Programowanie funkcyjne
Notatka z wykładu 1.

1. **Wartość** - Niezmienny (niemutowalny) obiekt typu prostego (np. liczba, litera, ciąg znakowy) lub niemutowalna kompozycja (struktura danych) innych wartości (np. lista, krotka, słownik, instancja klasy).
2. **Identyfikator** - Ciąg znakowy (zazwyczaj alfanumeryczny) identyfikujący stałą lub zmienną.
3. **Stała** - Niemutowalna relacja pomiędzy wartością a identyfikatorem. Przypisanie wartości do nazwy. Posiada tylko jeden stan (wartość się nie zmienia).
4. **Zmienna** - Mutowalna relacja pomiędzy wartością a identyfikatorem.
5. **Mutowalność** - Możliwość zmiany danej wartości (w miejscu).
6. **Stan** - Wartość identyfikatora w danych momencie czasu.
7. **Stanowość** (jako cecha języka) (ang. statefulness) - Posiadanie zmiennych oraz mutowalnych obiektów. Posiadanie stanu w aplikacji.
8. **Efekt uboczny** - Zjawisko zmiany stanu poza lokalnym kontekstem, tj. obserwowalny efekt poza zwróceniem wartości.
9. **Przeźroczystość referencyjna** - Brak efektów ubocznych, determinizm w stosunku do argumentów.
10. **Imperatywna ewaluacja** - sekwencyjna ewaluacja instrukcji (ang. statement).
11. **Funkcyjna ewaluacja** - ewaluacja kompozycji wyrażeń.
12. **Lambda calculus** - matematyczny model obliczeniowy, prekursor programowania funkcjnego (Lispa).
13. **Architektura von Neumanna/maszyna Turinga** - teoretyczna architektura komputera sekwencyjnego/sekwencyjny model obliczeniowy, prekursow programowania imperatywnego.
14. Przykłady języków funkcynych: Clojure, Lisp, Scheme, Common Lisp, Racket, Erlang, Scala, Haskell, ML, Idris, F#.