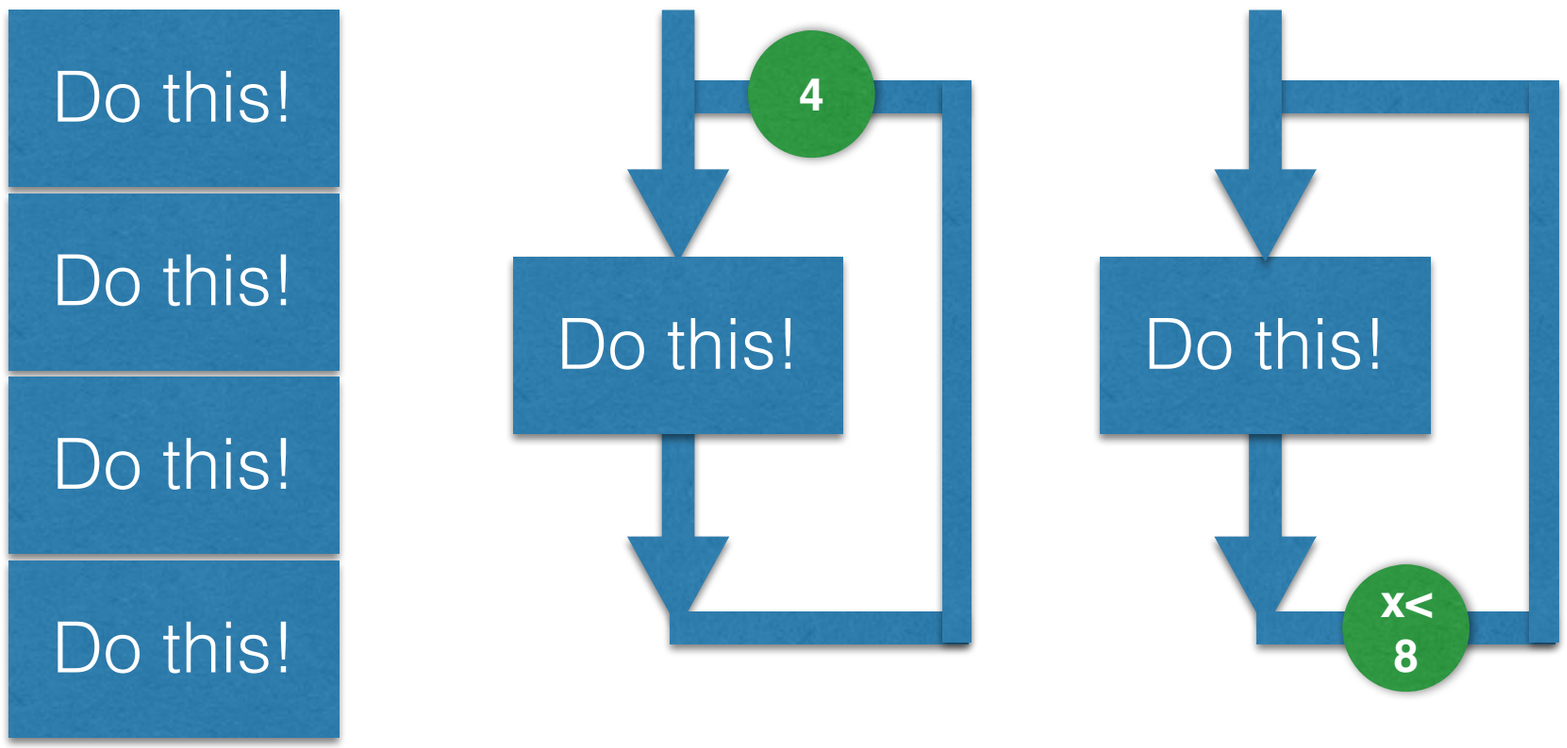
Introduction to Computer Programming Lecture 3.1:

