While Loops

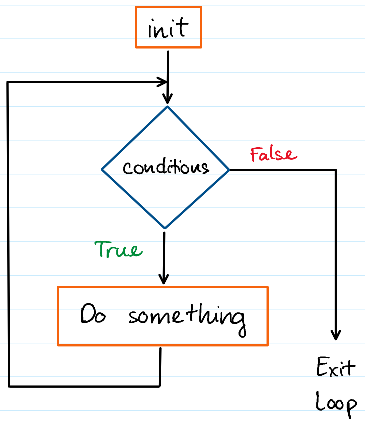
Tuesday, February 1, 2022 3:10 PM

1. While Loops

Looping means repeating something over and over until a particular condition is satisfied

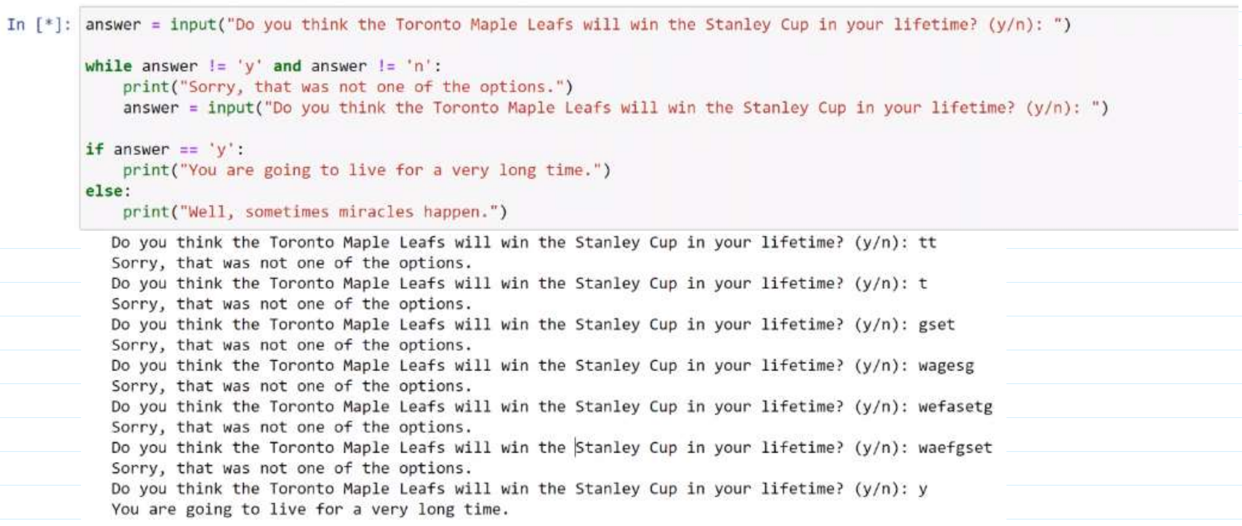
We will keep looping as long as condition is True
stop when it becomes False

```
        while expression :   ← colon
indent → do something
```



How it works :

- The condition that gets evaluated is just a boolean expression.
  It can include :
  +) Something that evaluates to True or False
  +) Logical operators (and , or , not)
  +) Comparision operators
  +) Function calls



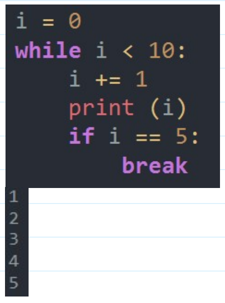
```
In [*]: answer = input("Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): ")

        while answer != 'y' and answer != 'n':
            print("Sorry, that was not one of the options.")
            answer = input("Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): ")

        if answer == 'y':
            print("You are going to live for a very long time.")
        else:
            print("Well, sometimes miracles happen.")
```

```
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): tt
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): t
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): gset
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): wagesg
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): wefasetg
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): waefgset
Sorry, that was not one of the options.
Do you think the Toronto Maple Leafs will win the Stanley Cup in your lifetime? (y/n): y
You are going to live for a very long time.
```

2. Break, continue

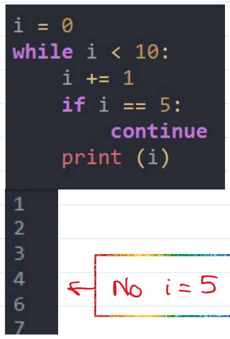
break is used to exit the while loop when met.

continue is used to immediately end the current loop and continue to next loop when met.



```
i = 0
while i < 10:
    i += 1
    print (i)
    if i == 5:
        break
```

