JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

EN.540.635
Software Carpentry

Lecture 10
Efficiency | Big O | Compiled Languages

Efficiency

- It is possible to write a program which could technically run forever, even though it is not technically wrong

- An efficient program is related to how the implemented algorithm uses computational resources:
  - Memory
  - Time

- You will not be writing your own algorithm; you will be implementing algorithms already developed for other problems

Order of Growth

- To compare the speed or time efficiency of algorithms, we might try comparing the time they take to complete

- The above metric is influenced by:
  - Speed of the machine
  - Python implementation
  - Input size

- Efficiency in the end depends on the number of steps needed to execute on a given size of input

Order of Growth

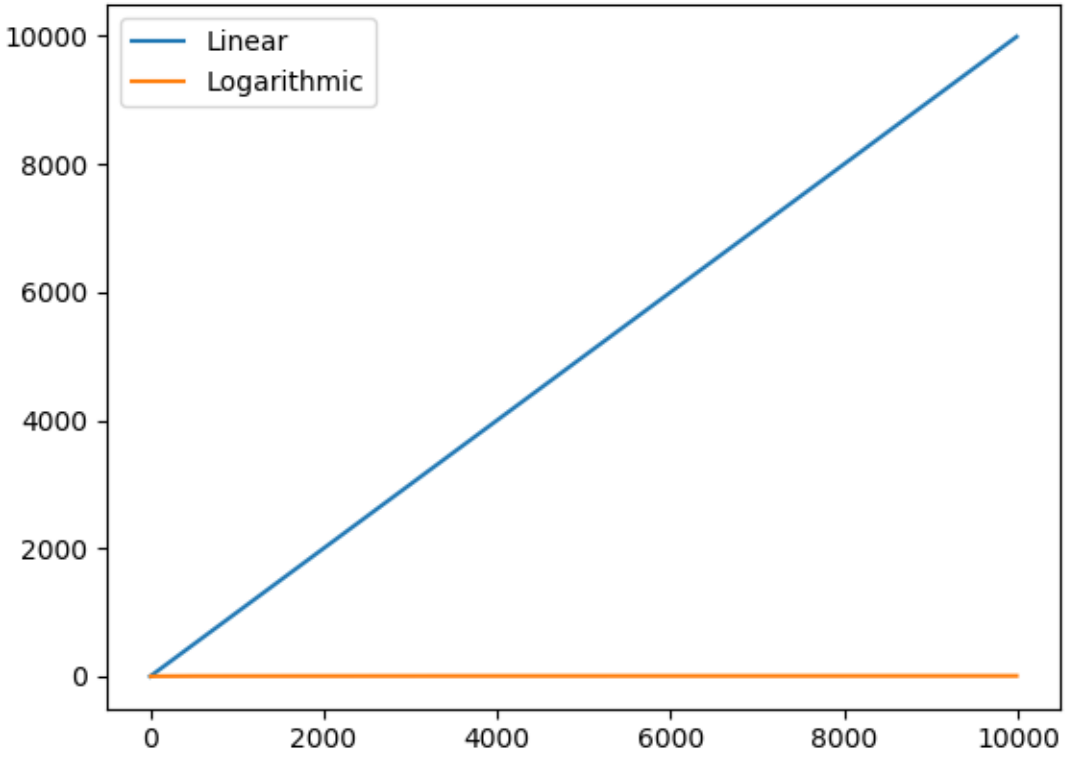
- How does a problem scale?
- Asymptotic growth models are used to calculate the growth, as it approaches a limit on the size of the input
- Useful to define some parameter describing how something scales
  - O notation – Upper bound
  - Ω notation – Lower bound
  - Θ notation – Exactly bound

