

LECTURE 3.3

Loops

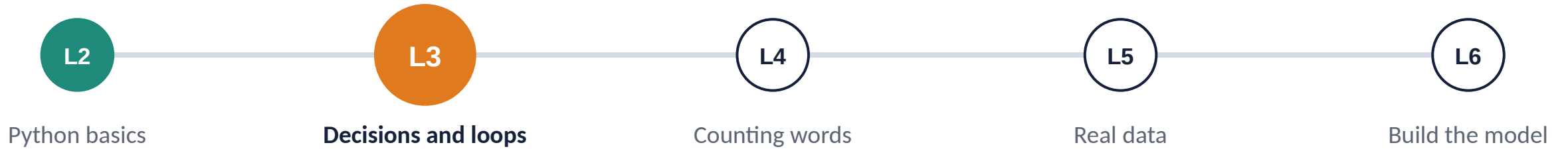
for, while, range, break: judging a whole inbox

Today's key line of code

```
for msg in inbox:  
    if "free" in msg.lower():  
        print("SPAM?", msg)
```

Where we are: Lecture 3, part 3 of 3

Part 3 of 3



Today you will

- 1 Read every part of a for loop and trace it by hand
- 2 Count with range()
- 3 Add up and count with the accumulator pattern
- 4 Repeat until a condition changes, with while
- 5 Stop early with break, skip with continue, and nest loops to build a spam filter

Why loops?

- Writing print five times does not scale
- A loop repeats the indented block for you
- Same two lines for 5 items or 5 million
- The loop variable `i` changes each round

Code

```
for i in range(1, 6):  
    print(f"Message {i} checked")
```

Output

```
Message 1 checked  
Message 2 checked  
Message 3 checked  
Message 4 checked  
Message 5 checked
```

Anatomy of a for loop

Code

```
inbox = ["Hi", "Win cash", "Bye"]  
for msg in inbox:  
    print(msg)
```

Output

```
Hi  
Win cash  
Bye
```

Keyword **for** starts the loop

Loop variable: holds the current item, a new one each round

The collection to walk through

Colon, as with if

Body: runs once for every item

Tracing a for loop, round by round

Each round, `msg` takes the next item. The body runs once per round.

Code

```
inbox = ["Hi", "Win cash", "Bye"]
for msg in inbox:
    print(len(msg), msg)
```

Output

```
2 Hi
8 Win cash
3 Bye
```

round	msg	len(msg)	prints
1	'Hi'	2	2 Hi
2	'Win cash'	8	8 Win cash
3	'Bye'	3	3 Bye

Click to reveal the next row. Predict it first!

range(): a sequence of numbers

`list()` is only here so we can see the numbers. In a for loop you write `range(...)` directly.

You type	Python gives	Meaning
<code>list(range(5))</code>	<code>[0, 1, 2, 3, 4]</code>	0 up to 5, stop not included
<code>list(range(1, 6))</code>	<code>[1, 2, 3, 4, 5]</code>	start at 1, stop before 6
<code>list(range(0, 11, 2))</code>	<code>[0, 2, 4, 6, 8, 10]</code>	third number = step
<code>list(range(5, 0, -1))</code>	<code>[5, 4, 3, 2, 1]</code>	negative step counts down
<code>len(range(1, 10))</code>	<code>9</code>	how many numbers: 9

Predict the range

Predict first

Code

```
for i in range(2, 11, 3):  
    print(i)
```



What will this print? Write your prediction as a comment before you run the cell.

Answer: Predict the range

Code

```
for i in range(2, 11, 3):  
    print(i)
```

Output

```
2  
5  
8
```

- Start at 2, add 3 each time
- 2, 5, 8 are below 11
- The next value, 11, is not included

The accumulator: adding up as you go

Start with 0 before the loop, then add each item inside the loop.

Code

```
nums = [4, 7, 1, 8]
total = 0
for n in nums:
    total = total + n
print(total)
```

Output

20

round	n	total before	total after
1	4	0	4
2	7	4	11
3	1	11	12
4	8	12	20

Click to reveal the next row. Predict it first!

