While Loops

while while_loop=="confusing":
read document

How many times do I have to do this?

Code is great for automating processes for us. We can ask our programs to run chunks of code again and again and again. Sometimes, we might not know how many times we want this code to run for. While loops allow us to loop code while a condition is true.

While - condition

What's the situation?

We might not know how many times we want our code to run - but we should know a condition we want it to run under. For example, if I'm generating a random number to match a number I've picked, I don't know how many times that will take - but I do know that **while** the numbers don't match, I want a new random number to be generated.

```
37  import random
38
39  my_num=20
40  random_num=random.randint(1,50)
41
42  while my_num != my_num:
43      print("the numbers don't match, we'll try again!")
44      random_num=random.randint(1,50)
45
46  print("the numbers matched!")
```

We've written our loop to run under a specific situation. **while** loops written in this way usually rely on variables, and their values. Line 42 is like an if statement. We're saying if my chosen number doesn't match the random number, run the code below - but an if statement would just run the code once. A **while** loop will run it again and again and again until the situation changes. In this case, the condition would be that the two numbers match. At this point, the condition we specified on line 42 is no longer True, and the loops breaks, and carries on to the code below.

When we write a **while** loop to say "**while** this condition is true: do this" - we might never see it run, or it might run forever and ever! Be aware you might need to create the chance for the situation to change! In our example above, we generate a random number on line 40 for the **while** loop to compare first, but as part of the loop we regenerate this number, and it's the regenerated number that is compared each time!

```
answer=input("Pick a number between 1 - 4")
while True:  #this is faster than writing out "while answer !="1" and answer!="2" and so on.
    if answer=="1":
        print("You picked 1!")
        break
    elif answer=="2":
        print("You picked 2!")
        break
    elif answer=="3":
        print("You picked 3!")
        break
    elif answer=="3":
        print("You picked 4!")
        break
    else:
        print("That is not between 1-4")
        answer=input("What is the capital of Spain?")
print("This will run after the break")
```

Have a break

You might see a while loop say just say while True. A while True loop will run forever! For a while True loop to work properly, we should combine it with a very important statement - break.

Break does just that - it breaks the loop and allows the program to run to the next executeable line. You'll usually see them around if statements! They could be more efficient than writing out a long list of conditions with and!

....or else

We can include else in our while loops to run once the loop is exhausted.

```
num=0

while num <10:
    num +=1
    print (num)
else:
    print("num is no longer less than 10")
```

Remember!

If you're not using break, the condition will need to change to stop your loop! If you get stuck in a loop, click into your terminnal and use ctrl+c to interupt!

You might feel like you're going loopy - but keep practising while loops!

{cn}®