1 Reeks 01: variabelen, expressies en statements

1.1 ISBN

```
# eerste negen cijfers van ISBN-10 code inlezen en omzetten naar integers
x1 = int(input())
x2 = int(input())
x3 = int(input())
x4 = int(input())
x5 = int(input())
x6 = int(input())
x7 = int(input())
x8 = int(input())
x9 = int(input())

# controlecijfer berekenen
x10 = (
    x1 + 2 * x2 + 3 * x3 + 4 * x4 + 5 * x5 + 6 * x6 + 7 * x7 + 8 * x8 + 9 * x9
) % 11

# controlecijfer uitschrijven
print(x10)
```

1.2 Som van twee getallen

```
# twee termen inlezen en omzetten naar integers
term1 = int(input('Geef een getal: '))
term2 = int(input('Geef nog een getal: '))

# som van twee termen berekenen
som = term1 + term2

# som uitschrijven
print(som)
```

1.3 Examens verbeteren

```
# gegevens inlezen
A = int(input())    # fouten gevonden door eerste examinator
B = int(input())    # fouten gevonden door tweede examinator
C = int(input())    # gemeenschappelijke fouten

# totaal aantal niet opgemerkte fouten in examen berekenen
niet_opgemerkte_fouten = (A - C) * (B - C) / C

# aantal niet opgemerkte fouten in examen uitschrijven
print(f'Er werden {niet_opgemerkte_fouten:.2f} fouten niet opgemerkt.')
```

1.4 The pudding guy

```
# parameters van probleem inlezen
stuks = int(input())        # aantal gekochte stuks
prijs = float(input())      # kostprijs per stuk
barcodes = int(input())     # aantal barcodes in een reeks
mijlen = int(input())       # aantal mijlen die men ontvangt per reeks barcodes

# totale kostprijs van aangekochte stuks berekenen
uitgave = stuks * prijs
```

```
# totaal aantal frequent flyer mijlen berekenen
inkomst = (stuks // barcodes) * mijlen

# opgemaakte uitvoer uitschrijven
print(f'Phillips spendeerde ${uitgave} voor {inkomst} frequentflyermijlen.')
```

1.5 Ontwikkelingsindex

```
# bijkomende wiskundige functionaliteit toevoegen aan Python
#
#   sqrt: vierkantswortel
#   log: (natuurlijke) logaritme
#
# opmerking: de math module maakt deel uit van The Python Standard Library
from math import sqrt, log

# parameters inlezen die gebruikt worden in de Human Development Index
land = input()          # naam van het land
LE = float(input())     # levensverwachting bij geboorte
MYS = float(input())    # gemiddeld aantal jaren scholing voor een 25-jarige
EYS = float(input())    # verwacht aantal jaren scholing voor een 5-jarige
GNI = float(input())    # bruto nationaal inkomen per hoofd van de bevolking

# drie genormaliseerde indices bepalen die gebruikt worden bij de berekening van de Human
    Development Index
LEI = (LE - 20.0) / (82.3 - 20.0)  # levensverwachtingsindex
MYSI = MYS / 13.2                  # gemiddelde scholingsjarenindex
EYSI = EYS / 20.6                  # verwachte scholingsjarenindex
EI = sqrt(MYSI * EYSI) / 0.951     # onderwijsindex
II = (log(GNI) - log(100.0)) / (log(107721) - log(100))

# Human Development Index berekenen als het meetkundig gemiddelde van de drie genormaliseerde
    indices
HDI = pow(LEI * EI * II, 1 / 3)

# waarde van Human Development Index afdrukken; format specifier .3f wordt gebruikt om een
    floating point getal af te
# drukken met drie cijfers na het decimale punt (hierbij wordt ook afronding toegepast)
print(f'De HDI van {land} bedraagt {HDI:.3f}.')
```

1.6 De gestopte klok

```
# tijd aflezen op vier opeenvolgende op 24-uursklokken (uu:mm)
u1, m1 = int(input()), int(input())
u2, m2 = int(input()), int(input())
u3, m3 = int(input()), int(input())
u4, m4 = int(input()), int(input())

# aantal minuten per uur en per dag
minuten_per_uur = 60
minuten_per_dag = 24 * minuten_per_uur

# tijd van alle 24-uursklokken omzetten naar minuten sinds middernacht
m1 += u1 * minuten_per_uur
m2 += u2 * minuten_per_uur
m3 += u3 * minuten_per_uur
m4 += u4 * minuten_per_uur

# correctie berekenen tussen foutieve klok en correcte klok
totale_tijd = (m4 - m1) % minuten_per_dag       # afgelezen op foutieve klok
bezoek_tijd = (m3 - m2) % minuten_per_dag       # afgelezen of correcte klok
reistijd = (totale_tijd - bezoek_tijd) // 2     # reistijd in één richting

# correcte tijd bij thuiskomst berekenen
```

```
# OPMERKING: modulobewerking zorgt ervoor dat het tijdstip altijd een tijd aangeeft tussen
    00:00 en 23:59
gecorrigeerde_tijd = (m3 + reistijd) % minuten_per_dag

# correcte tijd uitdrukken in uren en minuten
print(gecorrigeerde_tijd // minuten_per_uur)
print(gecorrigeerde_tijd % minuten_per_uur)
```

2 Reeks 02: voorwaardelijke opdrachten

2.1 ISBN

```
# tien cijfers van ISBN-10 code inlezen (elk op een afzonderlijke regel)
x1 = int(input())
x2 = int(input())
x3 = int(input())
x4 = int(input())
x5 = int(input())
x6 = int(input())
x7 = int(input())
x8 = int(input())
x9 = int(input())
x10 = int(input())

# controlecijfer berekenen
controlecijfer = (
    x1 + 2 * x2 + 3 * x3 + 4 * x4 + 5 * x5 + 6 * x6 + 7 * x7 + 8 * x8 + 9 * x9
) % 11

# controlecijfer controleren en uitschrijven
print('OK' if x10 == controlecijfer else 'FOUT')

# alternatieve oplossing:
#
# if x10 == controlecijfer:
#     print('OK')
# else:
#     print('FOUT')
```

