Computing GC content
with for loops

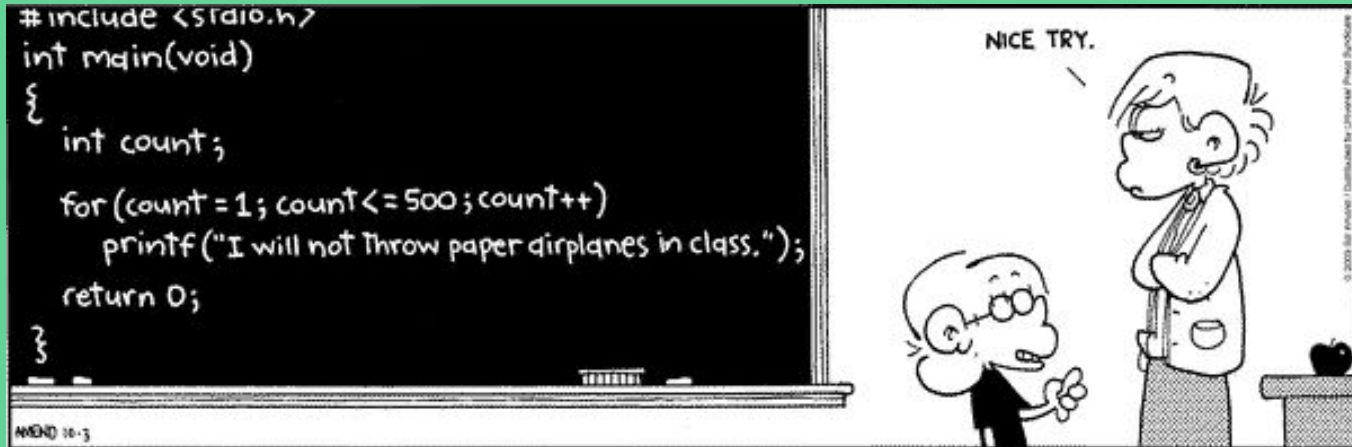**Recap of last time**: computing GC content with conditionals

Can we do this with `elif` statements?
Can we alert the user if the function gets incorrect input?

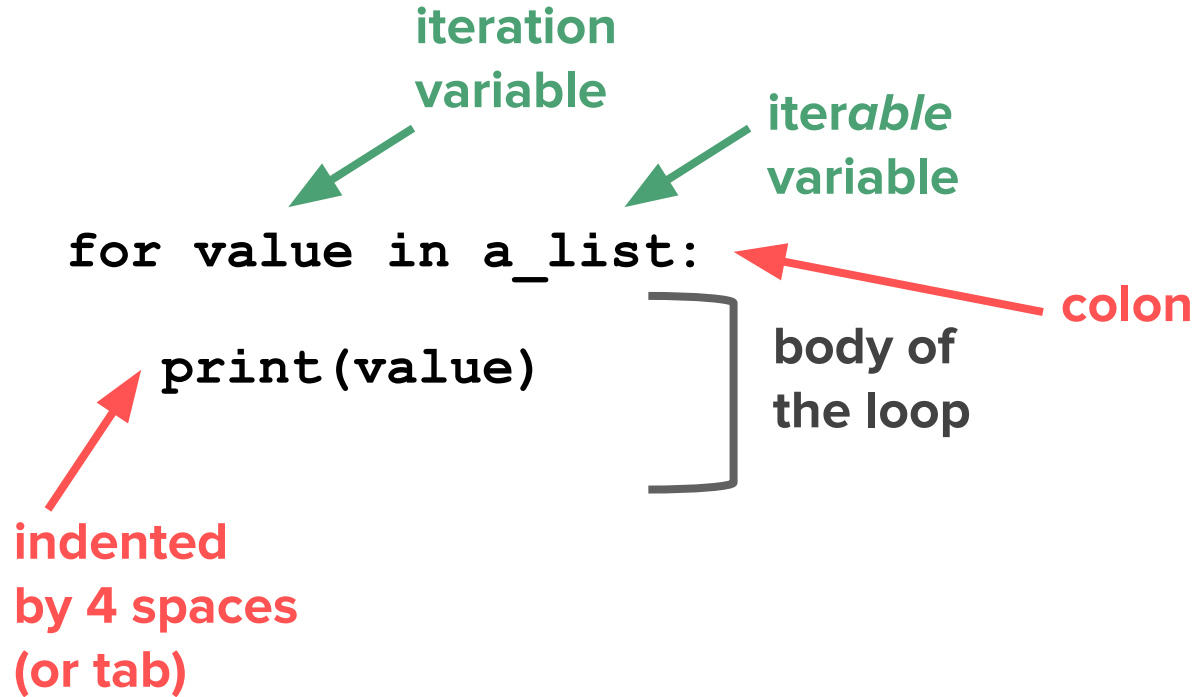A **loop** is a procedure to repeat a piece of code.

