

 [learn-co-curriculum](#) / [dsc-while-loops-break-and-continue-lab](#) Public [View license](#) 0 stars  136 forks [Star](#) [Watch](#) ▼[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Insights](#) [solution](#) ▼

...

This branch is [8 commits ahead](#), [8 commits behind](#) master. [Contribute](#) ▼[lmcm18 fixed learning goals](#) ...on Oct 14, 2019  9[View code](#) [README.md](#)

# While Loops, Break and Continue - Lab

## Introduction

In this lab, we will practice using `while` loops, and `break` and `continue` statements in our code. We will use our control flow statements to iterate through collections and filter out or selectively operate on each element. We'll use `while` loops to perform operations until a given condition is no longer true.

## Objectives

You will be able to:

- Use a `while` loop
- Use `break` and `continue` to add control flow to a `while` loop

## Instructions

## While Loops

Imagine a person named Agnes is finally ready to achieve her dream of becoming a competitive eater! Because the competition is so fierce, she knows she'll need to do lots of training to become number one.

Her first regiment of training consists of eating pizza. Below is a `for` loop version of her training process. Using your knowledge of `while` loops, translate the pizza eating code below from a `for` loop to a `while` loop.

```
slices_of_pie = 6
slices_eaten = 0

for slice in range(slices_of_pie):
    print('Another slice eaten!')
    slices_eaten += 1
    print('Now eaten {} slices!'.format(slices_eaten))
```

```
Another slice eaten!
Now eaten 1 slices!
Another slice eaten!
Now eaten 2 slices!
Another slice eaten!
Now eaten 3 slices!
Another slice eaten!
Now eaten 4 slices!
Another slice eaten!
Now eaten 5 slices!
Another slice eaten!
Now eaten 6 slices!
```

```
slices_of_pie = 6
slices_eaten = 0

# use a while loop to eat each slice of pie
# add each slice to the slices_eaten variable

while slices_eaten < slices_of_pie:
    print('Another slice eaten!')
    slices_eaten += 1
    print('Now eaten {} slices!'.format(slices_eaten))
```

```
Another slice eaten!  
Now eaten 1 slices!  
Another slice eaten!  
Now eaten 2 slices!  
Another slice eaten!  
Now eaten 3 slices!  
Another slice eaten!  
Now eaten 4 slices!  
Another slice eaten!  
Now eaten 5 slices!  
Another slice eaten!  
Now eaten 6 slices!
```

After a long night of training with pizza, Agnes sleeps like a rock. When she wakes up in the morning, she realizes that her journey is not over. It has only just begun! In the next cell, continue her training; this time she'll be eating pancakes. None of the pancakes are prepared yet, and she wants to determine how much time she'll have left over after making all the pancakes. Here are the important details:

- Agnes has 1468 seconds allotted for breakfast today
- Agnes will be making herself 5 pancakes
- Each pancake takes 27 seconds to cook on each side
- It takes an average of 5 seconds to either flip a pancake, add it or remove it from the pan
- There is only room for one pancake at a time on the frying pan
- Remember there are two sides to every pancake!

After Agnes cooks the 5 pancakes, how much time will she have left over to eat them? Use a while loop to find out below.

```
time_for_breakfast = 1468 # in seconds  
number_of_cooked_pancakes = 0  
# use a while loop to make 5 pancakes for breakfast  
# each pancake takes 27 seconds to cook on each side  
# it takes an average of 5 seconds to flip a pancake, add or remove a pancake from t  
# you must decrease the time_for_breakfast each time you  
# add a pancake to the skillet (frying pan) or flip a pancake (i.e. 2 times per panc  
# there is only room for one pancake at a time  
# return how much time is left  
while time_for_breakfast > 0 and number_of_cooked_pancakes < 5:  
    #make a pancake  
    time_for_breakfast -= sum([5,27,5,27,5]) #Add to pan, side 1, flip, side 2, remc
```

```
number_of_cooked_pancakes += 1
time_for_breakfast
```

1123

## For Loops

Fast forward 5 years, and Agnes has become an international competitive eating superstar. Using her stardom, she decides to open up a restaurant chain. As part of a promotional deal, at a grand opening, she tells her fans that if they are part of the first 30 people to the restaurant, there is a 50% chance that they'll receive free food. Agnes executes this by giving each of the first 30 people a number ranging from 0-29. All people who have an even number will receive free food, and all those with odd numbers will sadly remain hungry :(. Use a `while` loop to create two lists below of:

- `hungry_patrons`
- `fed_patrons`

All people will start out in the list of `hungry_patrons`, and only the lucky ones will move to `fed_patrons`.

