Kandidat ID
Tilkoblet

TDT4110 - Konteeksamen ITGK - Python - august 2018

Oppgave 1: Flervalgsoppgave (25%)

Oppgave	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Løsning	b	b	a	b	d	b	b	c	c	a	b	a	c	c	a	b	c	d	d	a

Kodeforståelse (20%)

De følgende oppgavene består av fire kodesnutter. For hver av disse skal du:

- Skrive akkurat hva koden returnerer (oppgave 2a, 2c, 2e, 2g).
- Beskrive med en setning hva koden gjør (oppgave 2b, 2d, 2f, 2h).
- Oppgave 2b beskriver oppgave 2a, 2d beskriver 2c, og så videre.
- Forsøk gjerne å beskrive hva koden gjør selv om du er usikker på det faktiske resultatet. Det kan hende at du er inne på hva den gjør!

Oppgave 2a (3%)

Hva blir skrevet ut til skjerm når du kjører programmet vist under? (3%)

```
def myst1(x,y,z):
     t = x
     x = y
     y = t
     for i in range(z):
          z = x + y
          y*= 2
     return z

>>> print(myst1(2,4,6))
```

Svar: 68

Oppgave 2b (2%)

Beskriv med en setning hva koden i oppgave 2a gjør: (2%)

Svar: Returnerer 2 opphøyd i det "tredje elementet minus 1" pluss det andre elementet, før svaret returneres.

*Det som returneres er z = (innparameter y) + (innparameter x)*2^(z-1). (z = 4 + 2*2^5 = 68)*

Oppgave 2c (3%)

Hva blir skrevet ut til skjerm når du kjører programmet vist under? (3%)

```
def myst2(x,y):
     s = ""
     for i in range(y-1,len(x),y):
          s+= x[i]
     return s

s = "dsigbtmsgmbknrenvrpqkxcunvlbttmx"
>>> print(myst2(s,3))
```

Svar: itgkerkult

Litt trekk hvis svaret er skrevet med 'fnutter' eller liknende, da fnuttene faktisk ikke skrives ut.

Oppgave 2d (2%)

Beskriv med en setning hva koden i oppgave 2c gjør: (2%)

Svar: Returnerer en streng sammensatt av hver y-ende bokstav i teksten s.

Oppgave 2e (3%)

Hva blir skrevet ut til skjerm når du kjører programmet vist under? (3%)

```
def myst3(a):
     s = ''
     for r in a:
          for c in r:
               s+=chr(ord('A')+c)
     return s

z = ((2,0,11,8,5),(14,17,13,8,0))
>>> print(myst3(z))
```

Svar:

CALIFORNIA

Litt trekk hvis svaret er skrevet med 'fnutter' eller liknende, da fnuttene faktisk ikke skrives ut. Det er også skrevet med store bokstaver, så litt trekk hvis besvarelsen bruker små.

Oppgave 2f (2%)

Beskriv med en setning hva koden i oppgave 2e gjør: (2%)

Svar: Henter ut ett og ett tall fra et tuppel av tupler, og lager en sammenhengende tekst av store bokstaver hvor c angir hvor langt unna hver bokstav er fra A i alfabetet (ASCII-tabellen).

Oppgave 2g (3%)

Hva blir skrevet ut til skjerm når du kjører programmet vist under? (3%)

```
def myst4(a,b,c):
     s = []
     for i in range(len(a)):
          s.append(a[i]+b[i]+c[i])
     s = tuple(s)
     return s

x = (1,2,3,4,5,6)
y = (2,4,6,8,10,12)
z = (3,6,9,12,15,18)
>>> print(myst4(x,y,z))
```

Svar: (6, 12, 18, 24, 30, 36)

Oppgave 2h (2%)

Beskriv med en setning hva koden i oppgave 2g gjør: (2%)

Svar: Returnerer et nytt tuppel, like langt som a, hvor hvert tall er summen av alle tallene på samme plass i input-tuplene.

PasswordManager – Programmering (Oppgave 3)

Du kan anta at alle funksjonene mottar gyldige argumenter (inn-verdier), og at filen alltid lar seg åpne. Du kan benytte deg av funksjoner fra andre deloppgaver selv om du ikke har løst de deloppgavene.

Du sitter på eksamen i pythonprogrammering. Du blir bedt om å lage et program for å håndtere passord som brukes på nettsteder. Her skal du legge inn passord for ulike nettsteder du vil huske, liste dem opp, og endre passord hvis du må. Du skal til og med kunne sjekke om du bruker det samme passordet på ulike steder.

