Lecture 2

Comparison with C, C++ and Python

1) Code Complexity (lines of code and syntax) : C > C++ > Python (depends on the application)
2) Variable declaration: C and C++ (Yes) but Python (not needed)
3) C and C++ (Compiled) but Python (interpreted)
4) Paradigms: C (Procedural), C++ (Support Procedural and Object oriented) but Python (Support Procedural, Object-oriented and Functional)
   What is a programming paradigm? It is a style of programming, a way of thinking about software construction.
   Procedural programming (PP) also known as inline programming takes a top-down approach. It is about writing a list of instructions to tell the computer what to do step by step. It relies on procedures or routines.
   Object-oriented programming (OOP) is about encapsulating data and behavior into objects. An OOP application will use a collection of objects which knows how to perform certain actions and how to interact with other elements of the application. For example an object could be a person. That person would have a name (that would be a property of the object), and would know how to walk (that would be a method). A method in OOP can be considered as a procedure in PP, but here it belongs to a specific object. Another important aspect of OOP is classes. A class can be considered as a blueprint for an object.
   Functional programming (FP) is about passing data from function to function to function to get a result. In FP, functions are treated as data, meaning you can use them as parameters, return them, build functions from other functions, and build custom functions. Functions in FP have to be pure functions. A function is called pure function if it always returns the same result for same argument values and it has no side effects like modifying an argument (or global variable) or outputting something. The only result of calling a pure function is the return value. Examples of pure functions are strlen(), pow(), sqrt() etc. Examples of impure functions are printf() etc.
5) Inheritance: C(no) but C++ and Python (Yes)
6) Automatic memory allocation and deallocation: C(malloc, calloc and free) and C++ (new and delete) (no) but Python (Yes)
7) Exception handling: C (no) but C++ and Python (Yes)

Some basic terminologies:

Library functions and Header Files: As an example, to compute the square root of a number, you can use the sqrt() library function. The function is defined in the math.h header file. We use **#include** to use these header files in programs. (**#include <math.h>** in C)

Modules: A module is a file containing Python definitions and statements. A module can define functions, classes, and variables. Grouping related code into a module makes the code easier to understand and use **import** in python is similar to **#include** header file in C/C++. Python modules can get access to code from another module by importing the file/function using **import**. (**import tensorflow** in python)

Main method declaration

C

```
#include <stdio.h>
int main()
{

// Your code here

return 0;
}
```

Python

It is not necessary to declare main in python code unless and until you are declaring another function in your code. So we can declare main in python as follows.

```
def main():
    # write your code here

if __name__=="__main__":
    main()
```

Declaring Variables

In C and C++ we first declare the data type of the variable and then declare the name of Variables. A few examples of data types are int, char, float, double, etc. In Python, we do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it.

C

```
#include <stdio.h>

int main()
{
    // Declaring one variable at a time
```

```
    int a;

    // Declaring more than one variable
    char a, b, c, d;

    // Initializing variables
    float a = 0, b, c;
    b = 1;

    return 0;
}
```

Python

```
# Python program to demonstrate
# creating variables

# An integer assignment
age = 45

# A floating point
salary = 1456.8

# A string
name = "John"
```

Printing to console

C

```
#include <stdio.h>
int main()
{
   printf("Hello World");
   return 0;
}
```

Python

```
print("Hello World")
```

Taking Input

C

```
int main()
{
   int a, b, c;
   printf("first number: ");
   scanf("%d", &a);
   printf("second number: ");
   scanf("%d", &b);
   c = a + b;
   printf("Hello World\n%d + %d = %d", a, b, c);
   return 0;
}
```

Python

Code with main function ()

```
def main():
   a=5
   b=10
   c=a+b
   print(a, "+", b, "=", c)

   a = input("first number: ")
   b = input("second number: ")
   c = a + b
   print("Hello World")
   print(a, "+", b, "=", c)

   a = int(input("first number: "))
   b = int(input("second number: "))
   c = a + b
   print("Hello World")
   print(a, "+", b, "=", c)

if __name__=="__main__":
   main()
```

But main function () if there is no other functions in the code

The following code also work

```
a=5
b=10
c=a + b
```

```
print(a, "+", b, "=", c)

a = input("first number: ")
b = input("second number: ")
c = a + b
print("Hello World")
print(a, "+", b, "=", c)

a = int(input("first number: "))
b = int(input("second number: "))
c = a + b
print("Hello World")
print(a, "+", b, "=", c)
```