Python is better for the beginner

Lindsey Wingate

Introduction

Learning how to program is difficult. Even the simplest program is challenging to a beginner who is developing new problem solving skills. I decided to write this paper when I discovered UND's Computer Science department was moving away from using Python in beginner courses such as CSci 160.

Background

This report will focus on code examples and analysis of C, Python, and Java. The oldest language of the three, C, was "designed and implemented by Dennis Richie at Bell Laboratories in 1972" (77, 1). It was designed for systems programming and did not emphasize type checking when first released. For its age, C has remained one of the most commonly used computer languages.

Java was created in 1990 to fill the need for a reliable, portable language. It was needed to reduce recalls of mass-produced consumer electronic devices (91). C was not sufficient for the task mainly because it did not support object oriented programming. Another significant different between C and Java was the garbage collector, or "implicit storage deallocation for its objects" (93, 1). Garbage collection was a very important aspect of Java, eliminating the need to free pointers and guaranteeing a memory-leak free program.

Similarly, garbage collection was implemented in Python, which was also a product of the early 90's. Unlike C and Java, Python is an interpreted scripting language and is easily extended by any user (99, 1). It was originally developed by Guido van Rossum with extensibility, error handling, statement grouping, and inclusion of high-level data types in mind (2).

Analysis

According to the UND Computer Science Wiki page, CSci 160 intends to provide "an introduction to computer science, with problem solving, algorithm development, and structured programming… Emphasis on learning how to design, code, debug, and document programs, using techniques of good programming style." Python is the best language to accomplish all of these goals for a beginning programmer.

Confidence in the ability to create a working program is essential. Grasping concepts such as algorithms and learning to decode are more difficult with complex syntax. The words used in a computer language need to be similar to concepts students already understand without having previous coding experience. For example:

Python
```
print "Hello, World!"
```

C
```
#include <stdio.h>
int main() {
   printf("Hello, World!");
}
```

Java
```
Public class HelloWorld {
   Public static void main(String[] args) {
      System.out.println("Hello, World!");
   }
}
```

Looking at these examples through a beginner's eyes, the Python program is straightforward. The only concept required to understand this program is how *print* shows whatever is within the quotes on the screen when the program runs. C requires an explanation of functions and libraries. Java is more complicated. A student looks at this code and thinks "What does public mean? What is a class? Also, what is String[] args? Not to mention it takes a lot more work to type System.out.println." By the time a beginner reaches the "Hello, World" in the Java example they already prefer Python or C. Java and C are not too complicated for a beginning student, but Python is simpler. It could be argued that curly braces are easier to use instead of Python's required indentations, however it is generally accepted that code will be indented for organizational purposes regardless of language requirements. Python is a good choice for beginners because it is simple and it enforces the habit of indenting code and maintaining consistency throughout the program.

Beginners prefer dynamically typed languages because it only requires one round of trouble shooting. Compilation may improve running times but speed is not the priority with small programs; what does matter are clear error messages when debugging. In C, the most common error found is "segmentation fault," which indicates the program is attempting to access unallocated memory. This message is broad and can be extremely frustrating when attempting to track down the issue. Java has more specific error messages such as "IOException or HeadlessException," yet further explanation is still needed. Python has the best error messages for beginners of the three languages. Message errors can include SyntaxError, TypeError, and NameError. Immediately, the programmer knows what the error message means instead of having to reference another source.

When it comes to built-in error handling, Python and Java are ahead of C. C has libraries that can be added to the code for error handling, but they are not as descriptive. Java and Python both use "try" statements. See the examples below:

Python
```
try:
   file = open('testfile.txt', 'w')
except IOError:
   print "Error: File doesn't exist."
```

Java
```
public class Test {
   public int test(File f) {
      try {
         File file = new File("data.txt");
         Scanner input = new Scanner(file);
      }
      catch(FileNotFoundException e) {
         System.out.println("The file was not found");
      }
```

C
```
int main() {
   File* pf;
   int errnum;
   pf = fopen("datatxt", "rb");
   if (pf==NULL) {
      errnum = errno;
      fprintf(stderr, "Error opening file: %s\n",
         strerror(errnum));
   }
   else {
      fclose(pf);
   }
   return 0;
}
```
(3)

In these cases, the error handling is simplest in Python. Java makes sense once the student has learned to create a scanner object, but C is not intuitive. In most situations, the error has to be identified and the proper method must be used to print the result. This example also presents pointers, which can add further complexity for a beginner. Also, C pointers can be very dangerous once they are used and have not been freed properly. Java and Python are better suited for beginners because the garbage collectors will ensure memory leaks do not occur. Also, it helps students not crash computer systems.

Python, Java, and C were all designed with structured programming in mind. Each language has access to functions, various loops and control statements. See the example below:

C
```
int x;
for(x=0; x<sizeof(“Lindsey”); x++) {
   printf(“%c\n”, name[x]);
}
```

Java
```
for(int x=0; x<”Lindsey”.length(); x++) {
   System.out.println(name.charAt(x));
}
```

Python
```
for x in ‘Lindsey’:
   print x
```

The for loops in C and Java require an initialized counter, a base case for when the loop should stop, and the increment for each iteration of the counter. Python is much more flexible because the programmer simply needs to indicate the iterating variable within the object with an iterable method. Once Python reaches the end of the iterated object it stops. A common error with C and Java is starting or ending the loop before or after it should, causing more errors.

Conclusion

Python is the perfect language to allow students develop self-confidence early on in their careers. The examples provided above are only a sample of Python’s capabilities. It’s simple syntax and flexibility with different data types allows them to learn the basics of data structures, data types, code organization, and debugging processes. Although C and Java are useful languages, they both introduce topics that are unnecessary for beginners; they require a more active understanding of compilers, objects, data types, and specifics on how the language communicates with the computer. These concepts can be learned with time but exaggerate the learning curve for novice programmers. The initial months of coding and programming concepts should be taught with Python.

References:

1. Concepts of Programming Languages: Tenth Edition. Robert W. Sebesta.
2. General Python FAQ: https://docs.python.org/2/faq/general.html. Python Software Foundation.
3. Tutorials Point: http://www.tutorialspoint.com/cprogramming/c_error_handling.htm. Tutorials Point (I) Pvt. Ltd.