2.2 Reynoldsgetal

```
# grootheden inlezen
snelheid = float(input())
lengte = float(input())
dichtheid = float(input())
viscositeit = float(input())

# Reynoldsgetal berekenen
reynoldsgetal = (snelheid * lengte * dichtheid) / viscositeit

# stromingstype berekenen
if reynoldsgetal < 2000:
    stromingstype = 'laminaire stroming'
elif reynoldsgetal <= 4000:
    stromingstype = 'omslagstroming'
else:
    stromingstype = 'turbulente stroming'

# resultaat uitschrijven
print(f'{reynoldsgetal} ({stromingstype})')
```

2.3 Blad, steen en schaar

```
# handgebaar van beide spelers inlezen
speler1 = input()
speler2 = input()

# resultaat van spelletje blad-steen-schaar bepalen
if speler1 == speler2:
    resultaat = 'gelijkspel'
```

```
elif (
    (speler1 == 'blad' and speler2 == 'steen')
    or (speler1 == 'steen' and speler2 == 'schaar')
    or (speler1 == 'schaar' and speler2 == 'blad')
):
    resultaat = 'speler1 wint'
else:
    resultaat = 'speler2 wint'

# uitkomst van spelletje blad-steen-schaar uitschrijven
print(resultaat)

# alternatieve oplossing (gebruikt een tuple)
#
# # reeks van mogelijke handgebaren vastleggen, zodat elk handgebaar uit de reeks telkens wint
    tegen het volgende
# # handgebaar uit de reeks (waarbij het laatste element uit de reeks terug gevolgd wordt door
    het eerste element)
# handgebaren = ('blad', 'steen', 'schaar')
#
# # handgebaar van beide spelers inlezen en omzetten naar positie in reeks van mogelijke
    handgebaren
# speler1 = handgebaren.index(input())
# speler2 = handgebaren.index(input())
#
# # resultaat van spelletje blad-steen-schaar bepalen
# resultaten = ('gelijkspel', 'speler1 wint', 'speler2 wint')
# resultaat = resultaten[(speler2 - speler1) % 3]
#
# # uitkomst van spelletje blad-steen-schaar uitschrijven
# print(resultaat)


# alternatieve oplossing (gebruikt een dictionary)
#
# # dictionary aanmaken die winnaars (sleutel) en verliezers (waarde) vastlegt
# wint_tegen = {'blad': 'steen', 'steen': 'schaar', 'schaar': 'blad'}
#
# # resultaat van spelletje blad-steen-schaar bepalen
# if speler1 == speler2:
#     resultaat = 'gelijkspel'
# elif wint_tegen[speler1] == speler2:
#     resultaat = 'speler1 wint'
# else:
#     resultaat = 'speler2 wint'
#
# # uitkomst van spelletje blad-steen-schaar uitschrijven
# print(resultaat)
```

2.4 Seizoenen

```
# dag en maand van de gegeven datum inlezen
dag = int(input())
maand = input()

# naam van maand omzetten naar volgnummer van die maand in het jaar
maanden = [
    'januari', 'februari', 'maart', 'april', 'mei', 'juni', 'juli',
'augustus', 'september', 'oktober', 'november', 'december'
]
volgnummer = maanden.index(maand)

# seizoen bepalen voor de gegeven datum
if volgnummer < 2 or (volgnummer == 2 and dag < 21):
    seizoen = 'winter'
elif volgnummer < 5 or (volgnummer == 5 and dag < 21):
    seizoen = 'lente'
```

```
elif volgnummer < 8 or (volgnummer == 8 and dag < 23):
    seizoen = 'zomer'
elif volgnummer < 11 or (volgnummer == 11 and dag < 21):
    seizoen = 'herfst'
else:
    seizoen = 'winter'

# alternatieve oplossing (gebruikt tuples)
#
# if (volgnummer, dag) < (2, 21) or (volgnummer, dag) > (11, 20):
#     seizoen = 'winter'
# elif (volgnummer, dag) < (5, 21):
#     seizoen = 'lente'
# elif (volgnummer, dag) < (8, 23):
#     seizoen = 'zomer'
# else:
#     seizoen = 'herfst'

# corresponderend seizoen uitschrijven
print(f'Het is {seizoen} op {dag} {maand}.')
```

2.5 Trilateratie

```
# coördinaten van eerste cirkel inlezen
x1 = float(input())
y1 = float(input())
r1 = float(input())

# coördinaten van tweede cirkel inlezen
x2 = float(input())
y2 = float(input())
r2 = float(input())

# Euclidische afstand tussen twee middelpunten bepalen
afstand = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5

# foutmarge vastleggen voor het vergelijken van floating point getallen
eps = 1e-6

# onderlinge positie van beide cirkels bepalen
if afstand < eps:
    positie = 'concentrisch'
elif abs(afstand - abs(r1 - r2)) < eps:
    positie = 'binnen rakend'
elif abs(afstand - (r1 + r2)) < eps:
    positie = 'buiten rakend'
elif afstand < abs(r1 - r2):
    positie = 'omsluitend'
elif afstand < r1 + r2:
    positie = 'snijdend'
else:
    positie = 'gescheiden'

# onderlinge positie van beide cirkels uitschrijven
print(positie)
```