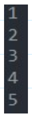
```
1
2
3
4
5
```



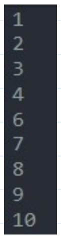
```
i = 0
while i < 10:
    i += 1
    if i == 5:
        continue
    print (i)
```

```
1
2
3
4
6
7
```

← No i = 5 printed !



```
1
2
3
4
5
```

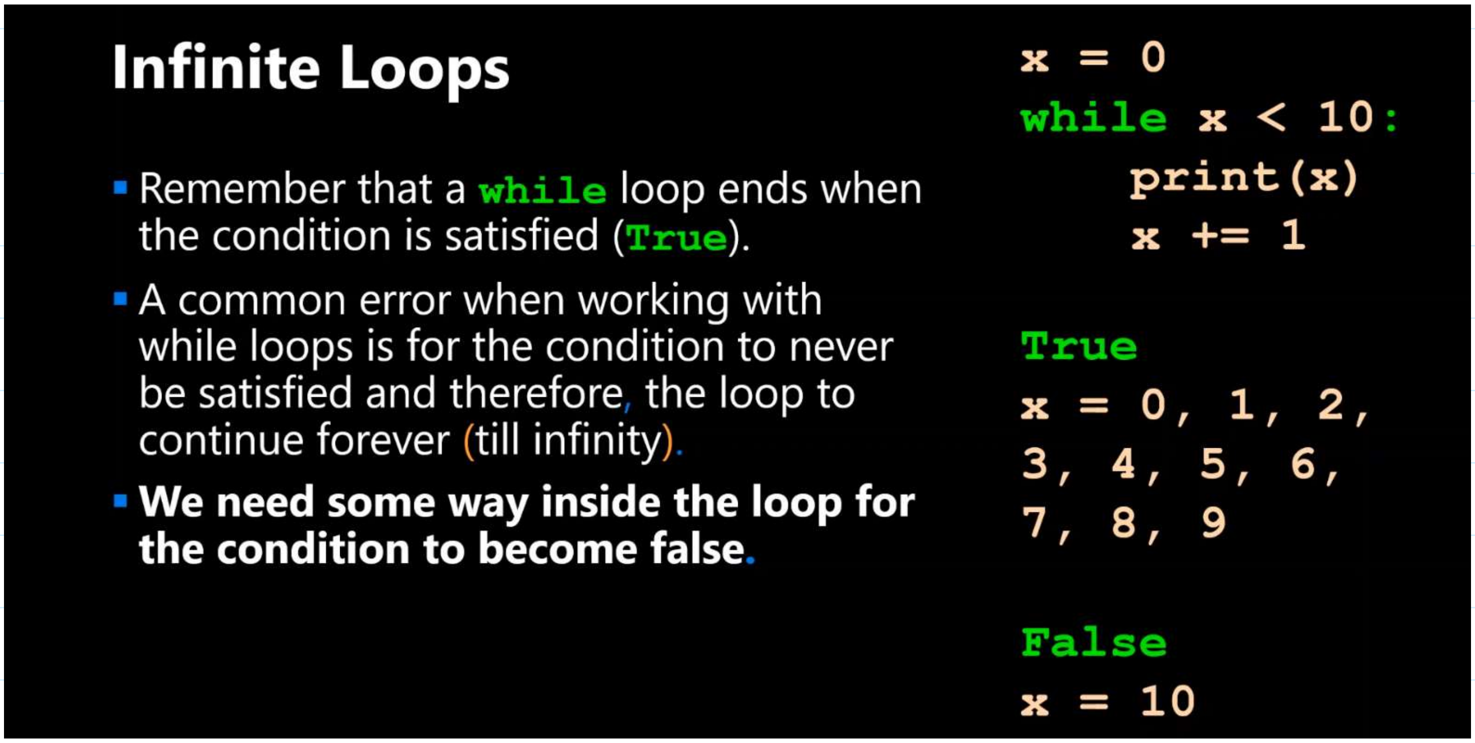


```
1
2
3
4
6
7
8
9
10
```

No i = 5 printed !