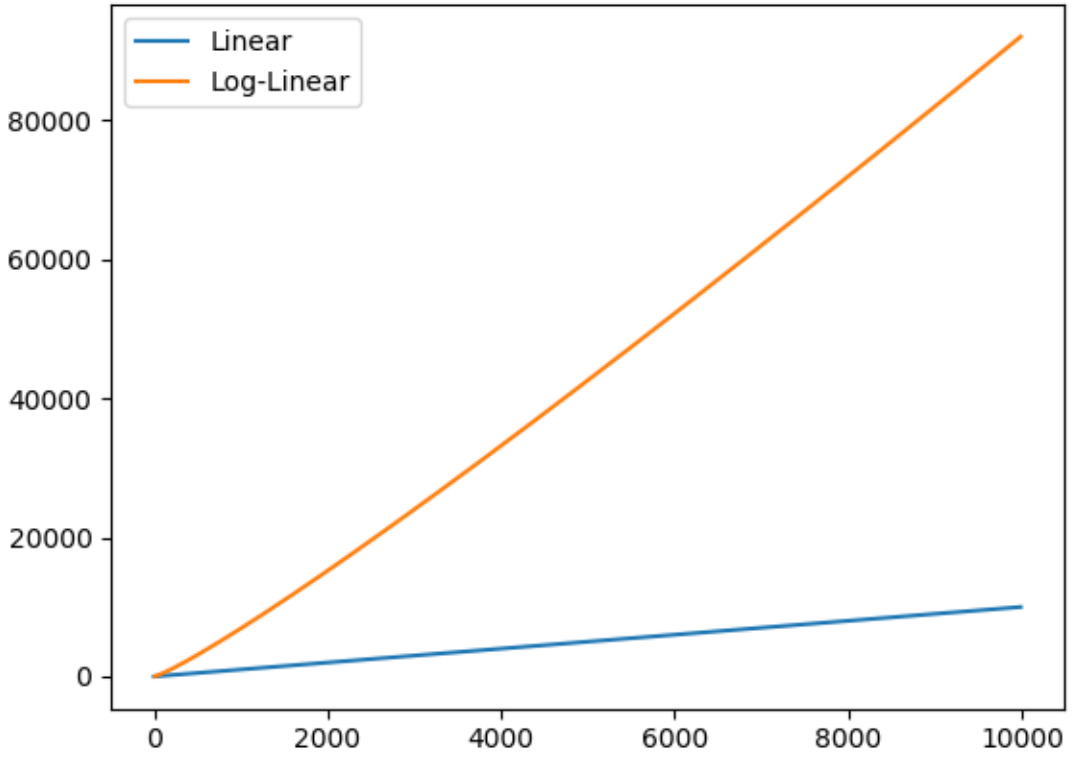
Big O Notation

- Worst case scenario and is the conventional notation used by computer scientists studying algorithms

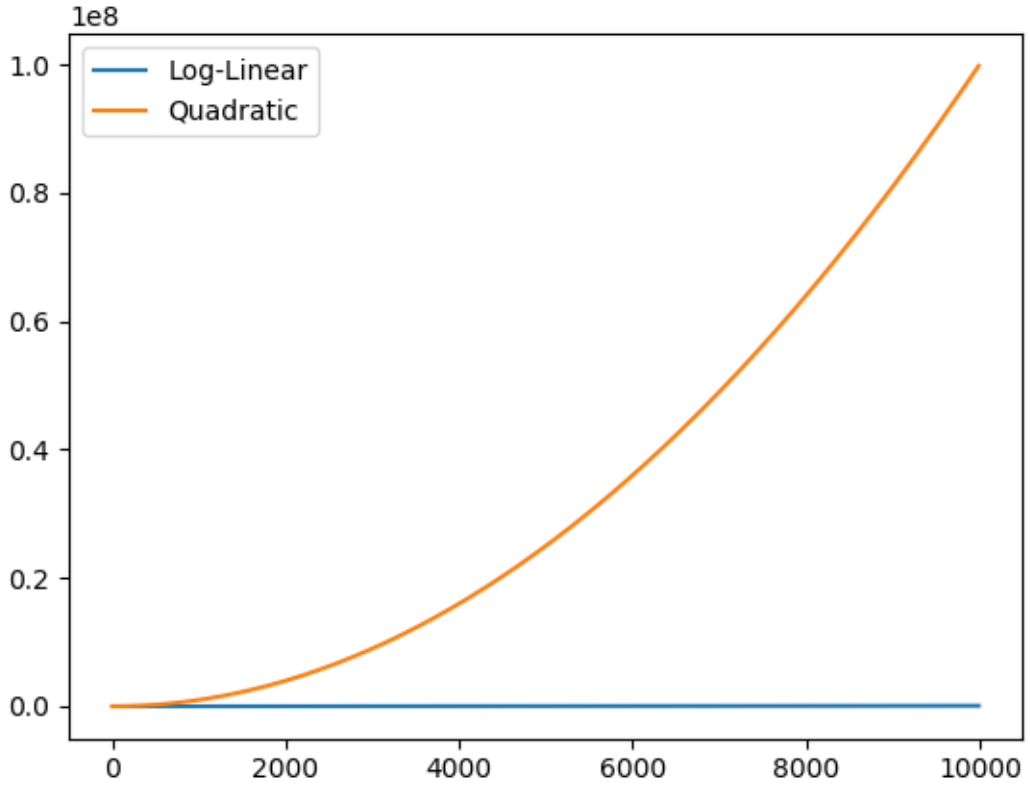
Fit Functions	Big O Notation
$y = 4 * x$	$O(N)$
$y = 12^{99999} * x$	$O(N)$
$y = 4 * \log_{10} 4x$	$O(\log(N))$
$y = x^4 + 12$	$O(N^4)$
$y = \mathrm{x} * \log_{10}(\mathrm{x} + 4) + \mathrm{x}$	$O\big(Nlog(N)\big)$

