Practicum vrijdag

Collapse context

Your answer passed the tests! Your score is 100.0%. [Submission #670a660a3baf09455ab781f8 (**2024-10-12 12:05:30**)]

Question 1: HMS

✓ Perfect

Schrijf een functie `hms` die een getal in seconden krijgt en een lijst teruggeeft (= gebruik return, geen print in je functie) met als eerste element het aantal uren, als tweede het aantal minuten en als derde het aantal seconden.

```
>>> hms(3601)
[1,0,1]
>>> hms(100)
[0,1,40]
```

```
1  def hms(s):
2      hours = s // 3600
3      minuten = s % 3600 // 60
4      seconden = s % 60
5
6      return [hours, minuten, seconden]
```

Submit

Question 2: En terug!

✓ Perfect

Schrijf een functie `tijdNaarSeconden` die drie getallen krijgt als parameter (resp. uren, min en sec) en teruggeeft hoeveel seconden het zijn.

```
>>> tijdNaarSeconden(1,0,1)
3601
>>> tijdNaarSeconden(0,1,40)
100
```

```
1  def tijdNaarSeconden(h, m, s):
2      return h * 3600 + m * 60 + s
```

Submit

Question 3: Combinatoriek (a)

✓ Perfect

Schrijf een functie `faculteit` die n! uitrekent (zonder gebruik te maken van recursie; als je niet weet wat dat is, dan doe je dat correct ;-)).

```
def faculteit(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

Submit

Question 4: Combinatoriek (b)

✓ Perfect

Schrijf een functie `variatie(n,k)` die het aantal mogelijke variaties van k verschillende elementen uit n mogelijke elementen berekent. Een variatie is in de combinatoriek een geordende keuze van k verschillende objecten uit een totaal van n. Hergebruik je functie faculteit, maar definieer die niet opnieuw. Je code uit de vorige opgave werd automatisch toegevoegd.

```
def variatie(n, k):
    return faculteit(n) // faculteit(n - k)
```

Submit

Question 5: Combinatoriek (c)

✓ Perfect

Schrijf een functie `combinatie(n,k)` die het aantal mogelijke combinaties van k verschillende elementen uit n mogelijke elementen berekent. Hergebruik je vorige functies, maar definieer die niet opnieuw. Je code uit de vorige opgaves werd automatisch toegevoegd.

```
def combinatie(n,k):
    return int((1 / faculteit(k)) * variatie(n, k))
```

Submit

Question 6: Veelvouden

✓ Perfect

Schrijf een functie `remove_multiples` met twee parameters `l` (een lijst van integers) en `n` (een integer) die alle veelvouden van `n` verwijdert uit de lijst `l`. Doe dit door de elementen effectief te verwijderen uit de lijst. De functie geeft niets terug.

```
def remove_multiples(l, n):
    for i in l:
        if not i % n: l.remove(i)

    for i in l:
        if not i % n: l.remove(i)
```

Submit

Question 7: Veelvouden vervolgd

✓ Perfect

Schrijf een functie `remove_multiples` met twee parameters `l` (een lijst van integers) en `n` (een integer) die alle veelvouden van `n` verwijdert uit de lijst `l`. Doe dit door een nieuwe lijst aan te maken en de originele lijst niet aan te passen.

```
def remove_multiples(l, n):
    result = []
    for i in l:
        if i % n: result.append(i)
    return result
```

Submit

Question 8: Even/oneven

✓ Perfect

Schrijf een functie `even` die `True` teruggeeft als en slechts als de meegegeven parameter even is.

```
def even(n):
    return not n % 2
```

Submit

Question 9: Piramide

✓ Perfect

Schrijf een functie `piramide` die een getal `n` als parameter krijgt en vervolgens een piramide van `n` rijen opstelt als een string met in elke rij een opeenvolging van natuurlijke getallen beginnend met 1. Voor `n = 5` wordt dit

```
      1
    1 2 3
  1 2 3 4 5
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8 9
```

Merk op dat er geen print in je functie voorkomt, maar dat je een string teruggeeft.

```
def piramide(n):
    for i in range(n):
        # Print the spaces first
        print(' ' * (2 * (n - i - 1)), end='')
        # Print the numbers
        for j in range(1, (i+1) * 2): print(j, end=' ')
        print()
```

Submit

Question 10: Som veelvouden

✓ Perfect

Als we alle natuurlijke getallen kleiner dan tien die veelvouden van 3 of 5 zijn oplijsten, krijgen we 3, 5, 6 en 9. De som van deze veelvouden is 23.

Schrijf een functie `som_veelvouden` om de som van alle veelvouden van 3 of 5 kleiner dan `n` op te lijsten.

```
def som_veelvouden(n):
    result = 0
    for i in range(n):
        if not i % 3 or not i % 5: result += i
    return result
```

Submit

Question 11: Som veelvouden veralgemeend

✓ Perfect

Maak de functie `som_veelvouden` meer algemeen: vertrek van 3 getallen `getal1`, `getal2` en `bovengrens`, en vind de som van alle veelvouden van `getal1` of `getal2` kleiner dan `bovengrens`.

```
def som_veelvouden(a, b ,c):
    result = 0
    for i in range(c):
        if not i % a or not i % b: result += i
    return result

```

Submit