Python CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
6/24/2016	1.0	Original Release	Monica Jeyasankar
7/7/2016	1.1	Removed cyclomatic complexity section. Modified examples to match current UCC	Monica Jeyasankar

Table of Contents

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program.

The following table lists the Python keywords that denote data declaration lines:

class

Table 1 Data Declaration Types

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile.

The following table lists the Ada keywords that denote data declaration lines:

import	use
do	require
from	package
no	

Table 2 Compiler Directives

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

Python comment delimiter is "#". A whole comment line may span one line and does not contain any compliable source code. An embedded comment can co-exist with compliable source code on the same physical line. Banners and empty comments are treated as types of comments. Triple quotes in Python are often considered to be multi-line comments. Actually the compiler considers these to be string literals that are not assigned to a variable. For consistency, triple quotes are counted as string literals and not comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

- An executable line of code may contain the following program control statements:
  - Selection statements (if, ? operator, switch)
  - Iteration statements (for, while, do-while)
  - Empty statements (one or more ";")
  - Jump statements (return, goto, break, continue, exit function)
  - Expression statements (function calls, assignment statements, operations, etc.)
  - Block statements
- An executable line of code may not contain the following statements:
  - Compiler directives
  - Data declaration (data) lines
  - Whole line comments, including empty comments and banners
  - Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) Lines	2	One per line	Defined in 1.4
Compiler Directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included	
Embedded	5	Not Included	
Banners	6	Not included	
Empty comments	7	Not included	
Blank lines	8	Not Included	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	“for”, “while”, or “if” statement	1	Count once	“while” is an independent statement
R02	Statemments ending by a new line character	2	Count once per statement, including empty statement	Semicolons within “for” statements are not counted. Semicolons used with R01 and R02 are not counted.
R03	Block delimiters, braces {…}	3	Count once per pair of braces {…}, except where a closing brace is followed by a semicolon, i.e., }; or an opening brace comes after a keyword “else”	Braces used with R01 and R02 are not counted. Function definition is counted once since it is followed by {…}
R04	Compiler directive	4	Count once per directive	

3. Examples

EXECUTABLE LINES

SELECTION Statements

ESS1 – if-else if-else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	if password == "pass":	1
if <expression>:	print "Access Granted"	1
<statements>		
	if password == "name":	1
if <expression>:	print "Access Granted"	1
<statement>	else:	0
else:	print "Access Denied"	1
<statement>		
	if num > 0:	1
if <expression>:	print 'positive'	1
<statements>	elif num < 0:	1
elif <expression>:	print 'negative'	1
<statements>	else:	0
else:	print 'zero'	1
<statements>		
	if x < 0:	1
if <expression>:	x = 0	1
<statements>	print 'Negative'	1
<statements>	else:	0
else:	print 'Positive'	1
<statements>		
NOTE: complexity is not considered		

ESS2 – try except finally statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
try:	try:	0
<do something>	try: 1/0	1
except Exception:	except:	1
<handle the error>	print "exception"	1
finally:	except ZeroError:	1
<cleanup>	print "divide-by-0"	1

ITERATION Statements

EIS1 – for Loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
for <expression>:	for x in a:	1
<statement>	print x,	1
NOTE: "for" statement counts as	for x in a:	1

one, no matter how many optional expressions it contains	{
	print 'x'
	}
		0
		1
		0

EIS2 – While Loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

while <boolean expression>:
<statement>
	while x <= 100:
	  print x
	  x += 1
		1
		1
		1

JUMP Statements
(are counted as they invoke action-pass to the next statement)

EJS1 - return

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

return *expression*	def knights():
	  title = 'Sir'
	  action = (lambda x: title + ' ' + x)
	  return action
	act = knights()
	print act('robin')
		1
		1
		1
		1
		1
		1

EJS2 – break

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

break	for x in range(1, 11):
	  if x == 5:
	  break
	  print x,
	print "\nBroke out of loop at x =", x
		1
		1
		1
		1
		1

EJS3 - continue

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

continue	for x in range(1, 11):
	  if x == 5:
	  continue
	  print x,
	print "\nUsed continue to skip printing the value 5"
		1
		1
		1
		1
		1

EJS4 - exit

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

os.exit(int return_code)	def outahere():
	  import os
	  print 'Bye os world'
	  os._exit(99)
		1
		1
		1
		1

	print 'Never reached' if __name__ == '__main__': outahere()	1 2

EXPRESSION Statements

EES1 – function and procedure call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name>(<paramters>);	read_file (name);	1

EES2 – explicit line joining

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<expression> \ <expression> \ <expression>	bar = 'this is ' \ 'one long string ' \ 'that is split ' \ 'across multiple lines' print bar	1 1

EES3 – implicit line joining (expressions in parentheses, square brackets, or curly braces can be split over more than one physical line without using backslashes)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

(<expression>,
…
<expression>)
 	
[<expression>,
…
<expression>]
 	
{<expression>,
…
<expression>}
	day = ['mon', 'tue',
	'wed', 'thur', 'fri',
	'sat', 'sun']
	
	def node(name):
	return {
	'Parent' : None,
	'LeftChild' : None,
	'RightChild' : None,
	'LeftRoutingTable' : list(),
	'Name' : name,
	'Level' : 0
	}
		1
		
		1
		1
		
		0

EES4 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
assignment_stmt = (target_list "=")+ expression_list target_list = target ("," target)* [","] target = identifier | "(" target_list ")" | "[" target_list "]" | attributeref | subscription | slicing	x = y name = "file1" a, b, c = 1,2,3	1 1 1

EES5 – empty statement (is counted and considered to be a placeholder for something to call attention)

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

Pass	if month == 1:
	pass
	else:
	print "late"
		1
		1
		0
		1

DECLARATION OR DATA LINES

DDL1 – data declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT

class ClassName:
<statement-1>
.
.
.
<statement-N>
	class MyClass:
	i = 12345
	def f(self):
	return 'hello'
		1
		1
		1
		1

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

MATH FUNCTIONS	TRIG	LOG	CALCULATIONS	CONDITIONALS	LOGIC	PRE-PROCESSOR	ASSIGNMENT
MATH.CEIL	MATH.COS	MATH.LOG	+	IF	==	DO	=
MATH.COPYSIGN	MATH.SIN	MATH.LOG10	-	ELSE	!=	FROM	
MATH.DEGREES	MATH.TAN	MATH.LOG1P	*	EXCEPT	>	NO	
MATH.E	MATH.ACOS	MATH.LOG2	/	FOR	<	USE	
MATH.EXP	MATH.ASIN		%	ELIF	>=	REQUIRE	
MATH.FABS	MATH.ATAN		**	TRY	<=		
MATH.FACTORIAL	MATH.ATAN2			WHILE			
MATH.FLOOR	MATH.ACOSH						
MATH.FMOD	MATH.SINH						
MATH.FREXP	MATH.TANH						
MATH.FSUM	MATH.ACOS						

	H						
MATH.HYPOT	MATH.ASINH						
MATH.LDEXP	MATH.ATANH						
MATH.MODF							
MATH.PI							
MATH.POW							