Watch it run: a for loop

WATCH FOR

- The highlighted line is the next one Python runs
- `n` takes a new value each round
- `total` grows by `n` each round
- `total = 0` runs only once, before the loop

WATCH THE CODE RUN, STEP BY STEP

A for loop in motion

Watch `n` change each round and `total` grow.

The highlighted line is the NEXT line Python will run.

Click the video to play, click again to pause. Silent: talk the class through it.

Counting: an if inside a loop

The count goes up only in rounds where the condition is True.

Code

```
count = 0
for i in range(1, 8):
    if i % 2 == 0:
        count = count + 1
print(count)
```

Output

3

i	i % 2	even?	count
1	1	False	0
2	0	True	1
3	1	False	1
4	0	True	2
5	1	False	2
6	0	True	3
7	1	False	3

Click to reveal the next row. Predict it first!

Anatomy of a while loop

Code

```
n = 3
while n > 0:
    print(n)
    n = n - 1
print("Go!")
```

Output

```
3
2
1
Go!
```

Keyword **while**: repeat as long as the condition is True

Condition, checked before every round

Body: runs while the condition is True

Update: moves towards the end. Without it, the loop never stops

After the loop, when the condition becomes False

Tracing the while loop

The condition is checked before each round. The loop ends the first time it is False.

Code

```
n = 3
while n > 0:
    print(n)
    n = n - 1
print("Go!")
```

Output

```
3
2
1
Go!
```

n	n > 0	what happens
3	True	print 3, then n = 2
2	True	print 2, then n = 1
1	True	print 1, then n = 0
0	False	leave loop, print Go!

Click to reveal the next row. Predict it first!

Watch it run: a while loop

WATCH FOR

- The condition is checked before every round
- n goes 3, 2, 1, 0
- When $n > 0$ is False, Python jumps past the loop
- Without $n = n - 1$ it would never end

WATCH THE CODE RUN, STEP BY STEP

A while loop counting down

The condition is checked before EVERY round.

Watch n get smaller until $n > 0$ becomes False.

Click the video to play, click again to pause. Silent: talk the class through it.

for or while?

for

- You know what to walk through
- Each item of a list, or a range of numbers
- Ends by itself
- Example: check every message

while

- Repeat until something changes
- You do not know how many rounds
- You must update the condition yourself
- Example: ask again until the password is right

Infinite loops, and how to stop one

Without `n = n - 1`, `n` stays 3 and `n > 0` is always True. In Colab, click the stop square next to the cell, or Runtime > Interrupt execution.

Code

```
n = 3
while n > 0:
    print(n)
    # forgot: n = n - 1    -> prints 3 forever
```

break: stop the loop early

- **break** leaves the loop immediately
- Useful when you have found what you were looking for
- Messages after the break are never checked
- It usually sits inside an if

Code

```
inbox = ["Hi", "See you", "WIN cash", "Bye"]
for msg in inbox:
    if "win" in msg.lower():
        print("First spam:", msg)
        break
    print("OK:", msg)
```

Output

```
OK: Hi
OK: See you
First spam: WIN cash
```

continue: skip to the next round

- `continue` skips the rest of this round only
- The loop carries on with the next item
- Useful for skipping empty or invalid data
- Real data always has some

Code

```
for msg in ["Hi", "", "Bye", ""]:  
    if msg == "":  
        continue  
    print("Checking:", msg)
```

Output

```
Checking: Hi  
Checking: Bye
```

enumerate(): position and item together

- `enumerate` gives (position, item) pairs
- Unpack them into two names, as with tuples
- Handy for numbered reports
- Positions start at 0, like indexing

Code

```
inbox = ["Hi", "Win cash", "Bye"]  
for i, msg in enumerate(inbox):  
    print(i, msg)
```

Output

```
0 Hi  
1 Win cash  
2 Bye
```

Nested loops: a loop inside a loop

- The inner loop runs completely for each round of the outer loop
- Here: every message is checked against every spam word
- **found** resets to False for each new message
- 3 messages x 2 words = 6 checks

Code

```
spam_words = ["free", "win"]
inbox = ["Win cash", "Class at 10",
         "Free gift"]
for msg in inbox:
    found = False
    for word in spam_words:
        if word in msg.lower():
            found = True
    print(found, msg)
```

Output

```
True Win cash
False Class at 10
True Free gift
```

Tracing the nested loop

Outer loop: one message at a time. Inner loop: every word for that message.