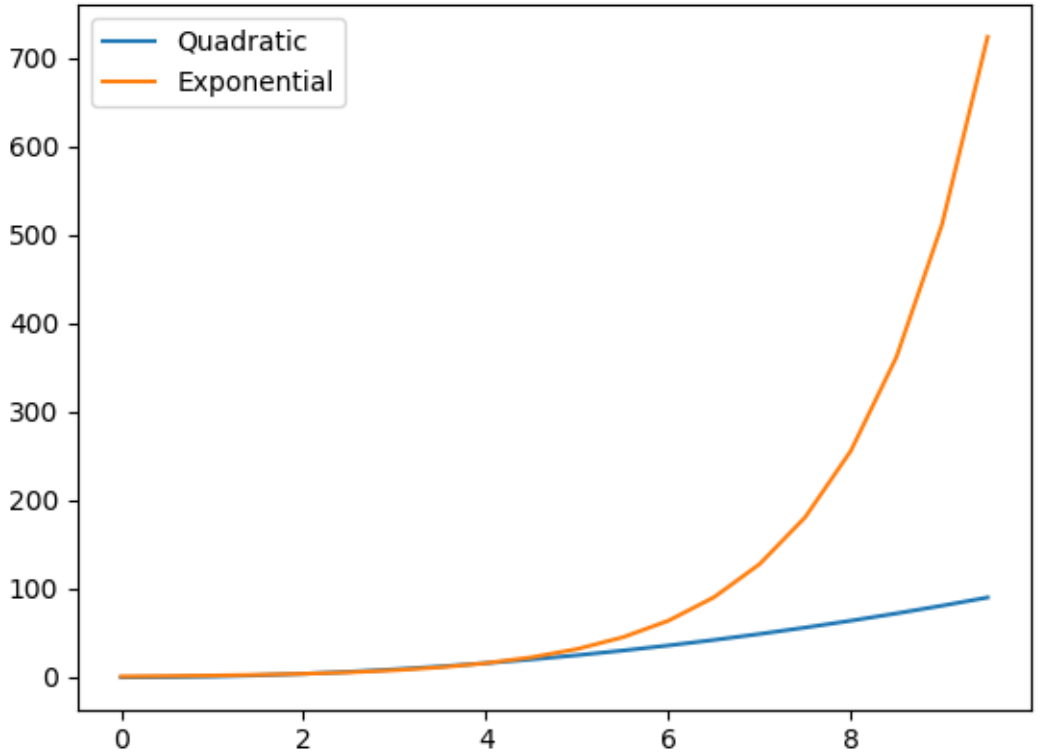
Examples


Linear
Logarithmic
10000
8000
6000
4000
2000
0
0
2000
4000
6000
8000
10000


Linear
Log-Linear
80000
60000
40000
20000
0
0
2000
4000
6000
8000
10000

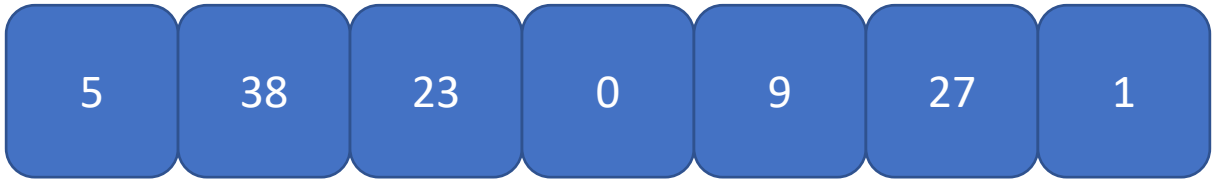
Examples


1e8
1.0
0.8
0.6
0.4
0.2
0.0
Log-Linear
Quadratic
0
2000
4000
6000
8000
10000


700
600
500
400
300
200
100