2.6 Darts

```
# extra wiskundige functies importeren
from math import sqrt, atan2, pi

# positie in carthesische coördinaten inlezen
x, y = float(input()), float(input())
```

```
# carthesische coördinaten omzetten naar poolcoördinaten
eps = 1e-6
if abs(x) < eps and abs(y) < eps:
    straal, hoek = 0.0, 0.0
else:
    straal, hoek = sqrt(x ** 2 + y ** 2), atan2(y, x)

# sommige scores kunnen direct bepaald worden op basis van de straal r, omdat ze onafhankelijk
     zijn van de hoek t;
# in die gevallen moeten de resterende berekeningen niet meer uitgevoerd worden
if straal >= 170.0:

    # pijltjes buiten het dartbord leveren geen punten op
    score = 0

elif straal < 12.7 / 2:

    # bull's eye
    score = 50

elif straal < 31.8 / 2:

    # bull
    score = 25

else:

    # hoek in poolcoördinaten omzetten naar beginhoek eerste sector
    # OPMERKING: modulo omdat functie atan2 hoek teruggeeft in interval [-, [
    sectoren = 20
    hoek = (pi / 2.0 + pi / sectoren - hoek) % (2 * pi)

    # index van sector berekenen
    index = int(hoek * sectoren / (2 * pi))

    # score is één hoger dan de index
    score = index + 1

    # score corrigeren op basis van afstand tot centrum
    if 107.0 - 9.6 < straal < 107.0:
        # triple ring
        score *= 3
    elif 170.0 - 9.6 < straal < 170.0:
        # double ring
        score *= 2

# score uitschrijven
print(score)
```

3 Reeks 03: controlelussen

3.1 ISBN

```
# eerste cijfer van eerste ISBN-10 code inlezen
# OPMERKING: er mag nog niet van uitgegaan worden dat deze regel een cijfer bevat, omdat deze
    regel ook het woord stop
#            kan bevatten
eerste_cijfer = input()

while eerste_cijfer != 'stop':

    # volgende acht cijfers inlezen en controlecijfer berekenen
    berekend_controlecijfer = int(eerste_cijfer)
    for index in range(2, 10):
        volgende_cijfer = int(input())
        berekend_controlecijfer += index * volgende_cijfer
    berekend_controlecijfer %= 11

    # gegeven controlecijfer inlezen
    gegeven_controlecijfer = int(input())

    # correctheid van gegeven controlecijfer uitschrijven
    print('OK' if gegeven_controlecijfer == berekend_controlecijfer else 'FOUT')

    # eerste cijfer van volgende ISBN-10 code inlezen
    # OPMERKING: er mag nog niet van uitgegaan worden dat deze regel een cijfer bevat, omdat
        deze regel ook het woord
    #            stop kan bevatten
    eerste_cijfer = input()
```

3.2 Chaos

```
# simulatieparameters inlezen
dichtheid = float(input())
vruchtbaarheid = float(input())
stappen = int(input())

# initiële dichtheid uitschrijven
print(dichtheid)

# dichtheid uitschrijven tijdens opeenvolgende simulatiestappen
for _ in range(stappen - 1):

    # nieuwe dichtheid berekenen
    dichtheid = vruchtbaarheid * dichtheid * (1.0 - dichtheid)

    # nieuwe dichtheid uitschrijven
    print(dichtheid)
```

3.3 Duitse tanks

```
# aantal serienummers en grootste serienummer initialiseren
aantal, maximum = 0, 0

# eerste serienummer inlezen
serienummer = int(input())

# serienummers verwerken totdat we een negatief serienummer tegenkomen
while serienummer >= 0:

    # aantal serienummers en grootste serienummer bepalen tijdens het inlezen
```

```
    aantal += 1
    maximum = max(maximum, serienummer)

    # volgende serienummer inlezen
    serienummer = int(input())

# schatting maken van het aantal geproduceerde tanks
tanks = ((aantal + 1) * maximum) / aantal - 1

# schatting uitschrijven
print(f'Het aantal geproduceerde tanks wordt geschat op {round(tanks)}.')
```

3.4 Pythagorese drietallen

```
import math

# som van drietallen inlezen
n = int(input())

# alle mogelijke drietallen overlopen, maar zorg ervoor dat we geen drietallen controleren die
    nooit aan de voorwaarden
# uit de opgave kunnen voldoen; voor a geldt immers naast 1 ≤ a (ondergrens) ook nog dat
#
#     a ≤ b ≤ c
#     a + b + c = n
#
# waaruit volgt dat
#
#     a + a + a ≤ n
#
# of (bovengrens)
#
#     a ≤ n / 3
#
for a in range(1, math.ceil(n / 3) + 1):

    # als a vastligt, dan geldt naast a ≤ b (ondergrens) ook nog dat
    #
    #     b ≤ c
    #     b + c = n - a
    #
    # waaruit volgt dat
    #
    #     b + b ≤ n - a
    #
    # of (bovengrens)
    #
    #     b ≤ (n - a) / 2
    #
    for b in range(a, math.ceil((n - a) / 2) + 1):

        # als a en b vastliggen, dan ligt ook c vast omdat a + b + c = n
        c = n - a - b

        # drietal enkel uitschrijven als de getallen de lengte van de zijden kunnen zijn van
            een rechthoekige driehoek
        if a ** 2 + b ** 2 == c ** 2:
            print(f'({a}, {b}, {c})')
```

3.5 Apen en kokosnoten

```
def aantal(noten):

    """
```

```
    Geeft omschrijving van een aantal noten terug en gebruikt daarbij de correcte enkelvouds-
        of meervoudsvorm.
    """

    return f'{str(noten) if noten else 'geen'} {'noot' if noten == 1 else 'noten'}'

# inlezen hoeveel piraten er zijn
piraten = int(input())

# inlezen hoeveel kokosnoten er initieel in de stapel liggen
kokosnoten = int(input())

# alle piraten nemen 's nachts kokosnoten voor zichzelf en voor de aap
template = '{} = {} voor piraat#{} en {} voor de aap'
for piraat in range(1, piraten + 1):

    # bepaal hoeveel noten de piraat aan zichzelf en aan de aap geeft
    kokosnoten_voor_piraat = kokosnoten // piraten
    kokosnoten_voor_aap = kokosnoten % piraten

    # uitschrijven hoeveel noten de piraat aan zichzelf en aan de aap geeft
    print(template.format(
        aantal(kokosnoten),
        aantal(kokosnoten_voor_piraat),
        piraat,
        aantal(kokosnoten_voor_aap)
    ))

    # kokosnoten van piraat en aap wegnemen uit de stapel
    kokosnoten -= kokosnoten_voor_piraat + kokosnoten_voor_aap

# bepaal hoeveel noten 's morgens voor elke piraat en voor de aap overblijven
print(f'elke piraat krijgt {aantal(kokosnoten // piraten)} en {aantal(kokosnoten % piraten)}
    voor de aap')
```

