Einleitung und Überblick

Der ILIAS Evaluator ist ein digitales Tool zur Auswertung und Nachkorrektur von Tests und Examen aus dem ILIAS- oder ILIAS-EA-System. Die Grundlagen des Tools wurden in https://github.com/P4ckP4ck/ILIAS_KlausurAuswertung /Bewerte ILIAS-Testergebnisse V1_5.py von Prof. Eberhard Waffenschmidt, TH Köln, geschaffen. Das Tool ILIAS Evaluator ist hier verfügbar und wurde von Silvan Rummeny verfasst und aufgebaut.

Für den vorgesehenen Export von Test und Examen aus dem ILIAS- oder ILIAS-EA-System nutzen Sie bitte folgendes Tool von Tobias Panteleit: https://github.com/Tpanteleit/ILIAS---Test-Generator. Für die Verwendung der Tools ist es empfehlenswert Python 3.8 oder Python 3.9 zu verwenden.

Im folgenden wird der typische Ablauf der Bonus- und Klausurauswertung beschrieben, wie er für das Fach Elektrotechnische Grundlagen (ETG) von Prof. Johanna May genutzt wurde. Dabei werden folgende Farbmarkierungen für folgende Dateitypen eingesetzt:

- Python-Skript (.py-Datei, z.B. skript_123.py)
- Excel-Datei zum Import (z.B. input_123.xlsx)
- Excel-Datei zum Export (z.B. output_123.xlsx)

Nach dem Überblick zum Ablauf der Bonus- und Klausurauswertung werden Skripte und den Stand ihrer Bearbeitung/Professionalisierung beschrieben, allen voran das Kern des ILIAS Evaluators: ilias_evaluator.py, aber auch die Anwendungsskripte evaluate_ETG_Praktikum.py, evaluate_ETG_Bonus.py und evaluate_ETG_Exam.py.

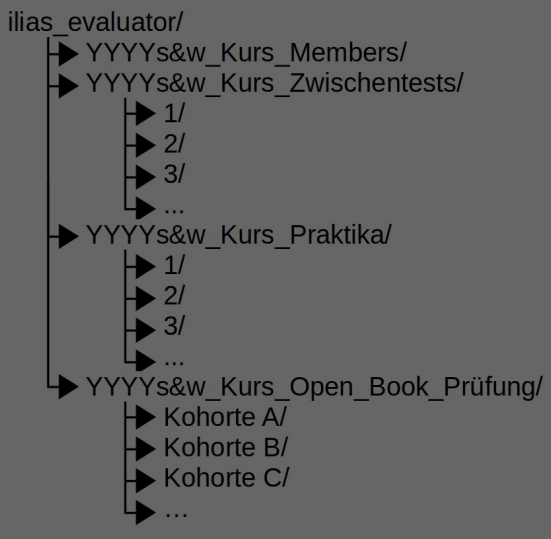
Überblick zum Ablauf der Bonus- und Klausurauswertung

1. ILIAS-Export der
   - 1.1 Testergebnis-Export der Teilnehmer als .xlsx
     - in ILIAS im Testelement unter **Statistik** --> **Evaluationsdaten exportieren als *Microsoft-Excel*** verfügbar. Bitte stellen Sie sicher, dass die Matrikelnummer im Export als Spalte enthalten ist!
   - 1.2 Test-Exportdatei als .zip
     - in ILIAS im Testelement unter **Export** --> **Erstelle Exportdatei** --> **Download** verfügbar. Bitte entzippen Sie diese Datei, sobald der Download erfolgreich war.

2. Einlesen der entzippten Test-Exportdatei via **ILIAS---Test-Generator** by Tobias Panteleit (https://github.com/TPanteleit/ILIAS---Test-Generator) und Export der Fragenpools für Formelfragen (**FF**) und Single-Choice-Aufgaben (**SC**)

3. Ablage der Fragenpools aus (2.) und der Testergebnisse aus (1.1) in einem selbst erstellten Arbeitsverzeichnis, dass je nach Testart varriiert. So z.B.:

   Die Unterverzeichnisse sollen so heißen wie es im jeweiligen Anwendungsskript in der Liste ***pra_experiment*** *(Seite 7)*, ***zt_test*** *(Seite 10)* oder ***considered_tests*** *(Seite 13)* genannt wird (empfohlen: ‘1‘, ‘2‘, ‘3‘...)



YYYYs&w_Kurs soll das Semester und das Kürzel des Kurses beinhalten (z.B. **2022s_ETG**). Für Zwischentests, Praktika und Open-Book-Prüfung werden jeweils alle ILIAS-Exporte wie in (1.) beschrieben heruntergeladen und in das jeweilige Unterverzeichnis gelegt. In **YYYYs&w_Kurs_Members/** wird für die Praktikaauswertung die ILIAS-Mitgliederliste und für die Bonus- und Examensauswertung die PSSO-Liste der Prüfungsteilnehmer gelegt.

4. Auswertung der Praktika
   siehe Skript: evaluate_ETG_Praktikum.py (siehe Details auf Seite 7)
   Hier werden die Dateien aus dem jeweiligen Unterverzeichnis (siehe 3.), sowie die Datei vorangegangener Praktika (z.B. 2021w_ETG_Bonuspunkte_Praktika.xlsx) importiert und verarbeitet. Am Ende des Skripts sollte immer eine Excel-liste mit allen (aktualisierten) Bonuspunkte aus Praktika erstellt werden, z.B.: 2022s_ETG_Bonuspunkte_Praktika.xlsx

5. Auswertung der Zwischentests und Ermittlung der resultierenden Bonuspunkte
   siehe Skript: evaluate_ETG_Bonus.py (siehe Details auf Seite 10)
   Hier werden die Dateien aus dem jeweiligen Unterverzeichnis (siehe 3.), sowie die Datei aus (4.) 2022s_ETG_Bonuspunkte_Praktika.xlsx, importiert und verarbeitet. Am Ende des Skripts sollte immer eine Excel-liste mit allen Bonuspunkte für alle Teilnehmer erstellt werden, die dann auch mit den Studierenden geteilt werden kann, z.B.: 2022s_ETG_Bonuspunkte_pub.xlsx

