For és While Ciklusok Cheatsheet

1. Egyszerű Lista/Tömb Bejárás

Kotlin

```kotlin
// Enhanced for (for-each)
val names = listOf("Hétfő", "Kedd", "Szerda")
for (name in names) {
    println(name)
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (i in 0 until names.size) {
    println(names[i])
}

// While
var i = 0
while (i < names.size) {
    println(names[i])
    i++
}
```

Java

```java
// Enhanced for (for-each)
String[] names = {"Hétfő", "Kedd", "Szerda"};
for (String name : names) {
    System.out.println(name);
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i < names.length; i++) {
    System.out.println(names[i]);
}

// While
int i = 0;
while (i < names.length) {
    System.out.println(names[i]);
    i++;
}
```

C++

```cpp
// Range-based for (C++11+)
std::vector<std::string> names = {"Hétfő", "Kedd", "Szerda"};
for (const auto& name : names) {
    std::cout << name << std::endl;
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i < names.size(); i++) {
    std::cout << names[i] << std::endl;
}

// While
int i = 0;
while (i < names.size()) {
    std::cout << names[i] << std::endl;
    i++;
}
```

C#

```cs
// foreach
string[] names = {"Hétfő", "Kedd", "Szerda"};
foreach (string name in names) {
    Console.WriteLine(name);
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i < names.Length; i++) {
    Console.WriteLine(names[i]);
}

// While
int i = 0;
while (i < names.Length) {
    Console.WriteLine(names[i]);
    i++;
}
```

2. Mátrix/2D Tömb Bejárás

Kotlin

```kotlin
// Enhanced for - 2D lista
val matrix = listOf(
    listOf("A", "B", "C"),
    listOf("D", "E", "F"),
    listOf("G", "H", "I")
)

for (row in matrix) {
    for (cell in row) {
```

```
        print("$cell ")
    }
    println()
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (i in 0 until matrix.size) {
    for (j in 0 until matrix[i].size) {
        print("${matrix[i][j]} ")
    }
    println()
}

// While
var i = 0
while (i < matrix.size) {
    var j = 0
    while (j < matrix[i].size) {
        print("${matrix[i][j]} ")
        j++
    }
    println()
    i++
}
```

Java

```
// Enhanced for - 2D tömb
String[][] matrix = {
    {"A", "B", "C"},
    {"D", "E", "F"},
    {"G", "H", "I"}
};

for (String[] row : matrix) {
    for (String cell : row) {
        System.out.print(cell + " ");
    }
    System.out.println();
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
}

// While
int i = 0;
while (i < matrix.length) {
    int j = 0;
    while (j < matrix[i].length) {
        System.out.print(matrix[i][j] + " ");
```

```
        j++;
    }
    System.out.println();
    i++;
}
```

C++

```
// Range-based for - 2D vector
std::vector<std::vector<std::string>> matrix = {
    {"A", "B", "C"},
    {"D", "E", "F"},
    {"G", "H", "I"}
};

for (const auto& row : matrix) {
    for (const auto& cell : row) {
        std::cout << cell << " ";
    }
    std::cout << std::endl;
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i < matrix.size(); i++) {
    for (int j = 0; j < matrix[i].size(); j++) {
        std::cout << matrix[i][j] << " ";
    }
    std::cout << std::endl;
}

// While
int i = 0;
while (i < matrix.size()) {
    int j = 0;
    while (j < matrix[i].size()) {
        std::cout << matrix[i][j] << " ";
        j++;
    }
    std::cout << std::endl;
    i++;
}
```

C#

```
// foreach - 2D tömb
string[,] matrix = {
    {"A", "B", "C"},
    {"D", "E", "F"},
    {"G", "H", "I"}
};

// Minden elem bejárása
foreach (string cell in matrix) {
    Console.Write(cell + " ");
```

```
}

// Sorok és oszlopok szerint
for (int i = 0; i < matrix.GetLength(0); i++) {
    for (int j = 0; j < matrix.GetLength(1); j++) {
        Console.Write(matrix[i, j] + " ");
    }
    Console.WriteLine();
}

// While
int i = 0;
while (i < matrix.GetLength(0)) {
    int j = 0;
    while (j < matrix.GetLength(1)) {
        Console.Write(matrix[i, j] + " ");
        j++;
    }
    Console.WriteLine();
    i++;
}
```

3. "Ati" Háromszög Minta (Fordított Piramis)

Kotlin

```
val name = "Ati"

// For ciklus (inicializálás; kifejezés; aktualizálás)
for (i in 3 downTo 1) {
    for (j in 1..i) {
        print(name)
    }
    println()
}

// While
var i = 3
while (i >= 1) {
    var j = 1
    while (j <= i) {
        print(name)
        j++
    }
    println()
    i--
}
```