3.6 Lifters

```
# aantal lifters inlezen
lifters = int(input())

# maximale score bepalen voor eerste helft van de lifters
maximale_score = 0
helft = lifters // 2
for _ in range(helft):
    maximale_score = max(maximale_score, float(input()))

# eerstvolgende lifter beoordelen
lifter = helft + 1
score = float(input())

# verdergaan zolang er nog lifters volgen en de huidige lifter een lagere score kreeg dan die
    in de eerste helft
while lifter < lifters and score < maximale_score:
    lifter += 1
    score = float(input())

# score uitschrijven van lifter die wordt meegenomen
print(score)
```

4 Reeks 04: strings

4.1 ISBN

```
# eerste ISBN-10 code (of het woord stop) inlezen
code = input()

# opeenvolgende ISBN-10 codes inlezen totdat regel met "stop" ingelezen wordt
while code != 'stop':

    # controlecijfer berekenen
    controlecijfer = int(code[0])
    for i in range(2, 10):
        controlecijfer += i * int(code[i - 1])
    controlecijfer %= 11

    # controlecijfer berekenen: alternatieve oplossing met generator expression, slicing en
        enumerate
    # controlecijfer = sum(index * int(digit) for index, digit in enumerate(code[-1], start=1)
        ) % 11

    # controlecijfer omzetten naar stringvoorstelling
    controlecijfer = 'X' if controlecijfer == 10 else str(controlecijfer)

    # nagaan of berekende en effectieve controlecijfers gelijk zijn
    controle = 'OK' if code.endswith(controlecijfer) else 'FOUT'
    print(controle)

    # volgende ISBN-10 code (of het woord stop) inlezen
    code = input()
```

4.2 Dagboek voor Stella

```
# gecodeerde boodschap inlezen
cijfertekst = input()

# boodschap decoderen door om de andere letter te verwijderen
# OPMERKING: alle karakters die geen letter zijn worden behouden
klare_tekst = ''
behouden = True
for karakter in cijfertekst:
    if karakter.isalpha():
        if behouden:
            klare_tekst += karakter
        behouden = not behouden
    else:
        klare_tekst += karakter

# boodschap uitschrijven
print(klare_tekst)
```

4.3 Een speld in een hooiberg

```
# op te zoeken woord inlezen en omzetten naar hoofdletters
woord = input()
WOORD = woord.upper()

# aantal rijen in het rooster inlezen
rijen = int(input())

# rooster rij per rij doorzoeken; stoppen als alle rijen verwerkt zijn of als het woord
    gevonden is
```

```
rij_index = -1
kolom_index = -1
while rij_index < rijen - 1 and kolom_index == -1:

    # volgende rij inlezen en omzetten naar hoofdletters
    rij = input().upper()
    rij_index += 1

    # positie van woord in rij zoeken
    kolom_index = rij.find(WOORD)

    # positie van omgekeerd woord in rij zoeken
    if kolom_index == -1:
        kolom_index = rij.find(WOORD[::-1])

# uitschrijven waar het woord in het rooster gevonden werd
if kolom_index == -1:
    print(f'{woord} werd niet gevonden')
else:
    print(f'{woord} staat op rij {rij_index} en kolom {kolom_index}')
```

4.4 Rustig aan

```
# sleutel en cijfertekst inlezen
sleutel = input()
cijfertekst = input()

# klare tekst berekenen
klare_tekst = ''
for cijfer1 in map(str, range(10)):
    for cijfer2, karakter in zip(sleutel, cijfertekst):
        if cijfer1 == cijfer2:
            klare_tekst += karakter

# cijfertekst uitschrijven
print(klare_tekst)
```

4.5 De letters van Utrecht

```
# zin inlezen
zin = input()

# inlezen waar er moet gekeken worden om te tellen: achteruit (verleden), overal (heden) of
    vooruit (toekomst)
kijken = input()

# zin omzetten naar kleine letters
# OPMERKING: om te kunnen tellen zonder onderscheid te maken tussen hoofdletters en kleine
    letters
zin_lowercase = zin.lower()

# letters uitlijnen met hun aantal voorkomens
zin_uitgelijnd = ''
aantal_uitgelijnd = ''
for index, karakter in enumerate(zin):
    if karakter.isalpha():
        aantal = str(zin_lowercase.count(
            karakter.lower(),
            # tel vanaf volgende positie als je naar de toekomst kijkt, anders vanaf het begin
            index + 1 if kijken == 'toekomst' else 0,
            # tel tot huidige positie als je naar het verleden kijkt, anders tot het einde
            index if kijken == 'verleden' else len(zin)
        ))
        zin_uitgelijnd += f'{karakter:-<{len(aantal)}s}'
```

```
        aantal_uitgelijnd += aantal
    else:
        zin_uitgelijnd += karakter
        aantal_uitgelijnd += karakter

# aantal voorkomens van de letters weergeven, uitgelijnd ten opzicht van de letters in de zin
print(zin_uitgelijnd)
print(aantal_uitgelijnd)
```

4.6 Wepe sprepekepen p

```
# zin in p-taal inlezen
zin = input()

# vertaling maken van p-taal naar gewone taal
index = 0
vertaling = ''
klinkercombinatie = ''
while index < len(zin):

    # bepaal de letter op de huidige positie (index)
    letter = zin[index]

    if letter.lower() == 'p' and klinkercombinatie:

        # letter p voorafgegaan door een klinkercombinatie geeft aan dat de
        # klinkercombinatie die erna komt (van dezelfde lengte als de
        # voorafgaande klinkercombinatie) moet overgeslaan worden; de letter
        # p wordt zelf ook overgeslaan
        index += len(klinkercombinatie)
        klinkercombinatie = ''

    else:

        # letter overnemen
        vertaling += letter

        # klinkercombinatie bijhouden
        if letter.lower() in 'aeiou' or (klinkercombinatie + letter).lower() == 'ij':
            klinkercombinatie += letter
        else:
            klinkercombinatie = ''

    # ga naar volgende positie
    index += 1

# vertaling van zin uitschrijven
print(vertaling)

# alternatieve oplossing
#
# # importeer module om met reguliere expressies te kunnen werken
# import re
#
# # zin in p-taal inlezen
# zin = input()
#
# # vertaling van zin uitschrijven
# print(re.sub(r'(ij|[aeiou]+)p\1', r'\1', zin, flags=re.IGNORECASE))
```

