Würgeschlange 3 - Lesson 9

Tobias Maschek, Viktor Reusch
https://github.com/jemx/wise1920-python

mit Materialien von Felix Döring, Felix Wittwer https://github.com/fsr/python-lessons

14. Januar 2020

Python-Kurs

Gliederung

1. Enums

2. Generatoren

Enums

Enumerationen

- Datentyp
- Unterscheidet fest definierte, benannte Zustände
- Zustände haben konstante Werte

```python
from enum import Enum

class Direction(Enum):
    NORTH = 'N'
    EAST = 'E'
    SOUTH = 'S'
    WEST = 'W'

direction = Direction.EAST
print(direction.name) # EAST
print(direction.value) # E
print(direction == Direction.EAST) # True
print(direction == Direction.SOUTH) # False
```

Weiteres Beispiel

```python
from enum import Enum, auto, unique


@unique  # ensure each value is unique
class Grammar(Enum):
    UNRESTRICTED = auto()  # generate numbering 1, 2, ...
    CONTEXT_SENSITIVE = auto()
    CONTEXT_FREE = auto()
    REGULAR = auto()

    def __str__(self):
        return f"{self.name} is from type {self.value - 1}"

print(Grammar.CONTEXT_SENSITIVE) # CONTEXT_SENSITIVE is from type 1
```

Aufgabe 9-1

Programmiert eine Fußgänger-Ampel.

- Der innere Status der Ampel soll durch eine Enumeration repräsentiert werden.
- Dabei gibt es den Zustand *STOP* und *GO*. Der Startzustand ist immer *STOP*.
- Der Zustand soll als Inhalt die entsprechende Ampelfarbe besitzen *'red'* und *'green'*
- Die Methode *switch()* soll entsprechend die Ampel umschalten.
- Die Stringrepräsentation (*__str__*) des Objekts soll sowohl den Status, als auch die aktuell aufleuchtende Farbe ausgeben.

```
1 signal = Signal()
2
3 print(signal) # The signal has the status STOP and shows red
4 signal.switch()
5 print(signal) # The signal has the status GO and shows green
```

Generatoren

Generatoren für mehr Power!

- Generatoren geben eine folge von Werten zurück
- Berechnung jedes Wertes erst, wenn benötigt
- Spart Arbeitsspeicher bei großen Datenmengen
- Ermöglicht erst potentiell unendliche Datenströme
- Läuft schneller bei Suchen oder Filtern
- Elemente werden mit yield zurückgegeben

Beispiel Generatoren

```python
import random

LOREM_WORDS = [
    'consectetur', 'adipiscing', 'elit', 'sed', 'do', 'eiusmod',
    'tempor', 'incididunt', 'ut', 'labore', 'et', 'dolore', 'magna', 'aliqua'
]

def lorem():
    yield from ['Lorem', 'ipsum', 'dolor', 'sit', 'amet']
    while True:
        yield random.choice(LOREM_WORDS)


for i, word in enumerate(lorem()):
    print(word, end=" ")
    if i > 10:
        break

print() # Lorem ipsum dolor sit amet et et dolore labore [...]
```

Aufgabe 9-2

Programmiert zwei Generatorfunktionen.

- Die erste Funktion generiert unendlich lange eine festgelegte Ziffer.
- Diese Ziffer wird der Funktion als Argument übergeben.
- Die zweite Funktion soll die Elemente einer Liste/eines Generators aus Zahlen zurückgeben, solange die Summe der zurückgegebenen Elemente unter einem festgelegten Maximal ist.
- Der Maximalwert wird der Funktion zu Beginn übergeben.
- Testet Kombinationen dieser Funktionen aus.

```
1 gen_list = gen_numbers(3) # [3, 3, 3, 3, 3, 3, 3, 3, 3, ...]
2
3 my_list = [1, 2, 3, 4, 5]
4 result = up_to_max_sum(my_list, 7) # [1, 2, 3]
```