# Recursion – Exercises
# ENSEA/FAME Computer Science

Linked lists are not native in Python. It is much more common to work with arrays, which are called in the Python jargon lists...

We will consider the following simple implementation of a linked list:

- the empty linked list is represented by `None`,
- a nonempty linked list of head `t` and tail `t` is represented by the tuple `(h, t)`.

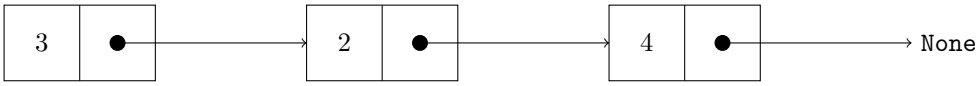


Figure 1: A linked list

For example, the linked list of figure 1 is represented by `(3, (2, (4, None)))`.

**Question 1.** Create a function for each of the following operations:

1. `def head(l):` returns the head of list `l`.
2. `def tail(l):` returns the tail of list `l`.
3. `def cons(e, l):` returns the list `e::l`.
4. `def length(l):` returns the length of list `l`.
5. `def nth(l, n):` return the `n`-th element of list `l`.
6. `def addlast(e, l):` returns a list with element `e` added at the end of list `l`. For example, `addlast(3, (1, (2, None)))` returns `(1, (2, (3, None)))`.
7. `def reverse(l):` returns the reverse of list `l`.
8. `def concat(l1, l2)`: returns the concatenation of `l1` and `l2`.
9. `def concats(ls):` returns the concatenation of all elements of `ls` (`ls` is a list of lists).
10. `def map_(f, l):` applies the function `f` to the elements of list `l` and returns the list of the evaluations. For example,

```
>>> map(lambda x: x+1, (3, (2, (4, None))))
(4, (3, (5, None)))
```

11. `def foldleft(f, start, l):` if we denote by `a1`, ..., `an` the elements of list `l`, then the value returned by `foldleft(f, start, l)` is `f( ... (f(f(f(start, a1), a2), a3), ...), an)`. Test with the following example

```
>>> foldleft(lambda x,y: x+y, 0, (1, (2, (3, (4, None)))))
10
```

12. `def forall(p, l):` checks if all elements of list `l` satisfy predicate `p`.
13. `def exists(p, l):` checks if at least one element of list `l` satisfy predicate `p`.
14. `def mem(e, l):` checks if `e` belongs to list `l`.
15. `def sort(l):` returns a sorted version of list `l`. Try to find a naive recursive function with quadratic complexity.

**Question 2.** Give the complexity of those functions. Which are tail recursive?