Loops enable you to re-run blocks of code for as many times as you need.

Python has two main ways to run loops: `for` & `while`

Bill Amend, FoxTrot, October 3, 2003

for loop syntax

iteration
variable

iter*able*
variable

```
for value in a_list:
    print(value)
```
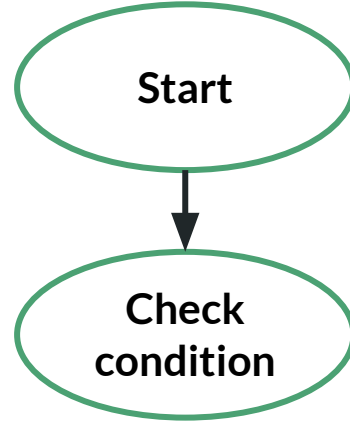
colon

body of
the loop

indented
by 4 spaces
(or tab)

A **for loop** is a
procedure to
repeat code for
every element in
a sequence.

for loop syntax

```
a_list = [1,2,3]

for value in a_list:

    print(value)
```
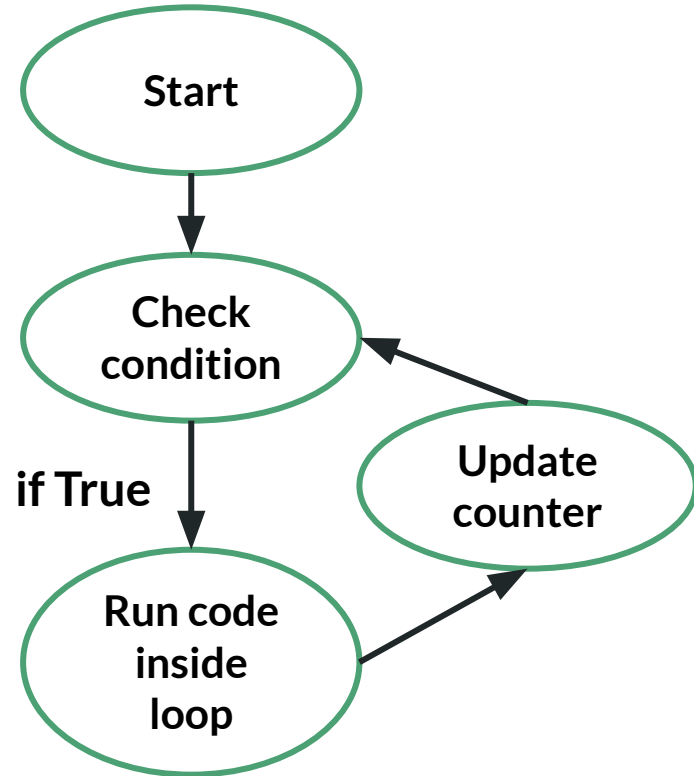
Start

Check condition

for loop syntax

```
a_list = [1,2,3]

for value in a_list:

    print(value)
```

1 | output

for loop syntax

```
a_list = [1,2,3]

for value in a_list:

    print(value)
```

```
1
2  output
```

