- **Starten der interaktiven OCaml-Umgebung:** Die interaktive OCaml-Umgebung wird mit dem Befehl `ocaml` gestartet. In dieser Umgebung können interaktiv Ausdrücke ausgewertet und Variablen definiert werden. Jeder Befehl in der OCaml-Umgebung muss mit `;;` abgeschlossen werden.

- **Komfortablere interaktive Ocaml-Umgebung:** Um besser mit der interaktiven Ocaml-Umgebung arbeiten zu können, empfiehlt sich der Einsatz von `ledit`. Dazu muß die Ocaml-Umgebung einfach nur mit dem folgenden Befehl auf der Kommandozeile gestartet werden: `ledit ocaml`

  `ledit` ist bereits auf den Praktikumsrechner vorinstalliert, sonst muß es heruntergeladen[1] und installiert werden.

  Damit funktionieren die Tasten ←, →und Del wie erwartet. Zusätzlich kann man mit ↑ und ↓ in den bereits eingebenen Befehlen blättern (History) und mit Ctrl + A und Ctrl + E an den Anfang bzw. das Ende der Zeile springen.

- **Laden von OCaml-Dateien**

  OCaml-Dateien sollten die Endung ".ml" tragen und können in der interaktiven OCaml-Umgebung mit dem Befehl `#use "<datei>"` geladen werden.

- **Elementare Funktionen:**

  - `print_string : string -> unit`
    Gibt den als Parameter übergebenen String aus und liefert `()` zurück.
  - `string_of_int : int -> string`
    Liefert eine String-Darstellung des als Parameter übergebenen `int`-Wertes zurück.
  - `int_of_string : string -> int`
    Liefert den `int`-Wert des als Parameter in String-Darstellung übergebenen Wertes zurück.
  - `read_line : unit -> string`
    Liest eine Zeile von der Standard-Eingabe und liefert sie als String zurück.

[1] `ftp://ftp.inria.fr/INRIA/Projects/cristal/Daniel.de_Rauglaudre/Tools/`

- **Typische Fehler:**
  - Listen-Elemente werden mit „;“und nicht mit „,“separiert. Der Wert `[1,2]` ist die Liste, die nur aus dem Paar `(1,2)` besteht. Der Wert `[1;2]` hingegen ist die Liste mit den Elementen `1` und `2`.

- **Executables erstellen**

  Um ein Executable zu erstellen muss die `.ml`-Datei zunächst compiliert werden. Zur Übersetzung des nachfolgenden Beispiels ist

  ```
  ocamlc -c fact.ml
  ```

  einzugeben. Dies erzeugt die Dateien `fact.cmi` und `fact.cmo`. Um schließlich die ausführbare Datei zu erzeugen ist

  ```
  ocamlc -o fact fact.cmo
  ```

  einzugeben. Dies erzeugt die ausführbare Datei `fact`. Das Programm `fact` berechnet die Fakultät des übergebenen Argumentes und gibt das Ergebnis aus. Bespielsweise wir bei der Eingabe

  ```
  ./fact 5
  ```

  folgendes ausgegeben:

  ```
  fact(5)=120
  ```

  Als Quell-Text ist beispielsweise nachfolgend angegebene Datei `fact.ml` dienlich:

```
open Sys
open Array

let rec fact = function
    0 -> 1
  | n -> n * (fact (n-1))

let _ =
    if length argv <= 1 then
        print_string "Argument erwartet!\n"
    else
        let str_n = argv.(1) in
        let n = int_of_string str_n in
        let result = fact n in
        let result = string_of_int result in
        print_string ("fact(" ^ str_n ^
                      ") = " ^ result ^ "\n")
```