Loops

Hemma Philamore

Department of Engineering Mathematics

Loops



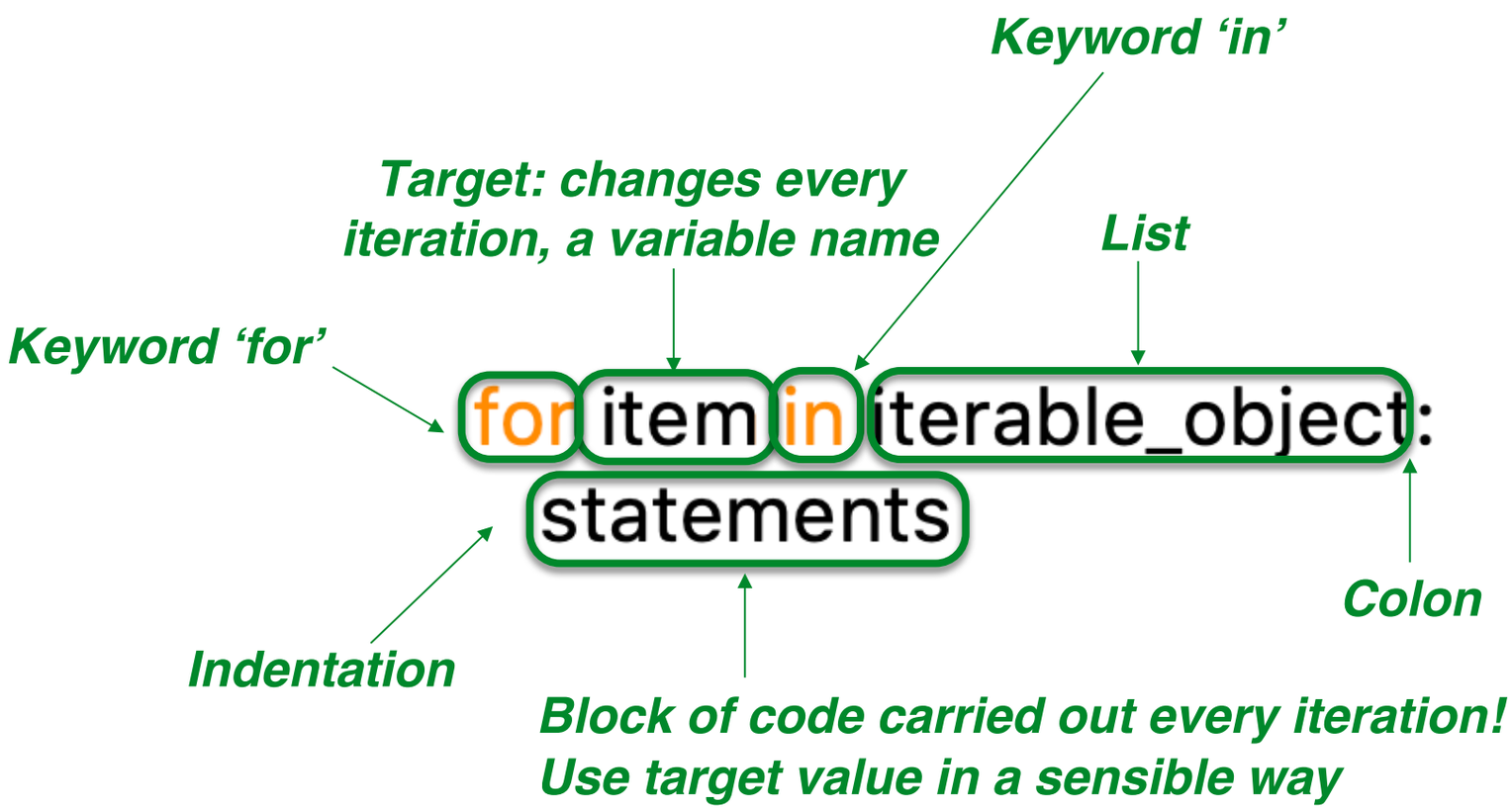
iterations

condition

For Loop

While Loop

For Loop


Keyword 'in'
Target: changes every iteration, a variable name
List
Keyword 'for'
for item in iterable_object:
statements
Colon
Indentation
Block of code carried out every iteration!
Use target value in a sensible way

