Ariel Valencia

Python Analysis

1. interpretation

Python is a purely interpreted language. This means that another program called the interpreter, interprets the program. The interpreter acts of a simulation of machine whose fetch-execute cycle deals with high-level language rather than machine instructions

It’s interesting that the code just runs without being compiled. It makes a lot faster to see how the code works but I don’t like that some errors are not caught until the program runs.

1. Boolean expressions

Both Java and Python use Boolean expressions to compare objects and both evaluate &&(and), ||(or) and !(not). In order to show that x must not be equal to y we write

not(x is y)

The usage is similar which I like since it makes it easy to pick up. I wish however that it used symbols instead of words since most other languages do that.

1. short circuit evaluation

Java and Python both do short circuit evaluation for the operators &&(and), ||(or). What this means is that the operator does not necessarily evaluate all of the operand. For example, in evaluating the statement 3==4 && 5+4=9, 3==4 is false so the short circuit operator stops evaluating the statement since for the statement to be true, both 3==4 and 5+4=9 must be true.

Similarly to my critique on Boolean expressions, I wish symbols where used instead of words to make it easier to remember.

1. numeric types

Numeric types in python are integers, floats, complex numbers and Booleans. This very is different from Java which has have byte, short, long, float, double and char.

I prefer the python numeric types since it’s much less to remember. It’s nice to control the size of the number but a pain to have to decide what number type I want to use.

1. strings

Strings in python have many features that are not included in Java. These features allow you to manipulate strings and process them. It also methods that are supposed to change the string does not actually change it since they are immutable.

There are so many features, that it is a bit overwhelming compared to Java, but it make the language way more powerful and allows much more flexibility.

1. arrays

While you can use an array data structure in Python, it is not native to the language. Python uses lists rather than arrays which are very similar. I think it was a good idea to leave them out since lists are very similar and offer more features than arrays.

1. lists

Lists work similarly to arrays and are native to Python. There are a few features that make them different from arrays. One difference is that while you can access an array from its index, you can also access an element using a negative index. For example, with x = [1,2,3], you can access the number 3 with x[2] or with x[-1]. You can also add elements to the list which you cannot with an array in Java and C++. Another nice feature of lists is that you can add any type of object. With arrays in Java and C++ you can only add one type of object and you will get an error if you try and add a string into an integer array.

I think lists are much more useful than an array since the size of the array can changed easily and it can old different types of objects rather than having to create a specific arrays that can only hold one type of object. It is also nice that if you want to go from a tuple to a list, there is a very simple way to do so.

1. tuples

Tuples are similar to lists except that they cannot be changed once they have created. In this aspect, tuples are similar to arrays in Java and C++. Tuples can also have elements that are different objects unlike arrays in java and C++. Tuples are mainly used as keys for dictionaries.

Since tuples have similar features as lists and since I like lists, I think they are a good feature. And since you can easily change tuples to lists and vice-versa, tuples are very useful.

1. slices

Slice is a feature that Java and C++ do not have. It is nice because it creates new list or tuple with the elements you want removed, removed.

This is an interesting feature but I feel like it’s not that useful unless you keep track of what is in the array and where it is.

1. index range checking

Since python is an interpreted language, it does not offer index range checking the way Java and other compiled languages do. What this means is that if an index that doesn’t exist is being accessed, then python will not find the error until it run the program.

This is a feature that would be nice to have but not really necessary since it is an easy mistake to catch.

1. dictionaries

Dictionaries are similar to hashes in Java. One of the differences is that the keys can be objects. Also, values can be accessed by both a key and indexes. Values in the dictionary are not store in any particular order because the keys are not only numbers.

Dictionaries are useful storing various types of objects and count words in a file. I find it to be a very powerful feature of the language.

1. if statement

Like in Java and C++, python has an if statement. It works the same except for in python you can choose to pass if would like to skip the body of the if statement.

It’s a handy feature to have in python and it was implemented well.

1. switch statement

Python has no switch statements unlike java and python. I don’t like that this was left out since it makes it easier when you have a loop for which you want to have evaluated differently for different instances. It makes cleaner to code instead of having multiple nested loops.

1. for loop

The for loop in python is a little different than how it works in Java or C++. Rather than having a variable be incremented throughout the duration of the loop, Python iterates from the values returned from an iterable object like a sequence.

I like that the for loop works a little differently in python. It’s convenient since you can go through a structure like a list without having to create a variable to increment and iterate your loop. You can instead use the list its self to iterate through the list.

1. while loop

The while loop in python is the same as it is in Java and C++. It uses indentation to delimit rather than braces. The statements in the body are separated by newlines that are the same level of indentation as the rest of the body.

While loops are useful and I am glad that it was put into python.

1. indentation to denote code blocks

Python uses indentation to delimit different blocks of the control-flow constructs. They also differentiate functions. The indentation is used in a similar way that braces and brackets are used in Java and C++. If the code is not indented correctly, errors occur in the interpretation similar to having forgotten a brace in Java.

I like using indentation rather than braces to delimit because then you don’t have to search 300 lines of code looking for one missing brace. However, it is very different from what Java and C++ coders are used to so it may take a while to get the hang of. It is also very easy to read. You have to indent correctly otherwise it will not work. However in Java and C++, indentation is preferred but not required. Since not everyone, indents there code well, it is nice that it has to be in python.

1. type binding

Variables in python are not bound to a specific type. So when a variable is created with an integer and later it is changed to string, that is perfectly legal. This is very different from Java and C++ since a variable has to be declared as a specific type.

It is nice not to have to keep track of variable types in code but requires the coder to remember what type was assigned to a variable, otherwise an illegal operation may be performed and ma cause an error later.

1. type checking

Type checking in python doesn’t happen at compile time. Variables are not required to reference objects of a certain type. Which means that at a variable can hold an integer one minute and a string the next. Since Java and C++ require a variable be of a certain type once declared, they need to type check at compile time to make sure the correct type was assigned to the variable.

I like this since you can reassign a variable if necessary without having to create a new variable. I don’t like that it eliminates a free check in my code but that is a small price to pay for the flexibility python variables allow.

1. functions

Like methods in Java, functions create a definition for a specific statements that you may want to have occur in the program more than once. These can have a return statement or not which means that it returns the value none.

Having functions is extremely useful as it saves from a lot of repetitive typing and makes code much cleaner and easier to read. It is very straight forward to use since it is similar to other languages.

1. garbage collection

Like Java, python also uses garbage collecting to reclaim objects that are no longer needed and frees up memory. This is a feature that C++ does not have.

It’s a nice feature to have in python since you as a programmer you don’t have to worry about garbage.

Overall critique of Python

While it took some getting used to, I found the language to be simple to learn, easy to understand and very useful. These are all qualities that make a new language appealing. There were something like the lack of arrays and index bounds checking but the good features outweigh those that were left out. I am sure that with time I would get used to the changes and find it a very fun and useful language to program in.