teaching demo

haley keglovits

Topic: Loops

goals:

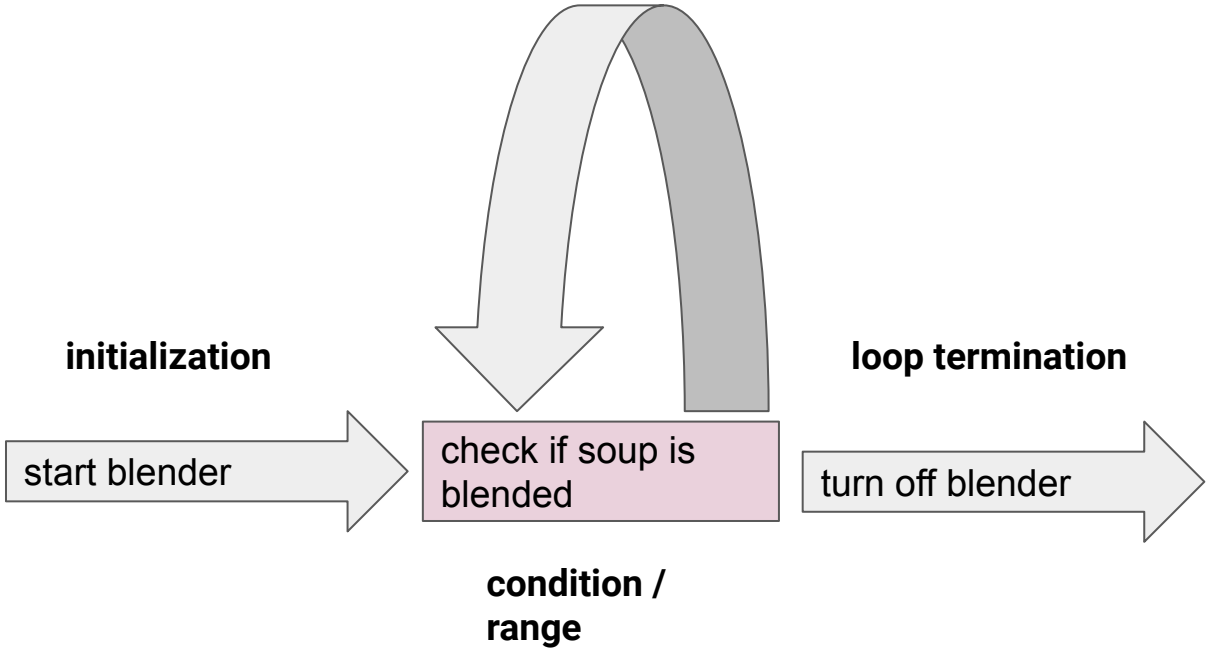
1. learn how to run code more than once
2. decide whether **for** or **while** loops are appropriate
3. introduction to code syntax for loops

