Microsoft

F# is a cross-platform, open source functional programming language for .NET

F# Basics

F# is supported in Visual Studio, Visual Studio for Mac, and Visual Studio Code with the Ionide plugin.

Type inference and lightweight syntax

F# infers types and lets you get a lot done with very little code

```
let square x = x * x
let isOdd x = x % 2 <> 0

// Use '|>' to pipe results into functions.
// Note that you can pass F# functions as parameters!
let oddSquares items =
    items
    |> List.filter isOdd
    |> List.map square
let result = oddSquares [1; 2; 3; 4; 5]
```

Rich data types

F# has advanced types for powerful domain modeling

```
type CartItem = { ProductCode: string; Qty: int }
type ActiveCartData = { UnpaidItems: CartItem list }
type PaidCartData =
    { PaidItems: CartItem list; Payment: decimal }

type ShoppingCart =
    | EmptyCart   // no data
    | ActiveCart of ActiveCartData
    | PaidCart of PaidCartData
```

Manipulate data with functions & pattern matching

Use **let** to define F# functions that work with data.

```
// Look up the price in a database somewhere
let getPrice cartItem =
    lookUpPrice cartItem.ProductCode

// Sum up the total price of each item
let calcCartPrice cart =
    cart.UnpaidItems
    |> List.sumBy (fun i -> i.Qty * (getPrice item))

// Use pattern matching to process a cart kind
let buyItems shoppingCart =
    match shoppingCart with
    | EmptyCart -> 0 // Nothing to pay for
    | PaidCart _ -> 0 // Already paid
    | ActiveCart cart -> calcCartPrice cart
```

Easily work with .NET objects

F# lets you define & interoperate with .NET objects, no matter how advanced they need to be. Use F# functions to manipulate objects.

```
type IMathable =
   abstract Add: int * int -> int
   abstract PI: float

// Define and implement an interface
type MathMachine(threePiDigits) =
    interface IMathable with
        member __.Add(x, y) = x + y
        member __.PI =
            if threePiDigits then 3.141
            else 3.14159

let machine =
    MathMachine(threePiDigits = true)
    :> IMathable
let result = machine.Add(1, 2)
```

Getting started is as easy as:

dotnet new console -lang F#

Learn more at:

F# homepage: aka.ms/fsharphome
F# docs: docs.microsoft.com/dotnet/fsharp
F# for Fun and Profit: https://fsharpforfunandprofit.com/why-use-fsharp/
F# Wikibook wikibooks.org/wiki/F_Sharp_Programming