Start

Check condition

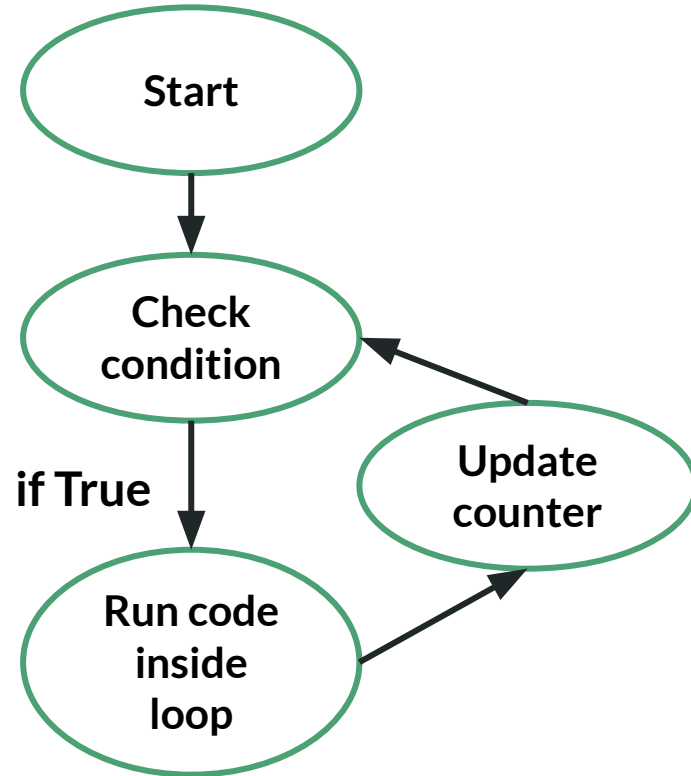if True
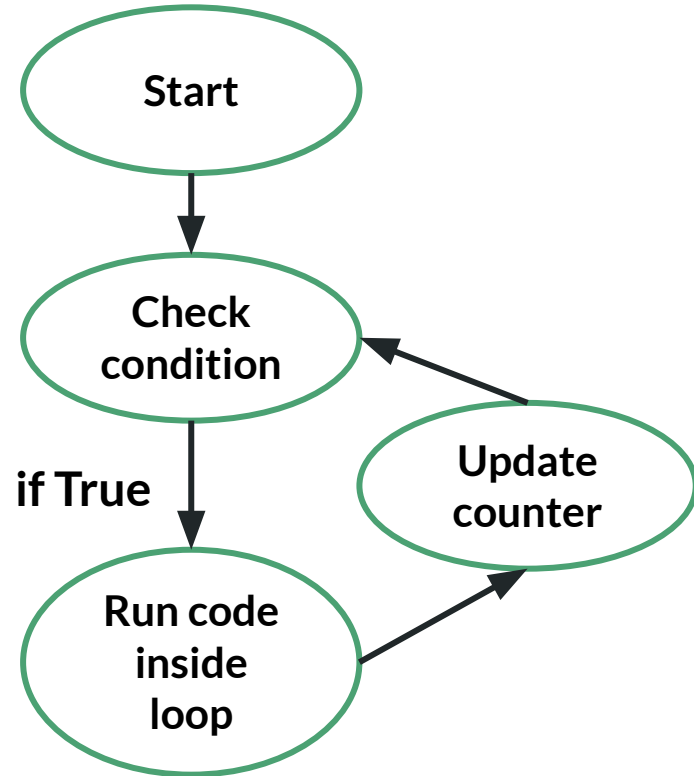
Run code inside loop