Infinite Loops

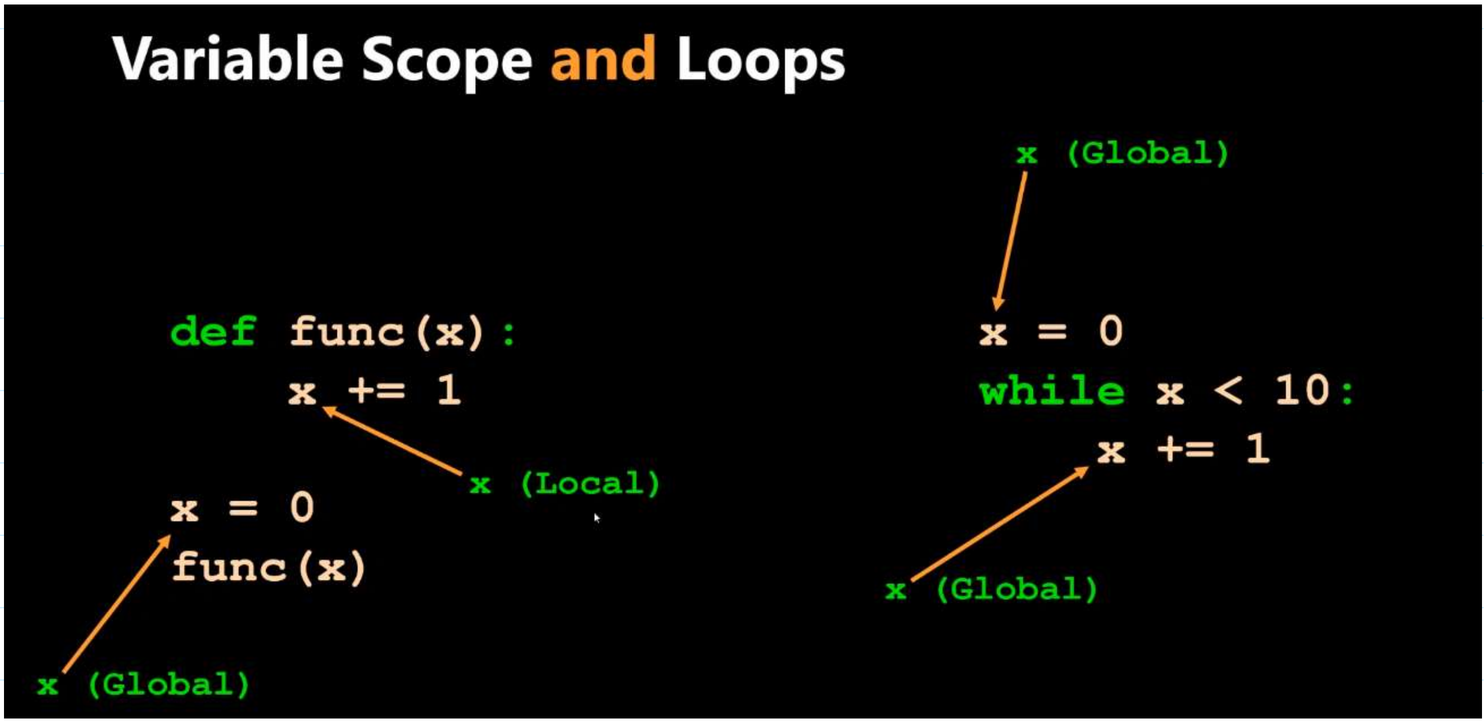
Tuesday, February 1, 2022 3:26 PM



Infinite Loops

- Remember that a `while` loop ends when the condition is satisfied (`True`).
- A common error when working with while loops is for the condition to never be satisfied and therefore, the loop to continue forever (till infinity).
- **We need some way inside the loop for the condition to become false.**

```
x = 0
while x < 10:
    print(x)
    x += 1

True
x = 0, 1, 2,
3, 4, 5, 6,
7, 8, 9

False
x = 10
```