5 Reeks 05: functies

5.1 ISBN

```
def isISBN(code):

    """
    Geeft True terug als het argument een string is die een geldige ISBN-10 code voorstelt.
        Anders wordt False
    teruggegeven.

    >>> isISBN('9971502100')
    True
    >>> isISBN('9971502108')
    False
    >>> isISBN('53WKEFF2C')
    False
    >>> isISBN(4378580136)
    False
    """

    # OPMERKING: isinstance is een ingebouwde functie die een Booleaanse waarde teruggeeft die
         aangeeft of het eerste
    #            argument een object is van het gegevenstype dat als tweede argument wordt
        doorgegeven aan de functie
    return (
        isinstance(code, str)                   # code moet string zijn,
        and len(code) == 10                     # en code moet bestaan uit 10 karakters,
        and code[:9].isdigit()                  # en eerste 9 karakters moeten cijfers zijn,
        and controlecijfer(code) == code[-1]    # en controlecijfer moet correct zijn
    )

def controlecijfer(code):

    """
    Berekent het controlecijfer voor een gegeven string die bestaat uit de eerste negen
        cijfers van een ISBN-10 code.
    Het controlecijfer wordt teruggegeven als een string, waarbij de waarde 10 wordt
        voorgesteld door de letter X.

    >>> controlecijfer('9971502100')
    '0'
    >>> controlecijfer('9971502108')
    '0'
    """

    # controlecijfer berekenen
    controle = sum(index * int(cijfer) for index, cijfer in enumerate(code[:9], start=1)) % 11

    # controlecijfer omzetten naar stringvoorstelling
    return 'X' if controle == 10 else str(controle)

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

5.2 Woordsommen

```
def letterwaarde(letter):

    """
    >>> letterwaarde('A')
    1
    >>> letterwaarde('j')
    10
```

```
    >>> letterwaarde('!')
    0
    """

    # letterwaarde = positie van gegeven letter in alfabet
    return ord(letter.upper()) - ord('A') + 1 if letter.isalpha() else 0

def woordwaarde(woord):

    """
    >>> woordwaarde('arm')
    32
    >>> woordwaarde('BEND')
    25
    >>> woordwaarde('elbow')
    57
    """

    # bepaal som van alle letters in het woord
    return sum(letterwaarde(letter) for letter in woord)

def iswoordsom(woord1, woord2, woord3):

    """
    >>> iswoordsom('arm', 'BEND', 'elbow')
    True
    >>> iswoordsom('KING', 'chair', 'THRONE')
    True
    >>> iswoordsom('Monty', 'Python', 'SHRUBBERY')
    False
    """

    # som van woordwaarden van eerste twee woorden =?= woordwaarde derde woord
    return woordwaarde(woord1) + woordwaarde(woord2) == woordwaarde(woord3)

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

5.3 De laatste banaan

```
def heeft_gewonnen(worp1, worp2, grens):

    """
    >>> heeft_gewonnen(3, 2, 4)
    True
    >>> heeft_gewonnen(1, 5, 4)
    False
    """

    return max(worp1, worp2) <= grens

def winnende_uitkomsten(zijden, grens):

    """
    >>> winnende_uitkomsten(6, 4)
    (16, 20)
    """

    speler1 = grens * grens
    speler2 = zijden * zijden - speler1
    return speler1, speler2

    # speler1, speler2 = 0, 0
    # for worp1 in range(zijden):
    #     for worp2 in range(zijden):
    #         if heeft_gewonnen(worp1 + 1, worp2 + 1, grens):
```

```
    #             speler1 += 1
    #         else:
    #             speler2 += 1
    #
    # return speler1, speler2

def winstkansen(zijden, grens):

    """
    >>> winstkansen(6, 4)
    (44.44444444444444, 55.55555555555556)
    """

    speler1, speler2 = winnende_uitkomsten(zijden, grens)
    totaal = speler1 + speler2
    return (speler1 / totaal) * 100, (speler2 / totaal) * 100

def winnaar(zijden, grens):

    """
    >>> winnaar(6, 4)
    2
    >>> winnaar(6, 6)
    """

    speler1, speler2 = winnende_uitkomsten(zijden, grens)
    return 2 if speler2 > speler1 else (1 if speler1 > speler2 else 0)

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

5.4 Achteruitkijkspiegel

```
def letter2positie(letter):

    """
    >>> letter2positie('a')
    0
    >>> letter2positie('Z')
    25
    """

    return ord(letter.lower()) - ord('a')

def positie2letter(positie):

    """
    >>> positie2letter(0)
    'a'
    >>> positie2letter(25)
    'z'
    """

    return chr(ord('a') + positie % 26)

def codeer_letter(letter, vorige):

    """
    >>> codeer_letter('d', 'Z')
    'c'
    >>> codeer_letter('T', 'e')
    'X'
    """

    gecodeerd = positie2letter(letter2positie(letter) + letter2positie(vorige))
    return gecodeerd.upper() if letter.isupper() else gecodeerd
```

```
def decodeer_letter(letter, vorige):

    """
    >>> decodeer_letter('c', 'Z')
    'd'
    >>> decodeer_letter('X', 'e')
    'T'
    """

    origineel = positie2letter(letter2positie(letter) - letter2positie(vorige))
    return origineel.upper() if letter.isupper() else origineel

def codeer(bericht):