Update counter

for loop syntax

```
a_list = [1,2,3]

for value in a_list:

    print(value)
```

```
1
2   output
3
```

for loop syntax

```
a_list = [1,2,3]

for value in a_list:

    print(value)
```
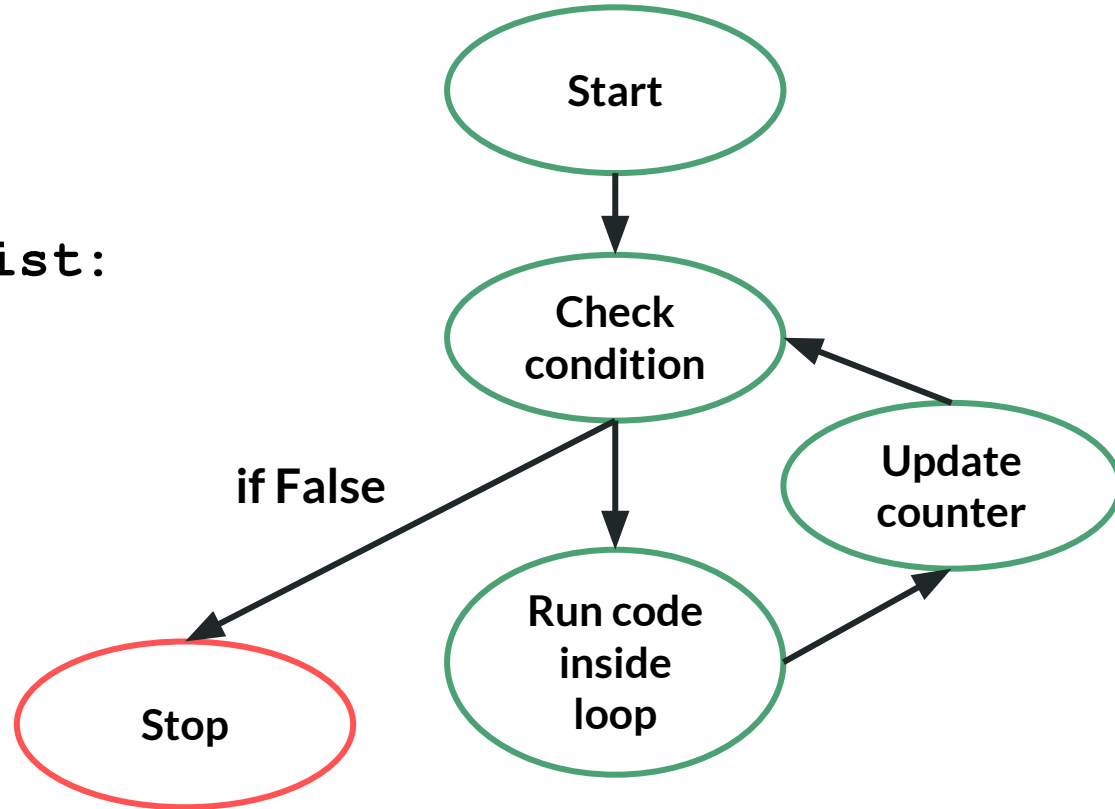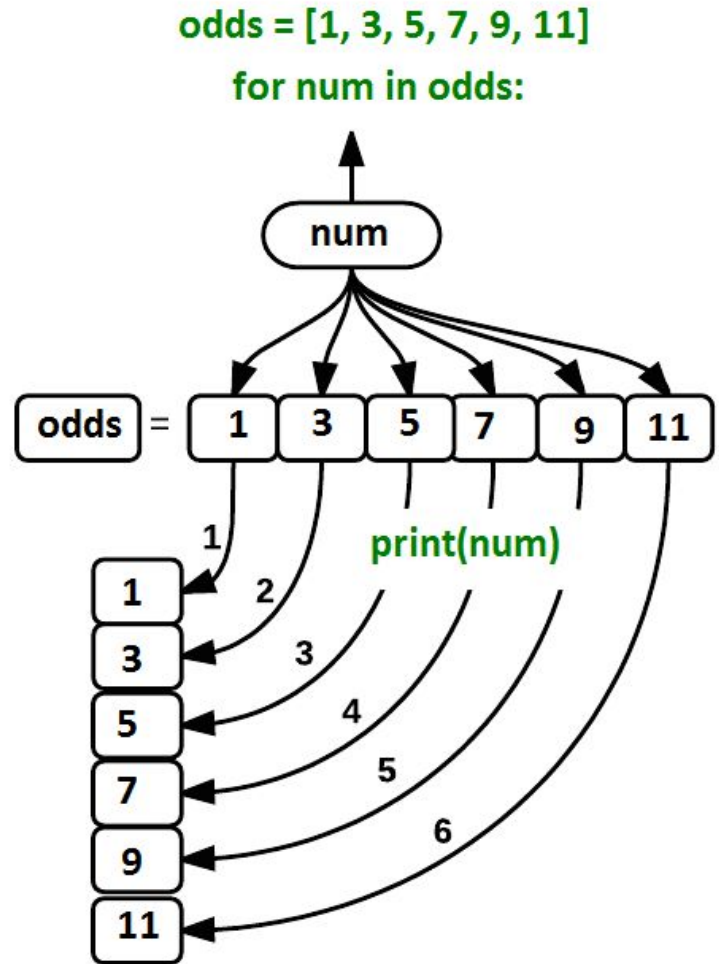
1 ⎤
2 ⎥ output
3 ⎦

