Assignment 4

L.A. Beukers: 5355281 | P.Wahi : 918540

Git Repository : https://github.com/lavibes/AS-4.git

Number of words : 514

Difference between Python and c++

At first sight, it was seen that the Python code file contained fewer lines than the C++ code file. Due to the static datatype of cpp and the dynamic datatype of python, there is a notable difference in how data is represented in the two programming languages. In addition, unlike Python, C++ functions require a return type during their definition, such as "void advance" or "double energy." Lastly, the representation of data linked to planetary bodies was more complicated in cpp than in python. The initial classes for vector 3d and body class were written in C++ to define the vectors' names, position, velocity, and mass. After this was accomplished, the body class was used to define the list/ 1d array containing the attributes of each planet's body. In contrast, using dictionaries in python, the entire procedure was accomplished efficiently by mapping the key (planet names) to the value (position, velocity and mass).

Reflection on assignment

For each iteration, the task required reading the positions of the five planetary bodies and appending them to a list or 1d array. The list was then traversed to write each element to a CSV file for each iteration. Both Python and C++ scripts utilised the same logic.

In previous versions of the code, the runtime was only measured for the specified number of iterations. However, the runtime should be calculated for the entire code, including the time required to write the CSV file. Therefore, a separate Python script, "benchmark script," was developed in which the user is first prompted to specify the script type, either Python, CPP_debug, or CPP_release. The script uses time.perf counter as a stopwatch to record the start and end time of the user-selected Python script or both CPP.exe files once input is received. The difference between the start and stop times yields the elapsed time for each of the four iterations, which is then used to generate the graph below. However, the.exe file generated in debug mode does not execute flawlessly. Therefore, the number of iterations was hard-coded into the.exe file created in debug mode to obtain the desired results.

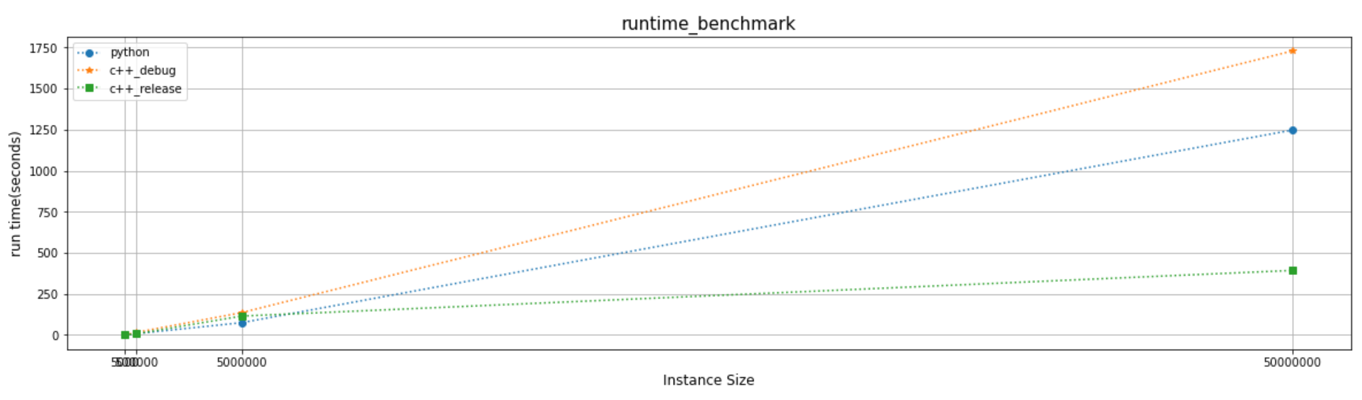
The git collaboration between the group members worked wonderfully, especially considering that one of the group members is working from abroad. The git platform provides sufficient tools for comparing the modifications made by two group members for effective collaboration. When stymied, Python, CPP, and Git-savvy friends were consulted. For syntax and conceptual issues, geek-to-geek websites or stack overflow was consulted.

The lack of knowledge in the nbody field makes it difficult to determine whether the screenshots provide satisfactory results. Concerning the distinction between an interpreted and a compiled language, it was observed that Python is slower than the CPP script in release mode but faster than the CPP script in debug mode. The CPP handled memory allocation more efficiently. For example, with 50,000,000 iterations, Python consistently crashed and could only be resolved by using a computer with a higher hardware configuration.

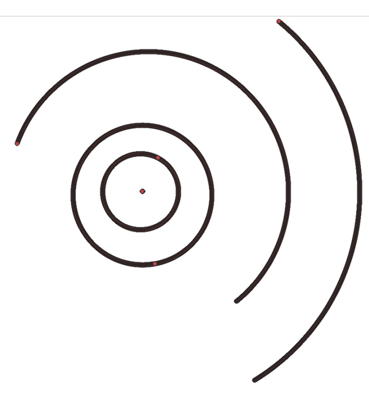
Runtime benchmarks (runtime in seconds)

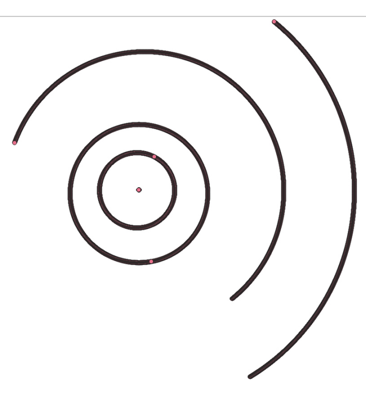
	Run time (seconds)		
Instance Size	**Python**	**C++(debug)**	**C++(release)**
5000	0.45	0	0
500000	8.2	16	7.2
5000000	74.8	136	115
50000000	1248	1730	393

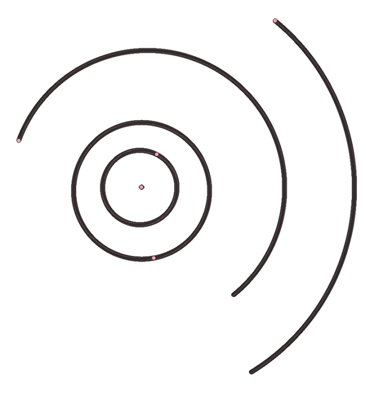
In [83]:
```python
import matplotlib.pyplot as plt

#setting the size of plot figure
fig = plt.figure()
fig.set_figheight(5)
fig.set_figwidth(20)

# instances on x axis
instance_size = [5000, 500000, 5000000, 50000000]

# runtime on y axis
#python
runt_py = [0.45, 8.2, 74.8, 1248]
#cpp_debug
runt_cppd = [0, 16, 136, 1730]
#cpp_release
runt_cppr = [0, 7.2, 115, 393]

#plotting lines
plt.plot(instance_size, runt_py, marker ='o', ls = ":")
plt.plot(instance_size, runt_cppd, marker = '*', ls = ":")
plt.plot(instance_size, runt_cppr, marker = 's', ls = ":")

#labelling
plt.title("runtime_benchmark", fontsize = 15)
plt.xlabel("Instance Size", fontsize = 12)
plt.ylabel("run time(seconds)", fontsize = 12)
plt.legend(['python', 'c++_debug', 'c++_release'])

#x ticks and grid
plt.xticks(instance_size, instance_size)
plt.grid()

#showing the graph
plt.show()
```



Overview of CSV files from QGIS


Python_5000


CPP_debug_5000


CPP_release_5000