## SICPy §3

## Khooi Xin Zhe, Ang Jun Jie 19 April, 2021

A SICPy¹ program is a *module*, defined using Backus-Naur Form² as follows:

## SICPy §3

module	::=	statement	statement sequence
statement	::=	name = expression	assignment statement
		function	function declaration
		return expression	return statement
		$if ext{-}statement$	conditional statement
		$while ext{-}statement$	while statement
		$for ext{-}statement$	for statement
		expression	expression statement
		<u>break</u>   <u>pass</u>   <u>continue</u>	
function	::=	$\underline{\mathtt{def}}$ name ( parameters ) :	
		$statement \dots$	function declaration
parameters	::=	$\epsilon \mid name \ [ \  extbf{,} \ name \ ] \ \dots$	function parameters
if-statement	::=	$\underline{\mathtt{if}}\ expression:$	
		$statement \dots$	
	[[	<pre>elif expression :</pre>	
		statement ]	
		<pre>else :</pre>	
		statement ]	conditional statement
while-statement	::=	<pre>while expression :</pre>	
		$statement \dots$	while statement
for-statement	::=	$\underline{\mathtt{for}}\ expression\ \underline{\mathtt{in}}\ expression$ :	
		$statement \dots$	for statement
expression	::=	number	primitive number expression
		<u>True</u>   <u>False</u>	primitive boolean expression
		None	primitive list expression
		string	primitive string expression
		name	name expression
		expression binary-operator expression	binary operator combination
		unary-operator expression	unary operator combination
		expression (expressions)	function application
		lambda name [, name]: expression	lambda expression

<sup>&</sup>lt;sup>1</sup>SICPy is an adaptation of Source - the official language of the textbook Structure and Interpretation of Computer Programs, JavaScript Adaptation.

<sup>&</sup>lt;sup>2</sup>We adopt Henry Ledgard's BNF variant that he described in A human engineered variant of BNF, ACM SIGPLAN Notices, Volume 15 Issue 10, October 1980, Pages 57-62. In our grammars, we bold and underline keywords, [] for optional syntaxes, italics for syntactic variables,  $\epsilon$  for nothing, x|y for x or y, and x... for zero or more repetitions of x.

```
expression <u>if</u> expression <u>else</u> expression
                                                                                                 conditional expression
                       list-expression
                                                                                                 list expression
                          expression \ [ \ expression \ ]
                                                                                                 list access
                       ( expression )
                                                                                                 parenthesised expression
  list-expression ::= [expressions]
                                                                                                 literal list expression
                     [ expression <u>for</u> expression <u>in</u> expression [ <u>if</u> expression ] ] list comprehension expression
                    ::= + | - | * | / | % | ==
binary-operator
                     | > | < | >= | <= | and | or
 unary\text{-}operator \ ::= \ \operatorname{not} \mid + \mid \text{-}
     expressions ::= \epsilon \mid expression [ , expression ] ...
                                                                                                argument expressions
```