leven-en-laten-sterven.py

Status:

Punten:

```python
def paren(element):
    '''
    >>> paren('Neon')
    ('Ne', 'Eo', 'On')
    >>> paren('Rutherfordium')
    ('Ru', 'Ut', 'Th', 'He', 'Er', 'Rf', 'Fo', 'Or', 'Rd', 'Di', 'Iu', 'Um')
    '''
    #opbouwen van een tuple en returnen
    return tuple(e.upper() + element[i+1].lower() for i, e in enumerate(element[:-1]))

def eerste(reeks, container):
    '''
    >>> eerste(['Ne', 'Eo', 'On'], {'Ne', 'On'})
    'Eo'
    >>> eerste(('Ne', 'Eo', 'On'), {'Ne', 'On', 'Eo'})
    '''
    #vanaf je een element vind dat niet in de container zit, return deze dan
    for e in reeks:
        if e not in container:
            return e

def toekennen(bestand, onmogelijk=None):
    '''
    >>> symbool = toekennen('elementen.txt', onmogelijk='???')
    >>> symbool['Neon']
    'Ne'
    >>> symbool['Rubidium']
    'Ru'
    >>> symbool['Ruthenium']
    'Ut'
    >>> symbool['Rutherfordium']
    'Rf'
    >>> symbool['Tin']
    '???'
    '''
    #een lege dict en een lege verz voor de toegekende symbolen aanmaken
    d = {}
    al_toegekend = set()

    bestand = open(bestand, 'r')

    #overloop alle elementen in het bestand
    for element in bestand:
        element = element.rstrip() #newlines en spaties weg aan de rechterkant
        #kies een symbool voor het element  met de functie 'eerste'
        symbool = eerste(paren(element), al_toegekend)
        if symbool: #als de functie 'eerste' niet None geeft
            al_toegekend.add(symbool)
            d[element] = symbool
        else:
            d[element] = onmogelijk

    return d

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

```

toegepaste-scheikunde.py

Status:

Punten:

```python
def isGeldig(symbool, element, lengte=None):
    '''
    >>> isGeldig('Zer', 'Zeddemorium')
    True
    >>> isGeldig('Zer', 'Zeddemorium', 2)
    False
    >>> isGeldig('di', 'Zeddemorium')
    False
    '''
    #Checken van criterium 1
    if not symbool[0].isupper() or (len(symbool) != 1 and not symbool[1:].islower()):
        return False

```

```
    #Checken van criterium 2
    element = element.lower()
    for s in symbool.lower(): #overloop de letters in het symbool
        index = element.find(s)
        if index == -1: #als find heeft gefaald (geeft dan -1 terug)
            return False
        element = element[index+1:] #verkort de string iedere keer na de gevonden letter

    #evt checken van criterium 3
    return len(symbool) == lengte if lengte else True


def symbolen(element):
    '''
    >>> symbolen('Iron') == {'Ir', 'Io', 'In', 'Ro', 'Rn', 'On'}
    True
    >>> symbolen('Neon') =={'Eo', 'Nn', 'No', 'En', 'Ne', 'On'}
    True
    '''
    v = set()
    #neem een letter in het element(behalve de laatste)
    for i, e in enumerate(element[:-1]):
        #overloop de rest van de letters na die letter
        for e2 in element[i+1:]:
            #voeg steeds een combinatie toe aan de verzameling
            v.add(e.upper() + e2.lower())
    return v

def voorkeur(element, laatste=False):
    '''
    >>> voorkeur('Iron')
    'In'
    >>> voorkeur('Iron', laatste=True)
    'Ro'
    >>> voorkeur('Neon')
    'En'
    >>> voorkeur('Neon', True)
    'On'
    '''
    #maak een alfabetisch gesorteerde lijst met mogelijke symbolen
    symbolen_list = sorted(list(symbolen(element)))

    #kies de eerste of de laatste
    return symbolen_list[-1] if laatste else symbolen_list[0]

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