6. Auswertung der Open-Book-Prüfung
   siehe Skript: evaluate_ETG_exam.py (siehe Details auf Seite 13)
   Hier wird die Datei aus (5.) 2022s_ETG_Bonuspunkte_pub.xlsx, wie auch die Dateien aus dem jeweiligen Unterverzeichnis (siehe 3.), importiert und verarbeitet. Am Ende des Skripts sollte immer eine Excel-liste mit allen Ergebnissen für alle Teilnehmer erstellt werden, die dann auch mit den Studierenden geteilt werden kann, z.B.:
   2022s_ETG_exp_results_review.xlsx
   Weitere Dateien werden ebenso erstellt, die für die Klausureinsicht für die Lehrenden und den PSSO-Upload gedacht sind, z.B.:
   2022s_ETG_exp_results_review_det.xlsx, 2022s_ETG_exp_results_psso.xlsx

7. Bereitstellung der Ergebnisdatei(-en) für Studierende
   Hier wird in der Regel die Datei YYYYs&w_Kurs_exp_results_review.xlsx mit Detailinformationen zu den einzelnen Aufgaben und Punkte der Studierenden mit den Studierenden zur Examenseinsicht als .xlsx und .pdf geteilt. Es sind jedoch auch andere Exportformate (z.B. nur eine Notenliste) verfügbar.

8. PSSO-Upload des generierten Datei „YYYYs&w_Kurs_exp_results_psso.xlsx“

Klassen und Funktionen von ilias_evaluator.py

Die folgende Beschreibung gibt einen Überblick der einzelnen Funktionen von ilias_evaluator.py, dem Hauptteil des Tools. Die folgenden Komponenten sind wie folgt farblich markiert:

eine Skriptdatei: script.py

- **eine Klasse oder Funktion** Status der Bearbeitung und Programmierlevel: **schnell und flexibel** oder **flexibel, aber noch offene ToDos** oder **nicht-professioneller work-around**

  kurze Beschreibung des Objekts
  Input Variabel oder Datei 1, Input Variabel oder Datei 2, ...
  Output Variabel oder Datei 1, Output Variabel oder Datei 2, ...

Code und Kommentierungen sind in englischer Sprache verfasst.

Klassen und Instanz-Funktionen

- **class Test**
  class of a test or exam to get evaluated
  - **def __init__** status**: fast and flexible**
    Parameters
    ----
    members: pd.DataFrame
    DataFrame of all course members incl. Name, Matrikelnr., etc.
    marker: list of str
    list of marker used to identify variables, results, etc. in self.d_ilias
    name: int or str
    Number or name of test (e.g. number of intermediate test or name of exam cohort)
    ilias_export: str
    path of the ILIAS result and data import file
    ff: str
    path of Formelfrage task pool, default is None
    sc: str
    path of SingleChoice task pool, default in None
    daIr: bool
    do you want to document aggregated ILIAS results? - default is False
  - **def process_ilias** status**: flexible, but open ToDos**
    process ILIAS Data for the test and saves it in self.ent for all members
  - **def process_pools** status**: flexible, but exceptions for faulty tasks are still included**
    process task pools, evaluate results and returns self.ent with evaluated results

Statische Funktionen

- **def drop_None** status: **fast and flexible**

  Drop all None Elements in a list (e.g. self.ent[(:,'Var' or 'R' or 'R_ref')][m])

  Parameters
  ---
  l : list
  input list which has to be reduced

  return l_out

- **def eval_ilias** status: **fast and flexible**

  Reformatting and calculation of ILIAS formula with given var and res inputs, returning calculated result
  @author: srummeny, 21.6.21 (edited Version of E. Waffenschmidt, 3.9.2020)

  Parameters
  ---
  formula_ilias: str
  formula string in ILIAS format
  var: list of float
  variable values as used as input in formula
  res: list of float
  result values as used as input in formula
  context: str
  context information of task and/or participant, test, etc.

  return result

- **import_psso_members** status: **fast and flexible**

  import and concatenation of all psso members. Return one complete psso member list, which is used for evaluation of an exam or course

  Parameters
  ---
  psso_import: list of str
  path str list of the files containing psso_members

  return psso_members

- **get_excel_files** status: **fast and flexible**

  algorithm to collect all excel input data for several considered tests

  Parameters
  ---
  considered_tests: list
  list of names of the considered tests (e.g. [1, 2, 4, 7] or ['Test1, Test_xy, Test_final'])
  import_dir: str
  directory containing import data
  identifier: list
  list of identifiers for ILIAS result files ('_results'), Formelfrage or Single Choice task pools

  return result_files, pool_ff_files, pool_sc_files

- **get_originality_proof** status: **flexible, but adjustments likely to be necessary**

  import originality proof including identity check and declaration of originality

  Parameters
  ---
  members: pd.DataFrame
  DataFrame of all members with columns 'Identitaetsnachweis' and 'Eigenstaendigkeitserklaerung', which has to get filled
  id_dir: str
  directory of the excel data of identity check (Identitaetsnachweis)
  doo_dir: str
  directory of the excel data of declaration of originality (Eigenstaendigkeitserklaerung). Default is None.

  return members

- **evaluate_intermediate_tests** status: **fast and flexible**

  evaluation of intermediate test bonus of a course and returns members['Bonus_ZT'] and full DataFrame of bonus_ges

  Parameters
  ---
  members: pd.DataFrame
  DataFrame of all course members incl. Name, Matrikelnr., etc.
  zt_tests: list of class Test
  List of evaluated intermediate tests
  d_course: pd.DataFrame
  empty DataFrame containing, Ges_Pkt and Note from each bonus test
  scheme: pd.Series
  scheme for intermediate test evaluation containing note str as index and corresponding percentage limits as values
  tests_p_bonus: int
  number of bonus tests to get 1 bonus point

  return members, d_course

- **evaluate_praktika** status: **flexible, but open ToDos**

  evaluation of praktikum bonus of a course and returns members['Bonus_Pra'] and full DataFrame of bonus_ges

  Parameters
  ---
  members: pd.DataFrame
  DataFrame of all course members incl. Name, Matrikelnr., etc.
  pra_prev: pd.DataFrame
  DataFrame of bonus achieved from Praktika of previous semesters
  pra_tests: list of class Test
  List of evaluated Praktika tests
  d_course: pd.DataFrame
  empty DataFrame containing, Ges_Pkt and Note from each bonus test
  scheme: pd.Series
  scheme for Praktika test evaluation containing note str as index and corresponding percentage limits as values
  tests_p_bonus: int
  number of bonus tests to get 1 bonus point

  return members, d_course

- **evaluate_bonus** status: **fast and flexible**

  evaluation of total bonus points of a course and returns members['Bonus_Pkt']

