For Loops in Python

Making a for loop

`for` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for
```

1. The `for` keyword

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for i
```

1. The `for` keyword
2. The **iterator**

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for i in
```

1. The `for` keyword
2. The **iterator**
3. The `in` keyword

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for i in range(3):
```

1. The `for` keyword
2. The **iterator**
3. The `in` keyword
4. The value being iterated over

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for i in range(3):
    print("Hey")
```

1. The `for` keyword
2. The **iterator**
3. The `in` keyword
4. The value being **iterated** over
5. The **body** of the loop

Making a for loop

`For` loops are used when we have an action we want taken a **specific number of times**.

There are 5 parts:

```
for i in range(3):
    print("Hey")
```

```
Hey
Hey
Hey
```

1. The `for` keyword
2. The **iterator**
3. The `in` keyword
4. The value being **iterated** over
5. The **body** of the loop

Using the value of i

```
for i in range(5):
    print(i)
```

```
0
1
2
3
4
```

We can make use of the **iterator** within our `for` loops, because it's a variable!

The first value of i will always be 0, it will go upward toward but ***not*** include the number in the range() function - let's call it **n**.

This will create a *number of values* equal to **n**.

Adding More Numbers

More Numbers in Range

If we add an additional number to the range, we can choose both the starting **and** ending numbers!

More Numbers in Range

If we add an additional number to the range, we can choose both the starting **and** ending numbers!

```
for i in range(1, 5):
    print(i)
```

More Numbers in Range

If we add an additional number to the range, we can choose both the starting **and** ending numbers!

```
for i in range(1, 5):
    print(i)
```

```
1
2
3
4
```

More Numbers in Range

If we add an additional number to the range, we can choose both the starting **and** ending numbers!

```
for i in range(1, 5):
    print(i)
```

```
1
2
3
4
```

Just like when we only use 1 number in our `for` loops, the ending number will never be reached - the numbers will always stop 1 short.

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(1, 10, 2):
    print(i)
```

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(1, 10, 2):
    print(i)
```

The order of those three values is:

1. Starting value (included)
2. Ending value (**not** included)
3. Step size

Just like the other 2 types of `for` loop, the ending value will never be included - the closest you can get is 1 short.

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(1, 10, 2):
    print(i)
```

```
1
3
5
7
9
```

The order of those three values is:

1. Starting value (included)
2. Ending value (**not** included)
3. Step size

Just like the other 2 types of `for` loop, the ending value will never be included - the closest you can get is 1 short.

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(1, 10, 3):
    print(i)
```

The order of those three values is:

1. Starting value (included)
2. Ending value (**not** included)
3. Step size

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(1, 10, 3):
    print(i)
```

```
1
4
7
```

The order of those three values is:

1. Starting value (included)
2. Ending value (**not** included)
3. Step size

If the next value in the sequence would be **greater than *or* equal to** the Ending value, it will not be included.

Even MORE numbers!

If we add a ***third*** number, it allows us to choose how much the **iterator** changes each time through the loop.

```
for i in range(10, 1, -2):
    print(i)
```

```
10
8
6
4
2
```

The order of those three values is:

1. Starting value (included)
2. Ending value (**not** included)
3. Step size

We can also have our Step size be **negative**. If we do, the Starting value must be **larger** than the ending value.

Using Variables

Anytime we can use a **literal** (programmer-written) value in our programs, we can instead use the value stored in a variable. This includes variables holding user input!

```
num = int(input("How many loops? "))
for i in range(num):
    print(i)
```