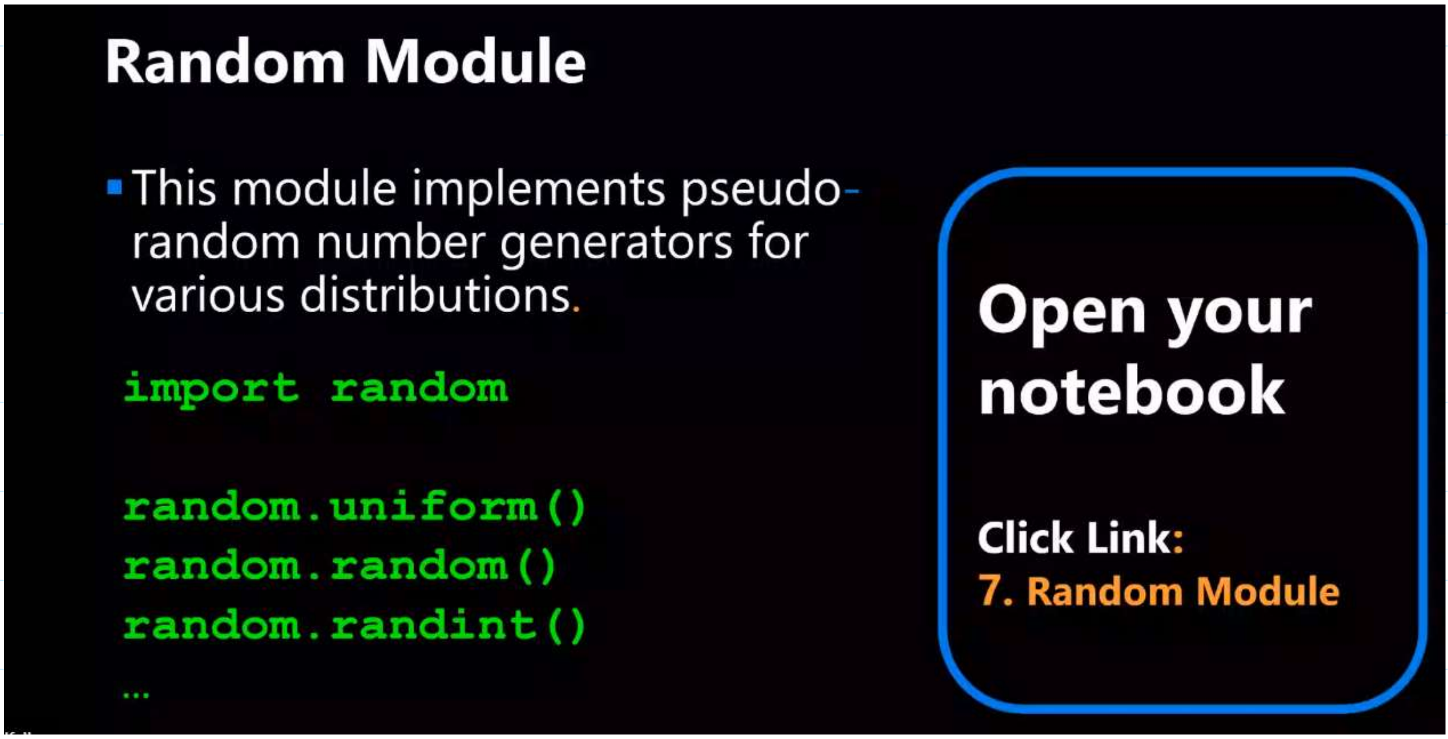
Variable Scope and Loops

```
def func(x):
    x += 1

x = 0
func(x)
```

```
x = 0
while x < 10:
    x += 1
```