  Parameters
  ----
  members: pd.DataFrame
  DataFrame of all course members incl. Name, Matrikelnr., etc.
  max_bonus: int
  maximum achievable bonus points

  return members

- **evaluate_exam** status: **fast and flexible**

  evaluation of the final exam and course note and returns members['Note'] and all_entries (all exam cohort data unified)

  Parameters
  ----
  members: pd.DataFrame
  DataFrame of all course members incl. Name, Matrikelnr., etc.
  exam: list of class Test
  List of evaluated cohorts of the exam
  scheme: pd.Series
  scheme for exam evaluation containing note str as index and corresponding percentage limits as values
  max_pts: int
  maximum achievable exam points, used as reference for note scheme

  return members, all_entries

Kommentare zu evaluate_ETG_Praktika.py

Im folgenden ist der gesamte Code (Stand: 19.07.2022) von evaluate_ETG_Praktika.py aufgeführt. Am rechten Rand sind Kommentare in Deutsch (grün hinterlegt) hinzugefügt, um die Funktion der Variablen und Funktionen zu erläutern.

```
"""
Script of ETG Praktikum evaluation
Author: Silvan Rummeny
"""
import pandas as pd
import numpy as np
import ilias_evaluator as ev

### important entries
                    Hier sind alle für die Praktikumauswertung nötigen Angaben gesammelt. Dazu zählen die
                    Teilnehmerliste (hier: '2022_07_12_09-511657612294_member_export_41962.xlsx') und
                        die Verzeichnisse der Teilnehmer und Praktikumstests . Die Teilnehmerliste und die
                                                    Verzeichnisse sollten immer entsprechend aktualisiert werden.
mem_dir = '2022s_ETG_Members/'
ilias_mem = pd.read_excel(mem_dir+'2022_07_12_09-511657612294_member_export_41962.xlsx',
                          sheet_name='Mitglieder')
pra_dir = '2022s_ETG_Praktikum/'
# Specific constants for Praktikum
# What Notes by what total percentage points?
                                                                    Hier findet sich das Benotungsschema
                                                    (hier: ab 50% der Punkte ist der Praktikumstest bestanden).
pra_scheme = pd.Series(data= [0,    50],
                       index=['NB','BE'])
                                            Im folgenden sind die zu untersuchenden Praktikaversuche aufgelistet, die
                                    entsprechend im Praktikumsverzeichnis als Unterordner vorhanden sein sollten.
pra_experiment = [1, 2, 3]
# read bonus list from old Praktika
                                                Einlesen der Liste von allen gültigen Praktika vergangener Semester
old_praktika = pd.read_excel(pra_dir+'2021w_ETG_Pra_Bonus.xlsx',
                             sheet_name='Sheet1')
print('Praktika import OK')
                                                                Benennungs-Vorsilbe für Exportdateien dieses Skripts
export_prefix = '2022s_ETG_'

###

                                                    Ab hier finden sich Konstanten, die sich nicht ändern und
                                                                    wichtig für die Auswertung sind
# General constants
result_identifier = '_results'
ff_pool_identifier = 'Formelfrage'
sc_pool_identifier = 'SingleChoice'
name_marker = 'Ergebnisse von Testdurchlauf '   # 'Ergebnisse von Testdurchlauf 1 für '
run_marker = 'dummy_text'   # run marker currently not used
tasks = ['Formelfrage', 'Single Choice', 'Lückentextfrage', 'Hotspot/Imagemap', 'Freitext eingeben']
res_marker_ft = "Ergebnis"
var_marker = '$v'
res_marker = '$r'
marker = [run_marker, tasks, var_marker, res_marker, res_marker_ft]
```

```
# Specific constants for members
                                                    Hier wird die ILIAS Teilnehmerliste danach gefiltert,
                                                    sodass sie nur Teilnehmer mit Matrikelnummer enthält
ilias_mem = ilias_mem.loc[ilias_mem['Matrikelnummer'].dropna().index].reset_index(drop=True)

# test data
                                                    Einlesen der Test-Dateien. Für weitere Infos siehe
                                                    def get_excel_files() in ilias_evaluator.py
[pra_ilias_result, pra_pool_ff, pra_pool_sc] = ev.get_excel_files(pra_experiment, pra_dir)
                                                    Definition des DataFrames members basierend auf ilias_mem.
members = ilias_mem
members['Matrikelnummer'] = pd.to_numeric(members['Matrikelnummer'])
                                                    Definition von neuen (noch leeren) Spalten im DataFrame members.
members['Name_'] = np.nan       # members['Name'].str.replace("'","")
members['Benutzername'] = np.nan
members['E-Mail'] = np.nan
members['Bonus_ZT'] = np.nan
members['Bonus_Pra'] = np.nan
members['Bonus_Pkt'] = np.nan
members['Kohorte'] = np.nan
members['Exam_Pkt'] = np.nan
members['Ges_Pkt'] = np.nan
members['ILIAS_Pkt'] = np.nan
members['Note'] = np.nan
                                                    Loop um Spalten ‘Name_‘, ‘Benutzername‘ und ‘E-mail‘ zu füllen.
for i in range(len(members)):
# add a space behind the komma of the name
    vorname = members.loc[i, 'Vorname']
    nachname = members.loc[i, 'Nachname']
    members.loc[i, 'Name_'] = nachname + ', ' + vorname
# get Benutzername and Email from ilias_mem
    mtknr_sel = ilias_mem['Matrikelnummer'].astype(int)==members['Matrikelnummer'][i]
    members.loc[i,'Benutzername'] = ilias_mem['Benutzername'][mtknr_sel].values.item()
    members.loc[i,'E-Mail'] = ilias_mem['E-Mail'][mtknr_sel].values.item()
## remove ' from Names to get ILIAS equivalent names
members['Name_'] = members['Name_'].str.replace("'", "")
print('PSSO member import OK')
                                        Loop um Multiindex c_tests und damit auch DataFrame course_data (noch leer)
                                            zu erzeugen mit den Gesamtpunkten und Note des jeweiligen Praktikumtests
i_lev1 = []
i_lev2 = []
subtitles = ['Ges_Pkt', 'Note']
for n in pra_experiment:
    i_lev1 += ['V'+str(n)]*len(subtitles)
    i_lev2 += subtitles
c_tests = pd.MultiIndex.from_arrays([i_lev1, i_lev2], names = ['test', 'parameter'])
course_data = pd.DataFrame(index=members.index, columns=c_tests)
                                                    Ab hier werden die Praktikumstests mittels ilias_evaluator (ev) ausgewertet
## disable here
############ LOOP of evaluating all considered Praktikum experiments #############
praktikum = []
for pra in range(len(pra_experiment)):
    praktikum.append([])
    for sub in range(len(pra_ilias_result[pra])):
        print('started evaluating Praktikum test', pra_ilias_result[pra][sub][21:])
        praktikum[pra].append(ev.Test(members, marker, pra_experiment[pra],
                              pra_ilias_result[pra][sub]))
        print("process ILIAS data...")
        praktikum[pra][sub].process_ilias()
```

```
print("evaluate pra bonus…")
                    Hier werden die erreichten Bonuspunkte von allen Praktikatests eines Teilnehmers ermittelt.
                    Für mehr Informationen siehe def evaluate_praktika in ilias_evaluator.py. Dort wird z.B. eine
                        aktualisierte Praktikumbonusliste mit entsprechendem semester_name exportiert/erzeugt.
[members, course_data]= ev.evaluate_praktika(members, pra_prev=old_praktika,
                                    pra_tests=praktikum,
                                    d_course=course_data,
                                    tests_p_bonus = 1,
                                    semester_name = export_prefix[0:5])
print("Done")
```

