Samenvatting programmeren evaluatie 2

Leerstof:
Boek, samenvatting, extra info oefeningen, oefeningen, les, nota's les, oefeningen gemaakt in les

HOOFDSTUK 7: Lijsten en Tuples

Eigenschappen over tuples en lijsten:
- lijsten zijn mutable, tuples niet (zoals strings)
- lijst -> lijst = list() of lijst = [] = False
-tuples -> tuple(lijst)
- lijsten kunnen allerlei types bevatten
- lijsten worden gescheiden door komma's in een string wordt elke letter apart in lijst gezet (moet itereerbaar zijn)

LIJSTEN
Slicen met lijsten:
- werkt hetzelfde als strings (ook for loop)
> lijst = [5,4,[6,3]]
> lijst[0]
>> 5
> lijst[2][0]
>> 6
> [1,2,3,4,5][2]
>> 3

Operaties op lijsten:
- + : enkel 2 lijsten kunnen plus elkaar gedaan worden
*- * : enkel een lijst met een int kunnen vermenigvuldigt worden*
- ook zip werkt zelfde bij lijsten als strings
- je kan lijsten vergelijken met elkaar:
> [1,2,3,4] < [1,2,3,4,0]
> [1,2,3,0] < [1,2,3,4]
> [1,2,'one','two'] < [3,4,5,6] -> er wordt gekeken naar het eerste getal
> [3,2,3,4] > [1,2,3,4,0]

Functies:
> len(list) -> lengte lijst
> min(list) -> minimum uit lijst (bij letters, kijkt het naar de ord)
> max(list) -> maximum uit lijst
> sum(list) -> som van alle elementen uit de lijst

Voorbeelden van mutability list:
> my_list = [1, 2, 'a', 'z']
> my_list[0] = True
>> [True, 2, 'a', 'z']
> my _list[:2] = [27]
>> [27, 'a', 7]
> my_list[:] = [1,2,3,4]
>> [1, 2, 3, 4]
> a,b,c,d = 'spam' -> a = 's'
*> a, *b, c = 'programmeren' -> b = 'rogrammere'*

Methoden die lijst niet gaan veranderen: (voorbeeldgebruik: lijst.index('a'))
> index(element_lijst) -> gaat de index geven van dat element uit lijst
> count(element_lijst) -> gaat tellen hoeveel keer element voorkomt in lijst

Methoden die lijst wel gaan veranderen:
> append(element) *-> element wordt toegevoegd aan einde lijst*
> pop() *-> haalt laatste element uit lijst en returned die*
> extend(lijst2) *-> lijst2 wordt toegevoegd aan lijst1*
> insert(i, element) *-> gaat element toevoegen in lijst op positie i*
> remove(element) *-> verwijdert element*
> sort() *-> lijst met zelfde types sorteren (OPLETTEN: verschil met sorted())*
> reverse() *-> gaat de posities van de elementen omdraaien*
> del lijst[1:2] *-> gaat stukje uit lijst deleten*

De split methode:
> zin.split() *-> zal automatisch splitsen op spaties, in haakjes specifiëren*
>a,b,c = 'Wortels zijn lekker'.split()
>a
>> 'Wortels'

Van string naar list (join method)
>string = ' '.join(lijst) *-> zal elk element uit lijst met elkaar verbinden via ' '*

Sorted functie
- gaat elementen uit lijsten sorteren + kan op strings gebruikt worden (sort alleen list)
> sorted('Hi mom')
>> [' ', 'H', 'i', 'm', 'm', 'o']
- sort is mutable, sorted is immutable (variabele gebruiken)
- sorted kanje ook reversen via reverse = True

Alle elementen in een lijst van datatype veranderen:
> lijst = ['1','2','3'] -> list(map(int, lijst)) = [1,2,3]

Controleren of alle elementen van een lijst aan een bepaalde voorwaarde voldoen:
> if all(b = Truefor b in lijst)

Controleren of minstens één element van een lisjt aan een voorwaarde voldoet:
> if any(b = Truefor b in lijst)

Pythonic tips:
- gelijkenis in return statement kan ook (return a == b) voor True of False returns

Value ERROR:
- try en except kan je proberen (vb in python_book p 333)

```
def decodeer(codes: str, c2k: dict) -> str:

    try:
        return ''.join(c2k[code] for code in codes.split())
    except KeyError as error:
        raise ValueError('ongeldige gecodeerde tekst') from error
```

```
if code not in dict_hippo:
    raise ValueError('ongeldige gecodeerde tekst')
```

