**Standard Fx**

**Fx is a language which design to use in some small occasion:**

**For example, you can print "Hello World" like this:**

**main:print("Hello World")**

**Also, you can sort a list named xs like this:**

**sort(xs,(x,y)=>{x<y})**

**A function you write could insert in any right places and means the same function in the same application's source code, like this function:**

**(x)=>{x<0:f(x),x=0:g(x),x>0:t(x)}**

**You can create an object like this:**

**pair{fst:0,snd:1}**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sign** | | **Form** | **Description** | **Note** | | |
| **Ei** | **Vi** | **Consist of Aa-Zz and \_ but no beginning with \_** | **An identifier** | **Vi is an variable in default**  **The variable Vi refers the global variable Vi's value in default** | | |
| **C** | **(V0,V1,...,Vm)=>{E-1:E1, E-2:E2,..., E-n:En}** | **A function** | **Note** | | **Type** |
| **The variable V0 in E-j or Ej must refers the global variable V0's value(It is Law A)**  **The variable Vi in E-j or Ej must refers the NO.i value (V0,V1,...,Vm)=>{E-1:E1, E-2:E2,..., E-n:En} received(Except disobey Law A)**  **As soon as E-j is \_true, it returns Ej 's value**  **If n=1 and E-1 always is \_true then E-1:E1 could write as E1**  **i=1,2,...,m,j=1,2,...,n,m>0,n>0** | | **\_func** |
| **E0(E1,E2,...,En)** | **A function call** | **E0 receives E1,E2,...,En in order and return a value as E0(E1,E2,...,En)'s value**  **i=1,2,...,n,n>0** | |  |
| **Consist of 0-9 and at most one . and e or e- in it** | **A number** |  | | **\_num** |
| **\_nan** | **\_nan refers nan** | |
| **\_inf** | **\_inf refers inf** | |
| **Consist of chars in ""** | **A string** | **"" means "** | | **\_str** |
| **Consist of chars in ''** | **An error message** | **'' means '** | | **\_err** |
| **\_true** | **A bool** | **If E0's value is \_true then what statement expressed by E0 is true** | | **\_bool** |
| **\_false** | **If E0's value is \_false then what statement expressed by E0 is false** | |
| **{}** | **A list** | **An empty list** | | **\_list** |
| **{E1,E2,...,En}** | **A list that has n(n>0) members** | |
| **V0{V1:E1,V2:E2,...,Vn:En}** | **An object or an error message** | **V0 is a type name**  **Vi is a member variable**  **T is a value which type is V0**  **The member variable V-1 of T refers 'undefined'**  **The member variable Vi of T refers Ei's value**  **If (?T)'s value is \_true then V0{V1:E1,V2:E2,...,Vn:En}'s value is T else V0{V1:E1,V2:E2,...,Vn:En}'s value is 'Create object error'**  **i=1,2,...,n,n>0** | |  |
| **E1.V1** | **A member variable** | **Get the member variable V1's value of E1's value** | |  |
| **(D)** | **(-E1)** | **The opposite of E1** |  | **If left char is ( or { or , or : then write D instead of (D)** | |
| **(?E1)** | **HERE** |
| **(E1^E2)** | **E1 to the power of E2** | **E1 and E2 has some type** |
| **(E1\*E2)** | **Multiply E1 by E2** |
| **(E1/E2)** | **E1 divided by E2** |
| **(E1+E2)** | **E1 plus E2** |
| **(E1-E2)** | **E1 subtract E2** |
| **(E1<E2)** | **E1 less than E2** |
| **(E1>E2)** | **E1 greater than E2** |
| **(E1=E2)** | **E1 equals to E2** |
| **(E1/=E2)** | **E1 not equals to E2** |
| **(E1<=E2)** | **E1 less than or equals to E2** |
| **(E1>=E2)** | **E1 greater than or equals to E2** |