Passord blir i dette systemet lagret i en dictionary kalt *notebook*. Hvert innslag i *notebook* er knyttet til ett brukernavn og en liste over passord til nettstedet:

```
notebook = {'Nettside': ['brukernavn',['gammeltpassord','nyttpassord']]}
```

I eksempelet over er det vist to passord, ett nytt og ett gammelt. Det kan være fra ett til et vilkårlig antall passord lagret, alt etter hver mange ganger det er endret for nettstedet. Det nyeste og aktive passordet er lagret sist i listen.

Vi vil bruke variabelen notebook til denne passordlisten i hele oppgaven. Du trenger ikke sjekke for alle typer feil i inputdata dersom det ikke er spesifisert.

Oppgave 3a) Legg til en nettside: (5%)

Lag funksjonen addSite(notebook)

Et nytt nettsted defineres ved et nettstedsnavn, et brukernavn, og et passord. Du skal lage en funksjon som har en dictionary kalt 'notebook' som spesifisert over som input. Funksjonen skal returnere notebook, der det er lagt til et nytt nettsted med et brukernavn og et passord. Dersom nettstedet allerede er lagt inn skal funksjonen gi beskjed om dette, og returnere notebook uten å ha endret den.

Eksempel på kall av funksjon (bruker-input er skrevet med fet font):

```
>>> notebook = {}
>>> notebook = addSite(notebook)
What is the site: reddit
Write username: urgle
Write password: asdf1234
Account added for reddit.
>>> print(notebook)
{'reddit': ['urgle', ['asdf1234']]}
```

Løsning:

```
# New site: ask for a site, and then for a password. Append site:password
# to the notebook and return it.
def addSite(notebook):
    sitename = input("What is the site: ")
    if sitename in notebook: # Already here
```

```
        print("That site is already there. Please use the function editSite()
             to edit your password.")
        input("Press <enter> to continue")
    else: #New sitename added
        username = input("Write username: ")
        password = input("Write password: ")
        notebook[sitename] = [username,[password]]
        print("Account added for {}.\n".format(sitename))
#        print("Account added for %s.\n"%(sitename))

    return notebook
```

Oppgave 3b) Skriv ut en liste over nettsider og deres nyeste passord: (5%)

Lag funksjonen showSites(notebook)

Funksjonen tar inn den samme notebook-dictionary, og skriver ut en oversikt over alle nettsteder, brukernavn og det **siste** passordet som er lagt inn (sist i listen). Hvis nettstedet har et navn som er mer enn 15 tegn langt skal du bare bruke de 15 første, mens selve feltet for nettstedet skal være 17 tegn langt. For brukernavnet holder det å bruke 15 tegn.

Eksempel: (input splittet på flere linjer for å få det mer leselig)

```
>>> notebook =
 {'Facebook': ['Urgle',['pwd1']],
   'reddit': ['urgle',['asdf1234','pwd1']],
'Aftenposten': ['ivrig',['pwdold','pwdnew']]}
>>> showSites(notebook)
Nettsted:         Brukernavn:     Passord:
Facebook:         Urgle           pwd1
reddit:           urgle           pwd1
Aftenposten:      ivrig           pwdnew
```

Løsning:

```
def showSites(notebook):
    print(str("Nettsted:").ljust(17),end="") # Sitename, max 15 plus some space
    print("Brukernavn:".ljust(15),end="")
    print("Passord:")
    for site in notebook:
#       site[:15] means length now 15 if it is longer, len(site) if it is shorter
        print(str(site[:15]+":").ljust(17),end="") # Sitename, 15 plus some space
        print(notebook[site][0].ljust(15),end="")  # username, max 15
        print(notebook[site][1][- 1]) # The last and active pwd.
```

The exam didn't ask for the print to be sorted in any way.Just a note: dictionaries weren't officially sorted as of the date of the exam. Starting with 3.7, python stores dictionary entries in the order they were added.
Presume no sites of more than fifteen characters, but strip above 15.
There are many ways of printing, here we show one with many print statements but each of them are finished with 'end="" ' to not add a newline. Could be done in one print, shown in other examples.

Oppgave 3c) Hjelpefunksjon: formattering av liste til streng (3%)

Lag funksjonen formatList(list)

Du vil snart ha bruk for å formatere en liste full av strenger til en mer lesbar tekst. Funksjonen tar imot en liste, og returnerer en streng med alle elementene. Hvert element skal være separert av komma og mellomrom ', '.