Java

```java
String name = "Ati";

// For ciklus (inicializálás; kifejezés; aktualizálás)
for (int i = 3; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        System.out.print(name);
    }
    System.out.println();
}

// While
int i = 3;
while (i >= 1) {
    int j = 1;
    while (j <= i) {
        System.out.print(name);
        j++;
    }
    System.out.println();
    i--;
}
```

C++

```cpp
std::string name = "Ati";

// For ciklus (inicializálás; kifejezés; aktualizálás)
for (int i = 3; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        std::cout << name;
    }
    std::cout << std::endl;
}

// While
int i = 3;
while (i >= 1) {
    int j = 1;
    while (j <= i) {
        std::cout << name;
        j++;
    }
    std::cout << std::endl;
    i--;
}
```

C#

```
string name = "Ati";

// For ciklus (inicializálás; kifejezés; aktualizálás)
for (int i = 3; i >= 1; i--) {
    for (int j = 1; j <= i; j++) {
        Console.Write(name);
    }
    Console.WriteLine();
}

// While
int i = 3;
while (i >= 1) {
    int j = 1;
    while (j <= i) {
        Console.Write(name);
        j++;
    }
    Console.WriteLine();
    i--;
}
```

4. Enum/Calendar Példa (Napok)

Kotlin

```
enum class Day(val message: String) {
    MONDAY("Hétfő van!"),
    TUESDAY("Kedd van!"),
    WEDNESDAY("Szerda van!"),
    THURSDAY("Csütörtök van!"),
    FRIDAY("Péntek van, majdnem hétvége!"),
    SATURDAY("Szombat van, pihenés!"),
    SUNDAY("Vasárnap van, nyugi nap!")
}

// Enhanced for
for (day in Day.values()) {
    println("${day.name} : ${day.message}")
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
val days = Day.values()
for (i in 0 until days.size) {
    println("${days[i].name} : ${days[i].message}")
}

// While
var i = 0
while (i < Day.values().size) {
    val day = Day.values()[i]
    println("${day.name} : ${day.message}")
```

```
    i++
}
```

Java

```
enum Day {
    MONDAY("Hétfő van!"),
    TUESDAY("Kedd van!"),
    WEDNESDAY("Szerda van!"),
    THURSDAY("Csütörtök van!"),
    FRIDAY("Péntek van, majdnem hétvége!"),
    SATURDAY("Szombat van, pihenés!"),
    SUNDAY("Vasárnap van, nyugi nap!");

    private final String message;

    Day(String message) {
        this.message = message;
    }

    public String getMessage() {
        return message;
    }
}

// Enhanced for
for (Day day : Day.values()) {
    System.out.println(day.name() + " : " + day.getMessage());
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
Day[] days = Day.values();
for (int i = 0; i < days.length; i++) {
    System.out.println(days[i].name() + " : " + days[i].getMessage());
}

// While
int i = 0;
Day[] daysArray = Day.values();
while (i < daysArray.length) {
    System.out.println(daysArray[i].name() + " : " + daysArray[i].getMessage());
    i++;
}
```

C++

```
#include <map>
#include <string>

enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
};

std::map<Day, std::string> dayMessages = {
```

```
    {MONDAY, "Hétfő van!"},
    {TUESDAY, "Kedd van!"},
    {WEDNESDAY, "Szerda van!"},
    {THURSDAY, "Csütörtök van!"},
    {FRIDAY, "Péntek van, majdnem hétvége!"},
    {SATURDAY, "Szombat van, pihenés!"},
    {SUNDAY, "Vasárnap van, nyugi nap!"}
};

// Range-based for
for (const auto& pair : dayMessages) {
    std::cout << pair.first << " : " << pair.second << std::endl;
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
Day days[] = {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};
for (int i = 0; i < 7; i++) {
    std::cout << days[i] << " : " << dayMessages[days[i]] << std::endl;
}

// While
int i = 0;
while (i < 7) {
    std::cout << days[i] << " : " << dayMessages[days[i]] << std::endl;
    i++;
}
```

