LEARN CODING

ale66

github.com/ale66/learn-coding

LOOPS

THE `while` INSTRUCTION

We saw `for` to *iterate* over a sequence

A block of code is executed as many times as the number of elements of the sequence

```
1  writers = ['Hemingway', 'Dickens', 'King']
2
3  for w in writers:
4    print(f)
```

__With the while loop we execute a block only as long as a given expression is true_

```
1  i = 1
2  while i < 6:
3    print(i)
4    i += 1
```

increment is not automatic

the expression is evaluated *before* executing the block of code

Try the example above on pythontutor.com

With the break statement we can stop the loop even if the while condition is true:

```
1  i = 1
2  while i < 6:
3    if i == 3:
4      break
5    i += 1
```

INDEFINITE LOOPS

Would execute forever, or until a break condition is met

```
1  while True:
2
3    command = input(''Please enter a command, type 'exit' to stop.'')
4
5    if command == 'exit':
6      break
7    else:
8      # implement commands here ..
```

What is this code doing?

```
1 n = 5
2
3 while n > 0 :
4
5       print(n)
6
7       n = n-1    # or n -= 1
8
9 print('Blastoff!')
```

QUIZ: WHAT'S WRONG WITH THIS LOOP?


```
1  n = 1
2
3  while n > 0 :
4
5          print("Hi")
6
7  print("End")
```

INDEFINITE LOOPS

- some `while` loops are called *indefinitE* because they keep going until a logical condition becomes False
- but will it ever happen?
- codes seen so far are easy to examine to see whether they will terminate or become *infinite*
- termination analysis is about checking, *before running it*, that our code will always reach the final nstruction and control will be back to the operating system
- this activity cannot be automated

`for` VS. `while`

Use `for` to automate work over collections, here represented by iterables

in principle, all elements will be examined

Use `while` to search for a specific element: it stops as soon as it's found

Also use `while` or to act *conditionally* and stop as soon as the condition is not true anymore

while IN LISTS

```
alist = [1, 3, 7, 9]

count = 0 # also works as index

while(True):
      print(alist[count])

      if count == len(alist)-1:
          print("end")
          break

      count += 1
```

```
for anum in alist:
        print(anum)
```