

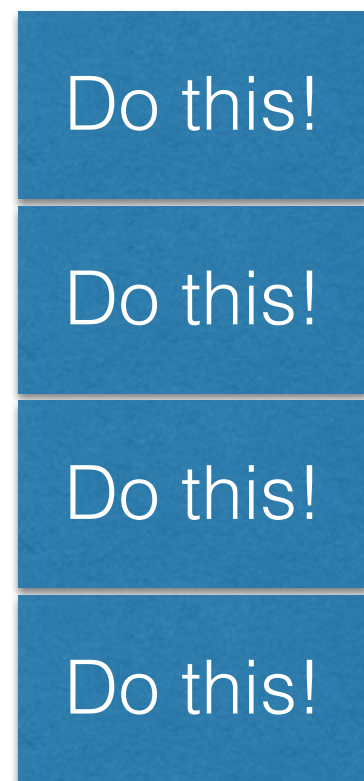
# Introduction to Computer Programming Lecture 3.1:

## **Loops**

Hemma Philamore

Department of Engineering Mathematics

# Loops



# iterations

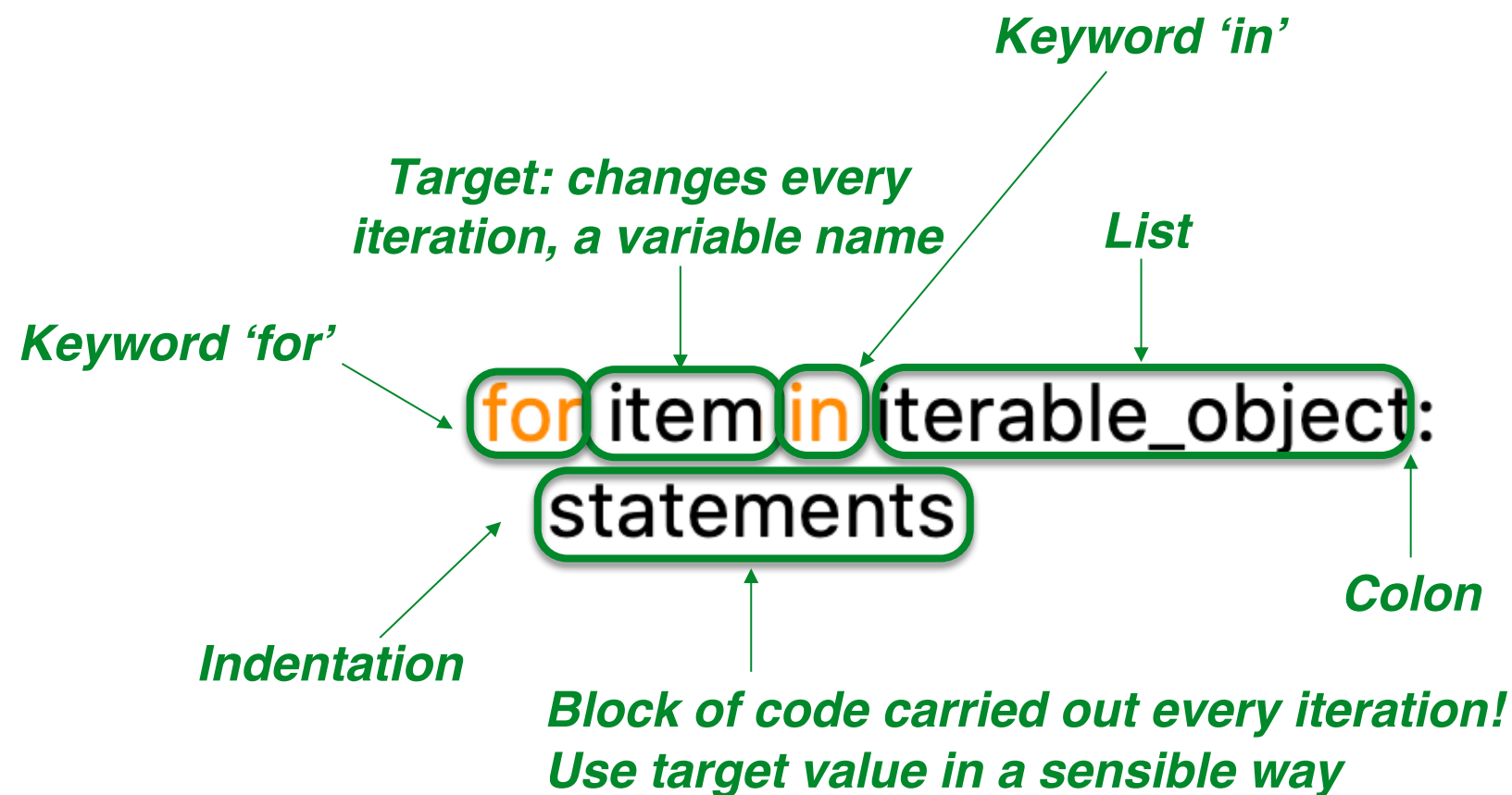