C#

```
enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}

static Dictionary<Day, string> dayMessages = new Dictionary<Day, string> {
    {Day.MONDAY, "Hétfő van!"},
    {Day.TUESDAY, "Kedd van!"},
    {Day.WEDNESDAY, "Szerda van!"},
    {Day.THURSDAY, "Csütörtök van!"},
    {Day.FRIDAY, "Péntek van, majdnem hétvége!"},
    {Day.SATURDAY, "Szombat van, pihenés!"},
    {Day.SUNDAY, "Vasárnap van, nyugi nap!"}
};

// foreach
foreach (Day day in Enum.GetValues<Day>()) {
    Console.WriteLine($"{day} : {dayMessages[day]}");
}

// Klasszikus for (inicializálás; kifejezés; aktualizálás)
Day[] days = Enum.GetValues<Day>();
for (int i = 0; i < days.Length; i++) {
    Console.WriteLine($"{days[i]} : {dayMessages[days[i]]}");
}

// While
```

```
int i = 0;
Day[] daysArray = Enum.GetValues<Day>();
while (i < daysArray.Length) {
    Console.WriteLine($"{daysArray[i]} : {dayMessages[daysArray[i]]}");
    i++;
}
```

5. Klasszikus For Ciklus Részletes Bontása (Inicializálás; Kifejezés; Aktualizálás)

Kotlin

```
// for (inicializálás; kifejezés; aktualizálás)
for (i in 0..4) {
    println("Érték: $i")
}
// Kimenet:
// Érték: 0
// Érték: 1
// Érték: 2
// Érték: 3
// Érték: 4

// Visszafelé
for (i in 5 downTo 1) {
    println("Visszafelé: $i")
}
// Kimenet:
// Visszafelé: 5
// Visszafelé: 4
// Visszafelé: 3
// Visszafelé: 2
// Visszafelé: 1

// Lépésközzel
for (i in 0..10 step 2) {
    println("Páros: $i")
}
// Kimenet:
// Páros: 0
// Páros: 2
// Páros: 4
// Páros: 6
// Páros: 8
// Páros: 10
```

Java

```
// for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i <= 4; i++) {
    System.out.println("Érték: " + i);
}
// Kimenet:
```

```
// Érték: 0
// Érték: 1
// Érték: 2
// Érték: 3
// Érték: 4

// Visszafelé
for (int i = 5; i >= 1; i--) {
    System.out.println("Visszafelé: " + i);
}
// Kimenet:
// Visszafelé: 5
// Visszafelé: 4
// Visszafelé: 3
// Visszafelé: 2
// Visszafelé: 1

// Lépésközzel
for (int i = 0; i <= 10; i += 2) {
    System.out.println("Páros: " + i);
}
// Kimenet:
// Páros: 0
// Páros: 2
// Páros: 4
// Páros: 6
// Páros: 8
// Páros: 10
```

C++

```
// for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i <= 4; i++) {
    std::cout << "Érték: " << i << std::endl;
}
// Kimenet:
// Érték: 0
// Érték: 1
// Érték: 2
// Érték: 3
// Érték: 4

// Visszafelé
for (int i = 5; i >= 1; i--) {
    std::cout << "Visszafelé: " << i << std::endl;
}
// Kimenet:
// Visszafelé: 5
// Visszafelé: 4
// Visszafelé: 3
// Visszafelé: 2
// Visszafelé: 1

// Lépésközzel
for (int i = 0; i <= 10; i += 2) {
```

```
    std::cout << "Páros: " << i << std::endl;
}
// Kimenet:
// Páros: 0
// Páros: 2
// Páros: 4
// Páros: 6
// Páros: 8
// Páros: 10
```

C#

```
// for (inicializálás; kifejezés; aktualizálás)
for (int i = 0; i <= 4; i++) {
    Console.WriteLine($"Érték: {i}");
}
// Kimenet:
// Érték: 0
// Érték: 1
// Érték: 2
// Érték: 3
// Érték: 4

// Visszafelé
for (int i = 5; i >= 1; i--) {
    Console.WriteLine($"Visszafelé: {i}");
}
// Kimenet:
// Visszafelé: 5
// Visszafelé: 4
// Visszafelé: 3
// Visszafelé: 2
// Visszafelé: 1

// Lépésközzel
for (int i = 0; i <= 10; i += 2) {
    Console.WriteLine($"Páros: {i}");
}
// Kimenet:
// Páros: 0
// Páros: 2
// Páros: 4
// Páros: 6
// Páros: 8
// Páros: 10
```

6. While Ciklus Részletes Bontása (Inicializálás előtte; Kifejezés; Aktualizálás benne)

Kotlin

```kotlin
// While: inicializálás előtte; kifejezés; aktualizálás benne
var i = 0                    // Inicializálás
while (i <= 4) {             // Kifejezés
    println("While érték: $i")
    i++                      // Aktualizálás
}
// Kimenet:
// While érték: 0
// While érték: 1
// While érték: 2
// While érték: 3
// While érték: 4

// Visszafelé
var j = 5
while (j >= 1) {
    println("While visszafelé: $j")
    j--
}
// Kimenet:
// While visszafelé: 5
// While visszafelé: 4
// While visszafelé: 3
// While visszafelé: 2
// While visszafelé: 1
```