    """
    >>> codeer('Saw things clearer once you were in my rear-view mirror.')
    'Ssw papvty unpervv fbpg cmi qavv mv zk pver-mdma iuziff.'
    >>> codeer('The past is always tense. The future perfect.')
    'Tal tpsl ba slhwyq lxrfw. Xal jznnlv ttvwjgv.'
    """

    gecodeerd = ''
    vorige = None
    for karakter in bericht:
        if karakter.isalpha():
            gecodeerd += codeer_letter(karakter, vorige) if vorige else karakter
            vorige = karakter
        else:
            gecodeerd += karakter

    return gecodeerd

def decodeer(bericht):

    """
    >>> decodeer('Ssw papvty unpervv fbpg cmi qavv mv zk pver-mdma iuziff.')
    'Saw things clearer once you were in my rear-view mirror.'
    >>> decodeer('Tal tpsl ba slhwyq lxrfw. Xal jznnlv ttvwjgv.')
    'The past is always tense. The future perfect.'
    """

    origineel = ''
    vorige = None
    for karakter in bericht:
        if karakter.isalpha():
            vorige = decodeer_letter(karakter, vorige) if vorige else karakter
            origineel += vorige
        else:
            origineel += karakter

    return origineel

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

5.5 Inventaris

```
import string

def inventaris(zin):

    """
    >>> inventaris('Data aequatione quotcunque fluentes quantitates involvente, fluxiones
        invenire; et vice versa.')
    '7accd14eff7i3l9n4o4qrr4s9t7u5vx'
```

```
    >>> inventaris('ut tensio sic vis')
    'ce3ino3sttuv'
    """

    # alle letters omzetten naar kleine letters
    zin = zin.lower()

    # codering toevoegen voor elke letter in het alfabet (die in de zin voorkomt)
    codering = ''
    for letter in string.ascii_lowercase:
        # tel hoe vaak letter in de zin voorkomt
        aantal = zin.count(letter)
        # codeer letter op basis van aantal voorkomens
        codering += f'{aantal}{letter}' if aantal > 2 else letter * aantal

    return codering

def uitpakken(zin):

    """
    >>> uitpakken('6accdae13eff7i3l9n4o4qrr4s8t12ux')
    'aaaaaaccdaeeeeeeeeeeeeeeffiiiiiiiilllnnnnnnnnnooooqqqqrrsssstttttttttuuuuuuuuuuuux'
    >>> uitpakken('ce3ino3sttuv')
    'ceiiinossstuv'
    """

    getal = ''
    resultaat = ''
    for karakter in zin:
        if karakter.isdigit():
            getal += karakter
        elif getal:
            resultaat += karakter * int(getal)
            getal = ''
        else:
            resultaat += karakter

    return resultaat

def vereenvoudigen(zin):

    """
    >>> vereenvoudigen('6accdae13eff7i3l9n4o4qrr4s8t12ux')
    '7accd14eff7i3l9n4o4qrr4s8t12ux'
    >>> vereenvoudigen('i2scvotu2ietns')
    'ce3ino3sttuv'
    """

    return inventaris(uitpakken(zin))

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

5.6 Blinkenlights

```
"""
>>> cijfertekst = '28 ZsTpaumnrhsrkotDscf nameggi cit rhi reenfeie  tüd enomntpeikadrtnenbu !g
    '
>>> sleutel = 'e = 2.7182818284590452353602874713526624977572470936999595749669676277240 76'

>>> aantal_cijfers(sleutel)
70

>>> positie_cijfer(3, 2, sleutel)
24
>>> positie_cijfer(6, 3, sleutel)
```

```
37
>>> positie_cijfer(9, 42, sleutel)
-1

>>> ontsleutel(cijfertekst, sleutel)
'Das komputermaschine ist nicht für der gefingerpoken und mittengraben!'

>>> cijfertekst = ' ftLsrnUisyTelvWya les   Kahen  Eu9ocr RRll 5ochyV.!s ynle5ITiRta'
>>> sleutel = '6630295(160n897o0338533q1=884868J7Z624$fJ035W3375E13016898B4X9.74'
>>> ontsleutel(cijfertekst2, sleutel2)
'Listen very carefully. I shall say this only once!'
"""

def aantal_cijfers(tekst):

    return sum(karakter.isdigit() for karakter in tekst)

def positie_cijfer(cijfer, herhaling, tekst):

    cijfer = str(cijfer)
    positie = -1
    for _ in range(herhaling):
        positie = tekst.find(cijfer, positie + 1)
        if positie == -1:
            return -1

    return positie

def ontsleutel(cijfertekst, sleutel):

    assert len(sleutel) == len(cijfertekst), 'ongeldige sleutel'

    aantal = aantal_cijfers(sleutel)
    klare_tekst = ''
    index = 0
    while len(klare_tekst) < aantal:
        herhaling, cijfer = index // 10 + 1, index % 10
        positie = positie_cijfer(cijfer, herhaling, sleutel)
        if positie != -1:
            klare_tekst += cijfertekst[positie]
        index += 1

    return klare_tekst

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

6 Reeks 06: lijsten en tuples

6.1 ISBN

```
def isISBN(code):

    """
    Gaat na of een ISBN-10 code geldig is.

    >>> isISBN('9-9715-0210-0')
    True
    >>> isISBN('997-150-210-0')
    False
    >>> isISBN('9-9715-0210-8')
    False
    """

    # controleer of de code een string is
    if not isinstance(code, str):
        return False

    # controleer of koppeltekens op juiste plaats staan en elke groep juiste aantal cijfers
        bevat
    groepen = code.split('-')
    if [len(e) for e in groepen] != [1, 4, 4, 1]:
        return False

    # verwijder koppeltekens uit de code
    code = ''.join(groepen)

    # controleer of alle karakters (behalve het laatste) cijfers zijn
    if not code[:-1].isdigit():
        return False

    # controleer controlecijfer van code
    return controlecijfer(code) == code[-1]

def controlecijfer(code):

    """
    >>> controlecijfer('997150210')
    '0'
    >>> controlecijfer('938389293')
    '5'
    """

    # controlecijfer berekenen
    controle = sum(index * int(cijfer) for index, cijfer in enumerate(code[:9], start=1)) % 11

    # controlecijfer omzetten naar stringvoorstelling
    return 'X' if controle == 10 else str(controle)

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

7 Reeks 07: meer over functies en modules

7.1 ISBN

```
def isISBN10(code):

    """
    Gaat na of de ISBN-10 code geldig is.