0
Quadratic
Exponential
0
2
4
6
8

Sorting Algorithms

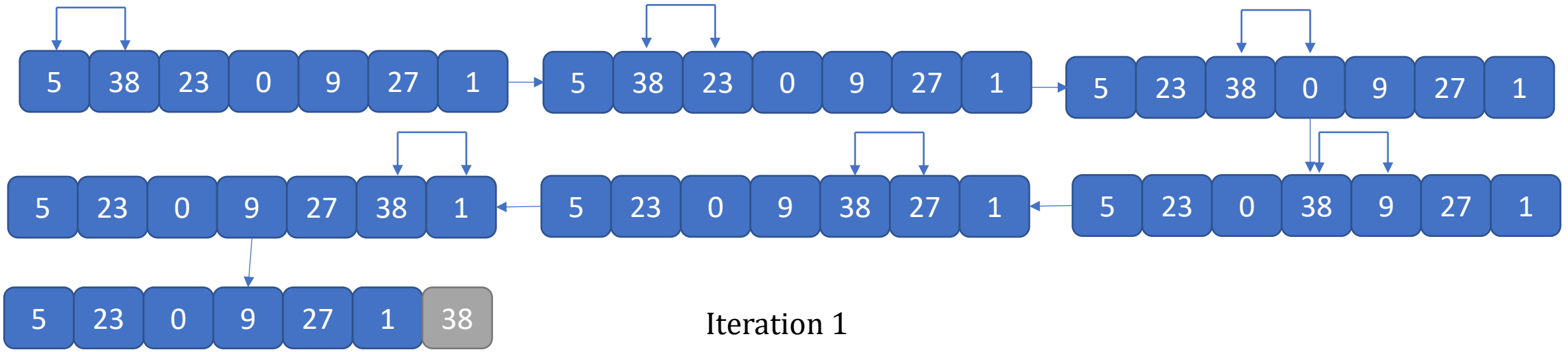
How can we sort the following array:



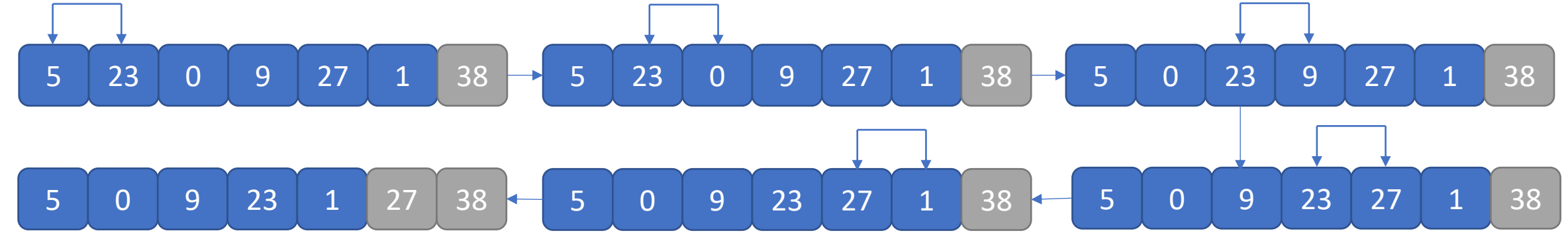
1. Insertion $O(N^2)$
2. Merge $O(Nlog(N))$
3. Quick $O(N^2)$
4. Bubble $O(N^2)$
5. Radix $O(wN) \rightarrow \mathrm{O(N)}, \mathrm{O(Nlog(N))}$, worse
6. More*

*Seizure Warning!

