**Programmierkurs Python I – WS 10/11**

# Übung 11

---

## 1 List Comprehensions (2 Punkte)

Schreibe eine Funktion `prime_list(n)`, die die Primzahlen von 1 bis n in einer Liste zurückgibt. Die Rückgabeliste soll nur über (möglichst wenige) List Comprehensions definiert werden.

## 2 Dekoratoren als Typ-Tester (2 Punkte)

Schreibe einen Dekorator der testet, ob die Parameter einer Funktion beim Aufruf einen bestimmten Typ haben, und andernfalls eine Ausnahme wirft.

```
>>> @check(int, int)
... def add(x, y):
...     return x + y
...
>>> add(17, 4)
21
>>> add(1.7, 4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 6, in wrap
Exception: 1.7 not a <class 'int'>
```

Der Dekorator soll so generisch sein, dass es egal ist, wie die zu dekorierende Methode aussieht und wie viele Parameter sie bekommt.

Hinweis: Ob ein Objekt `obj` einen bestimmten Typ `t` hat, überprüft man mit `isinstance(obj,t)` (siehe Übung 9).

## 3 Dekoratoren (1 + 3 Punkte)

Erweitere die Funktionalität des `log`-Dekorators aus der Vorlesung so, dass neben dem Funktionsnamen und den übergebenen Argumenten auch der Rückgabewert

ausgegeben wird:

```
>>> @log('add')
>>> def add(x, y):
...       return x + y
...
>>> add(3, 4)
Call: add(3, 4)
Result: 7
7
```

Bonus: erweitere die Funktionalität so, dass der Dokorator bei rekursiven Funktionen eine leicht lesbare Ausgabe produziert:

```
>>> @log('fib')
>>> def fib(n):
...       return n if n < 2 else fib(n - 1) + fib(n - 2)
...
>>> fib(3)
Call[1]: fib(3)
  Call[2]: fib(2)
    Call[3]: fib(1)
    Result[3]: 1
    Call[4]: fib(0)
    Result[4]: 0
  Result[2]: 1
  Call[5]: fib(1)
  Result[5]: 1
Result[1]: 2
2
```

---

## Abgabe bis Donnerstag, 03.02.2011, 14:00 Uhr