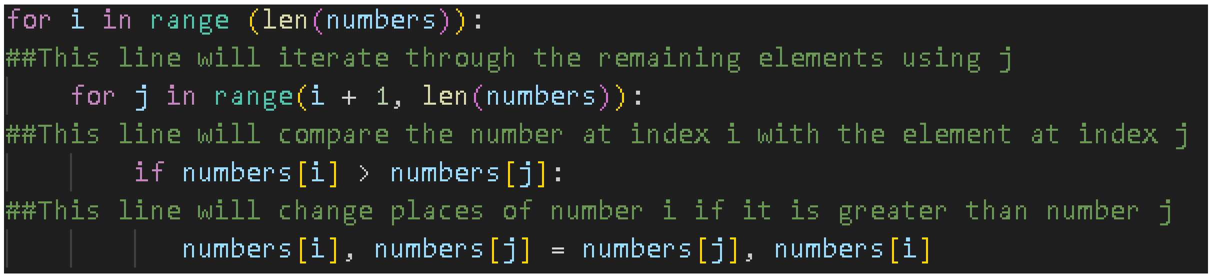
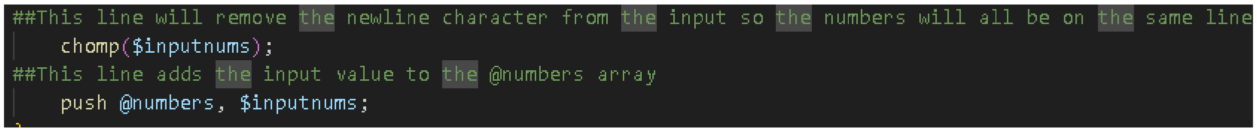
Readability:

Python: In terms of readability, I find that sort.py is more straightforward and easier to understand. Python's use of indentation for code blocks makes it clear, and the use of straightforward variable names like numbers, i, and j adds readability. I can quickly understand the code's structure and logic just by looking at the code. Adding on that idea, Python's lack of special characters in variable names simplifies reading the code. For example, In the code below, I can quickly understand that the numbers are being compared due to the position they are in the array and will be switched around.



```
for i in range (len(numbers)):
##This line will iterate through the remaining elements using j
    for j in range(i + 1, len(numbers)):
##This line will compare the number at index i with the element at index j
        if numbers[i] > numbers[j]:
##This line will change places of number i if it is greater than number j
            numbers[i], numbers[j] = numbers[j], numbers[i]
```

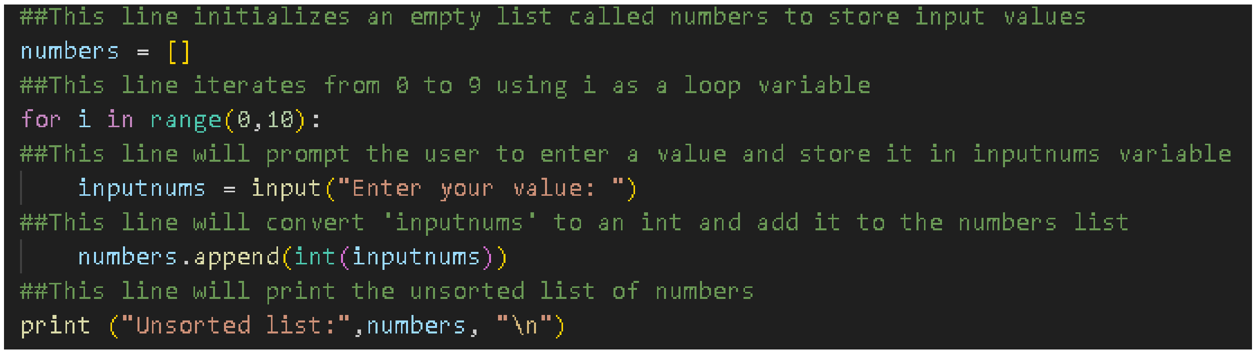
Perl: On the other hand, sort.pl may be less readable to those not familiar with Perl's syntax like me. It was my first time using this language and the use of special characters like @ and $ for variables made the code appear cluttered. Also, the reliance on built-in functions like chomp and Perl-specific syntax can require additional effort to understand. While Perl can be straightforward, it might benefit from more comments to improve readability. This point can be shown in this part of the program that requires words like chomp and push which are helpful in the code. Without comments, the user might not know what is going on.



```
##This line will remove the newline character from the input so the numbers will all be on the same line
    chomp($inputnums);
##This line adds the input value to the @numbers array
    push @numbers, $inputnums;
```