Bubble Sort

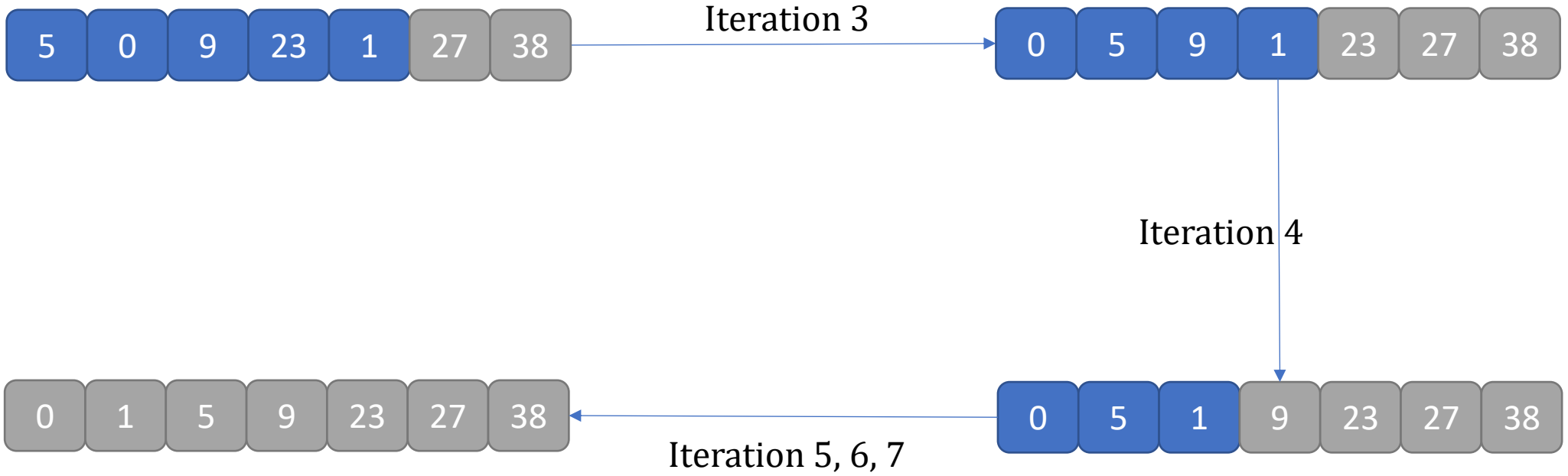


Iteration 1

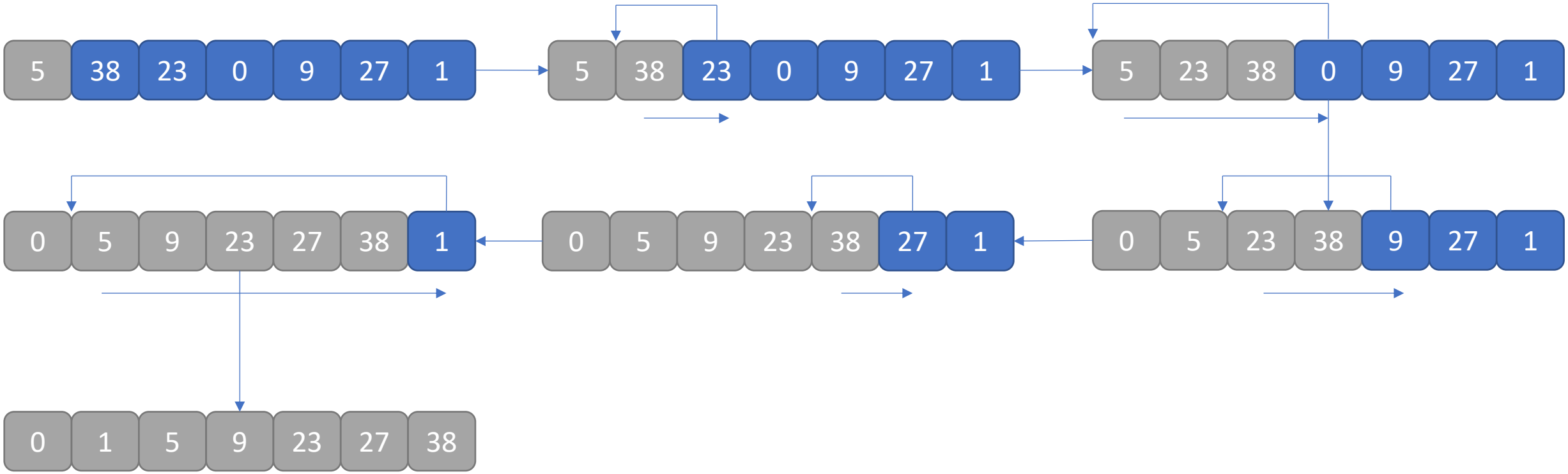


Iteration 2

