**A1)**

* Java and Python are both case-sensitive languages, so the same name with different cases can have different values.
* Java has keyword 'int' for integer, and 'double','float' for real, and these cannot be used as identifiers. Python does not have keywords for those, meaning you can use ‘int’ as an identifier.
* Java is a statically typed language, so the type of variable must be declared with the variable. Python is a dynamically typed language, meaning that the type of variable does not need to be declared by the user. It will instead be declared dep

**A2)**

Under static-scoping rules, x is 5.

Under dynamic-scoping rules, x is 10.

**A3)**

1. a - 1, b - 2, c - 2, d - 2
2. a - 1, b - 2, c - 3, d - 3, e - 3
3. a - 1, b - 2, c - 2, d - 2
4. a - 1, b - 2, c - 3

**B)**

The scope, in reference to a variable, are the locations where a variable is visible. There are three types of variables in reference to scope. Local variables which are declared in it’s block. Nonlocal variables which are declared outside of it’s block. And global variables, which are declared outside of all blocks, but can be used anywhere.

There is also static and dynamic scoping. Static Scoping is where variables declared inside a block is only visible inside that block and not outside. Dynamic scoping is where variables are visible by searching through functions that called that block.

The following are code snippets with explanations of the static scoping mechanism in the languages C, C++, and Java. They were made individually by Christopher Fallas and tested using Visual Studio Code.

C is a static scoping language. A variable made inside a block, such as a for loop, won’t be visible outside that loop.

for (int i = 0; i < 1; i++)

{  
 int inside = 5;

printf("Printing value inside the for loop: %i", inside);

}

printf("Printing value out the for loop: %i", inside);



The following code will give an error in the last line since it’s trying to print “inside”, but since it wasn’t declared inside it’s block, that variable is not visible.

Interestingly, if you declare the same variable name but outside of the loop, it will compile without errors. So in practice, you can have value “inside” be 10 when outside of the loop, and 5 when inside the loop.

C++ is a static scoping language, such as C. Thus, the local variable “inside” will not be visible after the for loop.

for (int i = 0; i < 1; i++)

{

int inside = 5;

std::cout << "Printing value inside the for loop: " << inside;

}

std::cout << "Printing value outside the for loop:" << inside;



The last print line will give an error since it’s not able it see the “inside” variable.

Java is another static scoping language, following the same rules as C and C++.

for (int i = 0; i < 1; i++)

{

int inside = 5;

System.out.println("Printing value inside the for loop: " + inside);

}

System.out.println("Printing value outside the for loop: " + inside);



The last print line will also give an error since the “inside” variable is not visible in that block.

In these three languages, if you were to declare the variable outside of the for loop, you would be able to use the variable inside and out of the loop. Although, if the variable gets changed inside the loop, it will not be reflected outside of the loop, the variable will stay the same as it was before.

This is not the same for all languages, LISP and Perl allow for dynamic scope variables. But for the most common languages nowadays, it will most likely be static scope.