Eksempel:

```
>>> tmp = formatList(['pwd','pwd1','pwd2'])
>>> print(tmp)
pwd, pwd1, pwd2
```

Løsning:

```
def formatList(list):
    # We use the function string.join. This takes an iterable (like a list) and
    # returns one string that contains all items in the list, separated by
    # whatever we specify. Could also be done using for loops and some checks.

    str = (', '.join(list))
    return str
```

Oppgave 3 d) Endre passord på nettsted: (6%)

Lag funksjonen editSite(notebook, site)

På grunn av sikkerheten bør en endre passordet på nettsteder innimellom. Funksjonen editSite tar inn notebook, samt navnet på nettstedet vi vil lage et nytt passord for. Vi ønsker også å lagre de gamle passordene. Hvis det nye passordet allerede har blitt brukt skal den liste opp passordene som er brukt for dette nettstedet før, og spørre etter et nytt. Hvis det nye passordet ikke er brukt før skal det legges til i slutten av passordlisten for dette nettstedet. Bruk funksjonen fra c) for å hjelpe deg. Funksjonen editSite skal returnere en oppdatert notebook.

Eksempel på kall av funksjon (bruker-input i det kjørende programmet er skrevet med **fet font**):

```
>>> notebook = {'Facebook': ['Urgle',['pwd1','pwd2']]}
>>> editSite(notebook,'Facebook')
Add new site password for Facebook: pwd2
'pwd2' has been used for Facebook already.

The following passwords have been used: pwd1, pwd2
Add new site password for Facebook: pwd3
Facebook has been updated with a new password.
```

```
>>> print(notebook)
{'Facebook': ['Urgle', ['pwd1', 'pwd2', 'pwd3']]}
```

Løsning:

```
def editSite(notebook, site):
    # Make a loop that will continue to ask for a password
    # until it the user adds a new one.
    old = True

    while old:
        password = input("Add new site password for %s: "%site)
        if password in str(notebook[site][1]): # Used already
            print("'%s' has been used for %s already.\n"%(password,site))
            print("The following passwords have been used:",
                                formatList(notebook[site][1]))
        else: # New password, go out of loop and append
            old = False
    notebook[site][1].append(password)
    print(site,"has been updated with a new password.")

    return notebook
```

Merk for øvrig at passordeksemplene som er vist i testkjøringen – pwd1, pwd2, pwd3 – er **dårlige** passord som det ikke ville være anbefalt å bruke noe sted (for korte, for lett å gjette). Kun brukt her for å gjøre eksempelkjøringen kort og kompakt.

Oppgave 3 e) Sjekke om passord er brukt flere steder: (6%)

Lag funksjonen secureSites(notebook)

Det er viktig å ikke bruke det samme passordet på flere steder. Denne funksjonen skal ta inn notebook, og gå igjennom hvert eneste av de sist innlagte passordene for hvert nettsted. Hvis et aktivt passord (altså ikke et gammelt) brukes på mer enn ett nettsted skal den vise hvilket passord dette gjelder, og hvilke nettsteder det er brukt på. Hvis alle de nyeste passordene er unike skal den gratulere brukeren med en god jobb.

Eksempel der samme passord 'pwd1' brukes to ganger:

```
>>> notebook = {'Facebook': ['Urgle',['oldpwd', 'pwd1']], \
                    'reddit': ['Urgle',['pwd1']]}
>>> secureSites(notebook)
You have used the password 'pwd1' at the following sites: Facebook, reddit.
```

Hvis du ikke har noen like passord:

```
>>> notebook = {'Facebook': ['Urgle',['oldpwd', 'pwd1']], \
        'reddit': ['Urgle',['pwd2']]}
>>> secureSites(notebook)
No sites had similar passwords. Good job!
```

Løsning:

```
def secureSites(notebook):
    # There are many ways to check if a password is used many times
    # One way is to make a dictionary with passwords as keys, pointing
    # to a list of sites that use it.
    pwds = {}

    for site, passlist in notebook.items():
        # We want the newest and active passwords, the ones in the end of the list
        currentpwd = passlist[1][- 1] # The most recent pwd for that site

        # Check if the password is already in our new password-key-based
        # dictionary:
        if currentpwd in pwds:
            pwds[currentpwd].append(site) # Add the current site name to
                                          # existing list
        else:
            pwds[currentpwd] = [site] # New password, first in list.

    duplicates = False # Only used this to check if there are no duplicates
    for currentpwd in pwds: # Check all passwords
        if len(pwds[currentpwd]) > 1: # Length more than one means it has
                                      # been used many times
            duplicates = True # Can't celebrate that all passwords are unique...
            print("You have used the password '{0}' at the following sites:
                  {1}.".format(currentpwd,formatList(pwds[currentpwd])))
    if not duplicates:
        print("No sites had similar passwords. Good job!")
```

