EPFL CIVIL-127, Lab 3 solution

3.1

```python
# Sieve of Eratosthenes implementation
# See https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes

N = 272
# False => number is prime
# True => number is composite
a = [False] * N
for i in range(2, N):
    if not a[i]:
        print(i, end=" ")
        for j in range(i+i, N, i):
            a[j] = True
```

3.2

```
# Simulate lots of Snakes & Ladders games and calculate average
# number of moves per game.
# Short implementation, using dict.

import random

snakes_and_ladders = {
    43:  17, 50:  5, 56:  8, 73:  15, 84:  63, 87:  49, 98:  40,
    2:  23, 6:  45, 20:  59, 52:  72, 57:  96, 71:  92, 101: 99,
    102: 40, 103: 97, 104: 96, 105: 95,
}
moves = 0
N = 1_000_000

for i in range(N):
    marker = 1
    while marker != 100:
        marker += random.randint(1, 6)
        moves += 1
        marker = snakes_and_ladders.get(marker, marker)

print("average: ", moves / N)
```

Another solution would be to use a list. As long as the number of `if` statements is reduced and you produce the correct answer in reasonable time, you are good.

3.3

When using `*` to expand a list, the elements are not copied. The list references the elements multiple times. This behavior often haunts new programmers; make sure you properly understand what's going on.

We can create a 5x8 grid by using a `for` loop. Inside each iteration, a new array of 8 elements is created.

```
grid = []
for i in range(5):
    grid.append([0] * 8)
print(grid)

grid[4][2] = 99
print(grid)
```

3.4

```
# Code to solve N people in a circle puzzle (Josephus problem)
# See https://en.wikipedia.org/wiki/Josephus_problem

N = 5
K = 2

active = K  # which person will be removed next
people = list(range(N))

while len(people) > 1:
    print(people)
    print("removing", people[active])
    people = people[:active] + people[active+1:]
    active = (active + K - 1) % len(people)
    print()

print("The answer is:", people[0])
```

3.5

```
# Solution for lab 3.5 from
# https://github.com/vita-epfl/civil127-2026/blob/main/lab3/lab3.pdf

for a in range(0, 10):
    for b in range(0, 10):
        for c in range(0, 10):
            for d in range(0, 10):
                # check that all digits are different by
                # putting the digits in a set and checking the
                # len
                ok = len({a, b, c, d}) == 4

                # check that the sum is 29
                ok = ok and (a + b + c + d == 29)

                # check that code is divisible by 13
                code = 1000 * a + 100 * b + 10 * c + d
                ok = ok and ((code * code - 1) % 13 == 0)

                if ok:
                    print(a, b, c, d)
```

We create a whole bunch of loops and use a `set` to check that we have 4 distinct values.

We could also have written several checks, `ok = (a != b and a != c and a != d and b != c …)`, but using a set is shorter and less error prone.

The answer you should get: 8 5 7 9

Alternatives:

```
# Solution for lab 3.5 from
# https://github.com/vita-epfl/civil127-2026/blob/main/lab3/lab3.pdf

for code in range(0, 9999):
    a = code // 1000  # // is integer division
    b = (code // 100) % 10
    c = (code // 10) % 10
    d = code % 10

    # check that all digits are different by
    # putting the digits in a set and checking the
```

```
    # len
    ok = len({a, b, c, d}) == 4

    # check that the sum is 29
    ok = ok and (a + b + c + d == 29)

    # check that code is divisible by 13
    ok = ok and ((code * code - 1) % 13 == 0)

    if ok:
        print(code)
```

And:

```
# Solution for lab 3.5 from
# https://github.com/vita-epfl/civil127-2026/blob/main/lab3/lab3.pdf

for code in range(0, 9999):
    s = str(code).zfill(4)
    a = int(s[0])
    b = int(s[1])
    c = int(s[2])
    d = int(s[3])

    # check that all digits are different by
    # putting the digits in a set and checking the
    # len
    ok = len({a, b, c, d}) == 4

    # check that the sum is 29
    ok = ok and (a + b + c + d == 29)

    # check that code is divisible by 13
    ok = ok and ((code * code - 1) % 13 == 0)

   if ok:
       print(code)
```