# 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 | |
| $E_i$ | $V_i$ | Consist of Aa-Zz or _ but no beginning with _ | An identifier | $V_i$ is an variable in default<br>The variable $V_i$ refers the global variable $V_i$'s value in default | |
| | C | $(V_1,V_2,...,V_m)=>\{E_{-1}:E_1, E_{-2}:E_2,..., E_{-n}:E_n\}$ | A function | Note: The variable $V_0$ in $E_{-j}$ or $E_j$ must refers the global variable $V_0$'s value(It is Law A)<br>The variable $V_i$ in $E_{-j}$ or $E_j$ must refers the NO.i value $(V_1,V_2,...,V_m)=>\{E_{-1}:E_1, E_{-2}:E_2,..., E_{-n}:E_n\}$ received(Except disobey Law A)<br>As soon as $E_{-j}$ is _true, it returns $E_j$ 's value<br>If n=1 and $E_{-1}$ always is _true then $E_{-1}:E_1$ could write as $E_1$<br>i=1,2,...,m,j=1,2,...,n,m>0,n>0 | Type: _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 $E_0$'s value is _true then what statement expressed by $E_0$ is true | _bool |
| | | _false | | If $E_0$'s value is _false then what statement expressed by $E_0$ is false | |
| | | {} | A list | An empty list | _list |
| | | $\{E_1,E_2,...,E_n\}$ | | A list that has n(n>0) members | |
| | | $V_0\{V_1:E_1,V_2:E_2,...,V_n:E_n\}$ | An object or an error message | $V_0$ is a type name<br>$V_i$ is a member variable<br>T is a value which type is $V_0$<br>The member variable $V_{-1}$ of T refers 'undefined'<br>The member variable $V_i$ of T refers $E_i$'s value<br>If (?T)'s value is _true then $V_0\{V_1:E_1,V_2:E_2,...,V_n:E_n\}$'s value is T else $V_0\{V_1:E_1,V_2:E_2,...,V_n:E_n\}$'s value is 'Create object error'<br>i=1,2,...,n,n>0 | |
| | | $E_0(E_1,E_2,...,E_n)$ | A function call | $E_0$ receives $E_1,E_2,...,E_n$ in order and return a value as $E_0(E_1,E_2,...,E_n)$'s value<br>i=1,2,...,n,n>0 | |
| | | $E_1.V_1$ | A member variable | Get the member variable $V_1$'s value of $E_1$'s value | |
| | | $(-E_1)$ | A calculation | If the char before $(-E_1)$ is ( or { or , or : then $(-E_1)$ could write as $-E_1$ | |
| | | $(?E_1)$ | | If the char before $(?E_1)$ is ( or { or , or : then $(?E_1)$ could write as $?E_1$ | |
| | | $(E_1\#E_2)$ | | $E_1$ and $E_2$ must have the same type<br>If the char before $(E_1\#E_2)$ is ( or { or , or : then $(E_1\#E_2)$ could write as $E_1\#E_2$ | |
| $P_i$ | $D_i$ | $V_0:E_0$ | Define a global variable | The global variable $V_0$'s value is $E_0$'s value | Blank chars will be ignored except in "" or '' |
| | | $V_0:$ | | The global variable $V_0$'s value is inexpressible by Fx | |
| | | $-V_1:E_0$ | Define a calculation | If $E_1$'s type is $V_1$($V_1$ is a type name here) then $(-E_1)$'s value is $(V_1)=>\{E_0\}(E_1)$'s value | |
| | | $?V_1:E_0$ | | If $E_1$'s type is $V_1$($V_1$ is a type name here) then $(?E_1)$'s value is $(V_1)=>\{E_0\}(E_1)$'s value | |
| | | $V_1\#V_2:E_0$ | | If $E_1$'s type is $V_1$($V_1$ is a type name here) then $(E_1\#E_2)$'s value is $(V_1,V_2)=>\{E_0\}(E_1,E_2)$'s value | |
| | | $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 | $D_1\|D_2$ | Multiple definitions or import files | $D_1\|D_2$ is the same as $D_2\|D_1$ | |