Kommentare zu evaluate_ETG_Bonus.py

Im folgenden ist der gesamte Code (Stand: 19.07.2022) von evaluate_ETG_Bonus.py aufgeführt. Am rechten Rand sind Kommentare in Deutsch (grün hinterlegt) hinzugefügt, um die Funktion der Variablen und Funktionen zu erläutern.

```
"""
Script of ETG Bonus evaluation
Author: Silvan Rummeny
"""
import ilias_evaluator as ev
import pandas as pd
import numpy as np

### Important entries
                                    Hier sind alle für die Bonusauswertung nötigen Angaben gesammelt. Dazu zählen die
                                                PSSO-Teilnehmerliste (hier: '20220712_Kohortenaufteilung_ETG_full_SR.xlsx'),
                                  das Verzeichnis der Zwischentests und die aktuelle Praktikumbonusliste. Die Teilnehmerliste,
                                 das Verzeichnis und die Praktikumsbonusliste sollten immer entsprechend aktualisiert werden.
export_prefix = '2022s_ETG'
# read psso member list
psso_members = pd.read_excel('2022s_ETG_Members/psso-2022-06-
21/20220712_Kohortenaufteilung_ETG_full_SR.xlsx',
                             sheet_name='Sheet1')
print('PSSO member import OK')
zt_dir = '2022s_ETG_Zwischentests/'
# read bonus list from Praktika
pra = pd.read_excel('2022s_ETG_Pra_Bonus.xlsx',
                    sheet_name='Sheet1')
print('Praktika import OK')
                                    Überschrift in der Export-Datei. Hier muss das Semester immer angepasst werden.
first_print_line = 'Bonuspunkte Elektrotechnische Grundlagen (ETG), SoSe 22'      # WiSe 21/22
# Specific constants for intermediate tests (Zwischentests=zt)
# What Notes by what total percentage points?
                                                                    Hier findet sich das Benotungsschema
                                                                  (hier: ab 70% der Punkte ist der Zwischentest bestanden).
zt_scheme = pd.Series(data= [0,    70],
                      index=['NB','BE'])
                        Im folgenden sind die zu berücksichtigenden Zwischentests und Praktikaversuche aufgelistet,
                                    die entsprechend im Zwischentestverzeichnis als Unterordner vorhanden sein sollten.
zt_test = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

# Specific constants for Praktikum
# What Notes by what total percentage points?
pra_experiment = [1, 2, 3]

###
                                                                  Ab hier finden sich Konstanten, die sich nicht ändern und
                                                                                                  wichtig für die Auswertung sind
# General constants
result_identifier = '_results'
ff_pool_identifier = 'Formelfrage'
sc_pool_identifier = 'SingleChoice'
Filename_Export_public = export_prefix+'_Bonuspunkte_pub.xlsx'
name_marker = 'Ergebnisse von Testdurchlauf '   # 'Ergebnisse von Testdurchlauf 1 für '
run_marker = 'dummy_text'   # run marker currently not used
tasks = ['Formelfrage', 'Single Choice', 'Lückentextfrage', 'Hotspot/Imagemap', 'Freitext eingeben']
res_marker_ft = "Ergebnis"
var_marker = '$v'
res_marker = '$r'
```

```
marker = [run_marker, tasks, var_marker, res_marker, res_marker_ft]

# Specific constants for members
                                        Definition des DataFrames members basierend auf psso_members.
members = psso_members
members['Matrikelnummer'] = pd.to_numeric(members['Matrikelnummer'])
                                        Definition von neuen (noch leeren) Spalten im DataFrame members.
members['Benutzername'] = np.nan
members['E-Mail'] = np.nan
members['Bonus_ZT'] = np.nan
members['Bonus_Pra'] = np.nan
members['Bonus_Pkt'] = np.nan
members['Kohorte'] = np.nan
members['Exam_Pkt'] = np.nan
members['Ges_Pkt'] = np.nan
members['ILIAS_Pkt'] = np.nan
members['Note'] = np.nan

# test data
                                                    Einlesen der Test-Dateien. Für weitere Infos siehe
                                                    def get_excel_files() in ilias_evaluator.py
[zt_ilias_result, zt_pool_ff, zt_pool_sc] = ev.get_excel_files(zt_test, zt_dir)
                                Loop um Multiindex c_tests und damit auch DataFrame course_data (noch leer)
                            zu erzeugen mit den Gesamtpunkten und Note des jeweiligen Praktikum- und Zwischentests
i_lev1 = []
i_lev2 = []
subtitles = ['Ges_Pkt', 'Note']
for n in zt_test:
    i_lev1 += ['ZT'+str(n)]*len(subtitles)
