Intermediate python - Control flow structures - 3

One should look for what is and not what he thinks should be. (Albert Einstein)

Chat question

- Let’s recap with a little bit of Python code
- We’ve been talking about the logic of **for loops**, which sometimes involve iterating over a sequence of numbers, or a subset of that sequence
- In the chat, write a string of code to generate a sequence of **odd numbers** from **11 to 30**

Module completion checklist

Objective	Complete
Implement while loops	
Add break/continue statements to the loop	

While loops

- `While` loops are used when you want your program to repeat an action an **unknown number of times**
- They are typically used when we iterate through an object based on a specific **condition**, instead of a concrete set of elements
- When the condition no longer holds true, the program will **exit** the loop

While loops in Python

- `While` loops are defined in Python using the `while` block
- To build a loop, you just need to have your **condition** defined at the top of your loop structure
- Consider our party invitation loop - we can write it as a `while` loop with minor modifications

```
# Set the index to starting point.
i = 0
```

```
# While  the index is less than the
# total number of contacts.
while i < num_contacts:

    print('Invite ' + contact_list[i] + '!')

    # Increase your index to advance
    # to the next name on list.
    i = i + 1
```

```
Invite Christian Bale!
Invite Bradley Cooper!
Invite Willem Dafoe!
Invite Rami Malek!
Invite Viggo Mortensen!
Invite Yalitza Aparicio!
Invite Glenn Close!
Invite Olivia Colman!
Invite Lady Gaga!
Invite Melissa McCarthy!
```

While loops involving a list

- We can perform a `while` loop on a list rather than a numeric comparison
- The condition is `truthy` if the list has elements in it and `falsy` if it is empty
  - This is because of how Python interprets list elements by default in Boolean terms
- Once all the items have been removed with the `.pop()` method and the list is empty, the loop terminates

```
# Create a list
my_list = ['I', 'love', 'Python']
```

```
# While there's element in the list
while my_list:
  # Print the last element
  # and remove it
  print(my_list.pop())
```

```
Python
love
I
```

Will these loops work?

- Consider the following code and the questions below

```
# Press `I` twice to interrupt the kernel and terminate the loop
# Note: Only works if you're in Command mode. If not already enabled, press `Esc` to enable it
k = 0

while k < 10:
    print(k)
```

- What do you think will happen if you try to run these chunks of code?
- Why do you think these loops are not going to work?
- What do you need to do to fix them?

Will these loops work? (cont'd)

- Now consider this code and the questions below

```
while i <= num_contacts:

    print('Invite ' + contact_list[i] + '!')
    i = i + 1
```

- What do you think will happen if you try to run these chunks of code?
- Why do you think these loops are not going to work?
- What do you need to do to fix them?

Module completion checklist

Objective	Complete
Implement while loops	✔
Add break/continue statements to the loop	

Break and continue statements

- The **break** statement immediately terminates a `while` loop
- When you use `break`, program execution proceeds to the first statement following the loop body

```
# break
n = 5
while n > 0:
  n -= 1
  if n == 2:
    break
  print(n)
print('Loop ended.')
```

```
4
3
```

```
Loop ended.
```

Break and continue statements (cont'd)

- The **continue** statement immediately terminates the current loop iteration
- When you use `continue`, execution jumps to the top of the loop, and the controlling expression is re-evaluated to determine whether the loop will execute again or terminate

```
# continue
n = 5
while n > 0:
    n -= 1
    if n == 2:
      continue
    print(n)
print('Loop ended.')
```

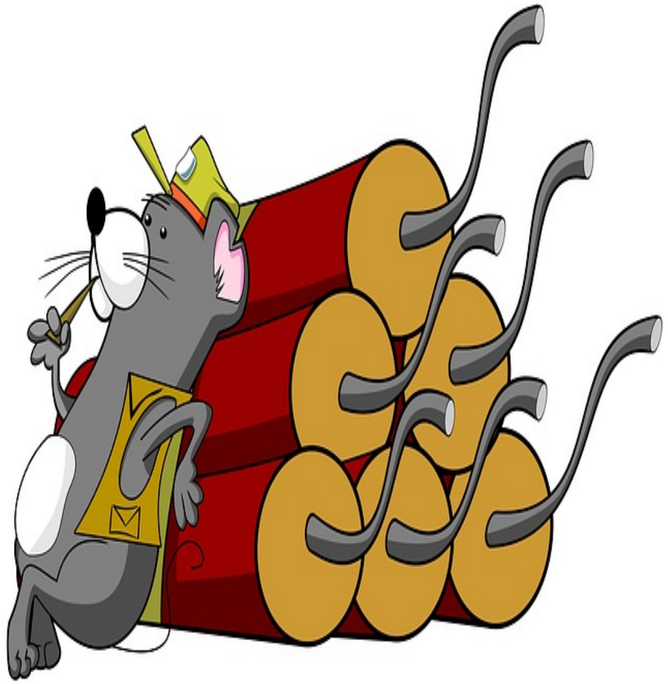
```
4
3
1
0
```

```
Loop ended.
```

Question

- **True or False:** Setting a starting number for a counter is only needed for a `while` loop
- **Type your answer** in the chat

While loops: danger zone!



- The biggest problem with `while loops` is a **poorly defined stopping condition**
- It can happen in cases where
  - the condition itself is faulty
  - the counter is not being increased
  - the index is incorrect
- This can lead to
  - *Infinite loops*
  - Letting the loop go *out of bounds* of a `list` or `array`, which will produce an error

Convert while loop into for loop

- If you can **convert** a `while` loop into a `for` loop, doing so may save you a lot of trouble
- Identify these three main components in the loop if you want to convert:
  - Initialization
  - Condition
  - Increment

Convert while loop into for loop (cont'd)

- Here is a simple example:

```
# while loop
# Initialization:
x=0
# Condition:
while x<5:
  print (x)
  # Increment:
  x=x+1
```

```
0
1
2
3
4
```

```
# for loop
# Initialization: x=0
# Condition: x = [0,1,2,3,4]
# Increment: 1
for x in range(5):
  print (x)
```

```
0
1
2
3
4
```

Knowledge check

Link: ***https://forms.gle/ucPYD7PJB2KM5m3D6***

Exercise

You are now ready to try Tasks 9-11 in the Exercise for this topic

Module completion checklist

Objective	Complete
Implement while loops	✔
Add break/continue statements to the loop	✔

Congratulations on completing this module!