Let us begin with the simple `hello world`.

Python

```python
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.

Java

```java
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

```java
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,

```java
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.