    i_lev2 += subtitles
for n in pra_experiment:
    i_lev1 += ['V'+str(n)]*len(subtitles)
    i_lev2 += subtitles
c_tests = pd.MultiIndex.from_arrays([i_lev1, i_lev2], names = ['test', 'parameter'])
course_data = pd.DataFrame(index=members.index, columns=c_tests)
                                        Ab hier werden die Zwischentests mittels ilias_evaluator (ev) ausgewertet
## disable here
########## LOOP of evaluating all considered intermediate tests (zt) ##############
zt = []
                                            Erstellung eines ErrorLogs und DiffLogs um Fehler in der Auswertung und
                                        Unterschiede zwischen ilias_evaluator und ILIAS System zu dokumentieren
# Log-Dataframe for occurance of Errors
errorlog = pd.DataFrame(columns=['Test', 'Matrikelnummer', 'Task', 'formula', 'var', 'input_res', 'tol', 'Error',
'Points'])
# Log-Dataframe for occurance of Differences between ILIAS result points and ilias_evaluator points
difflog = pd.DataFrame(columns=['Test', 'Matrikelnummer', 'Task', 'formula', 'var', 'input_res', 'tol',
'Points_ILIAS', 'Points', 'diff'])
for t in range(len(zt_test)):
    zt.append([])
    for sub in range(len(zt_ilias_result[t])):
        print('started evaluating intermediate test', zt_test[t])
        zt[t].append(ev.Test(members, marker, zt_test[t],
                             zt_ilias_result[t][sub], ff=zt_pool_ff[t][sub], sc=zt_pool_sc[t][sub]))
        print("process ILIAS data...")
        zt[t][sub].process_ilias()
        print("process task pools and evaluate...")
        zt[t][sub].process_pools()
        errorlog = errorlog.append(zt[t][sub].errorlog)
        difflog = errorlog.append(zt[t][sub].difflog)
```

```
print("evaluate zt bonus…")
                    Hier werden die erreichten Bonuspunkte von allen Zwischentests eines Teilnehmers ermittelt.
                                Für mehr Informationen siehe def evaluate_intermediate_tests in ilias_evaluator.py.
[members, course_data]= ev.evaluate_intermediate_tests(members,
                                        zt_tests=zt,
                                        d_course=course_data,
                                        scheme=zt_scheme,
                                        tests_p_bonus=3)
[members, course_data]= ev.evaluate_praktika(members,
                                    pra_prev = pra,
                                    d_course=course_data,
                                    tests_p_bonus=1,
                                    semester_name=export_prefix[0:5])

###### EVALUATE TOTAL BONUS ############
                            Hier werden die erreichten Bonuspunkte von allen Zwischentests und Praktika eines
                    Teilnehmers ermittelt. Für mehr Informationen siehe def evaluate_bonus in ilias_evaluator.py.
members = ev.evaluate_bonus(members)

#%%
######## Export for lecturer (detailed, not anonymous) ############
# TODO: Export hier noch nötig?

######## Export for participants (short, anonymous) #############
                                                Excel-Export der Bonusliste aller Prüfungsteilnehmer im Semester zur
                                                                    Veröffentlichung für die Studierenden.
Bonus_exp_pub = members[['Matrikelnummer','Bonus_ZT', 'Bonus_Pra', 'Bonus_Pkt']]
Bonus_exp_pub = Bonus_exp_pub.rename(columns={'Bonus_ZT':'Boni durch Zwischentests',
                                'Bonus_Pra':'Boni durch Praktika',
                                'Bonus_Pkt':'Summe'})
Bonus_exp_pub = Bonus_exp_pub.sort_values(by=['Matrikelnummer'])
writer = pd.ExcelWriter(Filename_Export_public)
Bonus_exp_pub.to_excel(writer ,index=False, na_rep='N/A', startrow=5)
workbook = writer.book
worksheet = writer.sheets['Sheet1']
title_format = workbook.add_format({'bold': True, 'font_size':16})
align = workbook.add_format({'align':'left'})
worksheet.write_string(0,0, first_print_line, title_format)
worksheet.write_string(1,0,'1 bestandenes Praktikum = 1 Bonuspunkt')
worksheet.write_string(2,0,'3 bestandene Zwischentests = 1 Bonuspunkt')
worksheet.write_string(3,0,'Die Summe der Bonuspunkte kann nur max. 5 Punkte betragen')
worksheet.write_string(4,0,'N/A = nicht teilgenommen')
worksheet.set_column(0,0,15, align)
worksheet.set_column(1,1,25, align)
worksheet.set_column(2,2,20, align)
worksheet.set_column(3,3,7, align)
writer.save()
print("Excel Export OK")
print ('### done! ###')
```

