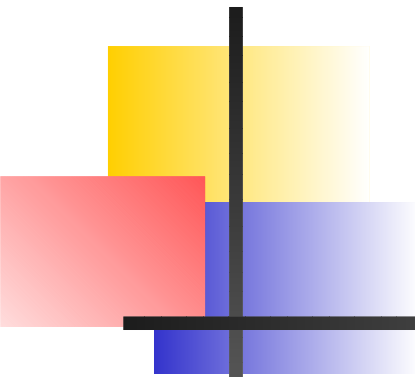




Functional Programming

Some properties of functional programming languages:

- allow the programmer to take a high-level view of *what* is to be computed rather than *how*
- programs are built out of *expressions* (whose meanings are *values*)
- there is no notion of a state (i.e. no variables or assignment)
- there is no sequencing
- functions do not have any *side effects* (any effect other than to return the value of the function)



Referential Transparency

- When we use an expression we are only interested in its value (and for functions, in its return value); nothing else can have any effect on the rest of our program.
- This property is known as *referential transparency*
- Basically, if we take an expression anywhere in a program, and replace it with another expression that gives the same value, we will not have changed the overall result of the program.



Functional Programming: Why?

- At first sight a language without variables, assignment and sequencing looks very impractical.
- We will show in these lectures how a lot of interesting programming can be done in the functional style.
- Imperative programming languages have arisen as an abstraction of hardware, from machine code, through assemblers and macro assemblers, to C, Java and beyond.
- Perhaps this is the wrong approach and we should approach the task by modelling the *problem* rather than the *solution*.



Advantages

- If $x = f(1)$, are the following true?
 - $x + x = f(1) + f(1)$
 - $f(1) = f(1)$
 - $x + f(1) = f(1) + x$
- Not if you are using an imperative language!
- These equalities all hold for functional languages because functions cannot have side-effects.
- This makes functional programs easier to reason about.
- Writing programs in a functional style also promotes good programming practice.



Disadvantages

- Functional programming is not without its deficiencies.
- Some things are harder to fit into a purely functional model:
 - Input-output
 - Interactive or continuously running programs (e.g. editors, process controllers)
- Functional languages also correspond less closely to current hardware, so they can be less efficient, and it can be hard to reason about their time and space usage.