commentators_total
 a__everything_Big_CSV_analyzer 14
 thread_amount_of_questioners_tier_1
 a__everything_Big_CSV_analyzer 14
 thread_amount_of_questioners_tier_x
 a__everything_Big_CSV_analyzer 14
 thread_amount_of_questioners_total

a__everything_Big_CSV_analyzer 14
 thread_author
 a__everything_Big_CSV_analyzer 15
 thread_average_comment_vote_score_tier_1
 a__everything_Big_CSV_analyzer 15
 thread_average_comment_vote_score_tier_x
 a__everything_Big_CSV_analyzer 15
 thread_average_comment_vote_score_total
 a__everything_Big_CSV_analyzer 15
 thread_average_question_vote_score_tier_1
 a__everything_Big_CSV_analyzer 15
 thread_average_question_vote_score_tier_x
 a__everything_Big_CSV_analyzer 15
 thread_average_question_vote_score_total
 a__everything_Big_CSV_analyzer 15
 thread_average_reaction_time_between_comments_ti
 er_1
 a__everything_Big_CSV_analyzer 15
 thread_average_reaction_time_between_comments_ti
 er_x
 a__everything_Big_CSV_analyzer 15
 thread_average_reaction_time_between_comments_t
 otal
 a__everything_Big_CSV_analyzer 15
 thread_average_reaction_time_between_questions_tie
 r_1
 a__everything_Big_CSV_analyzer 15
 thread_average_reaction_time_between_questions_tie
 r_x
 a__everything_Big_CSV_analyzer 16
 thread_average_reaction_time_between_questions_to
 tal
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_comment_time_iama_h
 ost_tier_1
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_comment_time_iama_h
 ost_tier_x
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_comment_time_iama_h
 ost_total
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_question_time_iama_ho
 st_tier_1
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_question_time_iama_ho
 st_tier_x
 a__everything_Big_CSV_analyzer 16
 thread_average_response_to_question_time_iama_ho
 st_total
 a__everything_Big_CSV_analyzer 16
 thread_creation_time_stamp
 a__everything_Big_CSV_analyzer 16
 thread_downs
 a__everything_Big_CSV_analyzer 16
 thread_id
 a__everything_Big_CSV_analyzer 16

thread_information	
a__everything_Big_CSV_analyzer	17
thread_life_span_until_last_comment	
a__everything_Big_CSV_analyzer	17
thread_life_span_until_last_question	
a__everything_Big_CSV_analyzer	17
thread_num_comments_answered_by_iama_host_tier_1	
a__everything_Big_CSV_analyzer	17
thread_num_comments_answered_by_iama_host_tier_x	
a__everything_Big_CSV_analyzer	17
thread_num_comments_answered_by_iama_host_total	
a__everything_Big_CSV_analyzer	17
thread_num_comments_tier_1	
a__everything_Big_CSV_analyzer	17
thread_num_comments_tier_x	
a__everything_Big_CSV_analyzer	17
thread_num_comments_total	
a__everything_Big_CSV_analyzer	17
thread_num_comments_total_skewed	
a__everything_Big_CSV_analyzer	17
thread_num_questions_answered_by_iama_host_tier_1	
a__everything_Big_CSV_analyzer	18
thread_num_questions_answered_by_iama_host_tier_x	
a__everything_Big_CSV_analyzer	18
thread_num_questions_answered_by_iama_host_total	
a__everything_Big_CSV_analyzer	18
thread_num_questions_tier_1	
a__everything_Big_CSV_analyzer	18
thread_num_questions_tier_x	
a__everything_Big_CSV_analyzer	18
thread_num_questions_total	
a__everything_Big_CSV_analyzer	18
thread_overall_correlation	
a__everything_Big_CSV_analyzer	13
thread_ups	
a__everything_Big_CSV_analyzer	18
thread_year	
a__everything_Big_CSV_analyzer	18
time_now_date	
PlotlyBarChart::PlotlyBarChart	66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	73
time_now_time	
PlotlyBarChart::PlotlyBarChart	66
PlotlyBarChart_5_Bars::PlotlyBarChart5Bars	73
time_shift_difference	
c_crawl_Threads_N_Comments	54
write_csv	
a_question_Answered_Yes_No_Tier_Percentage	35
a_question_Tier_Distribution	40
a_thread_Lifespan_N_Average_Commenttime	45
write_csv_and_count_unanswered	
a_question_Answered_Yes_No_Extrema	29
write_csv_data	
a_iAMA_Commenttime	23
d_create_Big_CSV	59
year_actually_in_progress	
a_iAMA_Commenttime	24
a_question_Answered_Yes_No_Extrema	30
a_question_Answered_Yes_No_Tier_Percentage	36
a_question_Tier_Distribution	41
a_thread_Lifespan_N_Average_Commenttime	46
d_create_Big_CSV	60
year_question_list	
a_question_Answered_Yes_No_Tier_Percentage	36