Random Module

Tuesday, February 1, 2022 3:54 PM



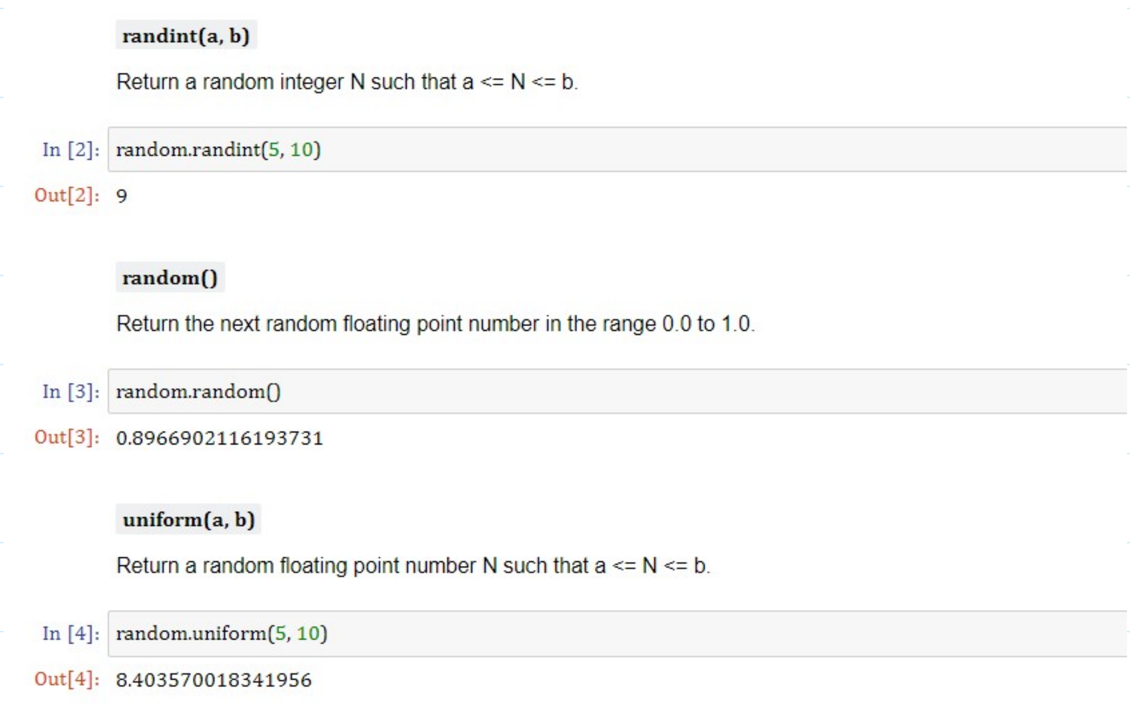
Random Module

- This module implements pseudo-random number generators for various distributions.

```
import random

random.uniform()
random.random()
random.randint()
...
```

Open your notebook

Click Link:
7. Random Module



randint(a, b)

Return a random integer N such that a <= N <= b.

```
In [2]: random.randint(5, 10)
Out[2]: 9
```

random()

Return the next random floating point number in the range 0.0 to 1.0.

```
In [3]: random.random()
Out[3]: 0.8966902116193731
```

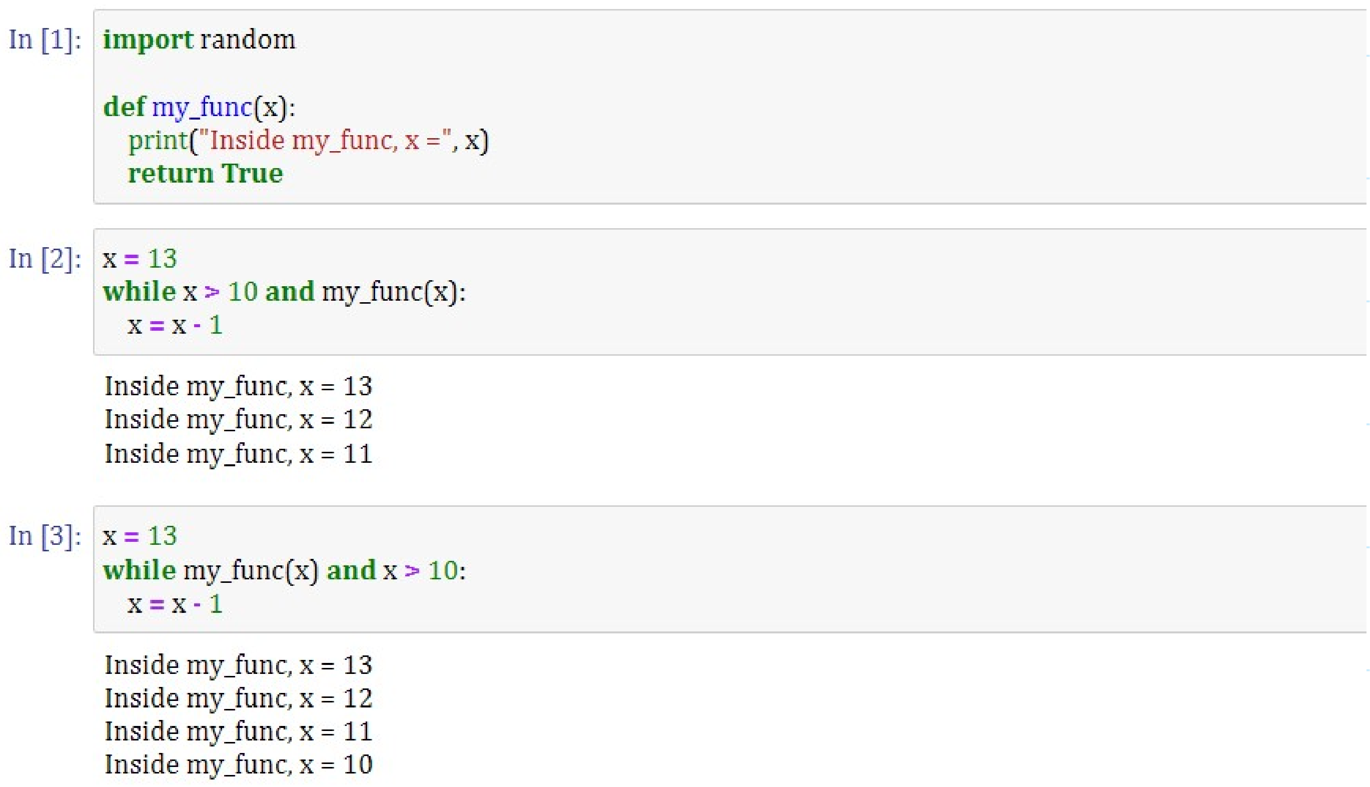
uniform(a, b)