**For** Loop



condition

**While** Loop

# For Loop



# For Loop

```
for item in iterable_object:  
    statements
```

# For Loop

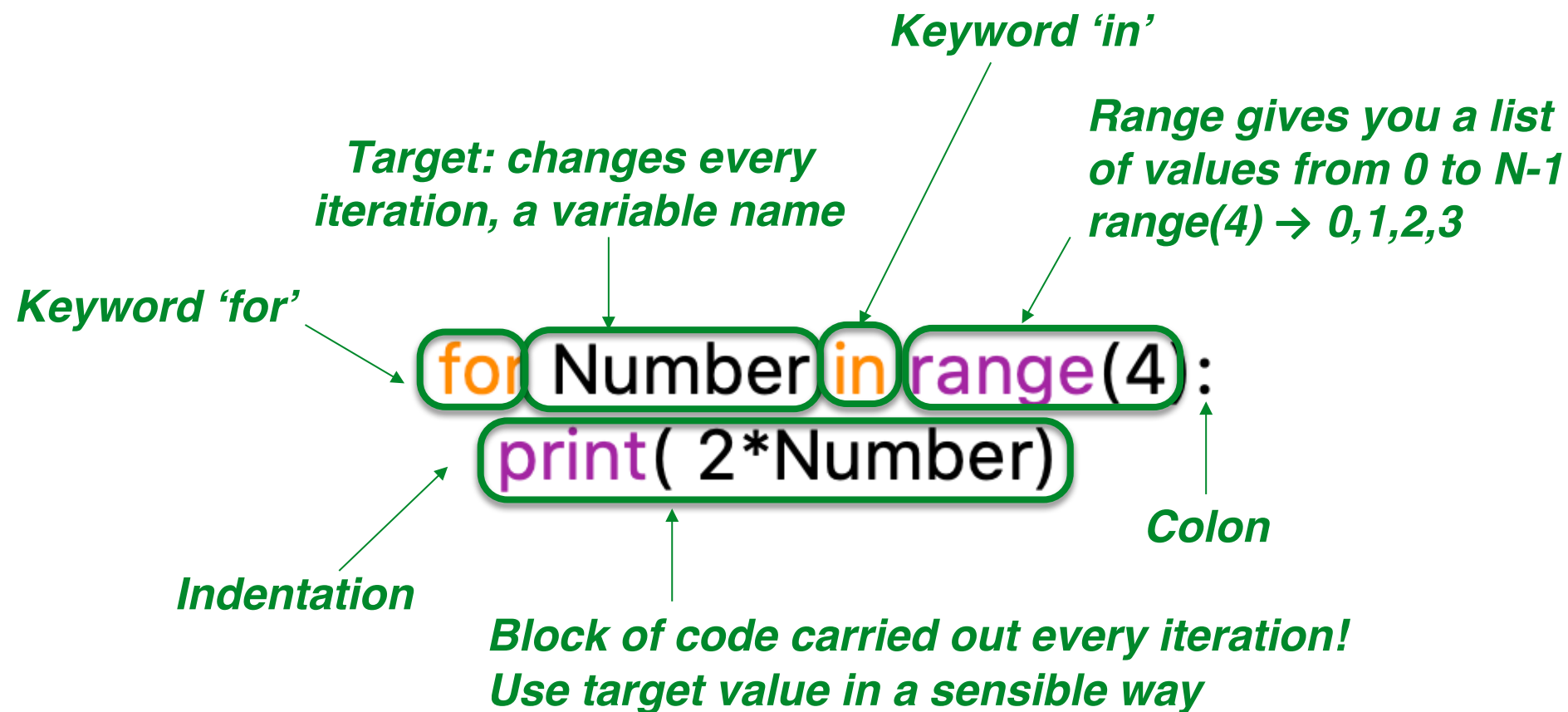
**for** item **in** iterable\_object:  
statements