follow along: haleyk.github.io/sheridan.pdf

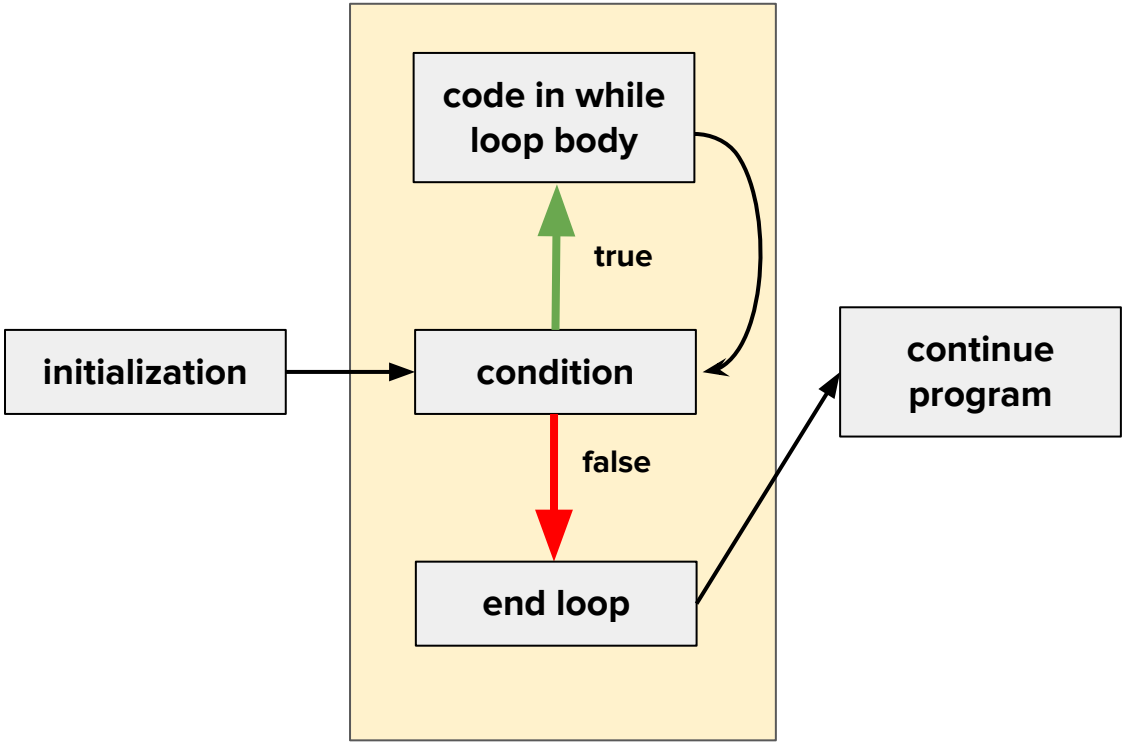
Last time…

- reviewed **logic** (decide if something is true or false) and **conditionals** (use a true/false decision to run only selected code)
  - recipe metaphor: **vegetarian? if yes, add vegetable stock, if no, add chicken stock. *not both!***
- today: what if we want to run code more than once? can we do this flexibly?
  - blend the soup until no chunks remain
  - for every serving, chop 3 carrots
  - line every cupcake tin with paper

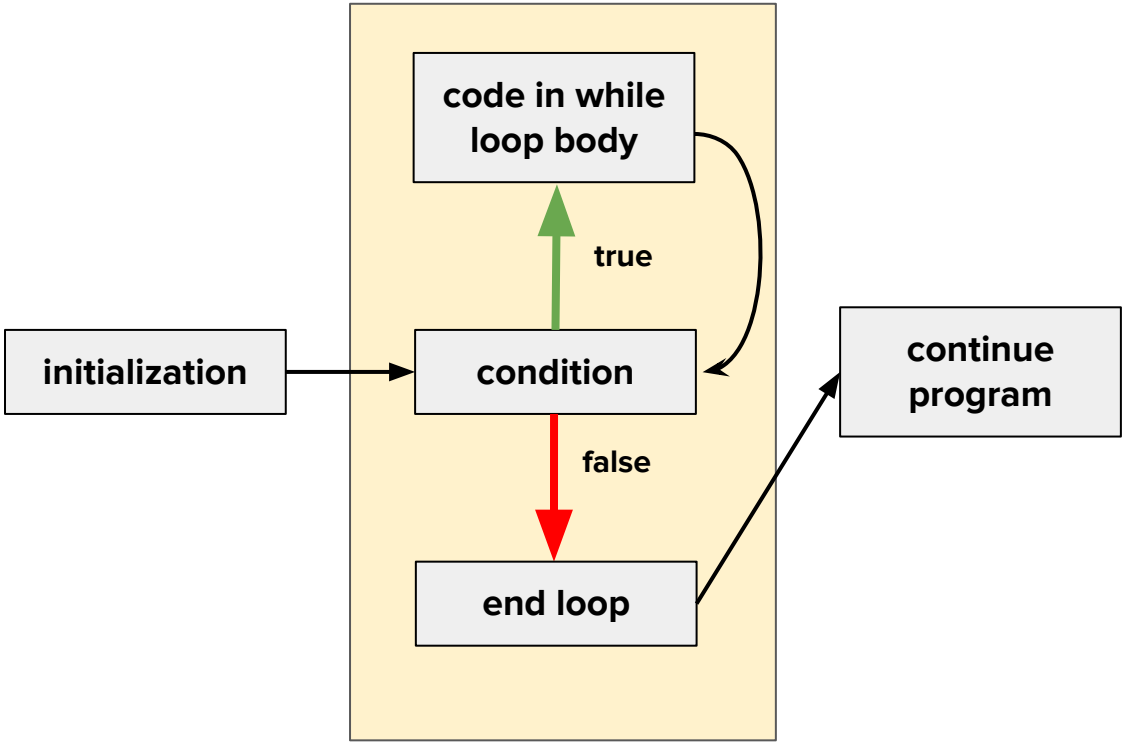
How does a loop work? Birds eye view and terminology



