Quick Reference to ISETL Syntax

Syntax Rules

`;`	a semicolon and <enter> or <return> must follow any statement.
`!`	an exclamation point must precede directives. They must not end with ;
`:=`	Variables may be assigned values.

Data Types

integer	`2130`
floating point	`3.5`
	`7.5e-001`
Boolean	`true / false`
string	`"embedded in double quotes"`
undefined	`OM /om`
tuple	`[2, true, 4.6]`
set	{7.5, false, "TOM", 6}

Arithmetic Operators

Unary

`+`	positive
`-`	negative

Binary

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation
`mod`	modular arithmetic

Relational Operators

`=`	equal
`/=`	not equal
`<`	less than
`<=`	less than or equal to
`>`	greater than
`>=`	greater than or equal to

Tuple Operators

`+`	concatenation
`%+`	sum elements of tuple

Set Operators

`+`	union
`*`	intersection
`-`	difference

Boolean Operators

`and`	conjunction
`or`	disjunction
`not`	negation
`impl`	implication

Statements

```
a := expression;
```

assigns the expression to variable a

```
expression;
```

evaluate the expression

```
read <list of variables>;
```

assign data entered from keyboard to the variables successively

```
read <list of variables> from file_id;
```

assign data from file, file_id, to the variables successively

```
print <list of expressions>;
```

evaluate the expressions successively print the values line by line to the screen

```
print <list of expressions> to file_id;
```

evaluate the expressions successively write the values to the file

```
print <list of expression-format pairs>;
```

evaluate each expression, use the format to determine how to display the value on the terminal screen

```
writeln <list of expressions>;
```

evaluate the expressions successively write the values in a single line on the screen

```
if <Boolean expression> then
            <statements>;
   elseif  <Boolean expression> then
            <statements>;
   else  <statements>;
  end if;
```

Perform the statements following the first Boolean expression which evaluates true, otherwise perform the statements after else. This is a conditional control statement.

```
for <variable> in <tuple> do
            <statements>;
     end for;
```

Assign each value in the tuple to the variable and perform the statements for each assignment. This is a loop control statement.

Quick Reference to ISETL Syntax–cont.

Syntax of Funcs

```
func (<list of parameters>);
    <statements>;
   end func;
```

The statements inside the func are executed as if they are a single process. A 'return' must be included in the statements.

`return <expression>;`

evaluate the expression, the resulting value is the value returned by the func. This statement can only be used inside a func.

Predefined Funcs

Mathematical

`abs(x)`	absolute value
`ceil(x)`	smallest integer $\geq x$
`even(x)`	true if x is even
`exp(x)`	exponential function
`fix(x)`	convert x to integer
`float(x)`	convert integer x to floating point number
`floor(x)`	largest integer $\leq x$
`ln(x)`	natural logarithm
`log(x)`	common logarithm
`max(x,y)`	max value
`min(x,y)`	min value
`odd(x)`	true if x is odd
`sgn(x)`	1 if $x > 0$, 0 if $x = 0$, -1 if $x < 0$
`sqrt(x)`	square root
`sin(x), cos(x), tan(x), sec(x), csc(x), cot(x)`	trigonometric functions
`asin(x), acos(x), atan(x)`	inverse trigonometric functions
`sinh(x), cosh(x), tanh(x)`	hyperbolic trigonometric functions
`asinh(x), acosh(x), atanh(x)`	inverse hyperbolic functions

Type-Testers

`is_boolean(x);`	true if x is Boolean
`is_floating(x);`	true if x is a floating point number
`is_func(x);`	true if x is a func
`is_integer(x);`	true if x is an integer
`is_number(x);`	true if x is either an integer or a floating point number
`is_om(x);`	true if x is undefined
`is_set(x);`	true if x is a set
`is_tuple(x);`	true if x is a tuple

Input/Output

`openr(<filename>);`

open the file named <filename> for reading, return a file_id value. If file does not exist, return `OM`.

`openw(<filename>);`

open the file named <filename> for writing, destroy any existing information , return a file_id value. If file does not exist, create it.

`opena (<filename>);`

open the file named <filename> for writing, do not destroy existing information, return a file_id value.

`close(file_id);`

close the file corresponding to file_id.