Programming language features

	Traditional, "static"	Static o-o	Scripting, "dynamic"	Mixed
Examples	C, Fortran	C++	Python, Ruby	Java, OCaml
Objects?	No	Yes	Yes	Yes
Automatic mem. mgt.?	No	No	Yes	Yes
Static types?	Yes	Yes	No	Yes
Tagged values?	No	No	Yes	Partially

Tuples

- Create "struct" by putting expressions in parentheses separated by commas:
  - `(3, 5.0)`: int * float
  - `(3, "abc", true)`: int * string * bool
- Exercise — fill in types:
  - ('a', 'b') : char * char
  - ('a', "a", 'a') : char * string * char
  - (5, ("a", 'a')) : int * (string * char)
- Exercise — create a value of the given type:
  - (3, (4, 5.0), "a") : int * (int * float) * string

Tuples (cont.)

- Use functions `fst` and `snd` to get elements of a pair. *Only works for pairs.* (We'll see how to deal with bigger tuples in next class.)

- Exercise: Write a function to add the elements of an int * int pair:
  - `let addelts p =` fst p + snd p

Lists (cont.)

- **Exercise — fill in types (or flag error):**
  - `['a'; 'b']:` char list
  - `['a'; 'b'; "c"]:` error
  - `[4; int_of_string "34"]:` int list
  - `[[4]; [5]; []]:` int list list
  - `[(1, 2); (3, 4)]:` (int * int) list
  - `[(1, 2); (3, 4, 5)]:` error
  - `[(1, [3]); (4, [5; 6])]:` (int * (int list)) list

Lists (cont.)

- Exercise — create a value of the given type (other than the empty list):
  - [3] : int list
  - [[3]] : (int list) list
  - [(3, "abc")] : (int * string) list
  - [["abc"]] : (string list) list
  - [(3, ["abc"])] : (int * string list) list
  - [[(3, ["abc"])]] : ((int * string list) list) list

Lists (cont.)

- **Exercise: Write a function to compute the sum of the first two elements of an int list:** `addfirsttwo [5; 3; 2; 6]` **= 8. You can assume the list is of length at least 2:**
  - `let addfirsttwo lis =` hd lis + hd (tl lis)

- **Exercise: Write a function to compute the sum of the *lengths* of the first two elements of an (int list) list:** `addfirsttwolengths [[5; 3]; [2]; [6; 2; 5; 3]]` **= 3. You can assume the list is of length at least 2:**
  - `let addfirsttwolengths lis =`
    length (hd lis) + length (hd (tl lis))

Polymorphic functions (cont.)

Ex: Write the polymorphic types of the following functions. (You can write either 'a, 'b, etc. or α, β, as you prefer.)

```
let mktriple p = (fst p, snd p, 3)
```

$\alpha * \alpha \to \alpha * \alpha * int$

```
let pair_of_first p = (fst p, fst p)
```

$\alpha * \beta \to \alpha * \alpha$

```
let double_first lis = [hd lis; hd lis]
```

α list $\to \alpha$ list

```
let pair2list p = [fst p, snd p]
```

$\alpha * \alpha \to \alpha$ list

addfirsttwolengths (defined above) (α list) list $\to$ int