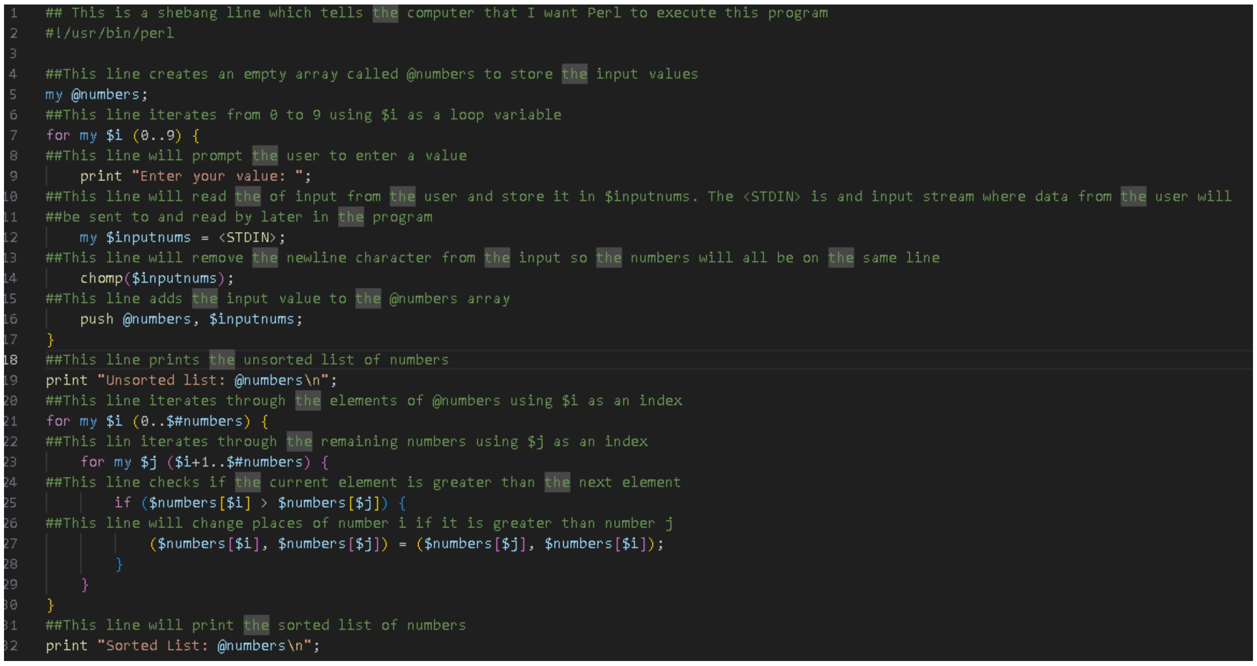
Writability:

Python: My file might have slightly longer code due to Python's emphasis on readability. However, it compensates for this with the self-explanatory code. The use of clear variable names and structured indentation makes it easy to write code that is both understandable and maintainable. I find that Python's writability is much better compared to Perl. This is shown in the example below with printing the unsorted list based on the user input.



```python
##This line initializes an empty list called numbers to store input values
numbers = []
##This line iterates from 0 to 9 using i as a loop variable
for i in range(0,10):
##This line will prompt the user to enter a value and store it in inputnums variable
    inputnums = input("Enter your value: ")
##This line will convert 'inputnums' to an int and add it to the numbers list
    numbers.append(int(inputnums))
##This line will print the unsorted list of numbers
print ("Unsorted list:",numbers, "\n")
```

Perl: Regarding writability, I noticed that sort.pl (Perl) is relatively short and concise, which is a characteristic of Perl. It allows me to perform various operations, such as reading input, modifying arrays, and swapping elements, with minimal code. However, this ideology came at the cost of self-documentation. I had to add more comments to clarify the code's functionality:

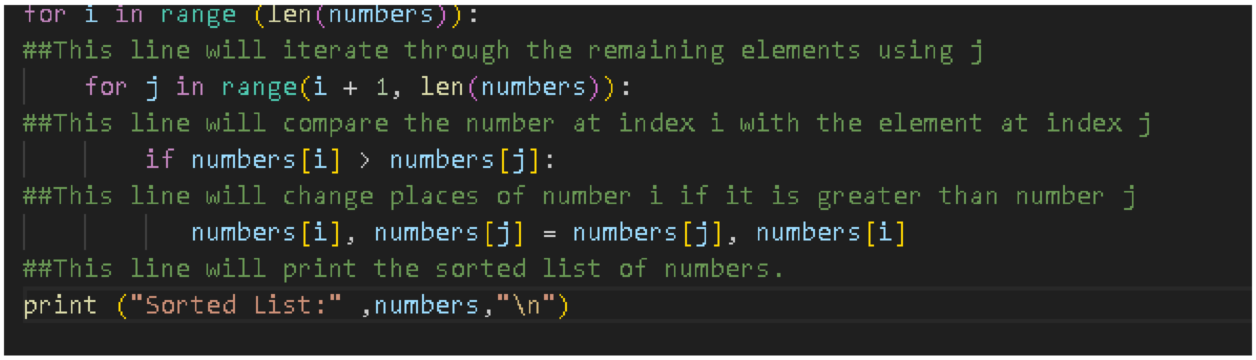


```perl
## This is a shebang line which tells the computer that I want Perl to execute this program
#!/usr/bin/perl

##This line creates an empty array called @numbers to store the input values
my @numbers;
##This line iterates from 0 to 9 using $i as a loop variable
for my $i (0..9) {
##This line will prompt the user to enter a value
    print "Enter your value: ";
##This line will read the of input from the user and store it in $inputnums. The <STDIN> is and input stream where data from the user will
##be sent to and read by later in the program
    my $inputnums = <STDIN>;
##This line will remove the newline character from the input so the numbers will all be on the same line
    chomp($inputnums);
##This line adds the input value to the @numbers array
    push @numbers, $inputnums;
}
##This line prints the unsorted list of numbers
print "Unsorted list: @numbers\n";
##This line iterates through the elements of @numbers using $i as an index
for my $i (0..$#numbers) {
##This lin iterates through the remaining numbers using $j as an index
    for my $j ($i+1..$#numbers) {
##This line checks if the current element is greater than the next element
        if ($numbers[$i] > $numbers[$j]) {
##This line will change places of number i if it is greater than number j
            ($numbers[$i], $numbers[$j]) = ($numbers[$j], $numbers[$i]);
        }
    }
}
##This line will print the sorted list of numbers
print "Sorted List: @numbers\n";
```

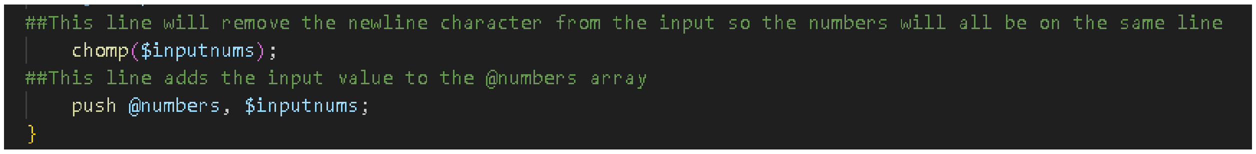
Expressiveness:

Python: In my opinion, Python places a strong emphasis on code readability and expressiveness. In the sort.py file, the code is clear and self-explanatory, making it easy to understand the sorting algorithm and the input/output process. Python's emphasis on clean and expressive code can lead to reduced debugging time and better maintainability with many other projects vs. this one.



```python
for i in range (len(numbers)):
##This line will iterate through the remaining elements using j
    for j in range(i + 1, len(numbers)):
##This line will compare the number at index i with the element at index j
        if numbers[i] > numbers[j]:
##This line will change places of number i if it is greater than number j
            numbers[i], numbers[j] = numbers[j], numbers[i]
##This line will print the sorted list of numbers.
print ("Sorted List:" ,numbers,"\n")
```

Perl: In my sort.pl script, I can appreciate Perl's built-in functions, such as chomp, and its ability to manipulate arrays directly through the push operation. These built-in functions significantly contribute to its expressiveness. I've also noticed that the presence of special characters and their brief syntax occasionally demand additional effort for readability, especially for people not familiar with the language.



```perl
##This line will remove the newline character from the input so the numbers will all be on the same line
    chomp($inputnums);
##This line adds the input value to the @numbers array
    push @numbers, $inputnums;
}
```