    >>> isISBN10('9971502100')
    True
    >>> isISBN10('9971502108')
    False
    """

    # hulpfunctie voor berekening van ISBN-10 controlecijfer
    def controlecijfer(code):

        # controlecijfer berekenen
        controle = sum(index * int(cijfer) for index, cijfer in enumerate(code[:-1], start=1))
             % 11

        # controlecijfer omzetten naar stringvoorstelling
        return 'X' if controle == 10 else str(controle)

    return (
        isinstance(code, str)                   # code is een string
        and len(code) == 10                     # code bestaat uit 10 karakters
        and code[:-1].isdigit()                 # eerste negen karakters van de code zijn
            cijfers
        and controlecijfer(code) == code[-1]    # controlecijfer van de code is correct
    )

def isISBN13(code):

    """
    Gaat na of de ISBN-13 code geldig is.

    >>> isISBN13('9789743159664')
    True
    >>> isISBN13('9787954527409')
    False
    >>> isISBN13('8799743159665')
    False
    """

    # hulpfunctie voor berekening van ISBN-13 controlecijfer
    def controlecijfer(code):

        # controlecijfer berekenen
        controle = sum((3 if index % 2 else 1) * int(cijfer) for index, cijfer in enumerate(
            code[:-1]))

        # controlecijfer omzetten naar één enkel cijfer
        return str((10 - controle) % 10)

    return (
        isinstance(code, str)                   # code is een string
        and len(code) == 13                     # code bestaat uit 13 karakters
        and code.isdigit()                      # code bestaat enkel uit cijfers
        and controlecijfer(code) == code[-1]    # controlecijfer van de code is correct
    )

def isISBN(code, isbn13=True):

    """
    >>> isISBN('9789027439642', False)
    False
```

```
    >>> isISBN('9789027439642', True)
    True
    >>> isISBN('9789027439642')
    True
    >>> isISBN('080442957X')
    False
    >>> isISBN('080442957X', False)
    True
    """

    return isISBN13(code) if isbn13 else isISBN10(code)

def zijnISBN(codes, isbn13=None):

    """
    >>> codes = ['0012345678', '0012345679', '9971502100', '080442957X', 5, True, 'The
        Practice of Computing Using Python', '9789027439642', '5486948320146']
    >>> zijnISBN(codes)
    [False, True, True, True, False, False, False, True, False]
    >>> zijnISBN(codes, True)
    [False, False, False, False, False, False, False, True, False]
    >>> zijnISBN(codes, False)
    [False, True, True, True, False, False, False, False, False]
    """

    # lijst met controles opbouwen
    return [
        False if not isinstance(code, str)
        else isISBN(code, len(code) == 13 if isbn13 is None else isbn13)
        for code in codes
    ]

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

8 Reeks 08: verzamelingen en dictionaries

8.1 ISBN

```
def isISBN13(code):

    """
    Gaat na of de ISBN-13 code geldig is.

    >>> isISBN13('9789743159664')
    True
    >>> isISBN13('9787954527409')
    False
    >>> isISBN13('8799743159664')
    False
    """

    def controlecijfer(code):

        """
        Hulpfunctie voor de berekening van het ISBN-13 controlecijfer.
        """

        # controlecijfer berekenen
        controle = sum(
            (3 if index % 2 else 1) * int(cijfer)
            for index, cijfer in enumerate(code[:12])
        )

        # controlecijfer omzetten naar één enkel cijfer
        return str((10 - controle) % 10)

    return (
        isinstance(code, str)                   # code is een string
        and len(code) == 13                     # code bestaat uit 13 karakters
        and code[:3] in {'978', '979'}          # code heeft geldige prefix
        and code.isdigit()                      # code bestaat enkel uit cijfers
        and controlecijfer(code) == code[-1]    # controlecijfer van de code is correct
    )

def overzicht(codes):

    """
    >>> codes = [
    ...     '9789743159664', '9785301556616', '9797668174969', '9781787559554',
    ...     '9780817481461', '9785130738708', '9798810365062', '9795345206033',
    ...     '9792361848797', '9785197570819', '9786922535370', '9791978044523',
    ...     '9796357284378', '9792982208529', '9793509549576', '9787954527409',
    ...     '9797566046955', '9785239955499', '9787769276051', '9789910855708',
    ...     '9783807934891', '9788337967876', '9786509441823', '9795400240705',
    ...     '9787509152157', '9791478081103', '9780488170969', '9795755809220',
    ...     '9793546666847', '9792322242176', '9782582638543', '9795919445653',
    ...     '9796783939729', '9782384928398', '9787590220100', '9797422143460',
    ...     '9798853923096', '9784177414990', '9799562126426', '9794732912038',
    ...     '9787184435972', '9794455619207', '9794270312172', '9783811648340',
    ...     '9799376073039', '9798552650309', '9798485624965', '9780734764010',
    ...     '9783635963865', '9783246924279', '9797449285853', '9781631746260',
    ...     '9791853742292', '9781796458336', '9791260591924', '9789367398012'
    ... ]
    >>> overzicht(codes)
    Engelstalige landen: 8
    Franstalige landen: 4
    Duitstalige landen: 6
    Japan: 3
    Russischtalige landen: 7
    China: 8
    Overige landen: 11
    Fouten: 9
```

```
    """

    # histogram van groepen opbouwen
    groepen = {groep: 0 for groep in range(11)}
    for code in codes:
        groep = int(code[3]) if isISBN13(code) else 10
        groepen[groep] += 1

    # samenvatting uitschrijven
    print(f'Engelstalige landen: {groepen[0] + groepen[1]}')
    print(f'Franstalige landen: {groepen[2]}')
    print(f'Duitstalige landen: {groepen[3]}')
    print(f'Japan: {groepen[4]}')
    print(f'Russischtalige landen: {groepen[5]}')
    print(f'China: {groepen[7]}')
    print(f'Overige landen: {groepen[6] + groepen[8] + groepen[9]}')
    print(f'Fouten: {groepen[10]}')

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

9 Reeks 09: tekstbestanden

9.1 ISBN

```
import urllib.request

def isISBN13(code):

    """
    Geeft een Booleaanse waarde terug die aangeeft of de ISBN-13 code geldig is.