Assertion ERROR:
- opmaak: assert (voorwaarde dat moet waar zijn, anders error), 'zin'
-voorbeeld (wanneer a niet gelijk is aan b, moet er komen 'fout'):
> assert a==b, 'fout'

Oppassen met mutable van lijsten:
> a = [1,2,3]
> b = a
> a.append(4)
>b
>> [1,2,3,5]

Standaardwaarde voor lijsten

```
>>> [' '] * 3
[' ', ' ', ' ']
>>> [1] * 5
[1, 1, 1, 1, 1]
```

Hoe mutable van lijsten overbruggen:

> b = a[:] -> kopie van a, nu zal b niet mee veranderen
- import copy
> b = copy.deepcopy(a)

Circulaire lijsten

> (i + 1) % len(lijst)

Functies en lijsten:

> woorden = ['Peer', 'appel', 'Banaan', 'pruim']
> sorted(woorden)
>> [Banaan, Peer, appel, pruim]
> sorted(woorden, key=sorteercriterium) -> sorteercriterium is functie die elk woord lower gaat returnen
>> [appel, Banaan, Peer, pruim]
> sorted(woorden, key = lambda woord:woord.lower())
>> [appel, Banaan, Peer, pruim]
> sorted(woorden, key = lambda woord: woord[1])
>> Banaan, Peer, appel , pruim -> sorteren op tweede letter
> sorted(woorden, key=lambda woord: woord[::-1]
>> appel, pruim, Banaan, Peer -> sorteren op laatste letter

TUPLES

Manieren om tuples te maken:

> a = 2,3
> b= (1,)
> x, y = 'a', 'b'
>> x,y -> ('a', 'b')
> tuple += (a,) -> element a toevoegen

Functies die horen bij tuples:

*- + en **
- slicing, in en for
- len, min, max, >, <, sum

Van lijst naar tuple en omgekeerd

tuple = tuple(lijst)
lijst = list(tuple)

Lijsten groeperen:

```
>>> lijst = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
>>> zip(lijst[::2], lijst[1::2])
[('a', 'b'), ('c', 'd'), ('e', 'f'), ('g', 'h')]
```

```
>>> lijst = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']
>>> [(lijst[i], lijst[i + 1]) for i in range(0, len(lijst), 2)]
[('a', 'b'), ('c', 'd'), ('e', 'f'), ('g', 'h')]
```

LIST COMPREHENSION

> [i for i in range(20) if i%2 == 0]
*> [(i, i**2, i**3) for i in range(20) if i%2 == 0]S*
> [v for v in 'solidarity' if v in 'aeiou']
> [(x,y) for x in range(3) for y in range(4)] -> 12 tuples in 1 lijst
> [(x,y) for x in range(3) for y in range(4) if x>y and x%2 == 0] -> 2 tuples in een lijst
> [int(c) for c in some_string if c.isdigit()] -> uit een zin alle getallen in lijst zetten
> [''.join(delen) for delen in zip((verdelen(woord, groepen) for woord in woorden))] -> sterretje gaat elke element uit lijst apart zetten*

> [index for index, az in enumerate(eiwit.upper()) if az == aminozuur]
> voorbeeld oefeningen hebben veel list comprehension

Andere voorbeelden
> 'Hi Mommy' if age < 10 else 'Hi Mother'
*> [i**2 if i%2 else i**3 for i in range(10)]*
> ''.join(nucleotide[(x2 - x1, y2 - y1)] for (x1, y1), (x2, y2) in zip(vector, vector[1:])
> getal1 < getal2 if teken == '<' else getal1 > getal2
> getallen.pop(0) if teken == '<' else getallen.pop() for teken in tekens] + getallen
> Opgeteld = [getal1 + getal2] for getal1, getal2 in zip(lijst1, lijst)]

HOOFDSTUK 8: Functies en modules

Functies met default parameters
> def functi(param_required, param_default = 2, param2_default = 4)
- je kan ook functie hebben met alleen maar default parameters
> functi(param2_default = 3) -> deze parameter specifiek veranderen
- nooit aan een default parameter een mutable gegeven meegeven (zoals lijst)

Meer te weten komen over een functie
> import math
>> dir(math) -> alle functies
>math.ceil.__doc__ -> uitleg over functie

Als je zelf een docstring wilt maken voor een functie:
- functie aanmaken en eerste lijntje : '''uitleg'''

Functies met meerdere argumenten
*> def som(*termen*)