Fotball – Programmering (Oppgave 4)

Din tante har en særegen interesse for spesielle fotballkamper. Som familiens datakyndige har du blitt engasjert for å hjelpe henne med å automatisere hobbyen hennes. Hun har skrevet ned noen kampresultater i en tekstfil, på dette formatet: (som du også kan se i filen Matches.txt) *(vedlegget er kopiert helt nederst på dette oppgavearket)*

```
Steinkjer,Byåsen,3-5
Byåsen,Steinkjer,2-1
Byåsen,Kvik Halden,2-10
Byåsen,Borg,0-0
…
```

Din jobb er å lage funksjoner som kan hjelpe henne med å få hente ut og tolke dette materialet.

Oppgave 4 a) Lesing fra fil: (5%)

Lag funksjonen importResults(file)

Alle resultater i serien ligger i en tekstfil kalt *Matches.txt* i den samme folderen som programmet, din funksjon skal lese fra denne filen. Hvis filen som er spesifisert ikke eksisterer må du be brukeren om å skrive inn rett filnavn eller trykke 'q' for å avslutte. Innparameter til funksjonen skal være filnavn, og funksjonen skal returnere en liste der hver tekstlinje (med unntak av linjeskift) blir ett element i listen, se eksemplet under:

Kall og resultat:

```
>>> results = importResults('Matches.txt')
>>> print(results)
['Steinkjer,Byåsen,3-5', 'Byåsen,Steinkjer,2-1', 'Byåsen,Kvik Halden,2-10',
'Byåsen,Borg,0-0', 'Borg,Lade,3-2', 'Byåsen,Borg,0-0', 'Lade,Steinkjer,5-
1', 'Nardo,Borg,3-3', 'Borg,Nardo,11-3', 'Steinkjer,Borg,2-2']
>>>
```

Hvis filnavn ikke finnes:

```
>>> results = importResults('Matches')
>>> 'Matches' could not be found. File name ('q' exits):
```

… og så avslutter funksjonen hvis det neste som skrives inn er 'q', ellers går den videre som over og sjekker om det som er skrevet inn er rett filnavn osv.

Løsning:

```
def importResults(file):
    foundFile = False

    # First we need to check if the filename exists. If not, ask for a new name.
    keepOn = True
```

```
    while keepOn and not foundFile:
        try:
            open_file=open(file, 'r') # Read only mode
            if open_file: # We have a match
                foundFile = True
        except IOError: # No such file.
            print("'"+file+"'",end="")
            file = input(" could not be found. File name (q exits): ")
            if file == 'q' or file == 'Q':
                print("Be that way")

    if foundFile:
        # Read from the opened file, and split it into separate items in a list
        # by using splitlines() (splitting at every new line \n
        textResults = open_file.read().splitlines() # [[line1][line2][line3][...]]
        open_file.close() # Closing to save memory
        return textResults

    else: return
```

Vi skulle egentlig også ha flyttet lesing av filen inn i try-catch, fordi den kan være på feil format. Dette ble derimot ikke spesifisert i oppgaven.

Oppgave 4 b) Analyse av resultater: (6%)

Lag en funksjon *analyzeResults(results)*

Funksjonen tar *results* fra oppgave a som input, og returnerer en *liste av lister* med 'hjemmelag', 'bortelag', hjemmemål og bortemål. Legg merke til at målene ikke lagres som tekststrenger men som tall.

Eksempel på kall av funksjon og resultat:

```
>>> results = importResults('Matches.txt')
>>> analyzed = analyzeResults(results)
>>> print(analyzed)
[['Steinkjer', 'Byåsen', 3, 5], ['Byåsen', 'Steinkjer', 2, 1], ['Byåsen',
'Kvik Halden', 2, 10], ['Byåsen', 'Borg', 0, 0], ['Borg', 'Lade', 3, 2],
['Byåsen', 'Borg', 0, 0], ['Lade', 'Steinkjer', 5, 1], ['Nardo', 'Borg', 3,
3], ['Borg', 'Nardo', 11, 3], ['Steinkjer', 'Borg', 2, 2]]
```

