# Uitwerking opdracht

Opdracht : *Tijdmeting op mars*

Weeknummer : *1*

Studentnummer : *S1096853*

Naam student : *Jordy Kruijer*

Specialisatie : *FICT*

Pogingnummer : *1*

# Vraagstelling

*Schrijf een programma waarin je sol (dagen op mars) omzet in aardse tijdwaardes.*

# Specificatie

## Invoer

*De invoer is een natuurlijk getal dat de sol(dagen op mars) uitdrukt*

## Uitvoer

*De uitvoer is een string waarin de berekende waardes staan verwerkt. “1874 sol = 1925 dagen, 12 uren, 26 minuten en 47 seconden”*

## Verband tussen in- en uitvoer

*De invoer is een bepaalde waarde waarmee verschillende berekeningen gedaan zullen worden om te zorgen dat de uitvoer bepaalt wordt.*

## Beperkingen

*Zowel de getallen bij de invoer als bij de uitvoer moeten natuurlijke getallen zijn. Dit houdt in dat ze afgerond moeten worden.*

## Voorbeelden (testscenario’s)

*Invoer: 1874 Uitvoer: 1874 sol = 1925 dagen, 12 uren, 26 minuten en 47 seconden*

*Invoer: 1 Uitvoer: 1 sol = 1 dagen, 0 uren, 39 minuten en 34 seconden*

# Ontwerp

1. *Vraag om de waarde van Sol*
2. *Reken de waarde van Sol om in (aardse)seconden*
3. *Bereken hoeveel dagen, uren, minuten en seconden dit zijn in aardse tijden*
4. *Geef de uitvoer weer*

# Pseudocode

*//invoer sol*

*Sol = invoer van de gebruiker*

*//omrekenen sol naar seconden*

*SolOmrekenenFormule = aantal uren,minuten,seconden die in een sol dag zitten*

*SolInSeconde = Sol \* SolOmrekenenFormule*

*//formules voor aardse dagen in seconden*

*AardDagInSeconde*

*AardUurInSeconde*

*AardMinInSeconde*

*//berekenen aantal dagen, uren, minuten, seconden*

*SolInSeconde gedeelt door Aard(…)InSeconde // berekend de waarde*

*SolInSeconde modulo Aard(…)InSeconde // berekend de rest om mee verder te werken*

*//uitvoer weergeven*

*Gebruik de round() functie voor het afronden van de getallen.*

*Dit is uiteindelijk de int() functie geworden*

# Code

# invoer sol (sol)  
Sol = int(input("Geef het aantal sol in: "))  
  
# omrekenen Sol omzetten in seconden  
SolOmrekenenFormule = 35.244 + (39 \* 60) + (24 \* 60 \* 60)  
SolInSeconde = Sol \* SolOmrekenenFormule  
  
# aarddag in seconden  
AardDagInSeconden = (24 \* 60 \* 60)  
AardUurInSeconden = (60 \* 60)  
AardMinInSeconden = 60  
  
# omrekenen resultaat naar dagen, uren, minuten, seconden  
AardDagen = SolInSeconde / AardDagInSeconden #hier bereken ik hoeveel dagen hier in zitten  
RestNaDag = SolInSeconde % AardDagInSeconden #hier bereken ik de rest waarmee ik verder kan rekenen  
AardUren = RestNaDag / AardUurInSeconden #hier bereken ik de uren  
RestNaUur = RestNaDag % AardUurInSeconden #hier bereken ik de rest  
AardMin = RestNaUur / AardMinInSeconden #hier bereken ik de minuten  
RestNaMin = RestNaUur % AardMinInSeconden #hier bereken ik de rest, dit zijn de seconden  
  
# uitvoer in aardtijd in natuurlijke getallen(uitvoer)  
print(Sol, "soldagen = ", int(AardDagen), "dagen, ", int(AardUren), "Uren, ",  
 int(AardMin), "Minuten en", int(RestNaMin), "Seconden") #int wordt gebruikt om de waardes te veranderen in int's  
  
#print(Sol, "soldagen = ", AardDagen, "dagen, ", AardUren, "Uren, ",  
# AardMin, "Minuten, ", RestNaMin, "Seconde") #dit is een controle lijn, om afrondingen te controleren

# Test

*Geef het aantal sol in: 1874*

*1874 soldagen = 1926 dagen, 12 Uren, 27 Minuten en 47 Seconden*

*Bij deze test valt op dat het opgegeven aantal minuten verschilt van het verwachtte resultaat. Dit komt door de round() functie. Wanneer er wordt gekeken naar de code zonder deze functie eindigt dit op 26.7….*

*De round functie rond dit dus omhoog af.*

*Geef het aantal sol in: 1*

*1 soldagen = 1 dagen, 1 Uren, 40 Minuten en 35 Seconden*

*Hierin is ook weer de afwijking te zien maar dit keer bij de uren.*

*Hierna heb ik hetzelfde programma geprobeerd maar de round() functie vervangen door de int() functie. Hier kreeg ik wel dezelfde resultaten dan ik al had verwacht. De code hierboven is dan ook de aangepaste code te zien.*