WHILE


code in while loop body
true
initialization
condition
continue program
false
end loop

WHILE



when to use:

- you want to continue doing something UNTIL a condition changes, and you *don't know how long* that will take
- considered 'indefinite' iteration

examples:

- collect coins up to a value
- braid hair
- while time hasn't run out, keep working

WHILE

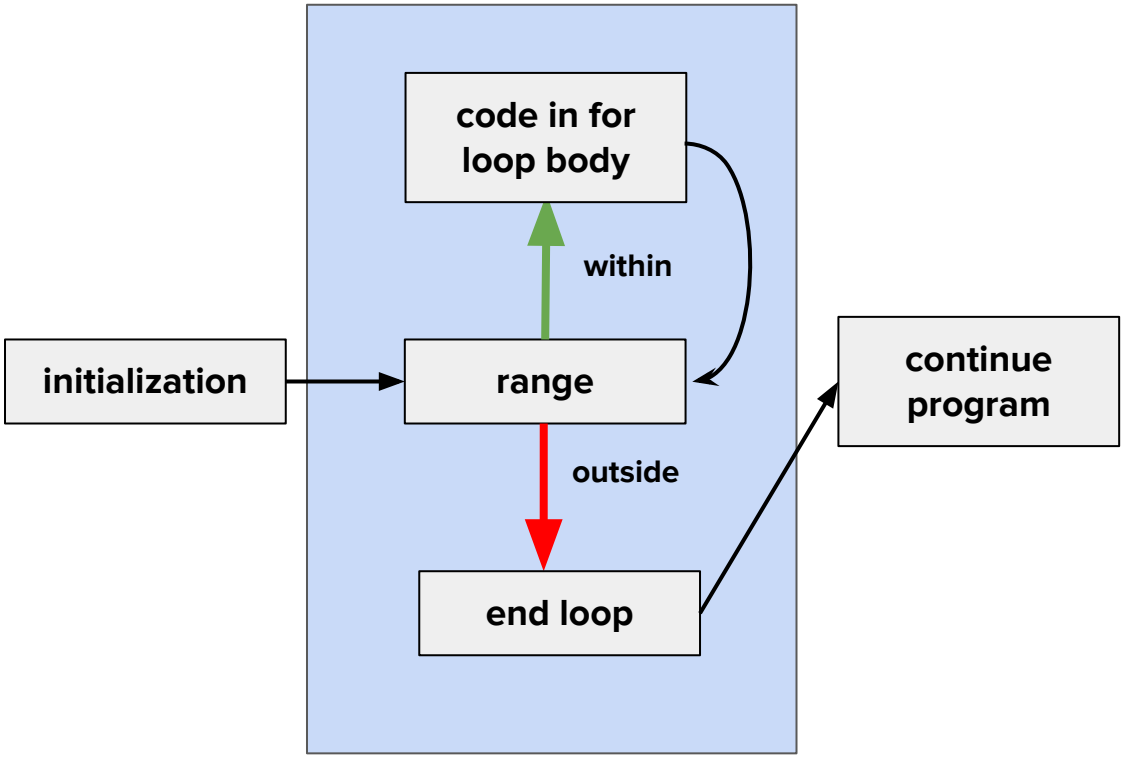
```
1  import random as r
2
3  myval = r.random()
4  print(myval)
5  while myval < .9:
6      myval = r.random()
7      print(myval)
8  print('found a large number')
```

```
0.7290243685083269
0.3869166751818798
0.49232939433446343
0.2174256616939827
0.9245344281906942
found a large number
```

