LOOPS

Objectives

- Understand what loops are and how they are useful
- Learn what an "iterable object" is
- Use ***for*** and ***while*** loops to iterate over ranges and strings
- Learn how to control exiting a loop

Print numbers 1 through 7

```
print(1)
1
print(2)
2
print(3)
3
print(4)
4
print(5)
5
print(6)
6
print(7)
7
```

That's way too much work!

We can simplify this with a ***for*** **loop** over a **range**.

for loops

In Python, ***for*** loops are written like this:

```
for item in iterable_object:
    # do something with item
```

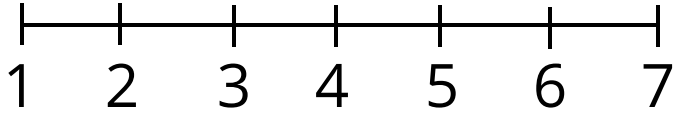
- An **iterable object** is some kind of collection of items, for instance: a list of numbers, a string of characters, a range, etc.
- *item* is a new variable that can be called whatever you want

- *item* references the current position of our **iterator** within the *iterable*. It will iterate over (run through) every item of the collection and then go away when it has visited all items

for loops with ranges

Let's print numbers 1 - 7 using our knowledge of looping thru ranges.

```
for number in range(1, 8):
    print(number)
```

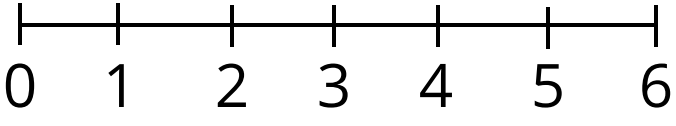



```
1
2
3
4
5
6
7
```

ranges

If we just want to print numbers, we can simply iterate over a **range**.

A **range** is just a slice of the number line.



Python ranges come in multiple forms:

range(7) gives you integers from 0 thru 6 (*shown*) *Count starts at 0 and is exclusive*

range(1, 8) will give you integers from 1 to 7 *Two parameters are (start, end)*

range(1, 10, 2) will give you odds from 1 to 10 *Third param is called the "step", meaning how many to skip. Also, which way to count, up + or down -*

range(7, 0, -1) will give you integers from 7 to 1

REPEATER EXERCISE

```
How many times do I have to tell you? 3
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!

How many times do I have to tell you? 5
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
```

EXERCISE

Is good for you

EXERCISE

Loop through numbers 1-20

- for 4 and 13, print "x is unlucky"
- for even numbers, print "x is even"
- for odd numbers, print "x is odd"

```
1 is odd
2 is even
3 is odd
4 is UNLUCKY!
5 is odd
6 is even
7 is odd
8 is even
9 is odd
10 is even
11 is odd
12 is even
13 is UNLUCKY!
14 is even
15 is odd
16 is even
17 is odd
18 is even
19 is odd
20 is even
```

while loops

We can also iterate using a ***while*** loop, which has a different format:

```
while im_tired:
    # seek caffeine
```

while loops continue to execute while a certain condition is truthy, and will end when they become falsy.

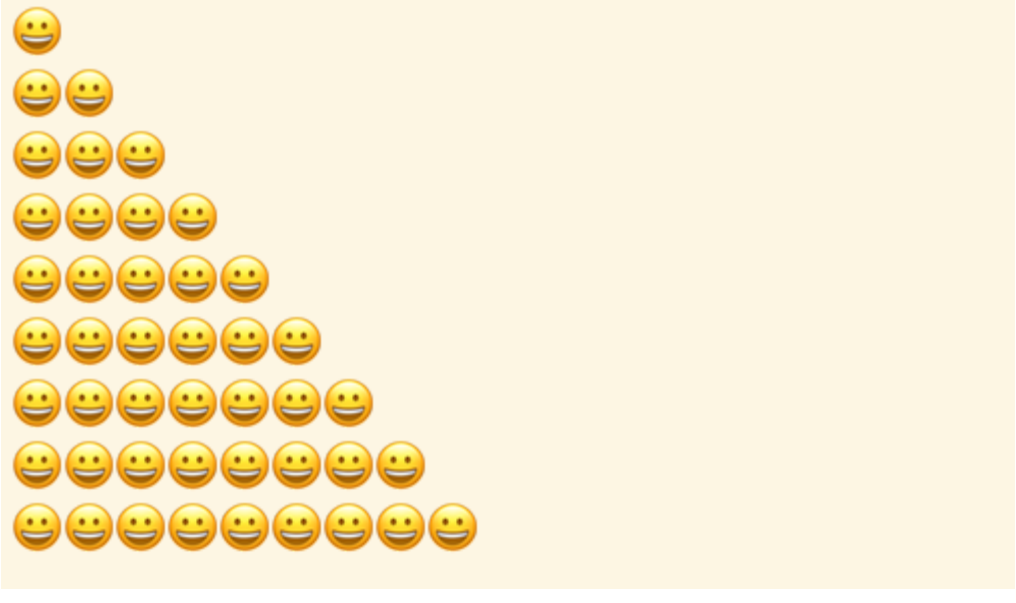
```
user_response = None
while user_response != "please":
    user_response = input("Ah ah ah, you didn't say the magic word: ")
```

while loops require more careful setup than *for* loops, since you have to specify the termination conditions manually.

Be careful! If the condition doesn't become false at some point, your loop will continue *forever*!

while loop exercise

Print the following beautiful art using **both** a for loop and a while loop:



print("\U0001f600")

The"Colt talking to his sister" Exercise

```
Hey how's it going? pretty good, you?
pretty good, you?
hahah
hahah
ok very funny
ok very funny
stop copying me
UGH FINE YOU WIN
```

repeat everything until the user says "stop copying me"

Controlled Exit

The keyword ***break*** gives us the ability to exit out of *while* loops whenever we want:

```
while True:
    command = input("Type 'exit' to exit: ")
    if (command == "exit"):
        break
```

We can also use it to end *for* loops early:

```
for x in range(1, 101):
    print(x)
    if x == 3:
        break
```

ADDING A BREAK

```
times = int(input("How many times do I have to tell you? "))

for time in range(times):
        print("CLEAN UP YOUR ROOM!")
        if time >= 4:
                print("do you even listen anymore?")
                break
```

```
How many times do I have to tell you? 100
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
CLEAN UP YOUR ROOM!
do you even listen anymore?
```

GUESSING GAME

MINI PROJECT

GUESSING GAME

```
Guess a number between 1 and 10: 1
Too low, try again!
Guess a number between 1 and 10: 10
Too high, try again!
Guess a number between 1 and 10: 22
Too high, try again!
Guess a number between 1 and 10: 8
Too high, try again!
Guess a number between 1 and 10: 7
Too high, try again!
Guess a number between 1 and 10: 6
Too high, try again!
Guess a number between 1 and 10: 5
You guessed it! You won!
Do you want to keep playing? (y/n) y
```

ADDING A LOOP TO ROCK PAPER SCISSORS

break

Recap

- Loops are sections of code that keep repeating
- For loops are useful for going through iterable objects
- While loops are more versatile but require more set up
- Any loop can be short-circuited with the break keyword

YOUR TURN