Kommentare zu evaluate_ETG_Exam.py

Im folgenden ist der gesamte Code (Stand: 19.07.2022) von evaluate_ETG_Exam.py aufgeführt. Am rechten Rand sind Kommentare in Deutsch (grün hinterlegt) hinzugefügt, um die Funktion der Variablen und Funktionen zu erläutern.

```
"""
Script of ETG Exam evaluation
Author: Silvan Rummeny
"""
import ilias_evaluator as ev
import pandas as pd
import numpy as np
import glob
import matplotlib.pyplot as plt

### Important entries
                                Hier sind alle für die Examensauswertung nötigen Angaben gesammelt. Dazu zählen die
                                    PSSO-Teilnehmerliste (hier: '20220712_Kohortenaufteilung_ETG_full_SR.xlsx'),
                                das Verzeichnis des Examens und die aktuelle Bonuspunkteliste. Die Teilnehmerliste,
                                das Verzeichnis und die Bonuspunkteliste sollten immer entsprechend aktualisiert werden.
import_dir = '2022s_ETG_Open_Book_Probeprüfung/'
# read psso member list
psso_members = pd.read_excel('2022s_ETG_Members/psso-2022-06-
21/20220712_Kohortenaufteilung_ETG_full_SR.xlsx',
                             sheet_name='Sheet1')
print('PSSO member import OK')
import_bonus = pd.read_excel('2022s_ETG_Bonuspunkte_pub.xlsx', header=5, sheet_name='Sheet1')
print('Bonus import of members OK')
                                                        Verzeichnis mit der Anwesenheitsliste bei dem Examen
identity_control_data =
'2022s_ETG_Members/22s_Probepruefung_Anwesenheit_Raumaufteilung_ed_SR.xls'
                                                    Hier findet sich das Benotungsschema
                                            (hier: ab 50% der Punkte ist das Examen bestanden mit entsprechender Note).
# What Notes by what total percentage points (referenced without Bonus)?
scheme = pd.Series(data= [0,   50,   55,   60,   65,   70,   75,   80,   85,   90,   95],
                   index=["5,0","4,0","3,7","3,3","3,0","2,7","2,3","2,0","1,7","1,3","1,0"])
                                        Im folgenden sind die zu berücksichtigenden Examenskohorten aufgelistet,
                                    die entsprechend im Verzeichnis des Examens als Unterordner vorhanden sein sollten.
considered_tests = ['Kohorte A']    #, 'Kohorte B', 'Kohorte C', 'Kohorte D', 'Kohorte E', 'Kohorte F']
                                                        Benennungsvorsilbe der Exportdateien
export_prefix = '2022s_ETG_Open_Book_Probeprüfung_'
                                                        Überschrift der Ergebnisübersicht für die Studierenden
title1='Ergebnisse der Open-Book-Probeprüfung vom 28.06.2022, Elektrotechnische Grundlagen (ETG),
SoSe 22'

###
                                                    Ab hier finden sich Konstanten, die sich nicht ändern und
                                                    wichtig für die Auswertung sind
# General constants
result_identifier = '_results'
ff_pool_identifier = 'Formelfrage'
sc_pool_identifier = 'SingleChoice'
result_import = []
import_pool_FF = []
import_pool_SC = []
Filename_Export = export_prefix+'exp_Ergebnisse.xlsx'
Filename_Export_public = export_prefix+'exp_Ergebnisse_pub.xlsx'
Filename_Export_PSSO = export_prefix+'exp_Ergebnissse_psso.xlsx'
Filename_Export_review_detailed = export_prefix+'exp_results_review_det.xlsx'
```

```
Filename_Export_review_public = export_prefix+'exp_results_review.xlsx'
name_marker = 'Ergebnisse von Testdurchlauf '   # 'Ergebnisse von Testdurchlauf 1 für '
run_marker = 'dummy_text'   # run marker currently not used
tasks = ['Formelfrage', 'Single Choice', 'Lückentextfrage', 'Hotspot/Imagemap', 'Freitext eingeben']
res_marker_ft = "Ergebnis"
var_marker = '$v'
res_marker = '$r'
marker = [run_marker, tasks, var_marker, res_marker, res_marker_ft]

# Specific constants for members
                                        Definition des DataFrames members basierend auf psso_members.
members = psso_members
members['Matrikelnummer'] = pd.to_numeric(members['Matrikelnummer'])
                                        Definition von neuen (noch leeren) Spalten im DataFrame members.
members['Benutzername'] = np.nan
members['bearbeitete Fragen'] = np.nan
members['Bearbeitungsdauer'] = np.nan
members['Startzeit'] = np.nan
members['Bonus_Pkt'] = np.nan
members['Kohorte'] = np.nan
members['Exam_Pkt'] = np.nan
members['Ges_Pkt'] = np.nan
members['ILIAS_Pkt'] = np.nan
members['% über Bestehensgrenze'] = None
members['Identitaetsnachweis'] = np.nan
members['Eigenstaendigkeitserklaerung'] = np.nan
members['Note'] = np.nan

bonus_mrg = pd.merge(members['Matrikelnummer'], import_bonus, how='left', on='Matrikelnummer')
members['Bonus_Pkt'] = bonus_mrg['Summe']
                                                    Einlesen der Test-Dateien.
# find all import data and pools in directory
for j in considered_tests:
    if len(glob.glob(import_dir+str(j)+'/*.xlsx')) > 3:
        print('### WARNING: There may be to much import data files in',
              import_dir+str(j), '###')
    for i in range(len(glob.glob(import_dir+str(j)+'/*.xlsx'))):
        if result_file_identifier in glob.glob(import_dir+str(j)+'/*.xlsx')[i]:
            result_import.append(glob.glob(import_dir+str(j)+'/*.xlsx')[i])
        if pool_FF_file_identifier in glob.glob(import_dir+str(j)+'/*.xlsx')[i]:
            import_pool_FF.append(glob.glob(import_dir+str(j)+'/*.xlsx')[i])
        if pool_SC_file_identifier in glob.glob(import_dir+str(j)+'/*.xlsx')[i]:
            import_pool_SC.append(glob.glob(import_dir+str(j)+'/*.xlsx')[i])
                                    Ab hier werden die Examkohorten mittels ilias_evaluator (ev) ausgewertet
#### disable here
########## LOOP of evaluating several cohorts (c) of the exam #############
exam = []
                                    Erstellung eines ErrorLogs und DiffLogs um Fehler in der Auswertung und
                                Unterschiede zwischen ilias_evaluator und ILIAS System zu dokumentieren
# Log-Dataframe for occurance of Errors
errorlog = pd.DataFrame(columns=['Kohorte', 'Matrikelnummer', 'Task', 'formula', 'var', 'input_res', 'tol', 'Error',
'Points'])
# Log-Dataframe for occurance of Differences between ILIAS result points and ilias_evaluator points
difflog = pd.DataFrame(columns=['Kohorte', 'Matrikelnummer', 'Points_ILIAS', 'Points', 'diff'])
for c in range(len(considered_tests)):
    print('started evaluating exam,', considered_tests[c])
    exam.append(ev.Test(members, marker, considered_tests[c],