MODULE RANDOM
> random.random() -> kiest willekeurig getal tussen 0 en 1
> random.randint(min, max) -> kiest willekeurig reëel getal tussen min en max
> random.choice(lijst) -> kiest een willekeurig element uit deze lijst
> random.sample(lijst, n) -> kiest n willekeurige elementen uit de lijst
> random.shuffle(lijst) -> schudt de elementen van de lijst willekeurig door elkaar

MODULE DATETIME
> datetime.datetime(jaar, maand, dag)
> datetime.timedelta(dagen, [uren, [minuten, [seconden, […]]]]) berekent tijdsverschil tussen twee tijdstippen
> datetime.datetime.time -> uur van de dag

MODULE FRACTIONS
> teller = breuk.numerator
> noemer = breuk.denominator

```
>>> from datetime import date
>>> geboortedatum = date(1990, 10, 3)
>>> geboortedatum = date(day=3, month=10, year=1990)
>>> geboortedatum.day          # day is een eigenschap
3
>>> geboortedatum.month        # month is een eigenschap
10
>>> geboortedatum.year         # year is een eigenschap
1990
>>> geboortedatum.weekday()    # weekday is een methode !!
2
>>> vandaag = date.today()
>>> vandaag
datetime.date(2015, 11, 10)    # uitgevoerd op 11 oktober 2015
>>> from datetime import timedelta
>>> morgen = vandaag + timedelta(1)
>>> morgen
datetime.date(2015, 11, 11)
>>> verschil = morgen - vandaag
>>> type(verschil)
datetime.timedelta
>>> verschil.days
1
```

Nog ppt slides kijken en demo's

HOOFDSTUK 9: Sets en dictionaries

DICTIONARIE

Basics van een dictionarie:

- mutable
> a = dict() -> dict maken
> a = {} -> dict maken
> a_dict = {1:'a', 2: 'b', 3:'c'}
> a_dict[1] -> 'a'
>a_dict.get(1) -> 'a'
> a_dict[1] = 'd' -> a_dict = {1:'d', 2: 'b', 3:'c'}
- ingeneste dictionaries lijken op ingeneste lijsten
- wanneer je een dict aan functie meegeeft, hoef je niet per se een return statement te gebruiken, als je in de functie de dict aanpast
- geen ordening

Operators:

> len() -> lengte van key/value paren
- in -> is de key in de dict
- for -> gaat door keys itereren

Dict methodes:

> items() -> alle key/value paren in tuples printen (zo kunje bv: for key, val in dict.items())
> keys() -> alle keys in dict
> values() -> alle values in dict
> copy() -> copy maken van dict
> update({key: value}) -> dict updaten
> del dict[key] -> key en value uit dict verwijderen
> clear() -> dictionary leeg maken
> get(element, waarde) -> geeft element terug, indien dit niet in dict zit, geeft het waarde terug

Basic dict maken:

```
count_dict = {}
for word in word_list:
    if word in count_dict:
        count_dict[word] += 1
    else:
        count_dict[word] = 1
```

```
count_dict = {}
for word in word_list:
    try:
        count_dict[word] += 1
    except KeyError:
        count_dict[word] = 1
```

```
count_dict = {}
for word in word_list:
    count_dict[word] = count_dict.get(word,0) + 1
```

CSV files:

> import csv
> reader = csv.reader(periodic_file)

Dictionary geordend uitschrijven:

> for element in sorted(list(dict)):
> print(element, dict[element])

Dictionary omkeren:

```
# dictionary omkeren
omgekeerd = {}
for sleutel, waarde in observaties.items():
    vogels = omgekeerd.get(waarde, [])
    vogels.append(sleutel)
    omgekeerd[waarde] = vogels

# dictionary geordend uitschrijven
for aantal in sorted(omgekeerd):
    print(aantal, omgekeerd[aantal])
```

SETS

Basics van een set

- mutable
> a = set() -> set maken

>set('abcd') -> {'a','b','c','d'}
- zal alle dubbele waarden negeren

Operaties:
> len() -> aantal elementen in set
> in/for

Methoden:
> a_set & b_set of a_set.intersection(b_set) -> nieuwe set waar gezamenlijke getallen inzitten
> a_set | b_set of a_set.union(b_set) -> nieuwe set waar alle elementen in zitten
> a_set – b_set of a_set.difference(b_set) -> elementen van a die niet in b zitten
> b_set – a_set of b_set.difference(a_set) -> elementen van b die niet in a zitten
> a_set ^ b_set of a_set.symmetric_difference(b_set) -> nieuwe set waar elementen zitten die niet gezamenlijk in a en b zitten
> small_set <= big_set of small_set.issubset('abcdef') -> gaat boolean geven of de subset kleiner is dan superset
> small_set >= big_set of small_set.issuperset('abcdef') -> gaat boolean geven of de small set een superset is