Løsning:

```
def analyzeResults(matches):  #(example graciously provided by Miriam)
    analyzedMatches = []
    for match in matches:
        lst = match.split(',')           # Split all by comma
        goals = lst[2].split('-')        # Split the goals
        lst[2] = int(goals[0])           # Adding home goal as int
        lst.append(int(goals[1]))        # Away goals
        analyzedMatches.append(lst)
    return analyzedMatches
```

Oppgave 4 c) Beregning av poeng fra kamper: (3%)

Lag funksjonen calculateScores(homeGoals, awayGoals)

Nå har du tilgang til alle kampene, og hvor mange mål hvert lag fikk. Din neste oppgave er å lage en funksjon som beregner poengene til de to ulike lagene i en enkelt kamp: Den skal ha som inputparametre målene fra en fotballkamp: (hjemmemål, bortemål). Funksjonen skal returnere to verdier: poengene til hjemmelaget og poengene til bortelaget.
Tap gir 0 poeng, uavgjort gir 1 poeng, seier gir 3 poeng.

Eksempel for første kamp, der Byåsen slo Steinkjer på bortebane:

```
>>> match_result = calculateScores(3, 5)
>>> print(match_result)
(0, 3)
```

Løsning:

```
def calculateScores(homeGoals, awayGoals):
    # Standard if clauses, and standard football rules apply
    if homeGoals > awayGoals:
        return 3, 0
    elif homeGoals < awayGoals:
        return 0, 3
    else:        return 1, 1
```

Oppgave 4 d) Analyse av resultat: (5%)

Lag funksjonen sumTeamValues(analyzed)

Det neste du skal gjøre er å lage en oversikt over hvor mange poeng hvert enkelt lag har, og hvor mange kamper de har spilt. Funksjonen skal ha som input *analyzed* fra oppgave b. I retur skal du få en dictionary der nøklene er lagnavn som peker til en liste med lagets totale poengsum og totalt antall kamper. Det er nok lurt å bruke calculateScores fra forrige oppgave for å forenkle prosessen.

```
>>> analyzed = analyzeResults(results)

>>> team_data = sumTeamValues(analyzed)

>>> print(team_data)
{'Steinkjer': [1, 4], 'Byåsen': [8, 5], 'Kvik Halden': [3, 1], 'Borg': [10, 6], 'Lade': [3, 2], 'Nardo': [1, 2]}
```

Løsning:

```
def sumTeamValues(analyzed):

    team_values = {} #Empty dictionary to store each team.

    for match in analyzed:
```

```
        # Each match has the following layout:
        # match = ['Steinkjer', 'Byåsen', 3, 5]
        # initialize some data from each match and calculated scores
        home, away, = match[0], match[1]
        homepoints, awaypoints = calculateScores(match[2], match[3])

        # If the home team has already been added as a team, increase
        # points and matches
        if home in team_values:
            team_values[home] = [team_values[home][0] + homepoints,
                                 team_values[home][1] + 1]
        else: # add the home team, with homepoints and 1 match played
            team_values[home] = [homepoints, 1]
        if away in team_values: # Away team already played, just add
                                # points and match + 1
            team_values[away] = [team_values[away][0] + awaypoints,
                                 team_values[away][1] + 1]
        else: # New away team, add points and one match
            team_values[away] = [awaypoints, 1]
    return team_values
```

Oppgave 4 e) Sammenstilling av resultater: (5%)

Lag funksjonen showResults(analyzed)

Din tante vil gjerne ha sammenstilt alle kampene på en fin og lesbar måte. Hun vil ha sammenstilt alle kampdata på formatet du ser under. Hjemmeseier markeres med (H), uavgjort markeres med (U), mens borteseier markeres med (B). Funksjonen bruker de analyserte kampene fra oppgave b som input. Du kan forutsette at ingen lag har navn som er over fjorten tegn lange.

Funksjonen skriver ut tabellen under. Den er 45 tegn bred, feltene for hvert lagnavn skal være femten tegn lange. Legg også merke til hvordan målene er høyrestilt.