                        result_import[c], import_pool_FF[c],
                        import_pool_SC[c], daIr=True))
    print("process ILIAS data...")
```

```
    exam[c].process_ilias()
    print("process task pools and evaluate...")
    exam[c].process_pools()
    errorlog = errorlog.append(exam[c].errorlog)

print("evaluate exam…")
                                    Hier wird die Eigenständigkeitserklärung und Identitätskontrolle mit berücksichtigt.
                                          Für mehr Informationen siehe def get_originality_proof in ilias_evaluator.py
members = ev.get_originality_proof(members, id_dir = identity_control_data )
                                                          Hier wird die erreichte Note von allen Teilnehmern ermittelt.
                                                  Für mehr Informationen siehe def evaluate_exam in ilias_evaluator.py.
[members, all_entries] = ev.evaluate_exam(members, exam, scheme, max_pts=41)
                                          Dokumentiere im Difflog, wenn Teilnehmer unterschiedliche Punkteanzahlen
                                                                    in ilias_evaluator und im ILIAS System haben
sel = members['Exam_Pkt'].dropna()!=members['ILIAS_Pkt'].dropna()
df = members.loc[sel[sel].index]
log = pd.DataFrame({'Kohorte':df['Kohorte'],
                    'Matrikelnummer':df['Matrikelnummer'],
                    'Points_ILIAS':df['ILIAS_Pkt'],
                    'Points':df['Exam_Pkt']})
log['diff'] = log['Points'] - log['Points_ILIAS']
difflog = difflog.append(log)
#%%
                                                          Ab hier werden die Ergebnisse in Excel-Dateien exportiert
print ('export results as excel...')
members[['Identitaetsnachweis','Eigenstaendigkeitserklaerung']] =
members[['Identitaetsnachweis','Eigenstaendigkeitserklaerung']].replace([True, False],['Ja','Nein'])
# sort members by Matrikelnummer
members = members.sort_values(by=['Matrikelnummer'])
# consider only members with Note
members = members.loc[members['Note'].dropna().index]

########## Export for PSSO ##############
                                                                                        PSSO-Export
exp_psso = members[members.columns[0:13]].rename(columns={'Matrikelnummer':'mtknr',
                                            'Name':'sortname',
                                            'Nachname':'nachname',
                                            'Vorname':'vorname'})
exp_psso.to_excel(Filename_Export_PSSO, index=False)

######### Export for lecturer (detailed, not anonymous) #############
                                    Export der Ergebnisübersicht für Prüfer inkl. Einzelaufgaben, -musterlösungen
                                                  und Punkten, sowie Formel, Variablen, Toleranzen, Teilnehmernamen.
                                    Diese Datei wird üblicherweise für die Klausureinsicht vom Prüfer verwendet.
cols_detailed = ['Matrikelnummer', 'Name', 'Benutzername', 'stg',
                 'pversuch', 'Kohorte', 'ILIAS_Pkt', 'Exam_Pkt', 'Bonus_Pkt', 'Ges_Pkt', 'bewertung']
exp_detailed = members[cols_detailed].rename(columns={'stg':'Studiengang',
                                            'pversuch':'Prüfungsversuch',
                                            'Kohorte':'Exam_Kohorte',
                                            'ILIAS_Pkt':'Exam_ILIAS_Pkt',
                                            'bewertung':'Bewertung'})
exp_detailed.to_excel(Filename_Export, index=False)

######### Export for participants (short, anonymous) #############
cols_public = ['Matrikelnummer', 'Note']
exp_public = members[cols_public]
exp_public.to_excel(Filename_Export_public, index=False)
```

```
######### Export for Exam review (detailed and short) ##############
                Export der Ergebnisübersicht für Studierende inkl. Einzelaufgaben, -musterlösungen und Punkten.
                                        Das ist die übliche Datei, die mit den Studierenden zur Einsicht geteilt wird.
exam_export = all_entries.copy()
exam_export.columns = exam_export.columns.map('_'.join)
exam_export = exam_export[~exam_export['A1_Title'].isnull()]
idx = exam_export.index
                                                                Formartierungs- und Sortierungseinstellungen der Tabelle
review = pd.concat([members.loc[idx,['Matrikelnummer','Name']], exam_export, members.loc[idx,
['Bonus_Pkt','Ges_Pkt','% über Bestehensgrenze','Identitaetsnachweis','Note']]],axis=1)
for i in range(42):
    review = review.drop(columns=['A'+str(i+1)+'_ID',
                                  'A'+str(i+1)+'_Type',
                                  'A'+str(i+1)+'_Pkt_ILIAS'])
    review = review.rename(columns={'A'+str(i+1)+'_R':'A'+str(i+1)+'_Student',
                                    'A'+str(i+1)+'_R_ref':'A'+str(i+1)+'_Musterloesung',
                                    'A'+str(i+1)+'_Pkt':'A'+str(i+1)+'_Punkte'})
review = review.rename(columns={'Bonus_Pkt':'Bonuspunkte', 'Ges_Pkt':'Gesamtpunkte'})
exp_review_detailed = review.copy()
# exp_review_detailed = exp_review_detailed.sort_values(by=['Matrikelnummer'])
exp_review_detailed = exp_review_detailed.transpose()
exp_review_detailed.to_excel(Filename_Export_review_detailed, header=False, index=True, na_rep='N/A')

exp_review_public = review.copy()
for i in range(42):
    exp_review_public = exp_review_public.drop(columns=['A'+str(i+1)+'_Title',
                                                        'A'+str(i+1)+'_Formula',
                                                        'A'+str(i+1)+'_Var',
                                                        'A'+str(i+1)+'_Tol'])
exp_review_public = exp_review_public.drop(columns=['Name'])
exp_review_public = exp_review_public.sort_values(by=['Matrikelnummer'])
exp_review_public = exp_review_public.transpose()
exp_review_public = exp_review_public.reset_index()
                                                                Erstellung und Formatierung eines ExcelWriter-Objekts,
                                                                        um formatierte Exceltabellen zu erstellen
writer = pd.ExcelWriter(Filename_Export_review_public)
exp_review_public.to_excel(writer , index=False, header=False, na_rep='N/A', startrow=5)
# add all different formats
title = writer.book.add_format({'bold': True, 'font_size':16, 'font_color':'#ff0000', 'fg_color':'#ffff00'})