e.g. [1,4,5,7]

e.g. print (2\*item)

#	current value of Item	printing out
1	item = 1	2
2	item = 4	8
3	item = 5	10
4	item = 7	14

# For Loop



# For Loop

```
w = "Hello"
```

```
for char in w:  
    print(char)
```

# Loops and Control Flow

## Printing even/odd numbers

additional  
indentation,  
part of if and  
for loop

```
for i in range(10):  
    if i%2 == 0:  
        print(i)  
    else:  
        print("-")
```

List of numbers  
from 0 to 9

even numbers

```
for i in range(10):  
    if i%2 != 0:  
        print(i)  
    else:  
        print("-")
```

!= "NOT EQUAL"

odd numbers



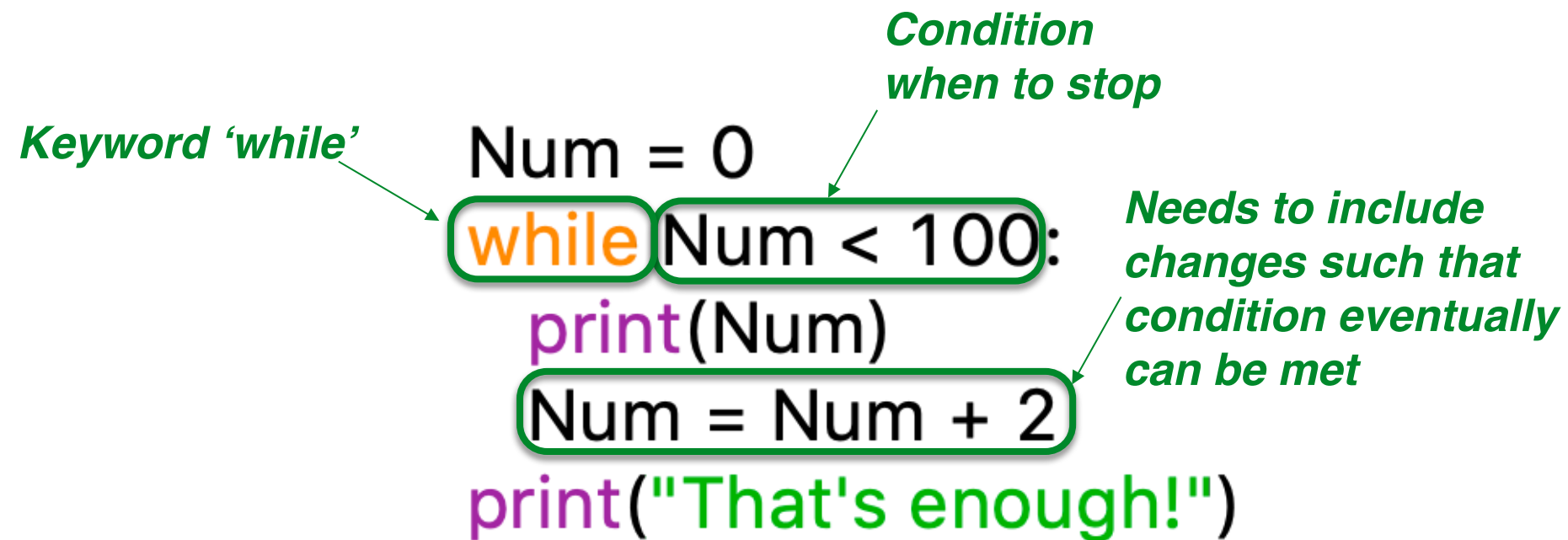
# While Loop

*Keyword 'while'*

```
Num = 0
while Num < 100:
    print(Num)
    Num = Num + 2
print("That's enough!")
```

