# For_Loops

| | |
|---|---|
| Created time | September 2, 2026 4:42 AM |
| Category | Loops_Sets |
| Last edited by | R.A R.A |
| Last updated time | September 4, 2026 9:15 AM |

```
for (int i = 0; i < 10; i ++)
{
    Console.WriteLine($"The value of i is {i} ");
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
string data = "Tim, Sue, Bob, Jane, Frank" ;
List<string> firstNames = data.Split(',').ToList();

for (int i = 0; i < firstNames.Count; i ++)
{
    Console.WriteLine($"{firstNames[i]} is in attendance");
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
List<decimal> charges = new();

charges.Add(23.78M);
charges.Add(15.89M);
charges.Add(125M);

decimal total = 0;

for (int i = 0; i < charges.Count; i++)
{
    total += charges[i];
}
```

```
Console.WriteLine($"Our Total charges: {total}");
```

יש `snippet` — אנחנו מקלידים `for` ולוחצים פעמיים על `TAB`, וזה יוצר עבורנו את המבנה של לולאת `for`.

לולאת `for` היא סוג מיוחד של לולאה, שנועדה לחזור על פעולה מספר מוגדר של פעמים.

יש לנו 3 חלקים שונים בתוך הסוגריים `()`.

### חלק 1:

אנחנו מגדירים משתנה שישמש אותנו לספירה.

זאת האתחול (`initialization`) שממנו המונה שלנו יתחיל:

```
int i = 0;
```

זכרו שהדברים הבאים משתמשים בספירה שמתחילה מ־0.

כלומר, הם מתחילים ב־`position 0` ומשם ממשיכים לעלות:

```
Arrays
Dictionaries
Lists
```

## חלק 2:

```
i < length;
```

זה החלק שבו אנחנו בודקים את התנאי.

לדוגמה, אם יש לנו את המספר `10`, הקוד ירוץ כל עוד `i` קטן מ־10:

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

}
```

### חלק 3:

```
i ++
```

המשמעות היא:

להוסיף `1` ל־`i`.

אחרי הסיבוב הראשון של הלולאה, `i` הופך ל־`1` במקום `0`.

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

}
```

האם `10 > 1`?

כן.

לכן הקוד בתוך הלולאה ירוץ.

לאחר מכן חוזרים ל־:

```
i++
```

מוסיפים שוב 1, בודקים שוב:

```
i < 10
```

מריצים את הקוד, וכך ממשיכים שוב ושוב.

---

כלל בסיסי הוא לתת למשתנים שמות שמתארים בצורה ברורה מה הם מייצגים.

אבל כאן יש יוצא מן הכלל:

```
i
```

`i` הוא שם מקובל וסטנדרטי למונה בתוך לולאת `for`.

## בואו נריץ את זה:

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

```
    Console.WriteLine($"The value of i is {i} ");
}
```

### וזה מה שנקבל מהקומפיילר:

```
The value of i is 0
The value of i is 1
The value of i is 2
The value of i is 3
The value of i is 4
The value of i is 5
The value of i is 6
The value of i is 7
The value of i is 8
The value of i is 9
```

ברגע שאנחנו מגיעים ל־`10`, התנאי אומר:

```
10 is not < 10
```

כלומר:

`10` **אינו קטן מ־10**.

לכן יוצאים מהלולאה.

---

### הדפסת מערך/רשימה של שמות פרטיים:

```
string data = "Tim, Sue, Bob, Jane, Frank";
List<string> firstNames = data.Split(',').ToList();

for (int i = 0; i < firstNames.Count; i ++)
{
    Console.WriteLine(firstNames[i]);
}
```

זכרו:

הסימן `>` אומר שאנחנו אף פעם לא נגיע לערך ששווה ל־`Count`.

אנחנו תמיד נעצור אחד לפניו.

```
Console.WriteLine(firstNames[i]);
```

המשמעות היא:

יש לנו 5 שמות פרטיים.

לכן יודפסו השמות שנמצאים במיקומים:

```
0
1
2
3
4
```

ואז הלולאה תיעצר.

בנוסף, הלולאה תתאים את עצמה באופן אוטומטי לגודל של הרשימה שלנו.

התוצאה:

```
Tim
Sue
Bob
Jane
Frank
```

## אם נכתוב:

```
Console.WriteLine($"{firstNames[i]} is in attendance");
```

## נקבל:

```
Tim is in attendance
Sue is in attendance
Bob is in attendance
Jane is in attendance
Frank is in attendanceTim is in attendance
Sue is in attendance
```

```
Bob is in attendance
Jane is in attendance
Frank is in attendance
```

---

דרך מקוצרת ליצור רשימה חדשה מסוג `decimal` :

```
List<decimal> charges = new();

charges.Add(23.78M);
charges.Add(15.89M);
charges.Add(125M);

decimal total = 0;

for (int i = 0; i < charges.Count; i++)
{
    total += charges[i];
}

Console.WriteLine($"Our Total charges: {total}");
```

נקבל:

```
Our Total charges: 164.67
```