Oefententamen Inleiding Programmeren voor Bèta-gamma

Dit tentamen bestaat in totaal uit vier programmeeropdrachten. Je hebt hier in totaal 3 uur voor. Je gaat het tentamen maken in een Sandbox-omgeving. Dit is een eenvoudigere variant van de IDE waarin je tot nu toe hebt geprogrammeeerd. De bedoeling is dat je alle uitwerkingen in één bestand zet: `tentamen.py`. Deze is al geopend in de editor van de Sandbox.

Ga naar de volgende website: `progbg.mprog.nl/oefententamen`

Op deze website vind je een link naar de Sandbox-omgeving en een link voor het inleveren van je uitwerkingen, als je klaar bent.

Ja mag allleen de de cursuswebsite (progbg.mprog.nl) als informatiebron raadplegen. **Tijdens dit tentamen mag je aleen de Sandbox omgeving en de cursuswebsite op je laptop open hebben staan. Sluit alle andere programmas en websites!**

Je kan geen assistentie krijgen. Je moet alles echt zelf doen. Je kan natuurlijk wel om opheldering vragen als een van de opdrachten niet duidelijk is.

Je wordt niet beoordeeld op op programeerstijl, maar alleen op de correctheid van de code. Dus je hoeft geen tijd te steken in commentaar schrijven en goede variabelenamen kiezen. (Alhoewel het voor je zelf handig kan zijn om dit wel te doen, natuurlijk.)

1. Getallenlijst (1 pt.)

Schrijf een functie `divisors(n)`. Deze functie berekent alle delers van `n` (inclusief 1 en `n` zelf). Return het resultaat in een lijst.

Gebruiksvoorbeeld:

```
divisor_list = divisors(80)
print(divisor_list)
```

Verwachtte output:

```
[1, 2, 4, 5, 8, 10, 16, 20, 40, 80]
```

2. Getaltheorie (2 pt.)

Sommige getallen n kan je schrijven in de vorm $n = 2 \cdot a + b$ waarbij a en b allebei delers van n zijn.

Schrijf een functie `print_decomposition(n)`. Deze functie print alle decomposities die aan de bovenstaande regel voldoen.

Gebruiksvoorbeeld:

```
print_decomposition(12)
```

Verwachte output:

```
12 = 2*3 + 6
12 = 2*4 + 4
```

3. Text (2 pt.)

Schrijf een functie `find_alliterations(text, letter)`. Deze functie krijgt als input twee strings: `text` en `letter`. De functie zoekt alle woorden uit `text` die beginnen met de letter `letter`. Je mag er van uitgaan dat de parameter `letter` altijd een enkele letter is.

Gebruiksvoorbeeld:

```
example_text = "David Donald Doo dreamed a dozen doughnuts and a duck-dog, too."
print(find_alliterations(example_text, "d"))
```

Verwachte output:

```
['david', 'donald', 'doo', 'dreamed', 'dozen', 'doughnuts', 'duck-dog']
```

4. Data (2 pt.)

Voor deze opdracht ga je het databestand `barca.txt` gebruiken. Deze is al aanwezig in de sandbox. Dit bestand bevat de resultaten van het voetbalelftal van Barcelona (seizoenen 11/12 tot en met 13/14). De inhoud ziet er als volgt uit:

```
date,opponent,result,goals_barcelona,goals_opponent,where
29/08/11,Villarreal,won,5,0,home
10/09/11,Sociedad,draw,2,2,away
17/09/11,Osasuna,won,8,0,home
...
03/05/14,Getafe,draw,2,2,home
11/05/14,Elche,draw,0,0,away
17/05/14,Ath Madrid,draw,1,1,home
```

Zoals je ziet zijn de velden gescheiden door komma's en bevat de volgende informatie:

1. de datum van de wedstrijd
2. de tegenstander
3. het resultaat (Barcelona heeft gewonnen/verloren/gelijkgespeeld)
4. het aantal goals voor Barcelona
5. het aantal goals voor de tegenstander (uit/thuis)

Schrijf een functie `longest_streak(filename)`. Deze functie betekent de lengte van langste *winning streak*. Een *winning streak* is een aaneengesloten periode met alleen maar gewonnen wedstrijden (dus niet onderbroken door een verlieswedstrijd of gelijkspel).

Gebruiksvoorbeeld:

```
streak = longest_streak('barca.txt')
print(streak)
```

Verwachte output:

```
13
```

Tips:

1. Je kan een bestand openen met `input_file = open(filename, 'r')`
2. Vergeet het bestand niet te sluiten met `input_file.close()`
3. De eerste regel van `barca.txt` bevat geen data maar header info. Je kan deze overslaan met de regel `next(input_file)` direct na het openen van het bestand.