All you need is Lambda

Definition : Functional Programming

A computer programming paradigm that relies on fucntions modeled on mathematical functions.

The essence of functional programming is that programs are combinations of expressions.

- **Expressions** conrete values, variables, and also functions.
- **Functions.** Expressions that are applied to an argument or input, and once applied, can be reduced or evaluated.
- **Purity**: Referential transparency - the same function, same `input`, always return the same result in pure functional programming, as they do in math.

Definition : What is a function?

A function is a relation between a set of possible inputs and a set of possible outputs.

- **The input set.** The Domain.
- **The output set.** The Codomain.
- **The image.** The subset of the codomain that contains possible outputs related to different inputs.

1 The structure of lambda terms

The lambda calculus has 3 basic components, or *lambda terms*: expressions, variables, and abstractions.

- **Expression** : refers to a superset of all those things.
  An expression can be a variable name, an abstraction, or a combination of those.
- **Abstraction** : a function. It is a lambda term that has a head (a lambda) and a body and is applied to an argument (an input value).

$$\lambda x.x$$

2 Alpha equivalence

Same meaning: $\lambda x.x \equiv \lambda y.y \equiv \lambda z.z$

3 Beta reduction

Definition : Beta reduction

Substitute `input` expression for all `variables` within the body of `abstraction`.

$$(\lambda x.x)2$$
$$2$$

This is the *identity* function. And applications in the lambda calculus are **left associative.**

- **Free variables.** Variable not in the head λ, not apply *alpha equivalence.*

$$\lambda x.xy$$

- **Multiple arguments.** $\lambda xy.xy \rightarrow \lambda x(\lambda y.xy)$

4 Evaluation is Simplification

- **Beta normal form.** The form which cannot be reduced any further.

  Eg: not `2000/1000`, but `2`.

5 Combinators

A **combinator** is a lambda term with **no free variables**. Combinators are only used **combine** the arguments.

- $\lambda zy.zzy$
- $\lambda x.x$
- $\lambda xy.y$

6 Divergence

- Reducible lambda terms **not** reduce neatly to a **beta normal form**. The reduction process **never** terminates or ends.

$$\texttt{omega:}\ (\lambda x.xx)(\lambda x.xx) \rightarrow (\lambda x.xx)(\lambda x.xx)$$

7 Definition

1. A **lambda abstraction** is an anonymous function or lambda term. $(\lambda x.x + 1)$
2. **Application** is reducing lambdas, which binds the argument to whatever the lambda was applied to.

$$(\lambda x.x)1$$

3. Lambda calculus is a formal system for expressing programs in terms of abstraction and application.
4. **Normal order** is a common evaluation strategy in lambda calculi. Normal order means evaluating (beta reducing) the leftmost outermost lambdas first.