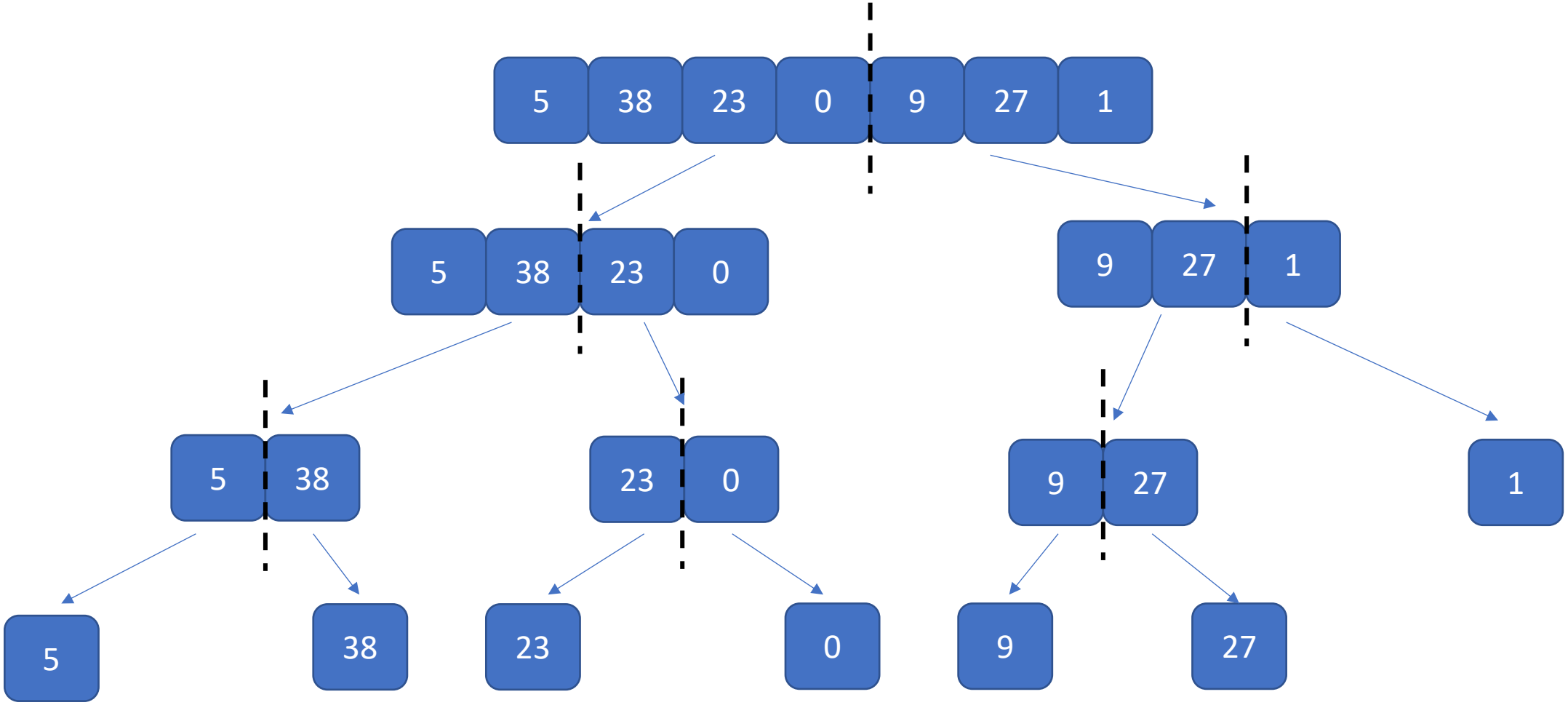
Bubble Sort


5 0 9 23 1 27 38
Iteration 3
0 5 9 1 23 27 38
Iteration 4
0 5 1 9 23 27 38
Iteration 5, 6, 7
0 1 5 9 23 27 38