Code

```
for msg in inbox:
    found = False
    for word in spam_words:
        if word in msg.lower():
            found = True
    print(found, msg)
```

msg	word	word in msg?	found
'Win cash'	'free'	False	False
'Win cash'	'win'	True	True
'Class at 10'	'free'	False	False
'Class at 10'	'win'	False	False
'Free gift'	'free'	True	True
'Free gift'	'win'	False	True

Click to reveal the next row. Predict it first!

Watch it run: nested loops

WATCH FOR

- Outer loop: msg changes slowly
- Inner loop: word changes quickly
- found resets for each new message
- 4 checks in total: 2 messages x 2 words

WATCH THE CODE RUN, STEP BY STEP

Nested loops: every word, every message

The inner loop runs completely for EACH round of the outer loop.
Watch msg change slowly and word change quickly.

Click the video to play, click again to pause. Silent: talk the class through it.

Build it: a rule-based spam filter

Loops + decisions + a list of suspicious words: the filter from Lecture 1's demo, in real code.

Code

```
spam_words = ["free", "win", "prize", "cash"]
inbox = ["Win a free phone",
         "Class moved to room 305",
         "Claim your cash prize",
         "See you at the library"]
spam_count = 0
for msg in inbox:
    is_spam = False
    for word in spam_words:
        if word in msg.lower():
            is_spam = True
    if is_spam:
        spam_count = spam_count + 1
        print("SPAM ", msg)
    else:
        print("OK ", msg)
print(f"Flagged {spam_count} of {len(inbox)}")
```

Output

```
SPAM  Win a free phone
OK    Class moved to room 305
SPAM  Claim your cash prize
OK    See you at the library
Flagged 2 of 4
```

Where rules break

Same rules, two trickier messages. Predict the labels before you run it.

Code

```
tricky = ["Claim your FR33 g1ft now",  
          "Free lunch after the seminar"]  
for msg in tricky:  
    is_spam = False  
    for word in spam_words:  
        if word in msg.lower():  
            is_spam = True  
    print(is_spam, msg)
```

Output

```
False Claim your FR33 g1ft now  
True Free lunch after the seminar
```

Common mistakes with loops

Read the last line of an error first. It names the problem.

What you typed	What Python says	What it means
<pre>for i in range(5) print(i)</pre>	<code>SyntaxError: expected ':'</code>	Missing colon after the for line.
<pre>for i in range(5): print(i)</pre>	<code>IndentationError: expected an indented block after 'for' statement on line 1</code>	The body must be indented.
<pre>for x in 5: print(x)</pre>	<code>TypeError: 'int' object is not iterable</code>	A number is not a collection. Use range(5).
<pre>for i in range(3): totl = totl + i</pre>	<code>NameError: name 'totl' is not defined</code>	totl was never created. Create it before the loop, and check spelling.

In-class exercises

Each task uses a loop. Test with $N = 20$ unless stated.

1 Even numbers

Print all even numbers from 1 to N .

2 Skip fives

Print 1 to N except numbers divisible by 5.

3 3 or 4, not 5

Print numbers from 1 to N divisible by 3 or 4 but not by 5.

4 Primes

Print all primes up to N . Expected for 20: 2 3 5 7 11 13 17 19.

5 Second largest

Find the second largest distinct value of [4, 9, 2, 9, 7] with a loop, no sort. Expected: 7.

6 Sum of digits

Given $n = 4721$, add its digits with a while loop. Hint: $n \% 10$ and $n // 10$. Expected: 14.

Solutions

Show these after students have tried the exercises.

Solutions 1 and 2

`range(1, N + 1)` includes N. `end=" "` prints on one line instead of a new line each time.

Code

```
N = 20
for i in range(1, N + 1):
    if i % 2 == 0:
        print(i, end=" ")
print()
for i in range(1, N + 1):
    if i % 5 != 0:
        print(i, end=" ")
```

Output

```
2 4 6 8 10 12 14 16 18 20
1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
```

Solution 3: brackets matter

and is evaluated before **or**. Without brackets, 15 would wrongly be printed.