subtitle = writer.book.add_format({'italic': True, 'font_color': '#00b050', 'fg_color':'#ffff00'})
remark = writer.book.add_format({'bold': True, 'fg_color':'#ffff00'})
matnr = writer.book.add_format({'bold': True, 'fg_color':'#b2b2b2', 'border': 1, 'align':'left'})
idx = writer.book.add_format({'bold':True})
ax_stud = writer.book.add_format({'fg_color':'#b7b3ca', 'border': 1, 'align':'left'})
ax_muster = writer.book.add_format({'fg_color':'#b2b2b2', 'border': 1, 'align':'left'})
ax_pkt = writer.book.add_format({'fg_color':'#ffdbb6', 'border': 1, 'align':'left'})
ax_stud_i = writer.book.add_format({'bold': True, 'fg_color':'#b7b3ca', 'border': 1, 'align':'left'})
ax_muster_i = writer.book.add_format({'bold': True, 'fg_color':'#b2b2b2', 'border': 1, 'align':'left'})
ax_pkt_i = writer.book.add_format({'bold': True, 'fg_color':'#ffdbb6', 'border': 1, 'align':'left'})
footer = writer.book.add_format({'fg_color':'#ffff00', 'border': 1, 'align':'left'})
footer_i = writer.book.add_format({'bold': True, 'fg_color':'#ffff00', 'border': 1, 'align':'left'})
note = writer.book.add_format({'bold': True, 'fg_color':'#81d41a', 'border': 1, 'align':'left'})
writer.sheets['Sheet1'].write_string(0,0,title1)
writer.sheets['Sheet1'].set_row(0,cell_format=title)
writer.sheets['Sheet1'].write_string(1,0,'A#_Student ist ihre getaetigte Antwort', subtitle)
writer.sheets['Sheet1'].set_row(1,cell_format=subtitle)
writer.sheets['Sheet1'].write_string(2,0,'A#_Musterloesung ist die Richtige Antwort', subtitle)
writer.sheets['Sheet1'].set_row(2,cell_format=subtitle)
writer.sheets['Sheet1'].write_string(3,0,'A#_Punkte sind die resultierenden Punkte aus der jeweilgen
Aufgabe', subtitle)
```

```
writer.sheets['Sheet1'].set_row(3,cell_format=subtitle)
writer.sheets['Sheet1'].write_string(4,0,'Die Bestehensgrenze betraegt für alle Teilnehmer 21 Punkte',
remark)
writer.sheets['Sheet1'].set_row(4,cell_format=remark)

# set columns width in pxl

writer.sheets['Sheet1'].set_column(1, len(exp_review_public.columns),11)
writer.sheets['Sheet1'].set_row(5,cell_format=matnr)
for i in range(42):
#    writer.sheets['Sheet1'].write_blank(0, 6+i*3, cell_format=ax_stud.set_bold())
    writer.sheets['Sheet1'].write_string(6+i*3,0, exp_review_public['index'][1+i*3], cell_format=ax_stud_i)
    writer.sheets['Sheet1'].write_string(7+i*3,0, exp_review_public['index'][2+i*3], cell_format=ax_muster_i)
    writer.sheets['Sheet1'].write_string(8+i*3,0, exp_review_public['index'][3+i*3], cell_format=ax_pkt_i)
    writer.sheets['Sheet1'].set_row(6+i*3,cell_format=ax_stud)
    writer.sheets['Sheet1'].set_row(7+i*3,cell_format=ax_muster)
    writer.sheets['Sheet1'].set_row(8+i*3,cell_format=ax_pkt)
writer.sheets['Sheet1'].set_column(0,0,27, cell_format=idx)
writer.sheets['Sheet1'].write_string(9+41*3,0, exp_review_public['index'][4+41*3], cell_format=footer_i)
writer.sheets['Sheet1'].write_string(10+41*3,0, exp_review_public['index'][5+41*3], cell_format=footer_i)
writer.sheets['Sheet1'].write_string(11+41*3,0, exp_review_public['index'][6+41*3], cell_format=footer_i)
writer.sheets['Sheet1'].write_string(12+41*3,0, exp_review_public['index'][7+41*3], cell_format=footer_i)
#writer.sheets['Sheet1'].write_string(13+41*3,0, exp_review_public['index'][8+41*3], cell_format=footer_i)
writer.sheets['Sheet1'].set_row(9+41*3,cell_format=footer)
writer.sheets['Sheet1'].set_row(10+41*3,cell_format=footer)
writer.sheets['Sheet1'].set_row(11+41*3,cell_format=footer)
writer.sheets['Sheet1'].set_row(12+41*3,cell_format=footer)
# writer.sheets['Sheet1'].set_row(13+41*3,cell_format=footer)
writer.sheets['Sheet1'].set_row(13+41*3,cell_format=note)
writer.sheets['Sheet1'].repeat_rows(5)
writer.save()

# TODO: export errorlog and difflog
                                Plot vom Vergleich der Punkte durch ilias_evaluator vs Punkte durch ILIAS System
### Plot comparison of Exam_Pkt and ILIAS_Pkt #####
a = members['Exam_Pkt'].value_counts().sort_index()
b = members['ILIAS_Pkt'].value_counts().sort_index()
ab = pd.merge(a, b, how='outer', on=a.index)
ab[['ILIAS_Pkt','Exam_Pkt']].plot.bar()
                                                    Plot der Verteilung der Gesamtpunkte (inklusive Bonuspunkte)
### Plot Ges_Pkt distribution
df=pd.DataFrame(index=np.linspace(0,41,num=42))
df['Ges_Pkt'] = members['Ges_Pkt'].value_counts().sort_index()
maxv = members['Ges_Pkt'].value_counts().max()
fig = plt.figure()
ax = fig.add_axes()
df.plot.bar(ax=ax)
plt.fill([-0.5, -0.5,       scheme['4,0']/100*41, scheme['4,0']/100*41],
         [0,    maxv+0.5,   maxv+0.5,            0],
         'r', alpha=0.25)
                                                  Überprüfung der Anzahl der Taxonomien eines jeden Teilnehmers
#### Check number of Taxonomies in Exam #####
tax = []
for i in range(42):
    tax.append('%02d' % i)
tax[0] = 'Bitte ignorieren!'
tax[-1] = 'SC'
taxonomies = pd.DataFrame(index=members.index, columns=tax)
# create empty Series for occurance of taxonomies (occurrance from 0 to 10 times per participant)
n_tax = pd.Series(index=range(11), dtype=object).fillna(0)
```

```
n_taxs = pd.DataFrame(index=range(11))
run_max = pd.Series(index=members.index, dtype=object)
for m in members['Note'].dropna().index.values:
    run = 0
    runs = []
    for i in range(len(tax)):
        if tax[i]=='SC':
            sel = all_entries.loc[m,pd.IndexSlice[:,'Title']].str.startswith(tuple(tax[:-1]))
            taxonomies.loc[m, tax[i]] = sum(~sel)
        else:
            taxonomies.loc[m, tax[i]] = sum(all_entries.loc[m,pd.IndexSlice[:,'Title']].str.startswith(tax[i]))
        if taxonomies.loc[m, tax[i]]==0:
            run += 1
        else:
            if run==0:
                continue
            else:
                runs.append(run)
                run = 0
    n_taxs[m] = n_tax.add(taxonomies.loc[m].value_counts().sort_index(), fill_value=0)
    if len(runs)==0:
        run_max[m] = runs
    else:
        run_max[m] = max(runs)
print('##### TAXONOMY ANALYSIS #####')
print(len(n_taxs.loc[1][n_taxs.loc[1]==len(tax)]), 'Participants (',
      len(n_taxs.loc[1][n_taxs.loc[1]==len(tax)])/len(n_taxs.columns)*100,'% ) had every Taxonomy once')
print(len(n_taxs.loc[0][n_taxs.loc[0]>0]), 'Participants (',
      len(n_taxs.loc[0][n_taxs.loc[0]>0])/len(n_taxs.columns)*100, '% ) had 1 or more Taxonomy missing!')
print(len(n_taxs.loc[2:,:].sum()[n_taxs.loc[2:,:].sum()>0]),'Participants (',
      len(n_taxs.loc[2:,:].sum()[n_taxs.loc[2:,:].sum()>0])/len(n_taxs.columns)*100,'% ) had 1 or more
Taxonomies more than once!')

print ('### done! ###')
```