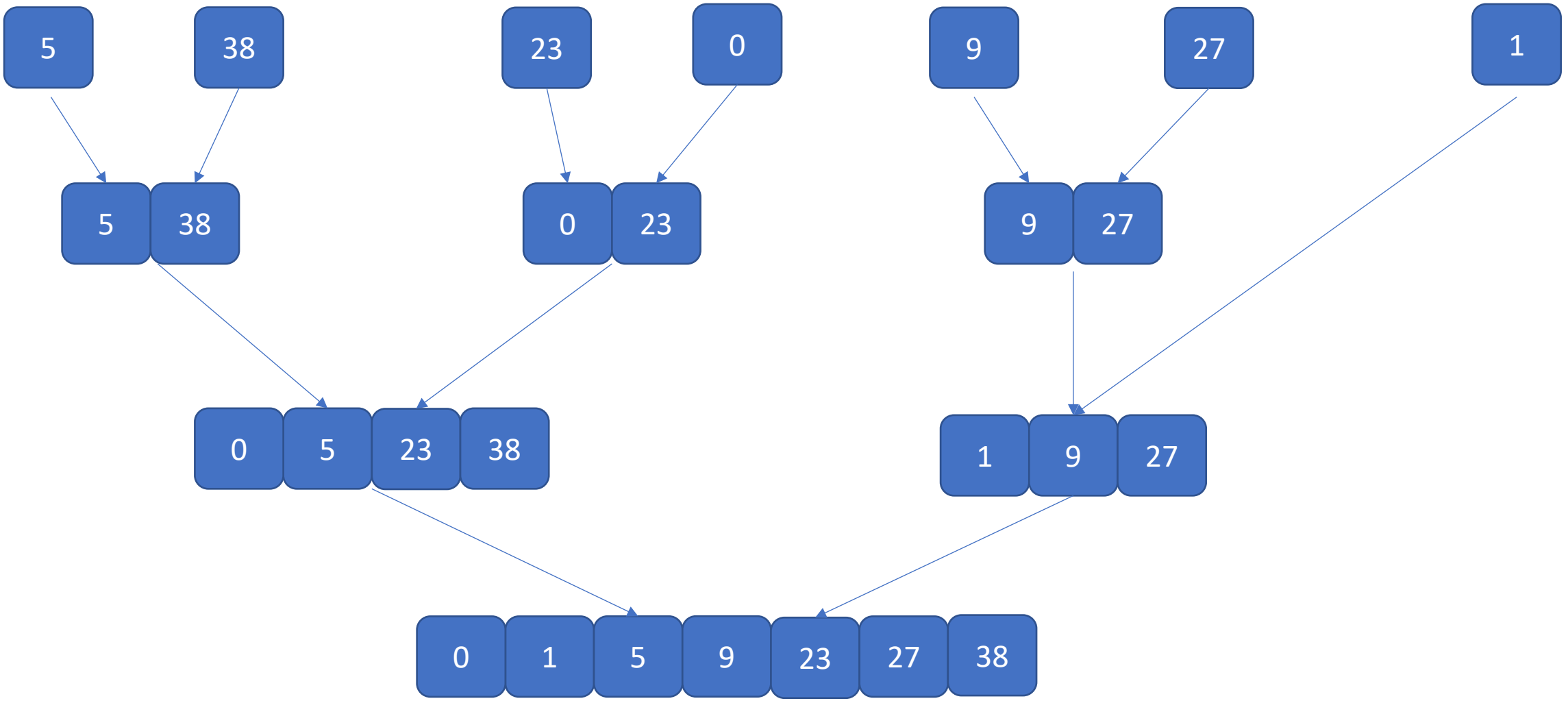
Insertion Sort



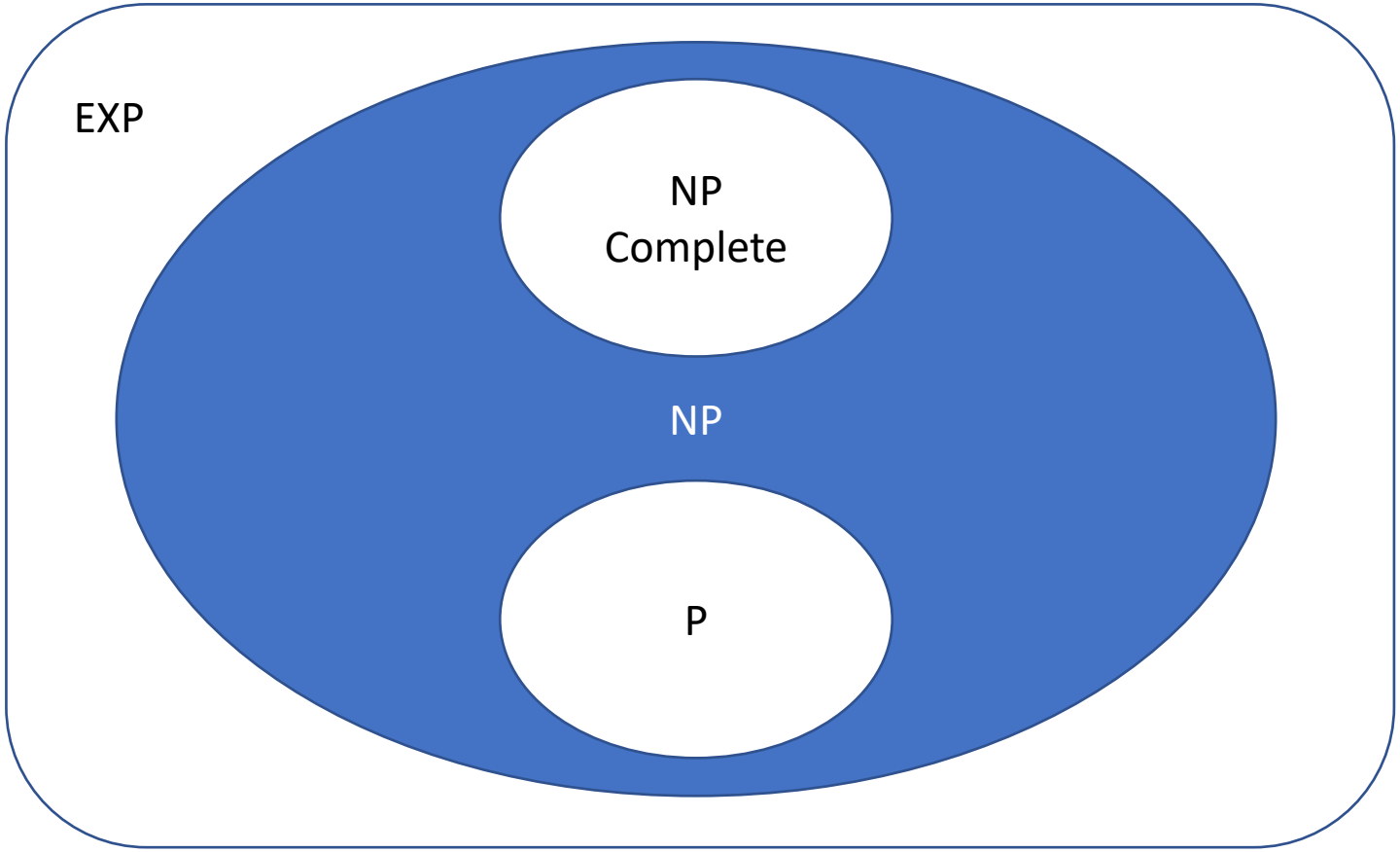
Merge Sort



Merge Sort



Computational Complexity



P stands for Polynomial time -> $\mathbf{O(n^k)}$
Examples: GCD and Prime Check

NP Problems

- The problem can be verified in polynomial time:
  - Problems with Efficient Algorithms for *Verifying* Proofs
  - As the input size grows, it becomes harder to solve the problem, but given a guess it is not that difficult to check the guess
  - Examples: Prime Factorization, Sudoku, Protein Folding, etc

- There's a $1,000,000 prize for something called the 'P vs NP' problem (which is itself an NP problem)

- Fun video for more information

How does Python work?

1. Code gets written (in a .py file or the command line).

2. On execution (running the code), it gets **parsed (compiled)**, verifying syntax and code.
   - source code is translated to something called byte code (.pyc files)

3. Finally, the **python interpreter** is called, running your code
   - The python interpreter is called the Python Virtual Machine

Hello World: Python vs C vs C++

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

hello.py

```
1  print "Hello World"
```

```
Hello World
[Finished in 0.1s]
```

hello.c

```
1  #include <stdio.h>
2
3  int main(){
4      printf("Hello World\n");
5      return 0;
6  }
```

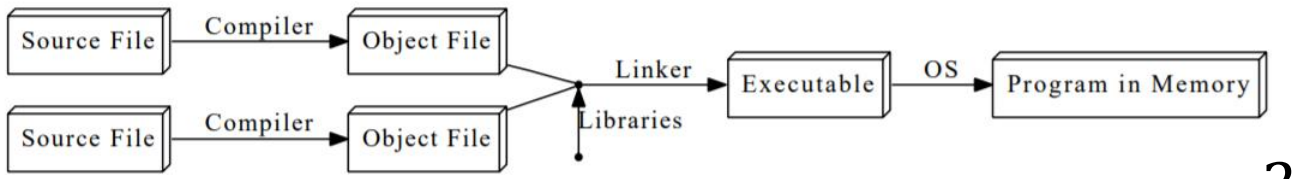
```
[Finished in 0.4s]
```

hello.cpp

```
1  #include <iostream>
2
3  int main(){
4      std::cout << "Hello World" << std::endl;
5      return 0;
6  }
```

```
Hello World
[Finished in 0.6s]
```

How do C/ C++ work?



Credit: MIT OpenCourseware

1. Have the compiler/linker installed (gcc and g++)

2. Compile using the following command:

   gcc –c hello.c

   This generates the hello.o object file.

```
hello.cpp
1  #include <iostream>
2
3  int main(){
4      std::cout << "Hello World" << std::endl;
5      return 0;
6  }

Hello World
[Finished in 0.6s]
```

Include I/O Functions

Start your Main Function
