Code Styling Conventions: MTEL Digital Literacy and Computer Science Test

The following conventions are followed in the code samples found on this test. Alternative styles are acceptable in candidate-created pseudocode in the open-response assignments. Candidate-created pseudocode will not be evaluated for syntax, but instead for how well it demonstrates programming concepts and solves the presented problem.

Category	Notation	Example
| Assignment | = | ```
x = 5
y = TRUE
my_string = "Some text."
my_list = [2, 3.14, "apple"]
``` |
| Spacing | indents indicate control flow | ```
while (x != 0)
    x = x - 1
    print(x)
``` |
| Mathematics | + addition<br>    (or string concatenation)<br>− subtraction<br>* multiplication<br>/ division<br>% modulo<br>^ exponentiation | ```
x + b
"Some " + "text."
x - b
x * b
x / b
x % b
x ^ b
``` |
| Relationships | < less than<br>> greater than<br><= less than or equal to<br>>= greater than or equal to<br>== equal to<br>!= not equal to<br>is string equality | ```
x < b
x > b
x <= b
x >= b
x == b
x != b
word = "bird"
word is "bird" == TRUE
``` |
| Logic | TRUE condition true<br>FALSE condition false<br>AND both conditions true<br>OR at least one condition true<br>NOT condition not true | ```
(5 < 7) == TRUE
(5 > 7) == FALSE
(5 < 7) AND (5 == 5) == TRUE
(5 > 7) OR (5 == 5) == TRUE
NOT (5 > 7) == TRUE
``` |
| Indexes | [0] the first item in a list or string<br><br>[n] the (n + 1)th item in a list or string<br><br>[x,y] the (x + 1)th row and (y + 1)th column in a 2d array | ```
my_sentence[0]

my_list[1] #accesses 2nd element in my_list

my_array[2,5]
``` |

Category	Notation	Example

Conditionals	if (condition)
	    block
	
	if (condition)
	    block
	else
	    block
		``
		if (x >= 100)
		    print("A large number.")
		
		if (x <= 100)
		    print("A small number.")
		else
		    print("A large number.")
		``

Iterables	while (condition)
	    block
	
	for (i = a; i < b; i = i + c)
	    block
		``
		while (x < 10)
		    print(x)
		    x = x + 1
		
		for (i = 1; i < 10; i = i + 1)
		    print(i ^ 2)
		``

Functions	function function_name(arg_1,…arg_n)
	    block
	
	function_name(val_1,...val_n)
		``
		function adder(num_1, num_2)
		    return num_1 + num_2
		
		adder(3, 4)
		``

Comments	# comment line	``` # some information ```
Object- Oriented Notation	object.class_method() object.attribute	``` my_sentence.length() my_tree.grow(12) my_house.color ```

Classes	class Class_Name(parent Class_Name, if any)
	
	public/private attribute_name = default_value
	
	public/private function class_function_name(arg_1,…arg_n)
	    block
	
	class_instance = Class_Name(attr_1 = value,…attr_n = value)
		``
		class Tree(Plant)
		
		    public height = 10
		    public type = "oak"
		    private kingdom = "Plantae"
		
		    public function grow(height)
		        height = height + 2
		
		my_tree = Tree(height = 9, type = "elm")
		``