RECURSION

Final exam

- Exam dates:
  January 6 at 12:15
  January 20 at 12:15
- Retake:
  January 27 at 12:15

- Quiz on the last part of the course (5 p)
- Task on the elements of programming (10 p)
- Programming task on recursion (5 p)
- Programming task on all aspects of the course (10 p)

There will be a sample exam in Moodle.

Recursive function

- A function that calls itself in its body is called recursive.
- In order for the sequence of calls to end, the function should contain branching according to the values of the arguments.
- The branch that doesn't contain recursive call is the *base case*.
- The branch that contains recursive call is the *recursive step*.

Examples of recursive functions

- Factorials

```
def fact(n):
    if n == 0:
        return 1                    ← base case
    else:
        return n * fact(n - 1)      ← recursive step
```

- Fibonacci numbers

```
def fib(n):
    if n == 0:
        return 0                    ← base case
    elif n == 1:
        return 1                    ← base case
    else:
        return fib(n - 1) + fib(n - 2)    ← recursive step
```

Using recursion

Always make sure that the recursion bottoms out.

- A recursive function must contain at least one base case.
- Sequence of recursive calls must eventually lead to the base case.
- Be careful not to create an infinite chain of function calls.

Missing base case:

```
def fact(n):
    return n * fact(n - 1)
```

Doesn't lead to the base case:

```
def fact(n):
    if n <= 1:
        return 1
    else:
        return n * fact(n + 1)
```

Recursive and non-recursive functions

Recursive solution can lead to very short and elegant code.
Compare the iterative solution with the recursive solution:

Iterative solution

```
def fact(n):
    prod = 1
    while n > 1:
        prod *= n
        n -= 1
    return prod
```

Recursive solution

```
def fact(n):
    if n <= 1:
        return n
    else:
        return n * fact(n - 1)
```

Recursive and non-recursive functions

But sometimes the recursive solution can be very slow.

Iterative solution

```
def fib(n):
    if n <= 1:
        return n
    else:
        a = b = 1
        for i in range(2, n):
            a, b = b, a + b
        return b
```

Recursive solution

```
def fib(n):
    if n <= 1:
        return n
    else:
        return fib(n - 1) +
                        fib(n - 2)
```

Summary

- Recursion is a way to decompose a task into smaller subtasks.
- Recursive step solves a smaller copy of the same task
- Base case of recursion corresponds to the trivial version of the task that can be solved directly.