Print to Screen
End the Function

3. Now, link the object file into an executable:

   gcc –o hello hello.o

4. Now, you can run the code hello.exe

Highest Common Factor


```python
def hcf(a, b):
    while b > 0:
        temp = b
        b = a % b
        a = temp

    return a


print(hcf(198, 168))
```

```cpp
#include <iostream>

using namespace std;

int hcf(int a, int b) {
    int temp;
    while(b != 0){
        temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}

int main(){
    cout << hcf(210, 45) << endl;
    return 0;
}
```

- Variable Declaration
- Semicolons and Parentheses
- Namespace specification

Python vs C++: Summary

Python

1. Commands are executed by the interpreter, after being compiled into 'byte code'
2. Source file can be compiled once, and then the byte code can be run on any machine with the interpreter
3. Dynamically typed: interpreter checks the variable types

C++

1. Source file in compiled into machine language
2. The source file must be compiled on different machines
3. Statically typed: programmer must ensure type compatibility

Memory Management

1. We can see that in C and C++, the user is 'declaring variables', and thus manually allocating memory before the variables were initialized

2. Deallocating memory:
   - Once done with the variable, the user is also responsible for freeing up the memory once occupied
   - Forgetting to deallocate can result in memory leaks for long-running applications

3. Deallocating memory space before your program is done with it can cause it to crash

Python Garbage Collection

1. Automatic memory management is available in Python, known as the Garbage Collector

2. It works by reference counting: when the number of references to a particular variable becomes 0, the memory is freed

3. This method uses extra memory to keep track of references, and can also stop the program during runtime to collect all the objects not being used

4. Python also uses another mechanism called Generational garbage collection