Start

Check condition

if False

Stop

Run code inside loop

Update counter

Another way to visualize working through a loop
(source)

efficiency benefit of **`for`** loops

Each of these would accomplish the same thing:

Option #1: 2+ lines of code

```
for value in a_list:

    print(value)
```

Option #2: as many lines of code as there are list entries

```
        print(a_list[0])
        print(a_list[1])
        print(a_list[2])
         ...
```

Second task: count the # of "CAT" boxes (`CCAAT`) in a string of DNA.

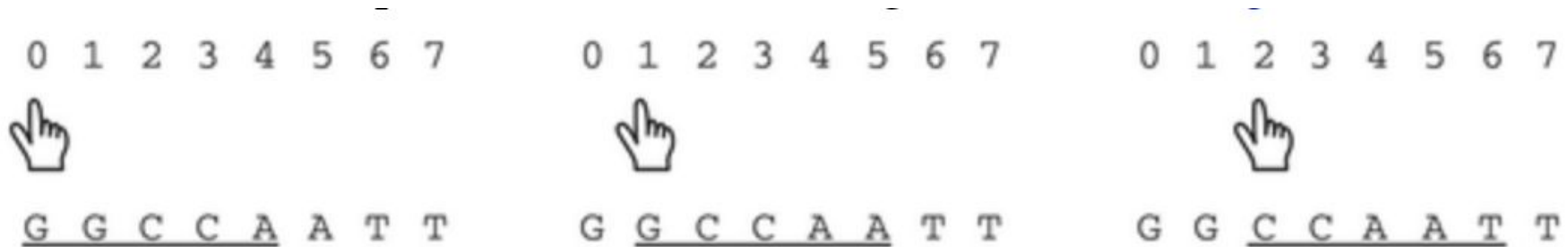The "CAT" box generally appears near the spot where transcription begins!


```
>>> countCCAAT('GGCCAATTGCCAAT')
>>> 2
```
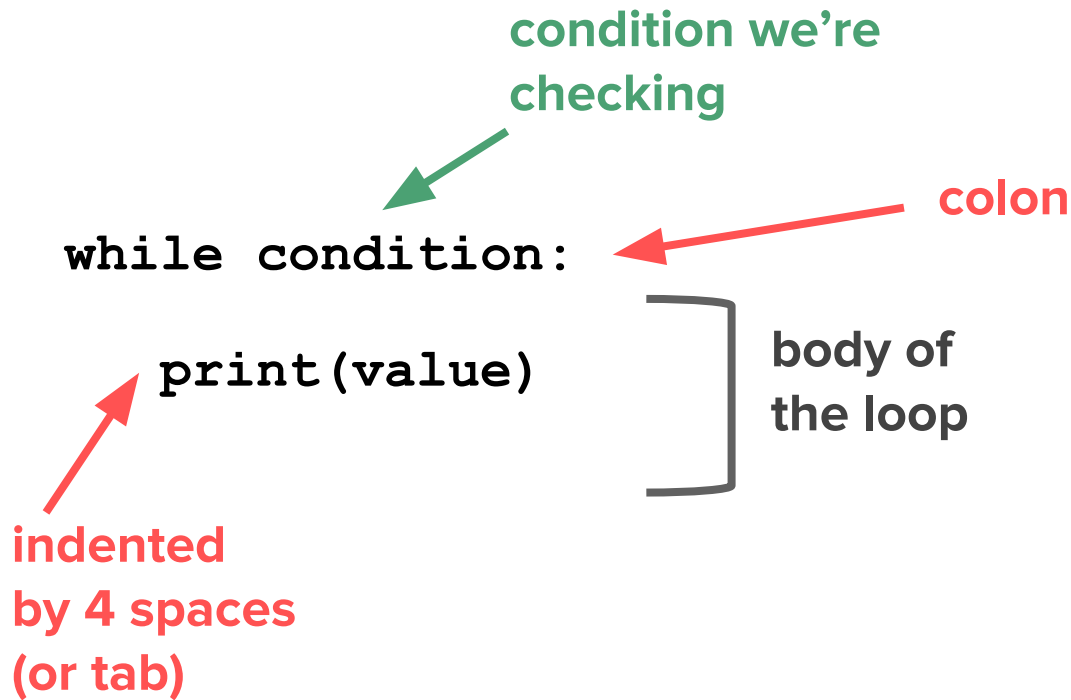
We can also loop over a list of indices!

