

TinyBASIC: Interactive programming system

What you need to know about F#

Tomas Petricek, Charles University

✉ tomas@tomasp.net

🦋 [@tomasp.net](https://tomasp.net)

🌐 <https://tomasp.net>

🌐 <https://d3s.mff.cuni.cz/teaching/nprg077>



TinyBASIC

What F# do you need to know

-  Project, console and tail recursion
-  F# language logic and data types
-  Records, functions, tuples, patterns
-  List processing using built-in functions

Demo

Project, console, recursion

```

let point = (1, 10)
let (x, y) = point

(* (int*int) -> (int*int) *)
let rotate (x, y) = (y, x)

(* int -> (int*int) -> (int*int) *)
let moveX by (x, y) = (x + by, y)

(* (int*int) -> int *)
let area (x, y) =
  match x, y with
  | 0, _ | _, 0 -> 0
  | x, y -> x * y

(* (int*int) -> int *)
let area pt =
  match pt with
  | ((0, _) | (_, 0)) -> 0
  | x, y -> x * y

```

Tuples, patterns and functions

Tuple type `int * int` is just another ordinary type of values

Pattern `(x, y)` can appear in multiple locations in code

Functions can mix arguments and tuples

SKETCH

Tuples and patterns

```
let l1 = [1; 2; 3; 4]
let l2 = 1::2::3::4::[]
let l3 = [1..4]
```

```
(* Pattern matching lists *)
match list with
| [e1; e2] -> (...)
| e1::els -> (...)
| [] -> (...)
```

```
(* Possible, but not very useful *)
let (e::es) = list
let foo [e1;e2] = (...)
```

```
(* Higher-order list functions *)
let twice x = x * x
List.map twice [1..10]
List.map (fun x -> x * x) [1..10]
List.sum [1..10]
```

List constructors and list patterns

List type written as
list<int> or **int list**

Constructed using **::**
(rare) and **[..]** (often)

Patterns **::** and **[..]**
can appear anywhere,
but are partial

Demo

Real-world list processing (1/2)



The pipe operator

Fluent style for
functional data
processing

```
let (|>) x f = f x
```

In bash scripting (`|`),
adopted by R (`%>%`),
maybe JavaScript

Demo

Real-world list processing (2/2)