WEEK 3

ONDERWERPEN

- any & all
- range, zip & enumerate
- higher-order functions
- classes and OOP
- exceptions
- assert
- file access
- working with CSV and JSON

OPGAVE 1 : WAT WORDT ER AFGEDRUKT ?

Gegeven de volgend programma's, wat wordt er afgedrukt ?

(a)

```
class A:
    def __init__(self, i = 0):
        self.i = i

    def m1(self):
        self.i += 1

class B(A):
    def __init__(self, j = 0):
        super().__init__(3)
        self.j = j

    def m1(self):
        self.i += 1

def main():
    b = B()
    print(b.i, b.j)
    b.m1()
    print(b.i, b.j)

main()
```

(b)

```
class A:
    def __init__(self, i):
        self.i = i

    def __str__(self):
        return "I am class A"

class B(A):
    def __init__(self, i, j):
        super().__init__(i)
        self.j = j
    def __str__(self):
        return "I am class B"
```

```
def main():
    a = A(1)
    b = B(1, 2)
    print(a)
    print(b)
    print(a.i)
    print(b.i, b.j)

main()
```

OPGAVE 2 : CLASS DIAGRAM

Teken een class diagram voor onderstaande code.

```
1  class PingPongParent:
2      pass
3
4  class Ping(PingPongParent):
5      def __init__(self, pong):
6          self.pong = pong
7
8  class Pong(PingPongParent):
9      def __init__(self, pings=None):
10         if pings is None:
11             self.pings = []
12         else:
13             self.pings = pings
14
15     def add_ping(self, ping):
16         self.pings.append(ping)
17
18 pong = Pong()
19 ping = Ping(pong)
20 pong.add_ping(ping)
```

OPGAVE 3 : IMPLEMENTEER DE CLASS CIRCLE

Implementeer de klasse Circle met de volgende methoden:

- constructor: definieert de straal
- area(): geeft oppervlakte van de cirkel
- perimeter(): geeft de omtrek van de cirkel

Test met onderstaande code:

```
NewCircle = Circle(8)
assert NewCircle.area()== 200.96
assert NewCircle.perimeter() == 50.24
```

OPGAVE 4 : IMPLEMENTEER DE CLASS ROMAN

Schrijf een klasse met (voorlopig) maar een class method, roman_to_int(cls, s), die een Romeins getal s omzet naar een integer. Kijk voor de regels eventueel op https://nl.wikipedia.org/wiki/Romeinse_cijfers. Je mag aannemen dat van een symbool hooguit één symbool wordt afgetrokken, dus 8 is niet niet IIX maar VIII.

Gebruik hiervoor de volgende dictionary: rom_val = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

Test met onderstaande code:

```
assert Roman.roman_to_int('C') == 100
assert Roman.roman_to_int('C') == 100
assert Roman.roman_to_int('XLIX') == 49
assert Roman.roman_to_int('MMMCMLXXXVI') == 3986
```

OPGAVE 5 : IMPLEMENTEER DE CLASS STACK

De Stack class maakt het mogelijk om objecten in een stapel te bewaren en biedt operaties om de stack te bewerken. De class Stack heeft de volgende attributen:

field	elements: list (private)	list to store the elements in the stack
method	constructor	creates an empty stack
method	is_empty()	returns True if the stack is empty
method	peek()	returns the element at the top of the stack without removing it from the stack
method	push(value: object)	stores an element at the top of the stack
method	pop()	removes the element at the top of the stack and returns it
method	getSize()	returns the number of elements in the stack

Het volgende testprogramma is gegeven :

```
from stack import Stack

stack = Stack()

for i in range(10):
    stack.push(i)

while not stack.is_empty():
     # prints 9 8 7 6 5 4 3 2 1 0
    print(stack.pop(), end = " ")
```

Maak het volgende programma af om de class Stack te implementeren:

```
class Stack:
    def __init__(self):
        self.__elements = []

    # Return True if the stack is empty
    def is_empty(self):
        # your code...

    # Return the element at the top of the stack
```

```
    # without removing it from the stack.
    def peek(self):
        # your code...

    # Store an element at the top of the stack
    def push(self, value):
        # your code...

    # Remove the element at the top of the stack and return it
    def pop(self):
        # your code...

    # Return the size of the stack
    def get_size(self):
        # your code...
```

OPGAVE 6 : EEN EENVOUDIGE STOPWATCH

De class StopWatch implementeert een eenvoudige stopwatch. De class StopWatch bevat het volgende:

fields	startTime en endTime (private)	
method	constructor	init start_time
method	start	reset start_time naar huidige tijd
method	stop	maakt end_time gelijk aan de huidige tijd
method	get_elapsed_time	geeft de verstreken tijd, dus end_time - start_time (in milliseconden)
method	getStartTime	
method	getEndTime	

De methode time.time() uit module time geeft de huidige tijd.

Het volgende testprogramma is gegeven :

```
from stop_watch import StopWatch
size = 1000000
stopWatch = StopWatch()
sum = 0
for i in range(1, size + 1):
      sum += i

stopWatch.stop()
print("The loop time is", stopWatch.get_elapsed_time(), "milliseconds")
```

Schrijf een implementatie van de class StopWatch.

OPGAVE 7 : EXCEPTIONS

Gegeven onderstaande pseudo-code.

```
try:
    statement1
    statement2
    statement3

except Exception1:
    # Handle exception

except Exception2:
    # Handle exception

except Exception3:
    # Handle exception

finally:
    statement4

statement5
```

a) Stel statement2 'werpt' een exceptie, wordt statement 3 dan nog uitgevoerd ?
b) Stel dat de exceptie niet wordt 'gevangen', wordt statement 4 dan ook uitgevoerd ? En hoe zit het dan met statement 5 ?
c) Stel dat de exceptie van het type Exception2 is, wordt statement 4 dan ook uitgevoerd ? En hoe zit het dan met statement 5 ?

OPGAVE 8 : WAT WORDT HET WEER ?

Het volgende programma toont het weer voor de komende dagen voor een opgegeven plaats. Alleen de url moet nog wel worden ingevuld.

a) Ga naar openweathermap.org, vraag aan API key aan en vul in het onderstaande programma de URL in.

```
import json, requests, sys
from pprint import pprint

# get command line arguments
if len(sys.argv) < 2:
    print('Usage: quick_weather.py location')
    sys.exit()

# argument 0 is program name
location = ' '.join(sys.argv[1:])

# download JSON data
# url = ='http://api.openweathermap.org/....?
response = requests.get(url)
response.raise_for_status()

# load JSON data into Python variable
weatherData = json.loads(response.text)

w = weatherData['list']
```

```
#pprint(w)

print('Current weather in %s:' % (location))
print(w[0]['weather'][0]['main'], '-', w[0]['weather'][0]['description'])
print()
print('Tomorrow:')
print(w[1]['weather'][0]['main'], '-', w[1]['weather'][0]['description'])
print()
print('Day after tomorrow:')
print(w[2]['weather'][0]['main'], '-', w[2]['weather'][0]['description'])
```

b) Wat doet de methode json.loads ?

OPGAVE 9 : PEP8

Installeer pep8 (of gebruik een online versie). Haal je programma stop_watch.py door de pep8 checker. In hoeverre is je programma PEP-8 compliant ? Wat valt je op ?