For Loop

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

For Loop

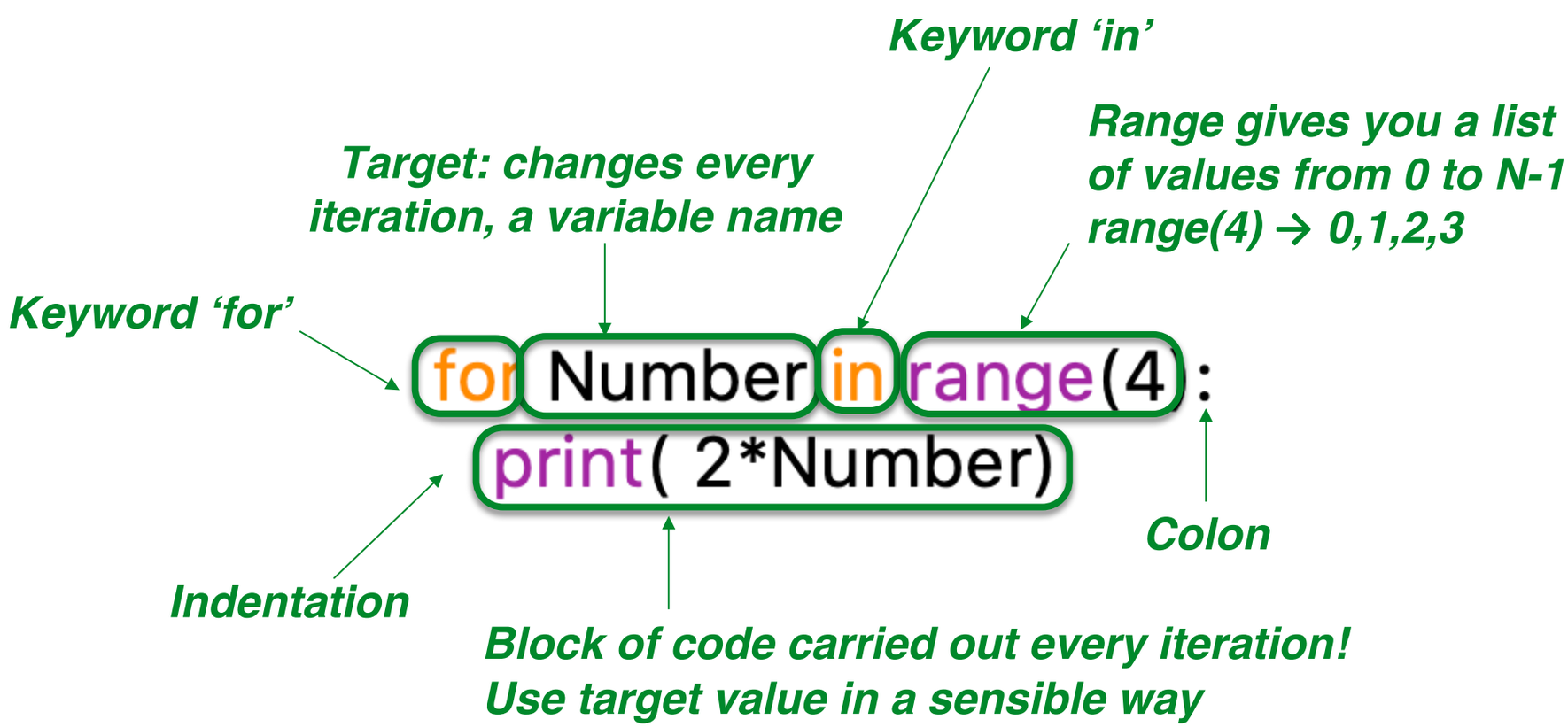
```
for item in iterable_object:
    statements
```

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


Keyword 'in'
Target: changes every iteration, a variable name
Range gives you a list of values from 0 to N-1
range(4) → 0,1,2,3
Keyword 'for'
for Number in range(4):
print(2*Number)
Colon
Indentation
Block of code carried out every iteration!
Use target value in a sensible way

For Loop

```
w = "Hello"

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

Loops and Control Flow

Printing even/odd numbers

List of numbers from 0 to 9

additional indentation, part of if and for loop

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

even numbers

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

odd numbers

!= “NOT EQUAL”

While Loop

Condition when to stop

Keyword 'while'

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

Needs to include changes such that condition eventually can be met

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
```

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
```

Break

```python
for j in range(1, 20):

    if j % 4 == 0:  # Check remainer of j/4
        break    # continue to next value of j

    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

```python
for j in range(1, 20):

    if j % 4 == 0:  # Check remainer of j/4
        continue    # continue to next value of j

    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.