    >>> isISBN13('9789743159664')
    True
    >>> isISBN13('9787954527409')
    False
    >>> isISBN13('8799743159664')
    False
    """

    def controlecijfer(code):

        """
        Hulpfunctie voor de berekening van het ISBN-13 controlecijfer.
        """

        # controlecijfer berekenen
        controle = sum(
            (3 if index % 2 else 1) * int(cijfer)
            for index, cijfer in enumerate(code[:12])
        )

        # controlecijfer omzetten naar één enkel cijfer
        return str((10 - controle) % 10)

    return (
        isinstance(code, str)                   # code is een string
        and len(code) == 13                     # code bestaat uit 13 karakters
        and code[:3] in {'978', '979'}          # code heeft geldige prefix
        and code.isdigit()                      # code bestaat enkel uit cijfers
        and controlecijfer(code) == code[-1]    # controlecijfer van de code is correct
    )

def verwijder_tags(s):

    """
    Verwijdert alle XML-tags uit de string en verwijdert daarna witruimte vooraan en achteraan
         de string.
    OPMERKING: er bestaan hiervoor ook gespecialiseerde modules zoals BeautifulSoup


    >>> verwijder_tags(' <Title> The Practice of Computing using <b>Python</b> </Title> ')
    'The Practice of Computing using Python'
    """

    # verwijder alle XML-tags uit de string s
    begin = s.find('<')
    while s.find('<') >= 0:
        einde = s.find('>', begin + 1)
        if einde == -1:
            einde = len(s)
        s = s[:begin] + s[einde + 1:]
        begin = s.find('<')

    # verwijder witruimte vooraan en achteraan en geef gewijzigde string terug
    return s.strip()

def print_boek_info(code):
```

```
    """
    >>> print_boek_info('9780136110675')
    Titel: The Practice of Computing using Python
    Auteurs: William F Punch, Richard Enbody
    Uitgever: Addison Wesley
    >>> print_boek_info('9780136110678')
    Foutieve ISBN-13 code
    """

    # eventuele witruimte rond code verwijderen
    code = code.strip()

    # geldigheid van ISBN-13 code nagaan
    if not isISBN13(code):

        # foutboodschap uitschrijven als de gegeven ISBN-13 code ongeldig is
        print('Foutieve ISBN-13 code')

    else:

        # URL opbouwen van nagebootste ISBNdb.com die informatie oplevert over boek met
            gegeven ISBN-13 code
        url = f'http://pythia.ugent.be/pythia-share/exercises/isbn9/books.php?isbn={code}'

        # informatie over boek uit antwoord filteren en uitschrijven
        with urllib.request.urlopen(url) as info:
            for regel in info:
                regel = regel.decode('utf-8')
                if regel.startswith('<Title>'):
                    print(f'Titel: {verwijder_tags(regel)}')
                elif regel.startswith('<AuthorsText>'):
                    print(f'Auteurs: {verwijder_tags(regel).rstrip(", ")}')
                elif regel.startswith('<PublisherText '):
                    print(f'Uitgever: {verwijder_tags(regel).rstrip(", ")}')

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```

10 Reeks 10: objectgericht programmeren

10.1 ISBN

```
class ISBN13:

    """
    >>> code = ISBN13(9780136110675)
    >>> print(code)
    978-0-13611067-5
    >>> code
    ISBN13(9780136110675, 1)
    >>> code.isgeldig()
    True
    >>> code.alsISBN10()
    '0-13611067-3'

    >>> ISBN13(9780136110675, 6)
    Traceback (most recent call last):
    AssertionError: ongeldige ISBN code
    """

    def __init__(self, code, lengte=1):

        # controleer geldigheid van argumenten
        assert (
            isinstance(code, int)        # code is een integer
            and isinstance(lengte, int)  # lengte is een integer
            and 1 <= lengte <= 5         # lengte behoort tot interval [1, 5]
        ), 'ongeldige ISBN code'

        # objecteigenschappen: ISBN-code en lengte van landaanduiding omzetten naar string van
             13 karakters met
        # voorloopnullen
        self.code = f'{code:013d}'
        self.lengte = lengte

    def __int__(self):

        return int(self.code)

    def __str__(self):

        # opgemaakte versie van ISBN-code teruggeven
        c = self.code
        return f'{c[:3]}-{c[3:3 + self.lengte]}-{c[3 + self.lengte:-1]}-{c[-1]}'

    def __repr__(self):

        # string teruggeven met Python expressie die nieuw object aanmaakt met dezelfde
            toestand als het huidige object
        return f'ISBN13({int(self)}, {self.lengte})'

    def isgeldig(self):

        def controlecijfer(code):

            # ISBN-13 controlecijfer berekenen
            controle = sum(
                (3 if index % 2 else 1) * int(cijfer)
                for index, cijfer in enumerate(code[:12])
            )

            # controlecijfer omzetten naar string
            return str((10 - controle) % 10)

        # nagaan of controlecijfer geldig is
        return self.code[12] == controlecijfer(self.code)
```

```
    def alsISBN10(self):

        def controlecijfer(code):

            # ISBN-10 controlecijfer berekenen
            controle = sum(
                index * int(cijfer)
                for index, cijfer in enumerate(code[:9], start=1)
            ) % 11

            # controlecijfer omzetten naar string
            return 'X' if controle == 10 else str(controle)

        # geen resultaat teruggeven voor ongeldige ISBN-13 codes
        if not self.isgeldig() or not self.code.startswith('978'):
            return None

        # ISBN-13 code omzetten naar ISBN-10 code
        code = self.code[3:-1]
        controle = controlecijfer(code)
        return f'{code[:self.lengte]}-{code[self.lengte:]}-{controle}'

if __name__ == '__main__':
    import doctest
    doctest.testmod()
```