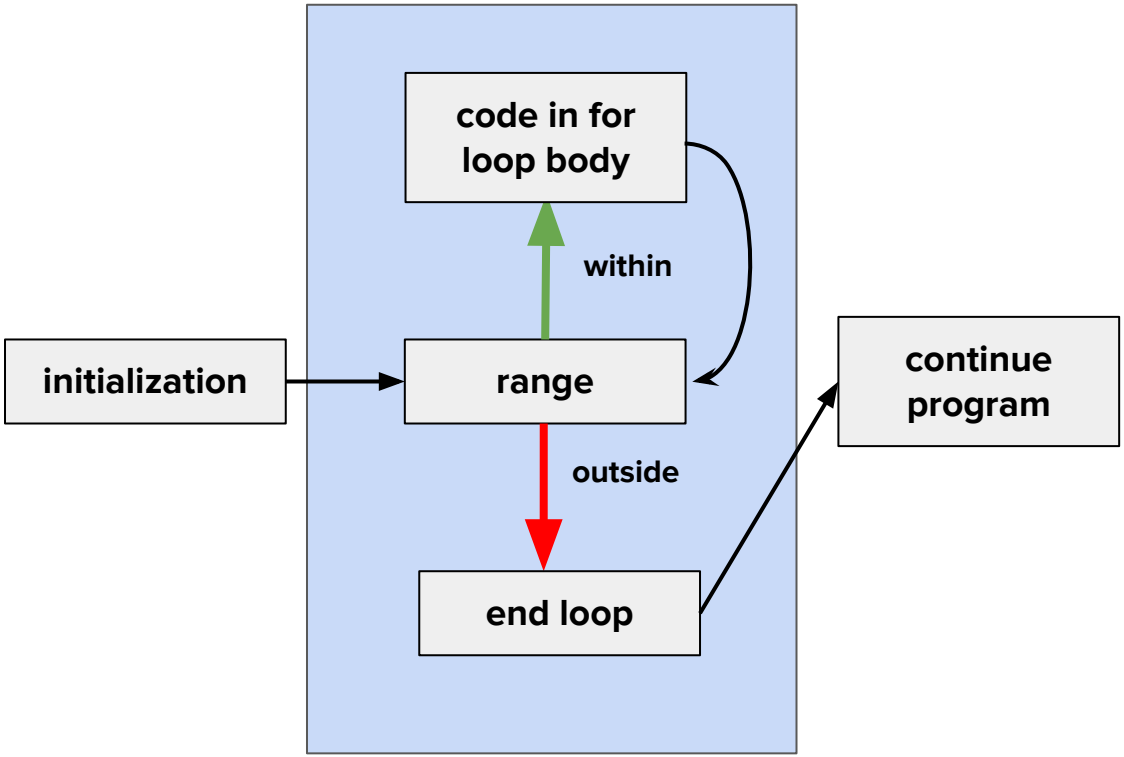
```
while LOGICAL_CONDITION_IS_TRUE:
    run_some_code
# hitting this line means the logical condition is false!
```

indentation is still important!

FOR


code in for
loop body
within
initialization
range
outside
end loop
continue
program

FOR



when to use:

- you want to do a process over a *KNOWN* set of information or repetitions
- considered 'definite' iteration

examples:

- put icing on each cupcake
- square a list of numbers
- check how many values in list exceed a certain number (think: points)
- push ups/cheer at football game

