Finally blocks (Java, C#, Python …)

```
try {
            statements
}           catch (ex-type1 identifier1) {
                  statements
} catch (ex-type2 identifier2) {
                  statements
}     finally {
                  statements
}
```

- Finally of a try block practically always gets executed
- Avoid having cleanup code accidentally bypassed by a return, continue, or break
- + readability and maintainability of code

Finally Example

```
try {
 throw null; // throws NullPointerException!
} catch (Exception e)  {
  int oops = 1/0; // throws ArithmeticException!
} finally {
  System.out.println("Finally!");
  // still gets executed!
}
System.out.println("What about me???");
  // doesn't get executed!
```

Finally and Return in Java

- From Specification:

  A return statement with an Expression attempts to transfer control to the invoker of the method that contains it; the value of the Expression becomes the value of the method invocation.

  The preceding descriptions say "attempts to transfer control" rather than just "transfers control" because if there are any try statements within the method or constructor whose try blocks contain the return statement, then any finally clauses of those try statements will be executed, in order, innermost to outermost, before control is transferred to the invoker of the method or constructor. Abrupt completion of a finally clause can disrupt the transfer of control initiated by a return statement.

Finally and Return in Java

```java
int i = 0;
try {
  i = 2;
  return i;
} finally {
  i = 12; // and what if here return 12

}
```

Finally and Return in Java

- With throw in both try and finally block
  - The try block throws SomeException
  - The finally block kicks in and throws SomeOtherException
  - The second throw “wins” and the first exception is discarded

- A return in a finally block will cause an exception thrown from a try block to be discarded
- Throw in a finally block will cause a normal return value from a try block to be discarded