Code

```
N = 20
for i in range(1, N + 1):
    if (i % 3 == 0 or i % 4 == 0) and i % 5 != 0:
        print(i, end=" ")
```

Output

```
3 4 6 8 9 12 16 18
```

Solution 4: primes with a nested loop

A prime has no divisor between 2 and itself. **break** stops checking as soon as one divisor is found.

Code

```
N = 20
for n in range(2, N + 1):
    is_prime = True
    for d in range(2, n):
        if n % d == 0:
            is_prime = False
            break
    if is_prime:
        print(n, end=" ")
```

Output

```
2 3 5 7 11 13 17 19
```

Solutions 5 and 6

Second largest: skip the maximum, then track the largest of the rest. Digits: take the last digit, then drop it.

Code

```
nums = [4, 9, 2, 9, 7]
largest = max(nums)
second = None
for x in nums:
    if x != largest:
        if second == None or x > second:
            second = x
print(second)

n = 4721
total = 0
while n > 0:
    total = total + n % 10
    n = n // 10
print(total)
```

Output

```
7
14
```

Interactive lab: Loops Lab

In the lab you can

1 Loop stepper

Step any loop round by round; the trace table fills itself.

2 range() explorer

Move start, stop and step and see the numbers on a line.

3 Nested loop grid

Watch every message checked against every spam word.

4 Predict the loop

8 short loops: choose the output, then see why.

LECTURE 3.3 · INTERACTIVE LAB

Loops Lab

Step through loops round by round, explore range(), and watch a spam filter check every word

Introduction to Data Science and Machine Learning
Md Mahbub E Noor · University of Barishal
[All Labs](#)

1 Loop stepper 2 range() explorer 3 Nested loop grid 4 Predict the loop

Loop stepper: watch every round

Pick an example from the lecture, or write your own. Step forwards and backwards; the trace table fills in exactly like the ones on the slides.

Example: Accumulator: add up [Edit code](#)

```
1 nums = [4, 7, 1, 8]
2 total = 0
3 for n in nums:
4     total = total + n
5 print(total)
```

round 3

◀ Back ▶ Next step ▶ Play ▶ Run to end

↺ Start again

Play speed Step 7 of 12 outer loop: round 3

Line 3: Round 3 of 4: n = 1.

VARIABLES

nums	list	total	int	n	int
[4, 7, 1, 8]		11		1	

OUTPUT

TRACE TABLE: EVERY STEP SO FAR

Open the lab ▶

Works offline in any browser

Labs_open_in_browser/
L3.3_Loops_Lab.html

Today's toolkit

for x in items:	Repeat for each item
range(a, b, step)	Numbers from a, stop before b
total = 0 ... total += x	Accumulator pattern
while condition:	Repeat while True

break	Leave the loop now
continue	Skip to the next round
for i, x in enumerate(items):	Position and item
loop inside a loop	Every pair of items

Homework 3.3

Due before Lecture 4.1 • [date]

1 Predict, then run

Predict first: the output of `for i in range(10, 0, -3): print(i)`, and `print(len(range(0, 100, 10)))`.

2 Multiplication table

Ask for no input: store `n = 7` and print `n x 1` up to `n x 10` in the form `7 x 3 = 21`.

3 Trace it

Draw a trace table by hand for the accumulator slide with `nums = [2, 5, 3]`. Then check by running.

4 Spam counter

Make 8 invented messages. Count how many contain any word from your `spam_words` list, using nested loops.

5 Guessing game

Store `secret = 7` and a list of guesses `[3, 9, 7, 5]`. Loop over guesses, print Too low / Too high, and break when correct.

Submit on [LMS / Google Classroom]: your notebook file `HW3_3_<your ID>.ipynb`, or a Colab link set to “Anyone with the link can view”.

Recap: Lecture 3.3 in 90 seconds

RECAP

Lecture 3.3 in 90 seconds

A short recap of today's lecture. Pause any time.

Click to play or pause. Watch it again at home to revise.