**Standard Fx**

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

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

**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}**

**However, you can’t write notes for your code!**

**Here is the definition of Fx:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sign** | | **Form** | **Description** | **Note** | | |
| **#** | | **One of + or - or \* or / or < or > or <= or >= or = or /= or /\ or \/ or \** | **An operator** | **There is no priority in operators** | | |
| **Ei** | **Vi** | **Consist of Aa-Zz or \_ 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** | **(V1,V2,...,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 (V1,V2,...,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** |
| **Consist of 0-9 and at most one . and e or e- in it** | **A number** | **Float 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** | | |
| **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** | | |
| **E1.V1** | **A member variable** | **Get the member variable V1's value of E1's value** | | |
| **(-E1)** | **A calculation** | **If the char before (-E1) is ( or { or , or : and the char after (-E1) is not . or ( then (-E1) could write as -E1** | | |
| **(?E1)** | **If the char before (?E1) is ( or { or , or : and the char after (?E1) is not . or ( then (?E1) could write as ?E1** | | |
| **(E1#E2)** | **If the char before (E1#E2) is ( or { or , or : and the char after (E1#E2) is not . or ( then (E1#E2) could write as E1#E2** | | |
| **Pi** | **Di** | **V0:E0** | **Define a global variable** | **The global variable V0's value is E0's value** | | **Blank chars will be ignored except in "" or ''** |
| **V0:** | **The global variable V0's value is inexpressible by Fx** | |
| **-V1:E0** | **Define a calculation** | **If E1's type is V1(V1 is a type name here) then (-E1)'s value is (V1)=>{E0}(E1)'s value** | |
| **?V1:E0** | **If E1's type is V1(V1 is a type name here) then (?E1)'s value is (V1)=>{E0}(E1)'s value** | |
| **T1#T2:E-1** | **If E-1 exists then (E1#E2)'s value is E-1(E1,E2)'s value**  **else if E-2 exists and E-3 does not exist then (E1#E2)'s value is E-2(E1,E2)'s value**  **else if E-2 does not exist and E-3 exists then (E1#E2)'s value is E-3(E1,E2)'s value**  **else if E-2 exists and E-3 exists then (E1#E2)'s value is 'T1#T2 is undefined.'**  **else if E-4 exists then (E1#E2)'s value is E-4(E1,E2)'s value**  **else (E1#E2)'s value is 'T1#T2 is undefined.'** | **T1 and T2 is a type name**  **E1's type is T1**  **E2's type is T2** |
| **T1#?:E-2** |
| **?#T2:E-3** |
| **?#?:E-4** |
| **$F** | **Import a file** | **F is a file path** | **Import the file what F refers** |
| **&F** | **F is the file name of a file in standard library** |
| **M** | **D1|D2** | **Multiple definitions or import files** | **D1|D2 is the same as D2|D1** | |