

# PEP Scala (2)

Email: christian.urban at kcl.ac.uk  
Office: N7.07 (North Wing, Bush House)  
Slides & Code: KEATS  
Office Hours: Mondays 12:00 – 14:00

# Scala on Lab Computers

```
$ /usr/share/scala/bin/scala
```

```
Welcome to Scala 2.12.6 (Java HotSpot(TM) 64-Bit  
Server VM, Java 10.0.1). Type in expressions for  
evaluation. Or try :help.
```

```
scala>
```

# Scala on Lab Computers

Avoid at all costs, even in comments

- `var`
- `return`
- `.par`
- `ListBuffer`
- `mutable`

# Scala on Lab Computers

Avoid at all costs, even in comments

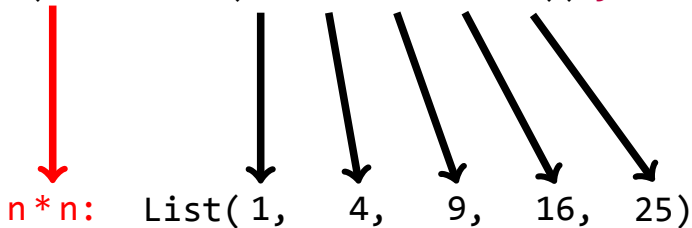
- `var`  $\Rightarrow$  `Var`
- `return`  $\Rightarrow$  `Return`
- `.par`
- `ListBuffer`
- `mutable`

# For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n*n
```

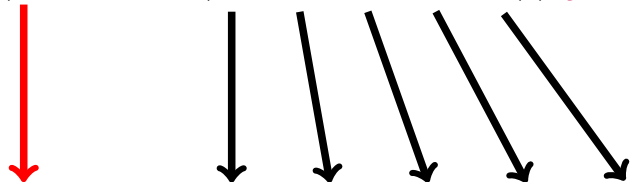
# For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n * n
```



# For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n * n
```



`n * n:` List(1, 4, 9, 16, 25)

This is for when the for-comprehension **yields** / **produces** a result.

# For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n * n
```

**VS**

```
for (n <- List(1, 2, 3, 4, 5)) println(n)
```

The second version is in case the for **does not** produce any result.



# Why Scala? No null!

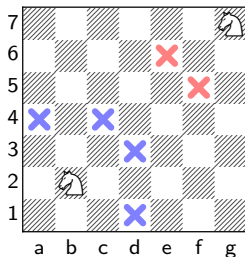


- **You can avoid null:**

“I call it my billion-dollar mistake. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (ALGOL W). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.”

Sir Tony (Hoare)

# Questions?



My Scala Office Hours: Thursdays 11 – 13