**Hint:** You may find the [remove method](#) to be useful for the next problem

```
line_of_hungry_patrons = list(range(0,30))
fed_patrons = []
# use a for or while loop to feed the hungry patrons who have an even number
# add the patrons with an even number to the fed_patrons list
# then remove the even numbered patrons from the line_of_hungry_patrons
# each list should contain 15 elements
for patron in line_of_hungry_patrons:
    if patron % 2 == 0:
        line_of_hungry_patrons.remove(patron)
        fed_patrons.append(patron)
print('Those hungry:', line_of_hungry_patrons)
print('Those fed:', fed_patrons)
```

```
Those hungry: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
Those fed: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28]
```

## break And continue Statements

Agnes decides that she wants to start creating targeted advertisements for people. Here is a list of customer objects with information about their name, age, job, pet, and pet name. You'll use loops to find people that meet certain requirements for Agnes' targeted marketing. Write `for` loops with conditional statements in conjunction with `break` and `continue` to get the desired output.

```
people = [  
    {'name': "Daniel", 'age': 29, 'job': "Engineer", 'pet': "Cat", 'pet_name': "Gato"},  
    {'name': "Katie", 'age': 30, 'job': "Teacher", 'pet': "Dog", 'pet_name': "Frank"},  
    {'name': "Owen", 'age': 26, 'job': "Sales person", 'pet': "Cat", 'pet_name': "Cc"},  
    {'name': "Josh", 'age': 22, 'job': "Student", 'pet': "Cat", 'pet_name': "Chat"},  
    {'name': "Estelle", 'age': 35, 'job': "French Diplomat", 'pet': "Dog", 'pet_name': "Helen"},  
    {'name': "Gustav", 'age': 24, 'job': "Brewer", 'pet': "Dog", 'pet_name': "Helen"}  
]
```

Use a `for` loop to find the first person in the list of people that has a dog as their pet. The iteration count shouldn't exceed 2. In your loop add a print statement that says

```
"{person} has a dog! Had to check {number} of records to find a dog owner."
```

The code has been started for you below:

```
first_dog_person = None  
iteration_count = 0  
for person in people:  
    iteration_count += 1  
    if person['pet'] == 'Dog':  
        print('{person} has a dog! Had to check {iteration_count} records to find a dog owner.'.format(person=person, iteration_count=iteration_count))  
        break
```

Katie has a dog! Had to check 2 records to find a dog owner.

Now, use a `for` loop to create a list of all the cat owners who are under the age of 28.


```
# use a for loop to create a list of the cat owners who are under the age of 28  
cat_owners = []
```

```
# for loop goes here
for person in people:
    if person['age'] < 28 and person['pet'] == "Cat":
        cat_owners.append(person)
cat_owners
```

```
[{'age': 26,
  'job': 'Sales person',
  'name': 'Owen',
  'pet': 'Cat',
  'pet_name': 'Cosmo'},
 {'age': 22,
  'job': 'Student',
  'name': 'Josh',
  'pet': 'Cat',
  'pet_name': 'Chat'}]
```

Use a `for` loop to find the first person who is above 29 years old. Use a print statement to state their name and how old they are.

```
# use a for loop to find the first person who is above 29 years old in our list of p
# remember to use a break and or continue statement
# for loop goes here
print("Who's over thirty?")
for person in people:
    if person['age'] > 29:
        print('{} is {}'.format(person['name'], person['age']))
        break
```



```
Who's over thirty?
Katie is 30!
```

Use a `for` loop to create a list of people's names and another list of pet names for all the **dog** owners.

```
# use a for loop to create a list of person names and another list of pet names for
dog_owner_names = []
dog_names = []
# for loop goes here
for person in people:
    if person['pet'] == 'Dog':
        dog_owner_names.append(person['name'])
```

```
dog_names.append(person['pet_name'])
else:
    continue
```

## Level Up

Use a `for` loop to create a list of odd numbers from the list of numbers from 0 to 100. Each time there is an odd number, add 10 to it and append it to `list_of_odd_numbers_plus_ten`. Stop adding numbers to the list when there are 35 numbers in it. Once you have reached 35 numbers, return the sum of the new list of numbers.

```
# use a for loop to create a list of odd numbers from the list of numbers from 0 to
# each time there is an odd number, add 10 to it and append it to the list_of_odd_nu
# stop adding numbers to the list when there are 35 numbers
# use break and continue statements in your code
list_of_numbers = list(range(0,100))
list_of_odd_numbers_plus_ten = []
for number in list_of_numbers:
    if number % 2 == 1:
        list_of_odd_numbers_plus_ten.append(number+10)
    else:
        pass
    if len(list_of_odd_numbers_plus_ten) >= 35:
        break
    else:
        continue
print('Sum:', sum(list_of_odd_numbers_plus_ten))
```

Sum: 1575

## Summary

In this lab, we practiced using `while` loops, which continue executing their block of code until the given condition is no longer true. This is useful for instances where we do not have a collection or do not need a collection to solve our problem, especially when we would only like to stop the process according to a certain condition. We then practiced using control flow statements, `break` and `continue`, to selectively operate on elements, append them to new lists, or assign them to new variables.

## Releases

No releases published

## Packages

No packages published

## Contributors 4



**mike-kane** Mike Kane



**LoreDirick** Lore Dirick



**fpolchow** Forest Polchow



**Imcm18**

## Languages

● Jupyter Notebook 78.5% ● Python 21.5%