

WEEK 2**ONDERWERPEN**

- tuple
- dictionaries
- sets
- functions
- scope
- comprehension
- modules and packages

OPGAVE 1 : TUPLES

- Maak twee tuples `tup1` en `tup2` en druk ze af. Het resultaat is `(1, 2, 3, 4, 5)` en `(5,)`.
- Tuple uitpakken. Gegeven een tuple `tup = ('xx', 'yy', 'zz')`. Toon als resultaat `yy`.
- Waarden toevoegen. Gegeven een tuple `tup1 = (4, 6, 2, 8, 3, 1)`. Toon als resultaat `(4, 6, 100, 2, 8, 3, 1)`.
- Van tuple naar string. Gegeven een tuple `tup = ('a', 'b', 'c')`. Toon als resultaat `'abc'`.
- Van list naar tuple. Gegeven een list `L = [5, 10, 7]`. Toon als resultaat een tuple `(5, 10, 7)`.
- Een lijst van tuples naar individuele lijsten (via `zip`). Gegeven een lijst `L = [(1,2), (3,4), (8,9)]`. Toon een lijst met twee tuples `[(1, 3, 8), (2, 4, 9)]`.
- Van een lijst van tuples naar een dictionary. Gegeven een lijst `L = [("x", 1), ("x", 2), ("x", 3), ("y", 1), ("y", 2), ("y", 3)]`. Toon een dictionary `{ 'x': [1, 2, 3], 'y': [1, 2, 3], 'z': [1] }`.

OPGAVE 2 : DICTIONARY UPDATES

Gegeven de dictionary `d = {"red": 4, "blue": 1, "green": 14, "yellow": 2}`. Wat is de waarde van `d` na de volgende statements (waarbij `d` na elke statement weer is als aan het begin).

- `d['red'] = d['blue']`
- `d['blue'] += 10`
- `d['yellow'] = len(d)`
- `d['green'] = {'orange': 6}`
- `d = dict.fromkeys(d, 0)`
- `d.pop('black', None)`
- `d.get('black', None)`
- `d.setdefault('black', None)`
- `d = {}`

OPGAVE 3 : DICTIONARIES

- a) Sorteren van een dictionary.

Gegeven de dictionary D = {'c':1, 'b':2, 'a':3, 'e':1, 'd':3}. Toon als resultaat:

a,3

b,2

c,1

d,3

- b) Alleen items met unieke waarden.

Gegeven de dictionary D = {'a':1, 'b':2, 'c':3, 'd':1, 'e':3, 'f': 5}

Toon als resultaat {'b': 2, 'f': 5}

Tip: het gebruik van een Counter is handig

- c) Unieke waarden in een lijst van dictionaries (via een set).

Gegeven de lijst L = [{"V": "S001"}, {"V": "S002"}, {"VI": "S001"}, {"VI": "S005"}, {"VII": "S005"}, {"V": "S009"}, {"VIII": "S007"}]

Toon als resultaat { 'S001', 'S005', 'S009', 'S007', 'S002' }

- d) Tel waarden met zelfde key bij elkaar op (via Counter).

Gegeven de lijst L = [{'a':1, 'b':2, 'c':3}, {'a':5, 'b':4, 'c':2}]

Toon als resultaat: Counter({'a': 6, 'b': 6, 'c': 5})

- e) Twee lijsten samenvoegen naar een dictionary (via zip).

Gegeven de lijsten keys = ['red', 'green', 'blue'] en values = ['#FF0000', '#008000', '#0000FF'].

Toon als resultaat: {'green': '#008000', 'blue': '#0000FF', 'red': '#FF0000'}

OPGAVE 4 : SETS

- a) De doorsnede van twee sets.

$s1 = \{1, 4, 5, 6\}$ en $s2 = \{1, 3, 6, 7\}$

Resultaat: $\{1, 6\}$

- b) Het symmetrisch verschil van twee sets.

$s1 = \{1, 4, 5, 6\}$ en $s2 = \{1, 3, 6, 7\}$

Resultaat: $\{3, 4, 5, 7\}$

- c) De elementen in grote lijst vinden (via set).

$L = [1, 7, 4, 8, 9, 9, 4, 1, 4, 11, 14, 21, 15, 5, 2, 5]$

Zitten 15 en 11 in de lijst?

