Comparison of running time of a program in different languages:

Program:

```
s=0
for i from 1 to 9999 do:
        for j from 1 to 9999 do:
                s=s+i/j
```

In Python:

```
import time
start_time = time.clock()
s=0
for i in range(1,9999,1):
       for j in range(1,9999,1):
              s=s+i/j
print (time.clock() - start_time, "seconds")
```

The output is: **20.980922 seconds**

In R:

```
start_time <- Sys.time()
s=0
for(i in 1:9999)
  for(j in 1:9999)
    s=s+i/j
end_time <- Sys.time()
x=end_time - start_time
x
```

The output is: **Time difference of 4.39877 secs**

In Java:

```
import java.util.*;
public class Time
{
       public static void main(String args[])
       {
              try
             {
                long start = System.currentTimeMillis( );
                long s=0;
                for(long i=1;i<10000;i++)
                {
                    for(long j=1;j<10000;j++)
                    {
                           s=s+i/j;
                    }
                }

                long end = System.currentTimeMillis( );
```

```
            long diff = end - start;
            System.out.println("Difference is " + diff/1000+" seconds");
        }
        catch (Exception e)
        {
            System.out.println("Got an exception!");
        }
         }
    }
```

The output is: **Difference is 2 seconds**

In C++:

```
#include <stdio.h>
#include <time.h>
int main ()
{
  time_t timer;
  struct tm y2k = {0};
  double seconds;
  time(&timer);
  float s =0;
  float i=1,j=1;
  for(i=1;i<10000;i++)
    for(j=1;j<10000;j++)
      s=s+i/j;
  time_t timer2;
  time(&timer2);
  seconds = difftime(timer2,timer);
  printf ("Difference is %.f seconds", seconds);

  return 0;
}
```

The output is: D**ifference is 1 seconds**

So, we see from this example that the execution time for same code in different languages is in the order:

Python > R > Java > C++

After seeing these results, I have a question in my mind "**If python is so inefficient then why most of the development in the field of machine learning and deep learning is in python?**"

One answer that I found is – Python performs vectorized operations. Using the CPU and GPU capabilities python performs SIMD (single instruction multiple data) instructions. Source: https://www.coursera.org/learn/neural-networks-deep-learning/lecture/NYnog/vectorization

Example:

```
import numpy as np
import time
a=np.random.rand(1000000)
b=np.random.rand(1000000)
```

```
start = time.time()
c=np.dot(a,b)
end=time.time()
print("Vectorized version: "+str(1000*(end-start))+" ms")

start =time.time()
c=0
for i in range(1000000):
      c+=a[i]*b[i]
end = time.time()
print("For loop version: "+str(1000*(end-start))+" ms")
```

The output is: **Vectorized version: 1.050710678100586 ms**
For loop version: 563.647985458374 ms

From this example we see that the vectorized computation in python is almost 500 times faster than running the same program using traditional for loop.

In Java:

```
import java.util.*;
public class LoopTime
{
      public static void main(String args[])
      {
            try
     {

        int[] a = new int[1000000];
        Random rand = new Random();
        for(int i=0;i<1000000;i++)
        {
            a[i]= rand.nextInt();
        }
        long start = System.currentTimeMillis( );
        long c=0;
        for(int i=0;i<1000000;i++)
        {
           c+=a[i]*a[i];
        }
        long end = System.currentTimeMillis( );
        long diff = end - start;
        System.out.println("Difference is " + diff+" ms");
     }
     catch (Exception e)
     {
        System.out.println("Got an exception!");
     }
      }
}
```

The output is: **Difference is 4 ms**
Clearly, vectorized operation in python is still more efficient than for loop in java.

In C++:

```
#include <stdio.h>
#include<iostream>
#include <ctime>
using namespace std;
int main ()
{

  int array[1000000];
  for( int i=0;i<1000000;i++)
  {
    array[i]=rand();
  }
  int start_s=clock();
  long c=0;
  for(int i=0;i<1000000;i++)
  {
    c+=array[i]+array[i];
  }
  int stop_s=clock();

  cout << "Difference is " << (stop_s-start_s)/double(CLOCKS_PER_SEC)*1000 <<"
ms"<< endl;

  return 0;
}
```

The output is: **Difference is 2.908 ms**
This shows that the vectorized version of python is more efficient than C++ loop version.

Therefore, this justifies the usage of vectorized code in python over other languages for running compute extensive code.

Credits: I want to thank my advisor Dr. Ovidiu Daescu for encouraging me to run this experiment.