Comparing python with prolog

Readability:

I think prolog has better readability than python.

Since in prolog, you can trace through the code one line by one line without going back and forward. While in python, since one function is being called multiple times, it requires to go back to the beginning of the code.

Writability:

I think python has better writability than prolog.

This is because python can write with different ways, while prolog is somewhat fixed in algorithm.

Therefore, when writing the same function, python is more fixable with the algorithms.

Maintainability:

I think python is easier than prolog.

Since in prolog, everything is chained together. If you want to change one part of the code, you will need to change the rest of the code too. While python can be written into small pieces of function, and you can change only one function as long as the input, output stays the same.

Adding nouns:

I think Both of them are pretty easy to add nouns. In python, we are adding new nouns at the end of list. For prolog, we are adding a new statement for each new noun.

Lang	Question Regarding Time to Perform Parameter-Passing	Summary	Explanation
Java	**What effect does array length have when calling function f ?**	**Call times are nearly identical, regardless of array length**	**The reason for this is that Java arrays are passed by reference. That is, the array itself is allocated on the heap and only a pointer to the array is sent to the function; the array itself is never copied onto the stack. The number of bytes in this pointer is the same for large or small arrays. Thus, the overhead of each function call is the same.**
Python	**The class MyStruct contains a list. What is the effect of the length of this list when calling the function f ?**	Call times are nearly identical, regardless of how many random numbers it generates.	When calling the f function, the length of the list will change, with what the parameter of the f function. If f(100), then the list of the self.data will become 100. But in function f, we only need the first and second variable, so regardless how long the list is, we only need the first two.
C#	**The MyStruct contains 10 doubles . What is the difference when passing this structure as a value parameter vs. a ref parameter?**	When I change the iteration into 1000000000, passing by value is 2.6 times slower than passing by reference	This is because passing by value in c#, means that we are actually creating new variables that have the same value as the original variables. This will takes a lot of time and spaces. While passing by reference, we are just using the original variable, by go to the address of the variables.
	What is the effect if the number of doubles inside the struct is switched from 10 to 20?	The calling time got twice longer, if we switch from 10 to 20	The reason why the time got extended twice is that, since the number of variable increased, the time that required to create the variable is cerated. In our case, variable increased by two, so is our calling time.
	What effect does defining MyStruct as a class instead of struct ?	The calling time of using class is 10 times faster than using struct.	The reason is that in struct, variables are stored into stack, while in class, variables are stored as reference. As we already discussed above, using reference has better performance.
C++	**For StructF , what is the effect of the array length? Consider all three parameter passing approaches.**	When passing by reference or ptr, the calling time is considered identical, while passing by value is nearly 2 times longer than fSmall	The reason is that in c++, passing by value needs to create a new variable to store the information of the same value, which takes memory and procedures. While passing by reference does not requires that.

Lang	Question Regarding Time to Perform Parameter-Passing	Summary	Explanation
	For StructG , what is the effect of the array length? Consider all three parameter passing approaches.	The calling time is nearly identical for passing by reference, pointers and value	This is because for StructG, the double data is a pointer. Therefore, no matter how big the array is, the only thing needs is just a pointer. So regardless the length of the array, the calling time is nearly the same.
	For pass by value, how do StructF and StructG compare, given identically sized arrays?	The calling time of structF is 2 times slower than structG	This is because structF is passed by value in the function, while structG is passed by reference as in the pointer. We already discussed the reason why pointer is faster than value above. For F, the value is being called on the stack, while for G, the value is called by the reference, which is the address of the value in the memory.
	For pass by reference, how do StructF and StructG compare, given identically sized arrays?	The calling time for either the big array size or small array size, are the nearly the same	When passing by reference, since G is already passed by pointer in the function, the calling stays the same. The change is in struct F, changed from passing by value to passing by reference. Right now two functions are all passed by reference. Therefore, the calling time is nearly the same.
	What is the effect of defining StructF as a class instead of struct ? Does this match with C#?	The calling time is nearly identical.	The reason is that the difference between struct and class in c++ is that struct has default public member and bases, and classes have default private members and bases. Therefore, it does not affect the performance of functions a lot.