Andere methodes:
> add(element) -> element toevoegen aan set
> clear() -> verwijdert elk element uit set
> remove(element) -> zal element verwijderen uit set, zal error geven wanneer element niet aanwezig is
> discord(element) -> zal element verwijderen uit set, zal geen error geven
> copy() -> gaat set kopiëren

Local/global namespace
- wanneer je in een functie een variabele wilt aanroepen die buiten de functie is, doe je:
> global mijn_variabele

Zip gebruiken voor dictionaries aan te maken
> dict = dict(zip(keys, values)) -

Dictionary en set comprehension
> a dict = {k:v for k,v in enumerate('abcdefg')}
> b dict = {v:k for k,v in a dict.items()}
> [(v,k) for v,k in b dict.items()]
> a set = {ch for ch in 'to be or not to be'}

Keys en values omwisselen:
> for key, value in dictionary.items(): key, value = value, key

De sleutel vinden voor gegeven waarde:
> key = list(d.keys())[list(d.values()).index(waarde)]

Voorbeeldoefeningen, specifieke tips en tricks, demo's, nota's les

HOOFDSTUK 10: Tekstbestanden

Files openen:
> invoer = open(file_name, 'r') -> bestandsobject aanmaken, r staat voor om te lezen
> inhoud = invoer.read() -> leest volledige inhoud van bestand in een string, inhoud bevat nu een kopie van alle bytes uit bestand

> invoer.close() -> object loskoppelen van bestand
> len(inhoud) -> geeft aan hoeveel bytes er zijn in het bestand (dus ook spaties)
> inhoud = invoer.read(64) -> zal de eerste 64 bytes lezen uit invoer
> regel = invoer.readline() -> eerste regel van bestand
> regel = invoer.readlines() -> alle regels in 1 keer uitlezen en in lijst plaatsen

- Wanneer tekstbestand niet te groot is, kunje alles uitlezen via read en elke regel overlopen via for lus

```
invoer = open('haiku.txt', 'r')
bytes, regels = 0, 0
for regel in invoer:
    regels += 1
    bytes += len(regel)
invoer.close()
print(f'gemiddelde: {bytes / regels}')
```

df

SPLIT verwijdert ook newline wanneer je geen element meegeeft

```
with open(bestand, 'r', encoding='utf-8') as data:

# bestand verwerken als CSV-bestand
csv = reader(data, delimiter=',')


# hoofding overslaan
next(csv)
```

```
woordwaarde_int += letterwaarden_dict.get(karakter)
```

```
[dict_letters_waarden[karakter] for karakter in woord]
```

```
aanwijzingen_list.append(f'{len(stuk)}{stuk[0]}')
```

```
eindlijst.append(str(len(eindletter)) + eindletter[0])
```

```
f blokkentoren(woord):
  delen_alfabet = sorted(fragmenten(woord), key=len)
  return tuple(delen_alfabet)
```

```
def rangschik(lijst_namen, locatie, veld=0, dalend=False):
    getal_n = veld
    if not dalend:
        lijst_namen.sort()
    else:
        lijst_namen.sort(reverse=True)
    waardes_bij_namen = waarden(lijst_namen, locatie, veld=getal_n)
    dict_namen = {woord: waarde for waarde, woord in zip(waardes_bij_namen, lijst_namen)}
    if not dalend:
        lijst_namen.sort(key=lambda waarde: dict_namen[waarde])
        # hij gaat elke naam van lijst namen afbeelden op waarde, waarde=naam
    else:
        lijst_namen.sort(key=lambda waarde: dict_namen[waarde], reverse=True)
```

```
else:
    gesorteerde_lijst_namen = sorted(collectie_namen, reverse=True)
waardes_bij_namen = waarden(gesorteerde_lijst_namen, locatie, veld=veld)
lijst_voor_sorteren = [(woord, waarde) for waarde, woord in zip(waardes_bij_namen, gesorteerde_lijst_namen)]
if not dalend:
    gesorteerd = sorted(lijst_voor_sorteren, key=lambda waarde: waarde[1])
else:
    gesorteerd = sorted(lijst_voor_sorteren, key=lambda waarde: waarde[1], reverse=True)
eindlijst =[karakters for karakters, cijfers in gesorteerd]
return eindlijst
```

Lipogram -> oef vr et uitsschijven vn file