**I. Tổng quan**

Các function như (**let, run, with,apply, also**). Chúng được nằm trong [Standard.kt](https://github.com/JetBrains/kotlin/blob/master/libraries/stdlib/src/kotlin/util/Standard.kt) - là một phần của Kotlin library.

Những function này rất hữu ích giúp cho ta có thể sử dụng các objects trong một ngữ cảnh cụ thể giúp cho đoạn code của bạn trông gọn gàng, clean hơn.

**1. let**

fun <T, R> T.let(f: (T) -> R): R = f(this)

let() là một hàm phạm vi (**scoping function**): Sử dụng nó khi bạn muốn **định nghĩa một biến trong một phạm vi cụ thể** trong đoạn code. Nó cực kì hữu dụng giúp cho đoạn code "tự đóng gói".

**let** cung cấp cho ta quyền truy cập vào đối tượng và trả về một giá trị bất kỳ tuỳ vào use case của ta.

**ví dụ 1:**

DbConnection.getConnection().let { connection ->

}

// biến connection sẽ không còn xuất hiện ở đây nữa

### Ví dụ 2: Chaining let functions

var a = 1 var b= 2 a = a.let { it + 2 }.let { val i = it + b i}

println(a) //5

Nó tham chiếu tới đối tượng person bằng từ khoá **it**. Và có ta hoàn toàn có thể thay thế giá trị của class hay variables thông qua **it**

### Ví dụ 3: Nesting let

var x = "Anupam" x.let { outer -> outer.let { inner -> print("Inner is $inner and outer is $outer") } }

//Prints //Inner is Anupam and outer is Anupam

### Text Description automatically generatedVí dụ 3.2: Nesting let

### Ví dụ 4: let for null checks

Graphical user interface, text, application

Description automatically generated

**2. with**

fun <T, R> with(receiver: T, f: T.() -> R): R = receiver.f()

with() khá thuận tiện khi bạn **có thể gọi nhiều methods khác nhau trong cùng một đối tượng**. Thay vì phải liên tục lặp lại việc gọi biến mỗi dòng, bạn có thể dễ dàng với with()

**ví dụ 1 :**

**Graphical user interface, text

Description automatically generated with medium confidence**

**ví dụ 2 :**

**Text

Description automatically generated**

**3. run**

fun <T, R> T.run(f: T.() -> R): R = f()

run() cùng là một hàm khá là thú vị và ta có thể hiểu đơn giàn hàm này là một sự kết hợp giữa with() và let().

### Ví dụ 1:

**Text

Description automatically generated**

**\* Note:**

**Graphical user interface, text, application

Description automatically generated**

**4. apply**

fun <T> T.apply(f: T.() -> Unit): T { f(); return this }

apply() định nghĩa một **extension function** cho tất cả các loại Object. **Khi bạn gọi nó, nó sẽ gọi đoạn closure đã được truyền vào trong tham số và sau đó trả về đối tượng sau khi đoạn closure đó được chạy.**

**ví dụ 1:**

Text

Description automatically generated

****

**ví dụ 2:**

**Graphical user interface, text, application

Description automatically generated**

**5. also**

As the name says, also expressions **does some additional processing on the object it was invoked.**

Unlike let, it **returns the original object** instead of any new return data. Hence the return data has always the same type.

Like let, also uses it too.

**ví dụ :**

A picture containing graphical user interface

Description automatically generated

**II. Conclusion**

let() là một hàm phạm vi (**scoping function**): Sử dụng nó khi bạn muốn **định nghĩa một biến trong một phạm vi cụ thể** trong đoạn code

apply() định nghĩa một **extension function** cho tất cả các loại Object. **Khi bạn gọi nó, nó sẽ gọi đoạn closure đã được truyền vào trong tham số và sau đó trả về đối tượng sau khi đoạn closure đó được chạy.**

**Tài Liệu Tham Khảo**

**[ 1 ]** [**https://viblo.asia/p/kotlin-cung-tim-hieu-ve-let-apply-with-run-also-vyDZOX0Olwj#\_apply-2**](https://viblo.asia/p/kotlin-cung-tim-hieu-ve-let-apply-with-run-also-vyDZOX0Olwj#_apply-2)

**[ 2 ]** [**https://www.journaldev.com/19467/kotlin-let-run-also-apply-with**](https://www.journaldev.com/19467/kotlin-let-run-also-apply-with)