*Condition when to stop*

*Needs to include changes such that condition eventually can be met*

A diagram illustrating a while loop. The code is shown with several annotations. The word 'while' is highlighted in orange and has a green arrow pointing to it from the text 'Keyword 'while''. The condition 'Num < 100:' is highlighted in a green rounded rectangle and has a green arrow pointing to it from the text 'Condition when to stop'. The loop body, consisting of 'print(Num)' and 'Num = Num + 2', is also highlighted in a green rounded rectangle. A green arrow points from the text 'Needs to include changes such that condition eventually can be met' to the 'Num = Num + 2' line. The final 'print' statement is in green.

# While Loop

```
Num = 0
while Num < 100:
    print(Num)
    Num = Num + 2
print("That's enough!")
```

# While Loop

Same loop  
implemented  
with while

```
Index = 0
while Index < len(w):
    print(w[Index])
    Index += 1
```

```
StoppingChar = "|"
i = 0
while w[i] != StoppingChar:
    print(w[i])
    i += 1
```


Until first  
"|" is found

# Break & Continue

## Break

Sometimes we want to exit a for or while loop prematurely i.e skip all remaining values

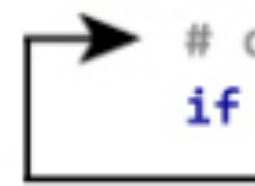
```
for var in sequence:
    # codes inside for loop
    if condition:
        break
    # codes inside for loop
# codes outside for loop
```

A diagram illustrating the 'break' statement. It shows a loop structure with a 'break' statement inside. An arrow originates from the 'break' statement, moves left, then down, and finally right to point at the code line '# codes outside for loop', indicating that the loop is exited prematurely.

## Continue

Sometimes, instead of *skipping all remaining values*, we want to skip *just one value* in a loop.

```
for var in sequence:
    # codes inside for loop
    if condition:
        continue
    # codes inside for loop
# codes outside for loop
```

A diagram illustrating the 'continue' statement. It shows a loop structure with a 'continue' statement inside. An arrow originates from the 'continue' statement, moves left, then up, and finally right to point at the start of the loop body, indicating that the current iteration is skipped and the loop proceeds to the next iteration.

# Break

```
1  for j in range(1, 20):  
2  
3      if j % 4 == 0: # Check remainder of j/4  
4          break      # continue to next value of j  
5  
6      print(j, "is not a multiple of 4")
```

```
1 is not a multiple of 4  
2 is not a multiple of 4  
3 is not a multiple of 4
```

# Continue

```
1 for j in range(1, 20):  
2  
3     if j % 4 == 0: # Check remainder of j/4  
4         continue # continue to next value of j  
5  
6     print(j, "is not a multiple of 4")
```

```
1 is not a multiple of 4  
2 is not a multiple of 4  
3 is not a multiple of 4  
5 is not a multiple of 4  
6 is not a multiple of 4  
7 is not a multiple of 4  
9 is not a multiple of 4  
10 is not a multiple of 4  
11 is not a multiple of 4  
13 is not a multiple of 4  
14 is not a multiple of 4  
15 is not a multiple of 4  
17 is not a multiple of 4  
18 is not a multiple of 4  
19 is not a multiple of 4
```

# Summary

Loops allow you to execute the same code over and over and over again.

There are two ways to do a loop: **for** and **while**.

Use **break** or **continue** to exit a loop, (or good stopping conditions).

Note: Check **help (range)** for a useful way to make lists of integers.