Eksempel på kjøring:

```
>>> showResults(analyzed)
#############################################
# Steinkjer      Byåsen          3 -  5 (B) #
# Byåsen         Steinkjer       2 -  1 (H) #
# Byåsen         Kvik Halden     2 - 10 (B) #
# Byåsen         Borg            0 -  0 (U) #
# Borg           Lade            3 -  2 (H) #
# Byåsen         Borg            0 -  0 (U) #
# Lade           Steinkjer       5 -  1 (H) #
# Nardo          Borg            3 -  3 (U) #
# Borg           Nardo          11 -  3 (H) #
# Steinkjer      Borg            2 -  2 (U) #
#############################################
```

Løsning:

```
def showResults(analyzed):
    print("#" * 45) # not important to get the number here right
```

```
    for match in analyzed:
        # First we need to calculate what the result was - draw,
        # homewin or awaywin
        homepoints, awaypoints = calculateScores(match[2], match[3])
        if homepoints == 1:
            score = 'U'
        elif homepoints == 3:
            score = 'H'
        else:
            score = 'B'
        print("# %s%s%s - %s (%s) #"%(match[0].ljust(15),
                                                match[1].ljust(15),
                                                str(match[2]).rjust(2),
                                                str(match[3]).rjust(2),
                                                score))
        # I think this one is prettier:
        # print("# {0}{1}{2} - {3} ({4}) #".format(match[0].ljust(15), ...
        # (and so on)


    print("#" * 45)
```

4 f) Sammenstilling av resultater: (6%)

Lag funksjonen savePoints(team_data)

Nå skal du bruke resultatene fra sumTeamValues (dictionary-resultatet fra oppgave d) til å lage en ordnet liste av alle lagene som har spilt, hvor mange poeng de har tilsammen og hvor mange kamper de har spilt til nå. Lagene skal skrives ut etter hvor mange poeng de har. Resultatet skal lagres i filen *Points.txt* i den samme katalogen som Python-programmet. Dersom filen finnes fra før skal du overskrive den. Filen skal se ut som vist under og i vedlegget Points.txt. Det vil ikke legges vekt på helt perfekt treff på lengde av strenger, det er å vise at en forstår prinsippene som er viktig.

Eksempel på kjøring og resultat:

```
>>> savePoints(team_data)
Team information saved to Points.txt.
```

Points.txt vil da inneholde: (Det er 35 tegns bredde totalt)

```
###################################
# Navn              Poeng   Kamper #
# Borg              10      6      #
# Byåsen             8      5      #
# Lade               3      2      #
# Kvik Halden        3      1      #
# Steinkjer          1      4      #
# Nardo              1      2      #
###################################
```

Løsning:

Her legger vi ved to versjoner: En kort og litt vanskeligere å lese, samt en lenger og mer forståelig. Bruk gjerne litt tid på å lese den vanskelige, og prøv å forstå hvordan den virker! Den bruker blant annet ulik print-syntaks.

```
# Version one: compact but less readable.
def savePoints(team_data):
    out = open('Points.txt', 'w')
    out.write("#" * 35)
    out.write("\n# {:<15}  {:<6}  {:<6} #\n".format("Navn", "Poeng", "Kamper"))
    # Go to https://docs.python.org/2/library/string.html
    # and search for :< and you will see a page full of lovely string fun.

    team_list = list(reversed(sorted((value[0],value[1],key) for (key,value) in
team_data.items())))
    for team in team_list:
        out.write("# {:<15}  {:<6}  {:<6} #\n".format(team[2],
                  str(team[0]).rjust(2), team[1]))
    out.write("#" * 35)
    out.close()

    # Version two: Simpler way, more elaborate and easier to understand
    # We also use print here instead of writing to file.
def savePoints2(team_data):
    print("#" * 35)
    print("# %s  %s  %s #"%("Navn".ljust(15), "Poeng".ljust(6),
          "Kamper".ljust(6)))

    team_list = [] # Reset the list for example purposes.
    for t in team_data.items():
        # Each dictionary item: {'Name', [points, matches]}
        # Since we need to sort on points, we add t[1][0] first, then
        # name and matches
        team_list.append([t[1][0], t[0], t[1][1]])
        # team_list now looks like this: [[points, teamname, matches],[p,t,m]...]

    # Now we need to sort the list, and in the reverse order:
    # Most points (at the top)
    team_list = sorted(team_list, reverse=True)

    for team in team_list: # First name, then points and matches
        print("# %s  %s  %s #\n"%(team[1].ljust(15),
              str(team[0]).rjust(2).ljust(6), str(team[2]).ljust(6)),end="")
    print("#" * 35)
```

Vedlegg

Matches.txt ser slik ut:

Steinkjer,Byåsen,3-5

Byåsen,Steinkjer,2-1

Byåsen,Kvik Halden,2-10

Byåsen,Borg,0-0

Borg,Lade,3-2

Byåsen,Borg,0-0

Lade,Steinkjer,5-1

Nardo,Borg,3-3

Borg,Nardo,11-3

Steinkjer,Borg,2-2