Resultaat: True

OPGAVE 5 : COMPREHENSION

- a) Schrijf de onderstaande functie opnieuw, waarbij je list comprehension gebruikt in plaats van de for-lus.

```
def capitalize_all(t):
    res = []
    for s in t:
        res.append(s.capitalize())
    return res
```

- b) Schrijf de onderstaande functie opnieuw, waarbij je list comprehension gebruikt in plaats van de for-lus.

```
def only_upper(t):
    res = []
    for s in t:
        if s.isupper():
            res.append(s)
    return res
```

OPGAVE 6 : FUNCTIES (DEF OF LAMBDA)

- a) Schrijf een functie `unique_list(L)`, die teruggeeft een lijst waarin alle element uit L precies één keer voorkomen.

Voorbeeld: $L = [1, 2, 3, 3, 3, 3, 4, 5]$ dan geeft `print(unique_list(L))` als resultaat: $[1, 2, 3, 4, 5]$.

- b) Schrijf een functie `even_elements(L)`, die teruggeeft een lijst met alle even elementen van L .

Voorbeeld: $L = [1, 2, 3, 4, 5, 6, 7, 8, 9]$ dan geeft `print(even_elements(L))` als resultaat: $[2, 4, 6, 8]$.

- c) Schrijf een functie `is_pangram(str)` die teruggeeft of de string `str` een pangram is. Een pangram is een woord of zin dat elke letter van het alfabet (tenminste een keer) bevat.

Voorbeeld: `str= "Filmquiz bracht knappe ex-yogi van de wijs"`. `Print(is_pangram(str))` geeft `True`.

Tip : `string.ascii_lowercase` geeft een string met alle letters van het alfabet.

- d) Schrijf een functie `sd(dict)` die teruggeeft een gesorteerde lijst met paren (key, value).

Voorbeeld: `dict = {'ed': 5, 'carl':3, 'alan':1, 'bob':2, 'dan':4}`

`print(sd(dict))` geeft als resultaat `[('alan', 1), ('bob', 2), ('carl', 3), ('dan', 4), ('ed', 5)]`

Dit kan je eenvoudig oplossen via `sorted(dict.items())`, maar doe dit ook door een ananieme functie mee te geven, dus met `key=lambda`.

OPGAVE 7 : MODULES A,B EN C

Bij het uitvoeren van `main.py` blijkt dat er een kleine fout is. Wat is deze fout ?

<pre>#main.py import moda from modb import * from modc import x,y f1('hi') f2() print(y) z=Simple() z.display()</pre>	<pre>#moda.py class Simple: x = 77 def display(self): print(self.x)</pre>
<pre>#modb.py import modc def f1(m): print(m) def f2(): x = 88 x = modc.inc(x) print(x)</pre>	<pre>#modc.py x = 88 y = 99 def inc(i): i += 1 return(i)</pre>

OPGAVE 8 : ZEESLAG

Schrijf een eenvoudige versie van het spel Zeeslag (Battleship), waarbij tegen de computer wordt gespeeld. Een deel van het programma is al gegeven.

```
from random import randint
BOARD_SIZE = 4
NR_GUESSES = 4

#initializing board
board = []

for x in range(BOARD_SIZE):
    board.append(["O"] * BOARD_SIZE)

def print_board(board):
    for row in board:
        print (" ".join(row))

#start the game and printing the board
print ("Let's play Battleship!")
print_board(board)

#define where the ship is
ship_row = randint(0, BOARD_SIZE-1)
ship_col = randint(0, BOARD_SIZE-1)

"""
    here your code :
    -ask the user for a guess
    -if the user's right, the game ends
    -warn if the guess is out of the board
    -warn if the guess was already made
    -if the guess is wrong, mark the point with an X and start again
    -print turn and board again here
"""

if turn == NR_GUESSES-1:
    print ("Game Over")
```

OPGAVE 9 : PERFECTE GETALLEN

Een perfect getal is een positief geheel getal dat de som is al van zijn eigen delers. Bijvoorbeeld $6 = 3 + 2 + 1$ en daarom een perfect getal. En 28 is ook een perfect getal. Schrijf een programma dat de 4 perfecte getallen vindt onder de 10.000.

OPGAVE 10 : TEL DE LETTERS

In deze opgave maak je een programma dat de frequentie van elke letter in een tekstbestand weergeeft in een histogram. Op Blackboard is de file "count_letters_histogram.py" te vinden. Dit programma is al bijna af. Waar staat # ... moet nog jouw code komen (dit kan met 7 regels). Op Blackboard is ook een bestand astronaut.txt te vinden dat je kan gebruiken als input.