Exercise Sheet 08 Solutions – Practical Python

Sebastian Höffner Aline Vilks

Deadline: Mon, 29 May 2017 08:00 +0200

Exercise 1: There's always more than one way to solve a problem!

File: `manyways.py`

```
import string
import itertools


def loops():
    list1 = []
    for i in string.ascii_lowercase:
        list1.append(ord(i))

    list2 = []
    for i in range(1, 100):
        if i % 3 and i % 5:
            list2.append(i)

    list3 = []
    for i in range(1, 4):
        for j in range(1, 4):
            list3.append((i, j))

    list4 = []
    for i in range(1, 4):
        for j in range(1, 4):
            list4.append(i + j)

    return list1, list2, list3, list4


def lambdas():
```

```
    list1 = list(map(ord, string.ascii_lowercase))

    list2 = list(filter(lambda x: x % 3 and x % 5, range(1, 100)))

    list3 = list(itertools.product(range(1, 4), range(1, 4)))

    # alternatively, you can reuse list3:
    # list4 = list(map(sum, list3))
    list4 = list(map(sum, itertools.product(range(1, 4), range(1, 4))))

    return list1, list2, list3, list4


def list_comprehensions():
    list1 = [ord(s) for s in string.ascii_lowercase]

    list2 = [x for x in range(1, 100) if x % 3 and x % 5]

    list3 = [(x, y) for x in range(1, 4) for y in range(1, 4)]

    list4 = [x + y for x in range(1, 4) for y in range(1, 4)]

    return list1, list2, list3, list4


def main():
    print('Loops:')
    print(*loops(), sep='\n')
    print('Lambdas:')
    print(*lambdas(), sep='\n')
    print('List comprehensions:')
    print(*list_comprehensions(), sep='\n')


if __name__ == '__main__':
    main()
```

Output:

```
Loops:
[97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
[1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32, 34, 37, 38, 41, 43, 44,
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
[2, 3, 4, 3, 4, 5, 4, 5, 6]
Lambdas:
[97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
```

```
[1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32, 34, 37, 38, 41, 43, 44,
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
[2, 3, 4, 3, 4, 5, 4, 5, 6]
List comprehensions:
[97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,
[1, 2, 4, 7, 8, 11, 13, 14, 16, 17, 19, 22, 23, 26, 28, 29, 31, 32, 34, 37, 38, 41, 43, 44,
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
[2, 3, 4, 3, 4, 5, 4, 5, 6]
```

Exercise 2: Passing functions

File: carsorter.py

```
class Car:

    def __init__(self, cost, maintenance, doors, seats, luggage, safety):
        self.cost = cost
        self.maintenance = maintenance
        try:
            self.doors = int(doors)
        except ValueError:
            self.doors = doors
        try:
            self.seats = int(seats)
        except ValueError:
            self.seats = seats
        self.luggage = luggage
        self.safety = safety

    def __repr__(self):
        return 'Car({}, {}, {}, {}, {}, {})'.format(self.cost,
                                                    self.maintenance,
                                                    self.doors,
                                                    self.seats,
                                                    self.luggage,
                                                    self.safety)

    def __str__(self):
        return 'Car: {} seats, {} luggage, {} doors'.format(self.seats,
                                                            self.luggage,
                                                            self.doors)


def read_cars(filename):
```

```
    """Reads cars and creates car instances.

    Drops the evaluation column.

    Args:
        filename: the name of the file containing the car data
    """
    with open(filename, 'r') as carfile:
        return [Car(*(car.split(',')[:-1]))
                for car in carfile.read().splitlines()]


def comfort_evaluation(car):
    """Calculates a comfort value for a car.

    The comfort value is the sum of mapped doors, seats and luggage:

        map_to    door    seats   luggage
           1        2       2      small
           2        3       4       med
           3        4      more     big
           4      5more     --       --

    Args:
        car: The car to evaluate.

    Returns:
        A comfort value. For a car with three doors, four seats and a small
        luggage boot, this would be 2 + 2 + 1 = 5.
    """
    doors = [2, 3, 4, '5more'].index(car.doors) + 1
    seats = [2, 4, 'more'].index(car.seats) + 1
    luggage = ['small', 'med', 'big'].index(car.luggage) + 1
    return doors + seats + luggage


def main():
    """Reads the cars and sorts them. Prints sorted cars."""
    cars = read_cars('car.data')

    cars = sorted(cars, key=comfort_evaluation)

    print(*cars[::100], sep='\n')
```

```
if __name__ == '__main__':
    main()
```

Output:

```
Car: 2 seats, small luggage, 2 doors
Car: 2 seats, small luggage, 3 doors
Car: more seats, small luggage, 2 doors
Car: 2 seats, big luggage, 2 doors
Car: 2 seats, med luggage, 3 doors
Car: 4 seats, small luggage, 4 doors
Car: 4 seats, big luggage, 2 doors
Car: more seats, med luggage, 2 doors
Car: 2 seats, big luggage, 3 doors
Car: 4 seats, med luggage, 4 doors
Car: more seats, small luggage, 4 doors
Car: 2 seats, med luggage, 5more doors
Car: more seats, big luggage, 2 doors
Car: more seats, small luggage, 5more doors
Car: more seats, med luggage, 4 doors
Car: more seats, big luggage, 3 doors
Car: more seats, big luggage, 4 doors
Car: more seats, big luggage, 5more doors
```