Let's say we want to look for a "CAT" box, a common motif in DNA, with the sequence `CCAATT`

Since we want to look at a **slice** of DNA, rather than looping through individual items in the string, we need the indices.
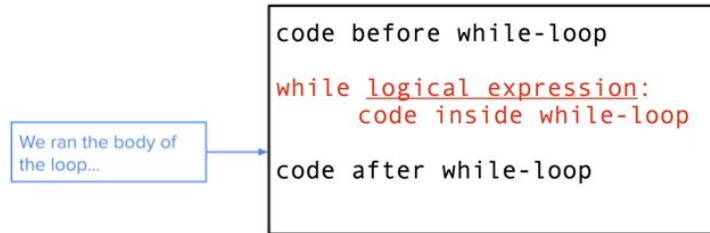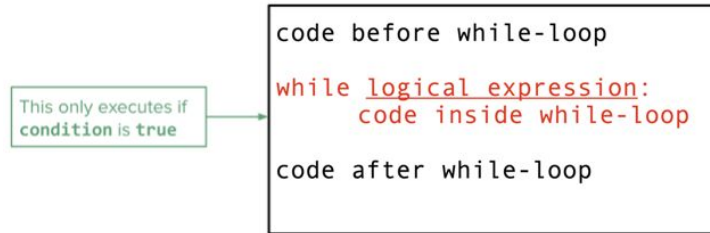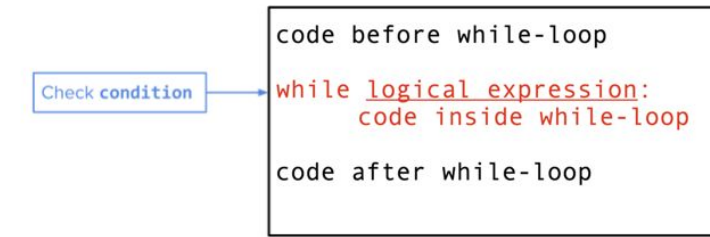
`while` loop syntax

condition we're
checking

colon

`while condition:`

`    print(value)`

body of
the loop

indented
by 4 spaces
(or tab)

While this
condition is
true, the loop
will run!