Java

```java
// While: inicializálás előtte; kifejezés; aktualizálás benne
int i = 0;                           // Inicializálás
while (i <= 4) {                     // Kifejezés
    System.out.println("While érték: " + i);
    i++;                             // Aktualizálás
}
// Kimenet:
// While érték: 0
// While érték: 1
// While érték: 2
// While érték: 3
// While érték: 4

// Visszafelé
int j = 5;
while (j >= 1) {
    System.out.println("While visszafelé: " + j);
    j--;
}
// Kimenet:
// While visszafelé: 5
// While visszafelé: 4
// While visszafelé: 3
// While visszafelé: 2
```

```
// While visszafelé: 1
```

C++

```cpp
// While: inicializálás előtte; kifejezés; aktualizálás benne
int i = 0;                                  // Inicializálás
while (i <= 4) {                           // Kifejezés
    std::cout << "While érték: " << i << std::endl;
    i++;                                    // Aktualizálás
}
// Kimenet:
// While érték: 0
// While érték: 1
// While érték: 2
// While érték: 3
// While érték: 4

// Visszafelé
int j = 5;
while (j >= 1) {
    std::cout << "While visszafelé: " << j << std::endl;
    j--;
}
// Kimenet:
// While visszafelé: 5
// While visszafelé: 4
// While visszafelé: 3
// While visszafelé: 2
// While visszafelé: 1
```

C#

```cs
// While: inicializálás előtte; kifejezés; aktualizálás benne
int i = 0;                          // Inicializálás
while (i <= 4) {                   // Kifejezés
    Console.WriteLine($"While érték: {i}");
    i++;                           // Aktualizálás
}
// Kimenet:
// While érték: 0
// While érték: 1
// While érték: 2
// While érték: 3
// While érték: 4

// Visszafelé
int j = 5;
while (j >= 1) {
    Console.WriteLine($"While visszafelé: {j}");
    j--;
}
// Kimenet:
// While visszafelé: 5
// While visszafelé: 4
```

```
// While visszafelé: 3
// While visszafelé: 2
// While visszafelé: 1
```

7. Ciklus Struktúrák Összehasonlító Táblázata

Ciklus Típus	Kotlin	Java	C++	C#
For Ciklus Szintaxis	`for (i in range)`	`for (init; condition; update)`	`for (init; condition; update)`	`for (init; condition; update)`
Inicializálás	`i in 0..4`	`int i = 0`	`int i = 0`	`int i = 0`
Kifejezés	automatikus	`i <= 4`	`i <= 4`	`i <= 4`
Aktualizálás	automatikus	`i++`	`i++`	`i++`
While Szintaxis	`while (condition)`	`while (condition)`	`while (condition)`	`while (condition)`
While Inicializálás	`var i = 0` (előtte)	`int i = 0;` (előtte)	`int i = 0;` (előtte)	`int i = 0;` (előtte)
While Kifejezés	`i <= 4`	`i <= 4`	`i <= 4`	`i <= 4`
While Aktualizálás	`i++` (benne)	`i++;` (benne)	`i++;` (benne)	`i++;` (benne)

8. Példa Kimenetek Összefoglalása

Egyszerű Számolás (0-tól 4-ig):

```
Érték: 0
Érték: 1
Érték: 2
Érték: 3
Érték: 4
```

Visszafelé Számolás (5-től 1-ig):

```
Visszafelé: 5
Visszafelé: 4
Visszafelé: 3
Visszafelé: 2
Visszafelé: 1
```

Páros Számok (0-tól 10-ig, 2-esével):

```
Páros: 0
Páros: 2
Páros: 4
Páros: 6
Páros: 8
Páros: 10
```

"Ati" Háromszög Kimenet:

```
AtiAtiAti
AtiAti
Ati
```

Napok Enum Kimenet:

```
MONDAY : Hétfő van!
TUESDAY : Kedd van!
WEDNESDAY : Szerda van!
THURSDAY : Csütörtök van!
FRIDAY : Péntek van, majdnem hétvége!
SATURDAY : Szombat van, pihenés!
SUNDAY : Vasárnap van, nyugi nap!
```

Összefoglalás

Nyelv	Enhanced/Range-based For	Klasszikus For	While
Kotlin	`for (item in collection)`	`for (i in 0 until size)`	`while (feltétel)`
Java	`for (Type item : collection)`	`for (int i = 0; i < size; i++)`	`while (feltétel)`
C++	`for (const auto& item : collection)`	`for (int i = 0; i < size; i++)`	`while (feltétel)`
C#	`foreach (Type item in collection)`	`for (int i = 0; i < size; i++)`	`while (feltétel)`