# 1 Hello World

Let us begin with the simple `hello world`.

```
def main():
    print("Hello World!")

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

The python code is a little bit "unusual", compared to what you have typically done. We define a function `main` and then call it if the script is run.

```

class HelloWorld{         // java uses { ... } to define block,
                          // while python use ':'
    public static void main(String[] args) {
        System.out.println("Hello world");    // ; is required
    }
}
```

Java vs Python

1. Python uses `:` and indent for defining a code block. Java uses `{. . .}`.
2. Python recognises line break as end of the statement. Java uses `;` to end the statement.

Conceptually, you can write the `HelloWorld.java` as

```
class HelloWorld{public static void main(String[] args){System.out.
    println("Hello world");}}
```

But really? It is too hard to understand, especially when the code becomes longer and longer.

A more substle and more important between Java and Python is that, Java is pure object-oriented. Evil Java king does not allow the verb slavers (`methods`) show in any public domain without a noun master (`class`). Therefore,

```
class HelloWorld{. . .
}
```

is required for Java to compile the code, though it looks like useless.

`if __name__ == "__main__"`

Python uses file name as module name. Therefore, one can reuse the function `main` by

```
import helloworld
helloworld.main()
```

When Python does `import`, it runs all the scripts define in `hello_world.py` by default. If we code the `helloworld.py` like the following,

```
def main():
    # do something
main()
```

`main()` in the last line would run when anyone does `import helloworld`. In most cases, this is not what we want. `if __name__ == '__main__'` prevents this from happening.

Run python code in a terminal,

```
python hello_world.py
```

Run java code in a terminal

1. Compile the Java code

   ```
   $ javac HelloWorld.java
   ```

2. Check the compilation result

   ```
   $ ls
   ```

   You will see there is a new file name `HelloWorld.class` showing up.

3. Run program

   ```
   $ java HelloWorld
   ```

**Exercise 1.1** *Change the* `HelloWorld` *java code, line 5 to*

```
System.out.println("Hello Sophie!");
```

*Compile and run the java code in a terminal.*

# 2 Data Types

Java data types are divided two groups:

| Data Type | Size | Description |
|---|---|---|
| byte | 1 byte | Stores whole numbers from -128 to 127 |
| short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| int | 4 bytes | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| boolean | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII values |

Table 1: Java Primitive data types

- Primitive data types - includes byte, short, int, long, float, double, boolean and char. Tabel 1 shows the primitive data types and the stored value description.
- Non-primitive data types - such as String, Arrays and Classes (you will learn more about these in a later chapter).

The ways that Java treats primitive data types and no-primitive data types are very different. We will discuss this once we are talking about class.

```
1  public class DataTypeExample {
2      public static void main(String[] args) {
3          // myNum = 5; // this doesn't work
4          int myNum;  // Declare a integer varaible
5          myNum = 5; // Assign Integer (whole number)
6
7          // myNum = 1.234; // This does not work
8
9          float myFloatNum = 5.99f; // Floating point number
10         double myDoubleNum = 5.99999999999; // Double
11         char myLetter = 'D'; // Character
12         boolean myBool = true; // Boolean
13         String myText = "Hello"; // String
14
15         System.out.println("My number: " + myNum + ", " + "My float
       : " + myFloatNum + ", " + "My double: " + myDoubleNum
16                 + ", " + "My letter: " + myLetter + ", " + "My Bool
       : " + myBool + ", " + "My Text: " + myText);
17
18         // myNum = 5.123; // this doesn't work
19         // myBool = "True"; // this doesn't work
```

```
20
21          myDoubleNum = myNum; // this works
22
23          System.out.println("My number: " + myNum + ", " + "My float
    : " + myFloatNum + ", " + "My double: " + myDoubleNum
24                  + ", " + "My letter: " + myLetter + ", " + "My Bool
    : " + myBool + ", " + "My Text: " + myText);
25
26      }
27  }
```

```
1
2   def main():
3       myNum = int(5)
4       myFloatNum = 5.99
5       myDoubleNum = 5.99999999999
6       myLetter = 'D'
7       myBool = True
8       myText = "Hello"
9
10      print("My number: " + myNum + ", " + "My float: " + myFloatNum
    + ", " + "My double: " + myDoubleNum
11              + ", " + "My letter: " + myLetter + ", " + "My Bool: "
    + myBool + ", " + "My Text: " + myText)
12
13      print(f"Before new assignment, the type of myNum value is {
    myNum} and type is {type(myNum)}")
14      myNum = 5.123   # this works
15      print(f"After new assignment, the type of myNum value is {myNum
    } and type is {type(myNum)}")
16      print(f"Before new assignment, the type of myNum value is {
    myBool} and type is {type(myBool)}")
17      myBool = "True" # this works
18      print(f"Before new assignment, the type of myNum value is {
    myBool} and type is {type(myBool)}")
19
20      myDoubleNum = myNum  # this is ok
21
22  if __name__ == "__main__":
23      main()
```

Notice that python also has similar data types: `int`, `float`, `bool` and `str`.

- Java language is designed to enforce type safety. A variable must be declared with a type before you can use it. In most cases, you are not allowed to assign a value with different type to this variable.
- Python is dynamically typed language. The type of a variable can be changed by the new assigned value.

**Exercise 2.1** *How to declare a variable in Java?*

**Exercise 2.2** *Can I assign a float value to a integer variable in Java? Why?*

**Exercise 2.3** *Can I assign a float value to a integer variable in Python? Why?*