```
fun main() {
    val bestFriends:ArrayList<String> = arrayListOf("Ryan","Emma")
    println("What is your BestFriends?")

    val bestfriendInput:String? = readlnOrNull()
    if (bestfriendInput != null) {
        bestFriends.add(bestfriendInput)
    }
    for (friend :String? in bestFriends){
        println(friend)
    }
}
```

vagy

```
fun main(){
    val bestFriends: ArrayList<String> = arrayListOf("Ryan", "Emma")
    println("Who is your best friend?")

    readlnOrNull()?.let { bestFriends.add(it) }

    bestFriends.forEach { println(it) }

}
```

Kotlin nyelvi elemek – Mit csinálnak?

Kifejezés / kulcsszó	Jelentése / Használata	Példa
`let`	Egy scope-függvény, **akkor fut le**, ha az érték **nem null**	`val name = input?.let { "Üdv, $it!" }`
`it`	A `let`, `forEach`, `map` stb. belső neve az aktuális elemre, ha nem nevezed el	`list.forEach { println(it) }` → `it` az adott listaelem
`forEach`	Bejár egy kollekciót (lista, tömb) és **minden elemre végrehajt egy kódblokkot**	`names.forEach { println("Hello $it") }`
`to`	Két értéket **párba** köt, és létrehoz egy `Pair`-t	`val pair = "key" to "value"`
`in`	Használható ciklusban (`for`) vagy tartalmazás ellenőrzésre	`for (i in 1..5)`, `if ("apple" in fruits)`
`?.`	**Nullbiztos elérés**: csak akkor hívja meg a metódust, ha az érték nem `null`	`name?.length`
`?:`	Elvis-operator: ha `null`, akkor ad vissza egy **alapértelmezett értéket**	`val length = name?.length ?: 0`
`!!`	Kijelented, hogy az érték **soha nem lehet** `null` (veszélyes, ha mégis az!)	`val nameLength = name!!.length`
`val`	**Változatlan** érték (mint a `final`)	`val pi = 3.14`
`var`	**Változtatható** érték	`var name = "Ati"`
`fun`	Függvény definíció	`fun sayHello() { println("Hello") }`
`when`	Kotlinos `switch-case`, **értékválasztás**	`when (value) { 1 -> "One" 2 -> "Two" else -> "Other" }`
`is`	Típusellenőrzés	`if (x is String)`
`as`	Típuskonverzió	`val text = x as String`
`::`	Referencia egy függvényre vagy property-re	`val lenGetter = String::length`
`with`, `apply`, `run`, `also`	Scope-függvények különböző céllal és viselkedéssel (pl. konfigurálás, láncolás, stb.)	`val user = User().apply { name = "Ati" }`

Példakód magyarázata:

```
readlnOrNull()?.let { bestFriends.add(it) }
```

- `readlnOrNull()` → bekér egy sort, de `null`-t ad vissza, ha nem írtál be semmit.
- `?.let { ... }` → **csak akkor** fut le a blokk, ha nem `null` az érték.
- `it` → maga a beírt szöveg (pl. `"Péter"`).
- `bestFriends.add(it)` → hozzáadja a listához.
