Recap: the "environment" maintains name-value pairs, where values are either constants or closures. To enable closures to reference itself in the defining environment, an additional dictionary (name-value pairs) is needed. The two association lists together form the enviornment: ["a list of name-value pairs" "a list of closure name-closure pairs"]	
To create names on-the-fly: gensym	
What does this support (independent of what the syntax allows)? - function parameters (partially allowed by the grammar) - anonymous functions (not allowed currently by the grammar) - function as returned value (partially allowed by the grammar) - normal order evaluation (unrelated to the grammar, but can be) - nested functions (not allowed currently by the grammar)	Work out examples on the board.
Note: Functions not just first-class, they can be the only class.	
Denotational Semantics: The semantics of a program is given by the "meaning" function [[...]]. Each part p of a program P is given a denotation, [[p]] -- a mathematical object. Ideally, the meaning (or denotation) of the program, [[P]], is the composition of the denotations of its constituent parts. (Compositionality)	
Structural Operational Semantics for Fpl (sibling to FP).	Notes by Matthew Hennessy (Chapter 4)

	Remarks: - Emphasis on "Big Step Semantics" (p. 69) - Complete semantics for Fpl on p. 88.