Return a random floating point number N such that a <= N <= b.

```
In [4]: random.uniform(5, 10)
Out[4]: 8.403570018341956
```

Lazy Evaluation

Thursday, February 3, 2022 3:16 PM



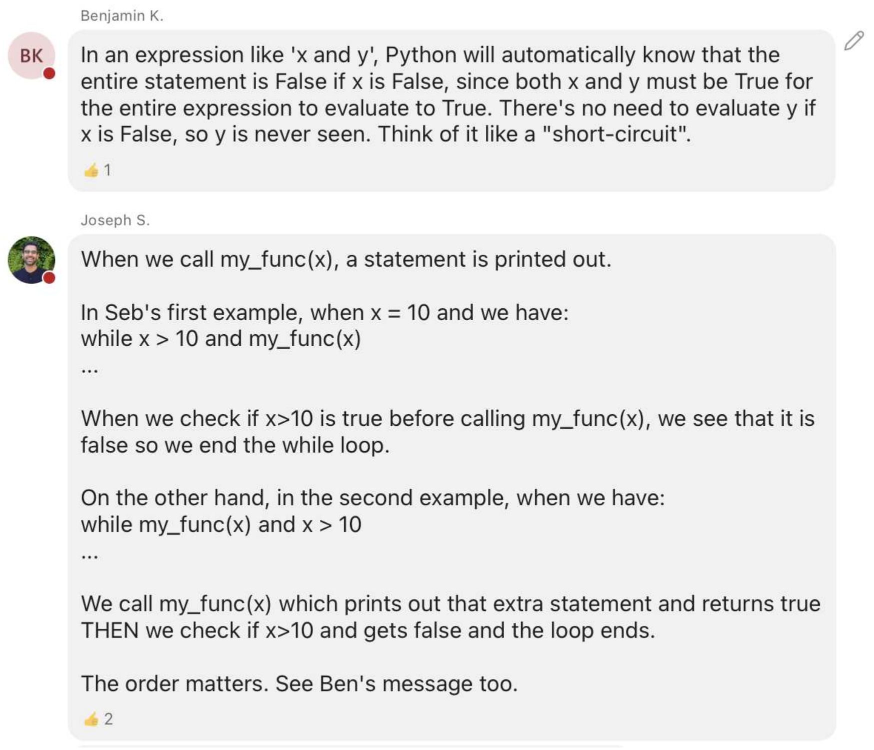
In [1]:
```python
import random

def my_func(x):
    print("Inside my_func, x =", x)
    return True
```

In [2]:
```python
x = 13
while x > 10 and my_func(x):
    x = x - 1
```

```
Inside my_func, x = 13
Inside my_func, x = 12
Inside my_func, x = 11
```

In [3]:
```python
x = 13
while my_func(x) and x > 10:
    x = x - 1
```

```
Inside my_func, x = 13
Inside my_func, x = 12
Inside my_func, x = 11
Inside my_func, x = 10
```



Benjamin K.

BK In an expression like 'x and y', Python will automatically know that the entire statement is False if x is False, since both x and y must be True for the entire expression to evaluate to True. There's no need to evaluate y if x is False, so y is never seen. Think of it like a "short-circuit".

👍 1

Joseph S.

When we call my_func(x), a statement is printed out.

In Seb's first example, when x = 10 and we have:
while x > 10 and my_func(x)
...

When we check if x>10 is true before calling my_func(x), we see that it is false so we end the while loop.

On the other hand, in the second example, when we have:
while my_func(x) and x > 10
...

We call my_func(x) which prints out that extra statement and returns true THEN we check if x>10 and gets false and the loop ends.

The order matters. See Ben's message too.

👍 2

While vs For loops (not on midterm)

Tuesday, February 1, 2022 3:27 PM

for loops	while loops
Number of iterations to be done is already know	Number of iterations not already known, only know when to stop.