It will repeat
until the
condition is no
longer True.

```
Check condition    →    code before while-loop

                        while logical expression:
                                code inside while-loop

                        code after while-loop
```

```
                        code before while-loop

This only executes if   while logical expression:
condition is true   →           code inside while-loop

                        code after while-loop
```

```
                        code before while-loop

                        while logical expression:
                                code inside while-loop

We ran the body of  →   code after while-loop
the loop...
```

Order of execution in a while loop (from Stepik)
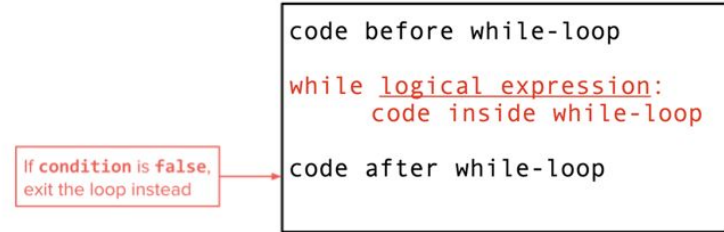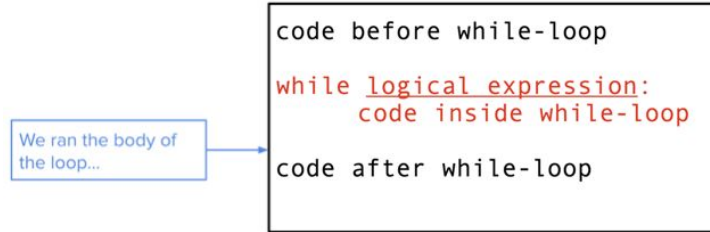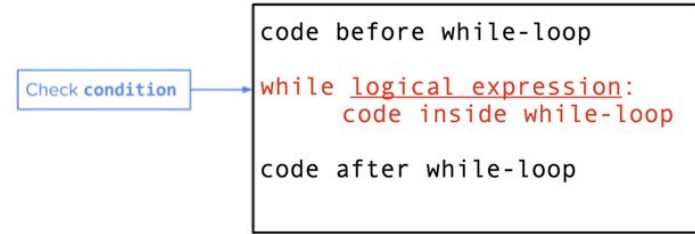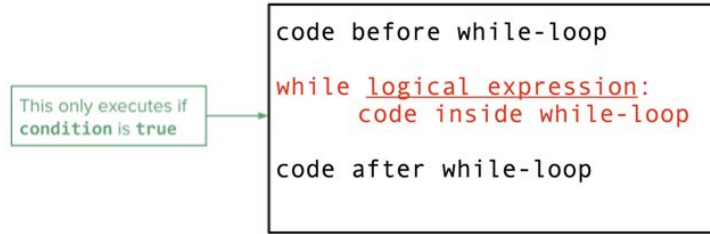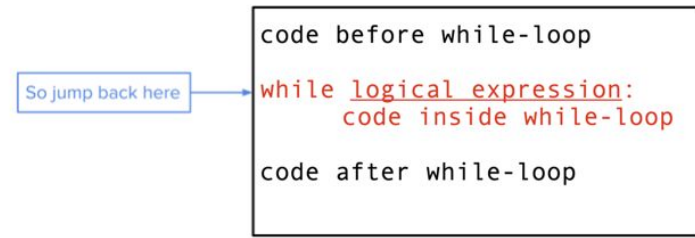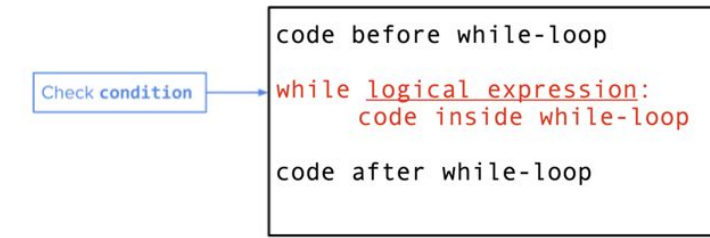
Order of execution in a while loop (from Stepik)

Resources

[Stepik Introduction to Python book, Chapter 3](#)

[Whirlwind Tour of Python: Control Flow](#)