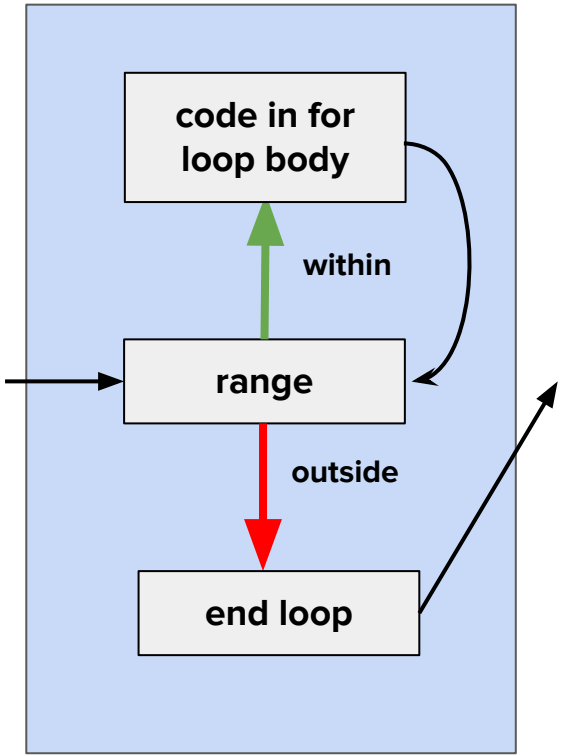
FOR

```
for SELECTION in ITERABLE:
    run_some_code
# hitting this line means you have done whole iterable!
```

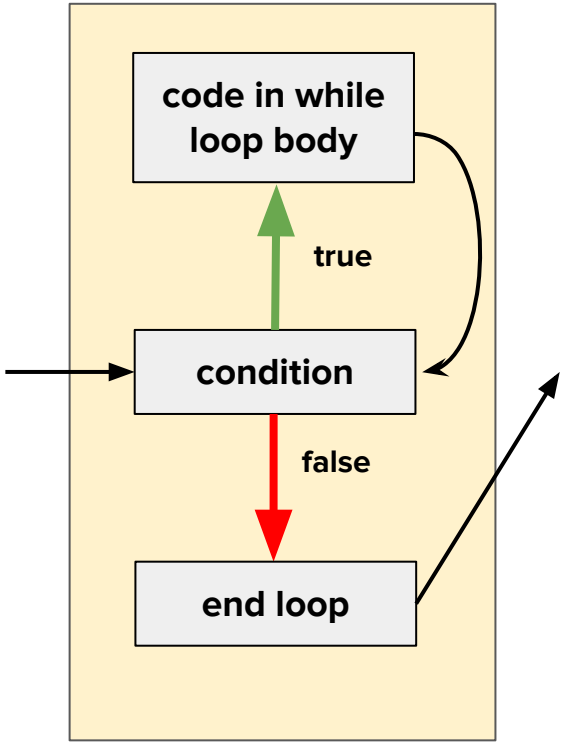
```
1  values = [1,2,3,4,5]
2  for v in values:
3      square_v = v**2
4      print('the square of '+str(v)+' is '+str(square_v))
```

```
the square of 1 is 1
the square of 2 is 4
the square of 3 is 9
the square of 4 is 16
the square of 5 is 25
```

FOR


code in for loop body
within
range
outside
end loop

WHILE


code in while loop body
true
condition
false
end loop

Think-Pair-Share

First: From the following list, can you identify which should be for loops and which should be while loops? *If you aren't sure, write down why you think it could be one or the other to discuss with your partner*

- you want to check how many of your kids is wearing a seatbelt
- you want to dig a hole to find buried treasure

If time: can you think of one new example (for code or from your life) that would fit in each type of loop?

Think-Pair-Share: Solution

First: From the following list, can you identify which should be for loops and which should be while loops? *If you aren't sure, write down why you think it could be one or the other to discuss with your partner*

- you want to check how many of your kids is wearing a seatbelt: **FOR**
- you want to dig a hole to find buried treasure**: WHILE**

class ideas:

-

Some loops can be formatted both ways!

Common Bugs and Looking Ahead

never ending code (especially common with **while** loops)

never starting code (not always a bug!)

next time:

- writing loops which don’t have to run
- variable initialization: before or within loops?
- see power of loops: same code for any length of list